Decimation of EM Volume Mesh Objects with Pyvista

Examples from MICrONS and H01 Volumes

Volume Examples

pinky	▪ MICrONS Layer 2/3 mouse visual cortex
minnie	▪ MICrONS 1mm^3 mouse visual cortex
H01	▪ Harvard/Google human temporal cortex

Decimation Process

Download

- Download mesh files to local disk *(or Google Drive for Google Colab)*

Decimate

- Process and decimate mesh files *(creates a new ply file 95% smaller)*

Visualize

- Visualize decimated mesh objects *(all examples shown here use vtk)*

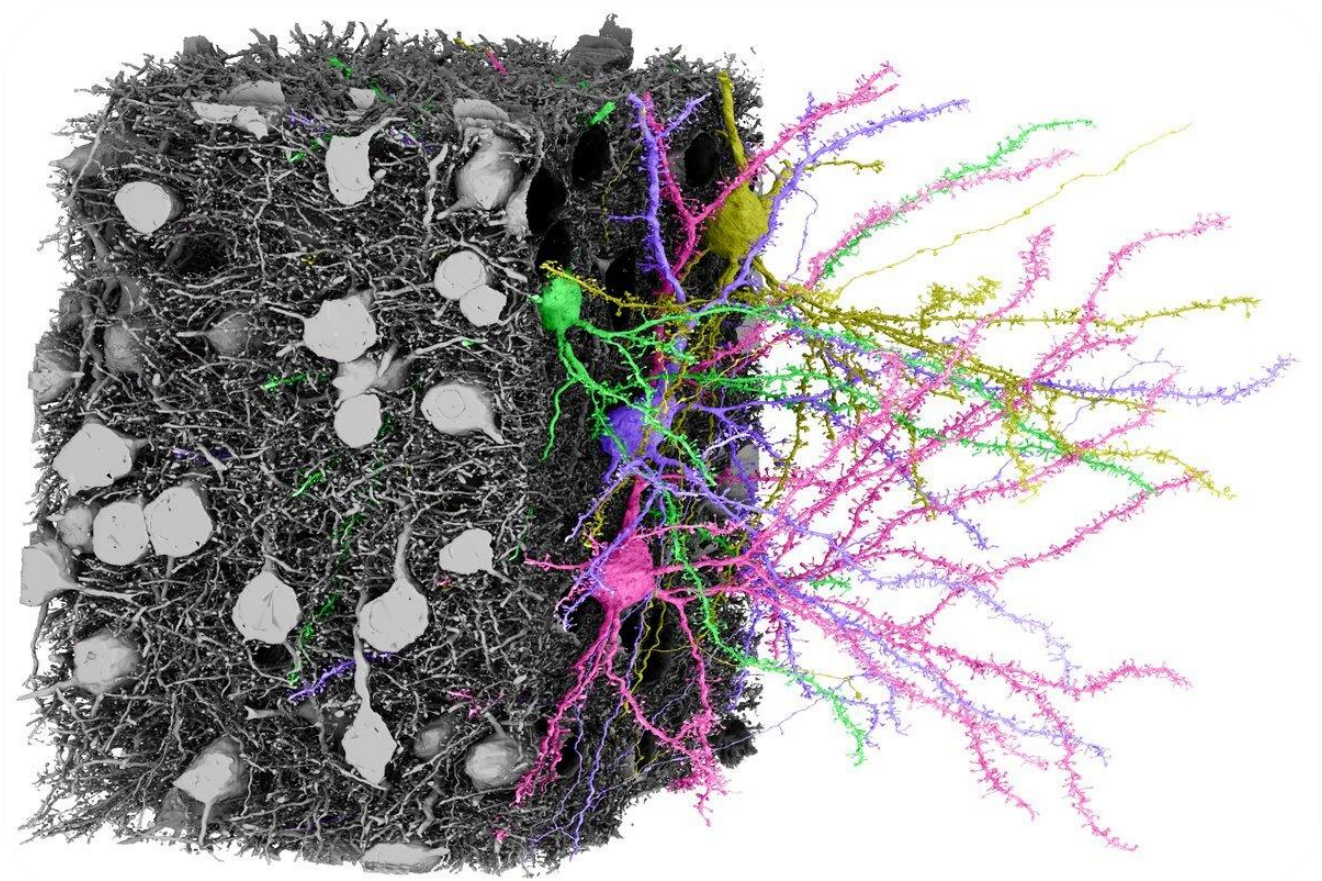
Source: downloading meshes

- Mesh download code from Forrest Collman's MeshExample.ipynb
- Thank you Bethanny Danskin (Allen Institute) for providing a workaround for meshparty legacy file handling error
- See Allen Institute Microns Binder github repository for more details and additional notebooks for working with the MICrONS volume datasets

Source: pyvista mesh decimation

- Mesh decimation code from Tyler Sloan's Mesh Decimation Pipeline.ipynb
- See Quorumetrix github repository for more details and additional scripts for working with volume meshes in Blender (however, all visualization examples shown here use vtk/OpenGL)

