Pipeline to Fixing Split-and-Merge Errors in X-Ray Nano-CT Annotations using WebKnossos

Marium Yousuf
University of Arizona

Mark Hereld
Argonne National Laboratory

1 Introduction

The report summarizes and describes the 3D dataset visualization softwares: Neuroglancer and WebKnossos. Neuroglancer and WebKnossos are both web-based visualization tools to view and investigate volumetric data. WebKnossos provides an additional, significant option to annotate existing labels (as compared to just viewing them). The focus of this report is on providing steps to proofread existing image labels using WebKnossos's live annotation features.

We use X-Ray Nano-CT images and their corresponding segmented images to investigate split/merge errors. Section 2 highlights the basic usage of the Neuroglancer package in Python. Section 3 provides an overview of view and annotation modes of WebKnossos. Section 4 provides steps to proofreading annotations, which involves fixing split and merge errors.

2 Neuroglancer

We accessed Neuroglancer using their Python module via Jupyter Notebook. The following are the steps and a brief example to install and use Python's *neuroglancer* package.

- Install the package:

```
pip install neuroglancer
```

- Initiate a neuroglancer viewer:

```
import neuroglancer
viewer = neuroglancer.Viewer()
```

- Assign a coordinate space according to the dataset. For example, for a 3D image dataset with a resolution of $50 \times 50 \times 50$ nanometers (nm), we could assign the coordinate space as follows:

```
neuroglancer.CoordinateSpace(
    names=['x', 'y', 'z'],
    units=['nm', 'nm', 'nm'],
    scales=[50, 50, 50])
```

- Read the image and labeling/segmentation stack:

```
from skimage import io
im = io.imread('imchunk022.tif') # image
# labeled segmentation created using scikit:
gt = io.imread('label0222conn.tif')
gt = gt.astype('uint16')
```

Preprint. Under review.

- Using coordinate space, the image, and the labeling, populate the neuroglancer viewer:

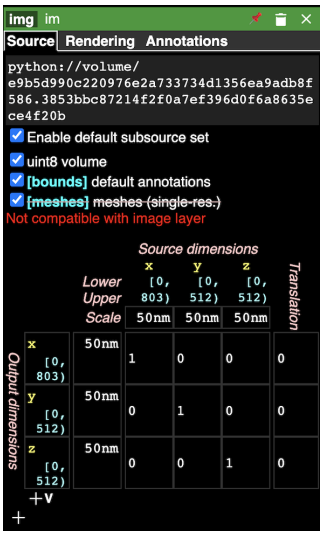
```
def ngLayer(data,res,oo=[0,0,0],tt='segmentation'):
    return neuroglancer.LocalVolume(data,dimensions=res,
        volume_type=tt,voxel_offset=oo)

with viewer.txn() as s:
    s.layers.append(name='im',layer=ngLayer(im,res,tt='image'))
    s.layers.append(name='gt',layer=ngLayer(gt,res,tt='segmentation'))
    s.layers['points'] = neuroglancer.LocalAnnotationLayer(dimensions=res)
```

- Print viewer instance to retrieve the URL to access neuroglancer:

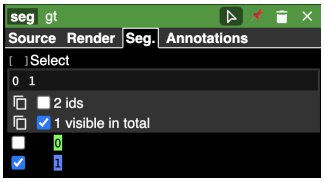
```
print(viewer)
## Output example:
## http://127.0.0.1:52678/v/e0d682c4f9f1626ddae296abe011977338957581/
```

To view the segmentations in Neuroglancer, we paste all the unique segmentation IDs in the segmentation tab in the *layer side panel*. This panel may be accessed by clicking the icon  on the top right corner of the Neuroglancer window.

Figure 1 shows the first view once the Neuroglancer is initiated (a), the view for the non-labeled binary mask (with unique IDs only 0 and 1), and (c) the view for labeled segmentation after pasting all the unique IDs into the segmentation tab. Figure 2 shows an instance of Neuroglancer panels of pairs XY (top-left), XZ (top-right), and ZY (bottom-right). On the bottom left, there is a 3D mesh computed for the binary mask. On the other hand, Figure 3 shows the 3D mesh for the labeled segmented components.



(a) Initial view from accessing the *layer side panel*.



(b) View after right-clicking (or control+click) the segmentation layer tab. This view corresponds to non-labeled segmentation.

(c) View after right-clicking (or control+click) the segmentation layer tab. This view corresponds to labeled segmentation.

Figure 1: Neuroglancer's views for segmentation and image from the *layer side panel* that is access clicking the following icon on the right: 

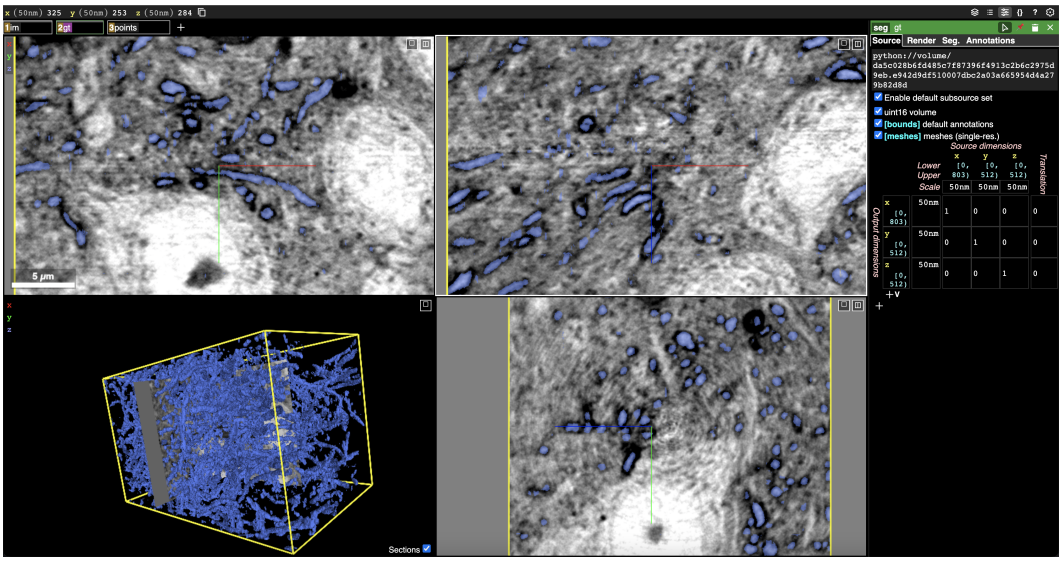


Figure 2: Neuroglancer view showing all non-labeled segmentation meshes.

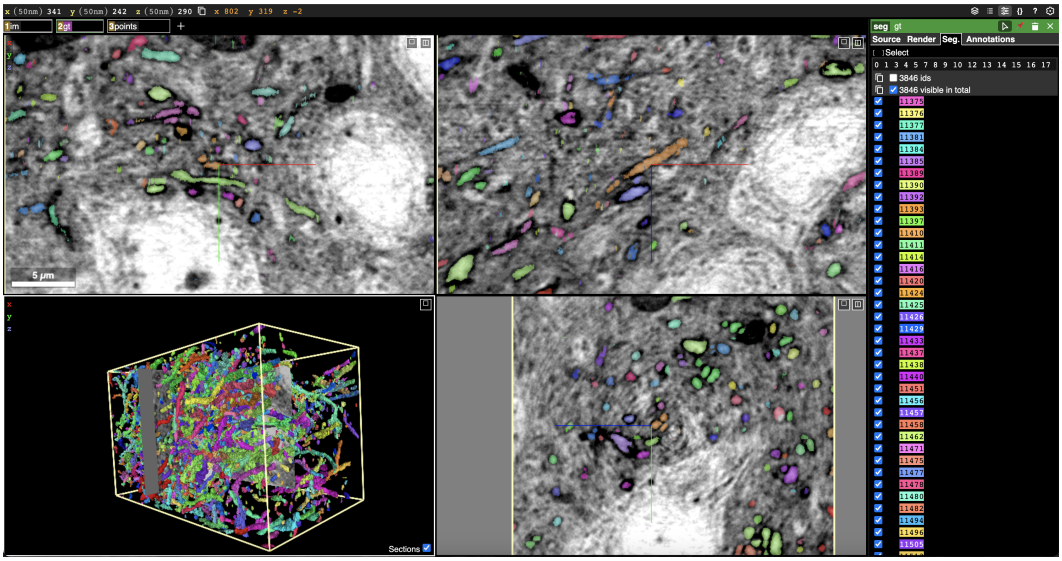


Figure 3: Neuroglancer view showing all labeled segmentation meshes. The labeling was generated with `scikit.measure.label` package with connectivity 2.

3 WebKnosses

WebKnossos is an open-source, in-browser visualization tool for large 3D image datasets. It provides an interface to explore the X-Ray nanoCT datasets with options to create and modify annotations. During the summer, we explored the available tools in WebKnossos in versions *20.03.0-12521* (virtual machine on a server) and *24144* (the current *webknossos.org* version).