MICrONS Layer 2/3 (pinky) volume



Explore EM data from layer 2/3 of the mouse visual cortex

Mesh Types in Layer 2/3 Volume

Cell	▪Cell soma meshes
Mito	▪Mitochondria meshes
Nuc	▪Nuclei meshes

Layer 2/3: download meshes

Visit the download_meshes folder on github

Use the respective notebooks to download and view the cell, mitochondria, and nuclei meshes

The cell body and nuclei notebooks also visualize the downloaded meshes with matplotlib and plotly plots

Mesh download for cell bodies

```
# setup the mesh meta to handle downloads and caching
mesh_dir = 'data/neuron_meshes_v185/'
seg_source = "precomputed://gs://microns_public_datasets/pinky100_v185/seg"
cv_obj = cloudvolume.CloudVolume(seg_source, use_https=True)
seg_id = [648518346349525821]
```

Workaround code for meshparty legacy issue with Layer 2/3 cell body mesh files

Mesh download for cell bodies

```
# Iterating through the seg_id list
for seg in seg_id:
    print(f"Processing seg_id: {seg}")

    # Retrieve the mesh object with tqdm progress bar
    print("Retrieving mesh...")
    cv_mesh = cv_obj.mesh.get(seg,
remove_duplicate_vertices=True)
```