Initially, we only focused on the VM Webknossos. Uploading image and segmentation datasets onto the VM introduced some limitations. The server version required uploading the datasets as WebKnossos wrapper (WKW) instead of TIF stacks (as allowed by the website version). The following are the steps to convert the TIF stack into a WKW dataset:

```
webknossos convert -voxel-size 50,50,50 data/source data/target,
```

where the `data/source` is the folder where the image stack is saved and `data/target` is the target folder for the conversion. This folder needs to be a non-existing folder containing the dataset name as suffix, for example:

```
webknossos convert -voxel-size 50,50,50  /Desktop/image/imtif/
/Desktop/image/wkwdataset,
```

for dataset name `wkwdataset`. Here `/Desktop/image/imtif` is a folder containing all of the TIF image slices. Doing the same for the segmented images:

```
webknossos convert -voxel-size 50,50,50  /Desktop/seg/segtif/
/Desktop/image/wkwdataset,
```

where `/Desktop/seg/segtif/` is the folder containing the segmented TIF images, would save two folders `segtif` and `imtif` inside a parent folder `wkwdataset`, together with a json file `datasource-properties.json`. The folder `wkwdataset` can be uploaded as is to the VM WebKnossos to view the images and corresponding segmentation. .

Note: the webknossos convert CLI requires a module pylibCZIrw. This module failed to install on macOS Ventura 13.4.1, Python 3.8-3.11 due to incompatibility with the required wheels. Figure 4 shows screenshots of errors from installing pylibCZIrw and webknossos[czi] (webknossos suggested module to install to use webknossos convert).

```
RuntimeError: Cannot import pylibCZIrw, please install it e.g. using 'webknossos[czi]'
(base) Mariums-MacBook-Pro-4:_Summer2023 mariumyousuf$ /Users/mariumyousuf/anaconda3/lib/python3.8/mul
tiprocessing/resource_tracker.py:216: UserWarning: resource_tracker: There appear to be 12 leaked sema
phore objects to clean up at shutdown
  warnings.warn('resource_tracker: There appear to be %d '
(base) Mariums-MacBook-Pro-4:_Summer2023 mariumyousuf$ pip install 'webknossos[czi]'
Requirement already satisfied: webknossos[czi] in /Users/mariumyousuf/anaconda3/lib/python3.8/site-pac
```

(a) Error from using `webknossos convert`.

```
  note: This error originates from a subprocess, and is likely not a problem wit
h pip.
  ERROR: Failed building wheel for pylibCZIrw
  Running setup.py clean for pylibCZIrw
Failed to build pylibCZIrw
ERROR: Could not build wheels for pylibCZIrw, which is required to install pypro
ject.toml-based projects
```

(b) Error from using `pip install webknossos[czi]`.

Figure 4: Errors from trying to install `pythonCZIrw` on macOS Ventura 13.4.1, Python 3.8-3.11.

3.1 View Mode

The view mode of WebKnossos has three panels: the left panel has layers and a settings tab, the middle panel displays the dataset, and the right panel shows the segmentation information and other existing information (regarding annotated segments).

The Layers and Settings panel shows a histogram of sampled color values of the dataset on a log scale. A slider below the histogram allows adjustment of the contrasts and brightness of the displayed colors. The panel also allows adjustment for opacity and gamma correction for the image stack and opacity and pattern opacity for the segmentation stack. The segmented components all have automatically generated patterns assigned to make them more visually distinguishable.

The middle panel has four windows including three planes: XY (the top-left window), YZ (the top-right window), and XZ (the bottom-left window). On the bottom right of the middle panel is the 3D window, which allows investigation of the segmented components as 3D meshes.

The right panel has four tabs: Info, BBoxes, Segments, and Connectome. The Info tab shows the dimensions and keyboard shortcuts, the BBoxes tab shows custom bounding boxes (if they exist), the

Segments tab shows the annotated segments that have been labeled, and the Connectome tab allows further investigation of incoming and outgoing synapses for a given segment/neuron.

Figure 5 shows a screenshot of WebKnossos view mode.

3.2 Annotation Mode

In the annotation mode, we have multiple tools to create new annotations and modify existing annotations. Table 1 lists all the tools available in the current version of *webknossos.org* and provides brief description for what each tool does.

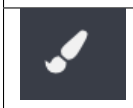
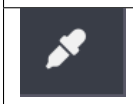
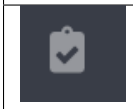
Tool	Function and notes
	Move tool: To move around within the XY, YZ, XZ, and 3D panels.
	Skeleton: To create/select/move nodes. The menu  for this tool includes options to create a new Tree, Single node Tree mode (enabling this creates a new Tree for each node), and Merger mode (enabling this merges the skeletons that connect different segments into a single segment).
	Brush: To draw over voxels for relabeling. More options  allow selecting a segment, picking brush width, overwriting (everything vs. empty areas), and interpolating the current segment between the last labeled and current slice.
	Erase tool: To erase the voxels by brushing over them. Similar to brush tools, there are options to resize and overwrite  .
	Trace tool: To draw an outline around voxels for labeling. More options  allow picking a segment and overwriting existing labels.
	Fill tool: To flood-fill a 2D/3D region; 2D fill is constrained to in-plane, while 3D fill is volumetric fill constrained to a small, regional bounding box. This tool is also used to re-label connected components. Menu options allow picking a segment and a choice of 2D or 3D fill: 
	Segment picker: To select a segment and make its label ID the active segment ID. This tool ensures that any volume annotations stay specific to the currently active segment. 
	Quick select tool: To draw a rectangle around a segment to automatically detect it. Menu options  allow selecting a new segment ID, overwriting existing IDs, and AI-supported quick selection and interpolation of a current segment between the last labeled and current slice (this option is only active when the recent label actions were performed on different slices).
	Bounding box tool: To create, resize, and modify bounding boxes. The menu option  allows creation of a new bounding box centered around the current position.
	Proofreading tool: To fix incorrectly labeled segments. This tool requires an agglomerate file mapping for all segmentation layers.

Table 1: Annotation tools available in WebKnossos (version 24194)

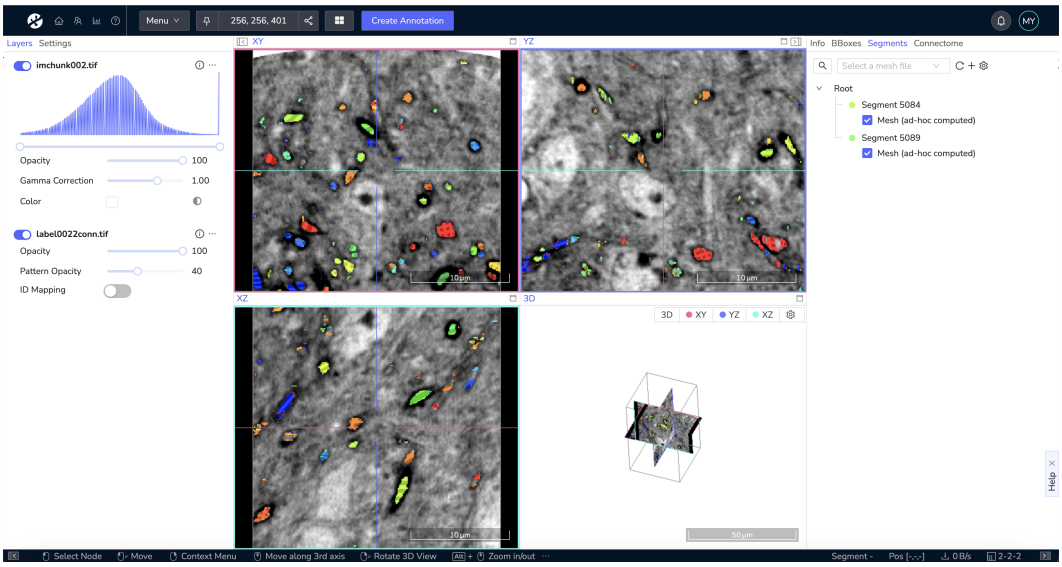


Figure 5: The view mode of WebKnossos with the three planes and 3D window. The left panel shows the settings for image and segmentation and right panel shows a couple of segments with meshes computed for.

4 Proofreading Annotations

4.1 Dataset

The x-ray nanoCT volume has dimensions $1606 \times 2560 \times 2560$ (with each voxel nominaly $50\,\mathrm{nm}$ on a side. This unit order is interpreted as 1606 slices, each composed of a 2560×2560 image plane. To minimize computational intensity while debugging the interfaces we broke this volume up into 50 chunks, each with dimensions $803 \times 512 \times 512$.

We used image stacks of size $1606 \times 2560 \times 512$ extracted from an HDF5 dataset (from file named `slabXX2.h5`) of size $1664 \times 2560 \times 512$. The segmentation stacks of size $2560, 1606, 2560$ were retrieved from a file named `xlbl7.tif`. The segmentation stack did not originally have the same orientation of axes and required swapping the first and second axes. To ensure the image and segmentation stacks matched voxel by voxel, we also used a cropped stack in the third dimension, only using slices 1024 through 1535. After these couple of changes, the segmentation and image stacks matched exactly, both with the dimensions $1606 \times 2560 \times 512$. We further distributed these stacks into 10 chunks with IDs: 002, 012, 022, 032, 042, 102, 112, 122, 132, 142.

Figure 6 show the cropped slab of size $1606 \times 2560 \times 512$ and its distribution into chunks of size $803 \times 512 \times 512$.

4.2 Split and Merge Errors

The proofreading of annotations involves fixing split and merge errors. The split error corresponds to a wrongful splitting of a component that should actually be a single component, while a merge error corresponds to a merge of two components resulting in an incorrectly labeled single component.

Figure 7 shows an example of components that are segmented, labeled, labeled with split/merge errors, and ground truth. The blue patches in the Split/Merge Repairs represent a fix of split errors, while green patches represent a fix of merge errors.

4.3 Steps: Fixing Split/Merge Errors

We focused on chunk 002 to explore WebKnossos annotation tools. Figure The following steps include how image and segmentation chunks were uploaded and examined in WebKnossos, as well as, how WebKnossos can be used to fix split/merge errors:

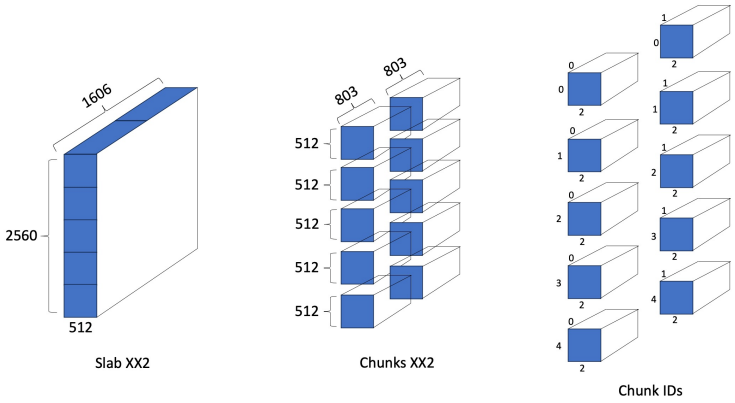


Figure 6: The distribution of provided slab of size $2560 \times 1606 \times 512$ into smaller chunks of sizes $803 \times 512 \times 512$ and their respective IDs, such as 002, 102, 012, 112, etc.

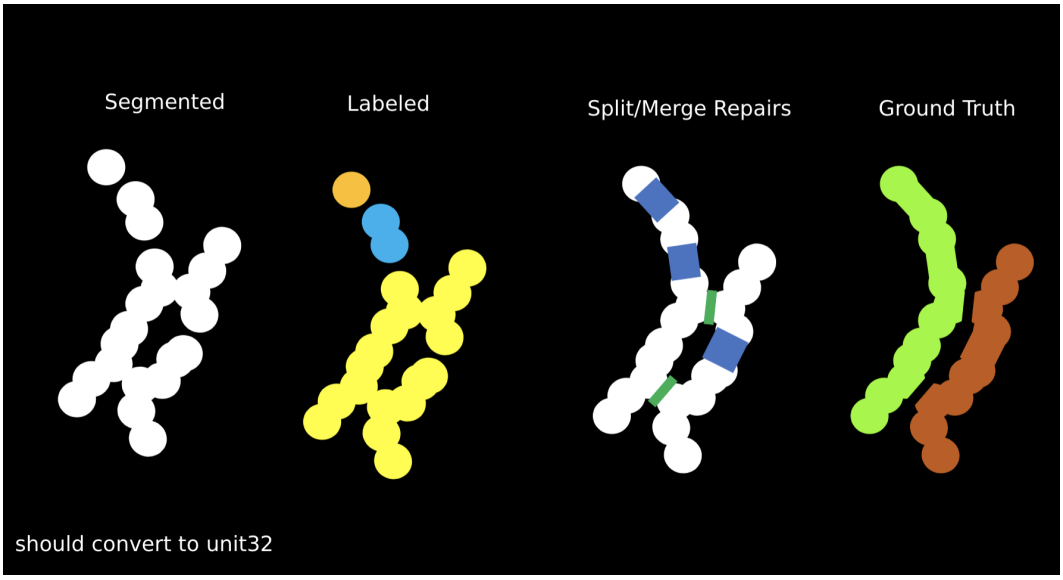


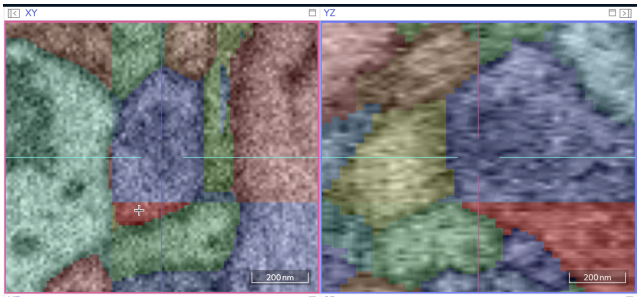
Figure 7: Segmented and labeled components with an example of split/merge repairs and corresponding ground-truth.

- Prepare image and segmentation stacks: The image and segmentation chunks as described in Section 4.1 were saved as TIF files. These TIF files can either be saved as two individual TIF files (one for image and one for segmentation) or they can be saved as multiple slices in folders (one for image and one for segmentation).
- Compute labels for the segmentation: The `scikit.measure` package allows assigning labels to the segmented components. We used `skimage.measure.label` with the connectivity of 2 and background 0 to label the available segmentation for chunk 002. The function `skimage.measure.label(seg_im, return_num=True, background=0, connectivity=2)` returns a labeled image and the count of total labeled components.
- Upload image and segmentation stacks onto WebKnossos: The image and segmentation files can either be uploaded as two TIF stacks with all the slices or two folders including all the slices. To add the dataset, the name of the dataset and the voxel size (in our case 50, 50, 50) is required. Once uploaded, go to the Dashboard and in the Actions tab go to the

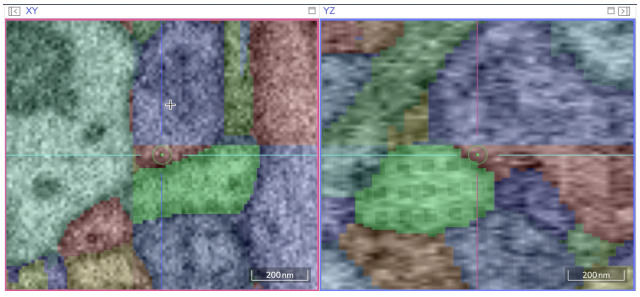
Settings for the uploaded dataset. In the settings, go the Layers and change the category of the segmentation layer to "Segmentation" from "Color/grayscale". Note that the Voxel Size can be updated in the settings, however, the current WebKnossos version does not allow changing the name of the dataset once uploaded.

- Annotation mode and 3D meshes: Finding the split/merge errors requires computation of 3D meshes for the labeled components. To do so, we manually selected components to compute the 3D mesh for. WebKnossos allows precomputing meshes for all the components, however, it requires upgrading to a Team plan or higher. To precompute meshes, go to the right panel then select the Segments tab. Next to the drop-down for "Select a mesh file" is a *plus* sign, clicking which gives opens a window for *Precompute Meshes* with several options of quality. Figure 10 shows the message pop-up on a free usage plan that doesn't allow pre-computation of meshes.

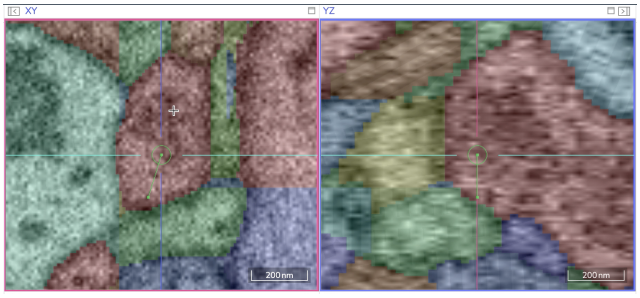
- WebKnossos tools to fix split/merge errors:
  Split: To fix the split error, we could either use a Skeleton tool, a Brush tool, or a Fill tool. Using the skeleton tool and New Tree tool, select one of the split pieces to make its ID the current active ID. Click first on the selected piece to create a node with the same ID and then the rest of the split pieces to assign them the same ID. This would create a skeleton/tree consisting of nodes, as many as the number of split pieces. Figure 8 shows this process in images.



(a) Select one of the pieces that are split.



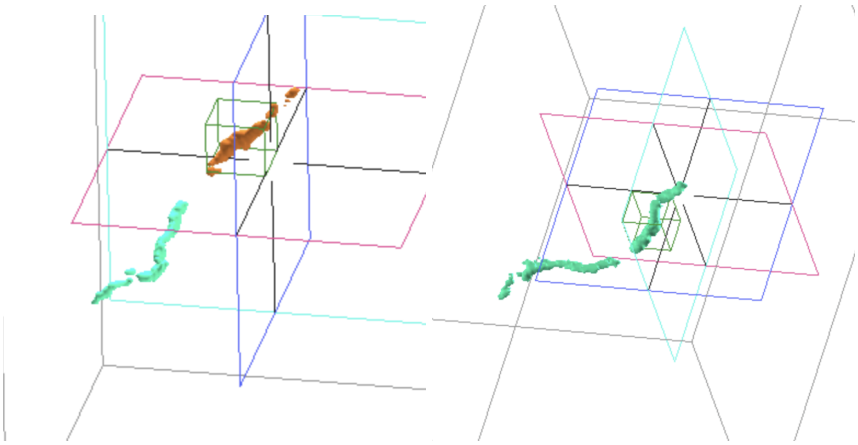
(b) Create a node on the selected piece by clicking on it.



(c) Click on the other pieces of the split to merge them into a single segment.

Figure 8: Example of a split fix using the Skeleton tool.

To avoid creating skeletons/trees of nodes, we can also use the Fill tool to fix the split errors. Like the skeleton tool, Fill Tool requires selecting one of the split pieces to make its ID the current active ID and then clicking to select all the other split pieces to assign them the same ID. Unlike the skeleton tool, this doesn't create trees with nodes. The Fill tool allows both 2D and 3D fills. The 3D fills require close inspections to ensure the IDs of the split pieces were merged for all the slices. Additionally, the Brush tool can be used in complement to the Fill tool to fill the apparent gaps/holes in the segments. Figure 9 shows an example of a split fix using Fill and Brush tools for a segment that was incorrectly identified as two separate segments.



(a) (b)

Figure 9: Example of a split error fix using the Fill and Brush tool.

Merge: If two segments are conjoined that should be separate, the Erase tool can be used to divide them into different segments and then the Fill tool can be used to assign them different IDs. In certain slices, the two segments are not conjoined but have the same ID; in this case, we can just use the Fill tool to fix the merge.

Figure 10 shows an example of segments incorrectly labeled due to split and merge errors. The YouTube video `https://www.youtube.com/watch?v=91fWa9otUbw` by WebKnossos shows a demo of fixing split and merge errors..

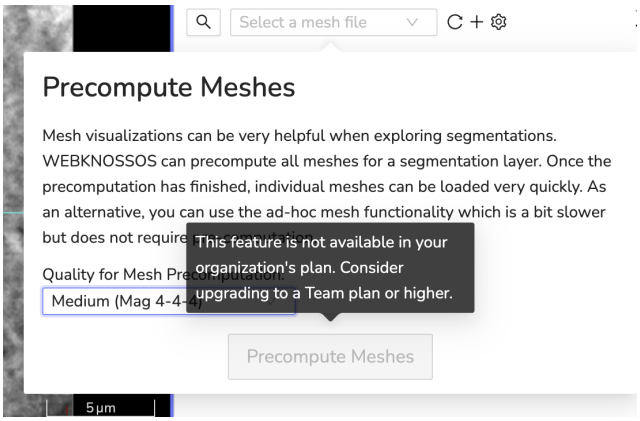


Figure 10: The message from WebKnossos shown while trying to precompute meshes for all existing segmented components: an upgraded WebKnossos usage plan is required to use this feature.

- Export updated annotation: To save new/modified annotations, go to the Menu dropdown on the left of the annotation toolbar. Click download and choose the TIFF Export tab. Two export formats are available: OME-TIFF and TIFF stack (as .zip). Choose the format and

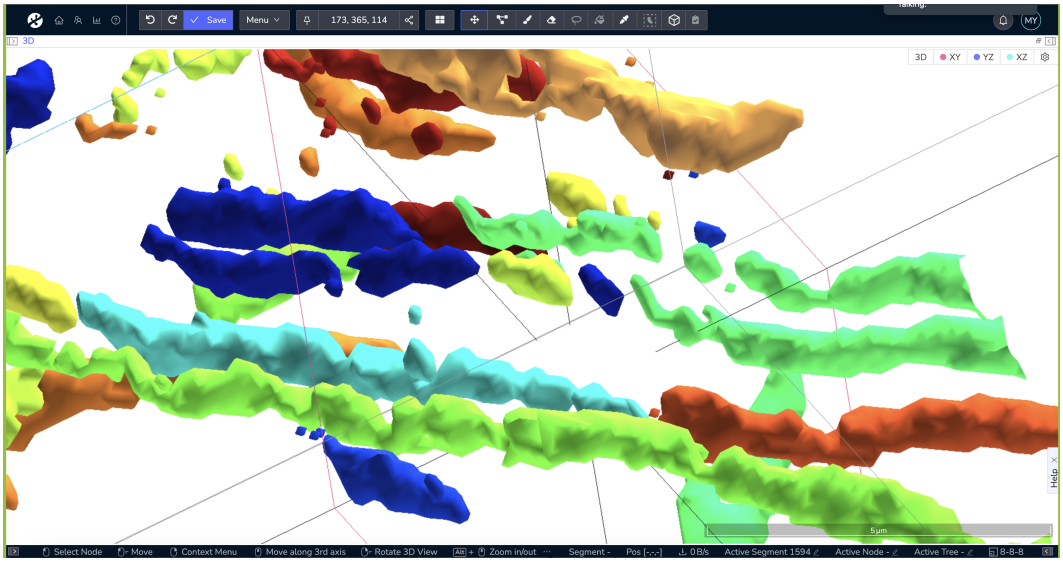


Figure 11: An example of merge and split error: the two myelinated axons in dark blue and light green in the middle are incorrectly labeled. The axons are split lengthwise, while they should have the same label. Moreover, the green segments are merged into one label, while they have two separate label IDs. These segments are from chunk 002 and have WebKnossos IDs: 2240 (the bigger dark blue segment on the left), 64 (the light green segment on the right), 1835 (the yellow component in the middle), 1594 (the smaller, dark blue component in the middle).

click `Export` on the bottom-right of the window. This would save a Zipped folder with new labeled slices with modified annotations.

Remarks

WebKnossos: Focus on precomputed files (mesh files, labeling, ID mappings), does not allow renaming the dataset once uploaded

Neuroglancer: No proofreading tools to fix split/merge errors

Original Goal: Learn how to repair merge and split errors on x-ray nanoCT data using available tools developed by the connectome community for EM stacks. Then apply this tech to annotating a reference volume of x-ray data.

Reason: The resulting ground truth data, showing all of the myelinated axons in the volume, would allow us to (1) evaluate statistics of axons for estimating requirements, (2) evaluate algorithms for identifying axons, and (3) evaluate algorithms for repairing imperfect identifications.

Findings, progress, and lessons learned:

- Neuroglancer is a very easy to use and powerful viewer for this problem. It allows overlay in 2D and 3D of original x-ray image data with segmentation data (axon identifications) and/or labeled data (one color per connected component, ideally an isolated axon). It can be quickly installed using pip and invoked within a python script on a Mac. It will generate the necessary meshing for 3D viewing of the axons. It doesn't provide support for annotation of the sort we need to identify and repair merge or split errors.
- webKnossos is meant to do all of the above and is the go-to tool for the connectome community as we understand it. It is available in two forms: (1) full feature online installation at webknossos.org, and (2) freely available github source for a critically watered down version.
- An installation of the github version on ALCF hosted virtual machine was made available to us, and we found it to be unusable for annotation because very little of the functionality was

available on this free version. We had hoped to work locally so that we could completely secure the x-ray data.

  - Even loading data into this open source version was onerously difficult. Only their home-grown format (.wkw) could be loaded, and the software intended to convert more conventional formats (image stack as folder full of tiffs) would not work on MacOS.
  - Using a virtualbox VM running linux on a Mac we could configure that VM to do the format conversions. But when we found that the functionality of the targeted open source version of webKnossos running on the ALCF VM was catastrophically under powered, we bailed on this line of work.
  - Ultimately and by necessity, we found that the security model of the online unpaid version at webknossos.org is sufficient, and that we only exposed small chunks of the data at a time in any case.

- The webknossos.org version was helpful, but would require purchase of features to enable convenient meshing (beyond one-at-a-time component meshing, essentially manual). Still, we used the time to work through process to understand how we might proceed. Left with the impression that even the best available tools don't measure up.
  - **split errors**: It is pretty easy to see that two isolated axon fragments ought to be connected. And it is also pretty easy to cause the two differently labeled (colored) components to be relabeled to match, there doesn't appear to be much available to facilitate filling in the gap between the fragments to create a continuous axon. The available tools enable operations in 1D and 2D that could be useful (tedious), but these fall very short.
  - **merge errors**: We found even less to support merge errors, which can be quite complicated, consisting of inadvertent connections between axons at many different points.
  - It is worth noting that a premium tool (requiring payment) called *Proofreading* is available that aims at both of these features. We were unable to try it. Documentation for the feature can be found at `https://docs.webknossos.org/webknossos/proof_reading.html`.
- Our net was an understanding of all of these components of the landscape, and specific steps required to perform most of the operations we would need to fully annotate the x-ray volume.