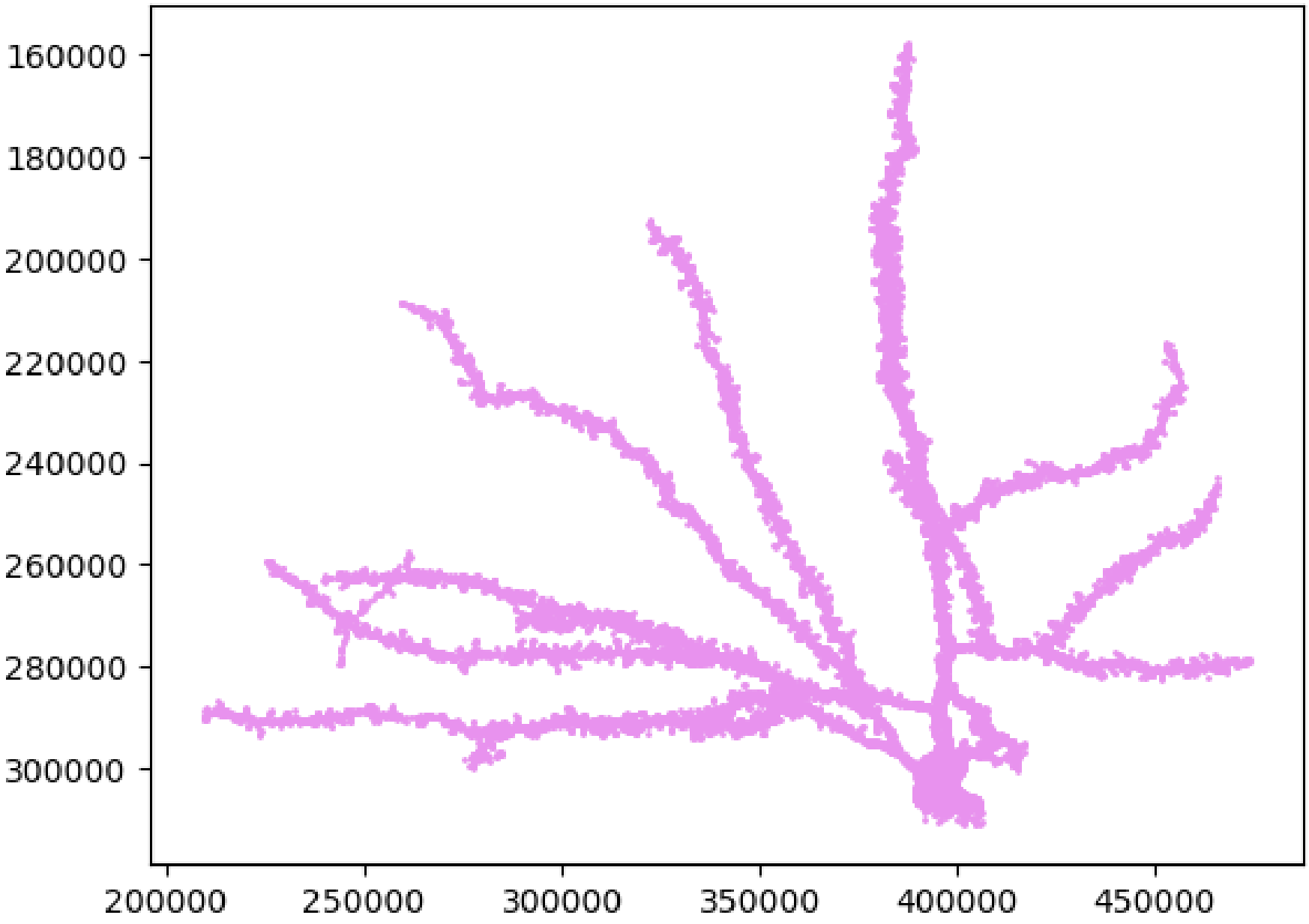
Mesh download for cell bodies

```
# Extracting and reshaping faces if necessary
print("Processing mesh faces...")
faces = np.array(cv_mesh.faces)
if len(faces.shape) == 1:
    faces = faces.reshape(-1, 3)

# Creating the Mesh object
print("Creating Mesh object...")
mesh = Mesh(vertices=cv_mesh.vertices, faces=faces)
```

Mesh download for cell bodies

```
# Constructing the file path
mesh_file = os.path.join(mesh_dir, str(seg) + '.h5')

# Writing the mesh to file
print(f"Writing mesh to file: {mesh_file}")
mesh.write_to_file(mesh_file)

print(f"Finished processing seg_id: {seg}\n")
```

matplotlib visualization of a pyramidal cell mesh



Mesh download for mitochondria

```
# setup the mesh meta to handle downloads and caching

mito_mesh_dir = 'data/meshes/'
mito_source = "precomputed://https://td.princeton.edu/sseung-archive/pinky100-mito/seg_191220"
mito_mm = trimesh_io.MeshMeta(cv_path=mito_source,

disk_cache_path=mito_mesh_dir)
```

Mesh download for mitochondria

```
cellid_list = [648518346349529031]

mitodf =
mito[mito['cellid'].isin(cellid_list)]
mitodf_mitoid_list = mitodf.mito_id.to_list()
```

Mesh download for mitochondria

```
for i in range(len(mitodf_mitoid_list)):
    mito_id = mitodf_mitoid_list[i]
    mito_seg_id = mito_id
    mito_downloadmesh = mito_mm.mesh(seg_id =
mito_seg_id, remove_duplicate_vertices=True)

    print(f'completed ' + str(i) + ' of ' +
str(len(mitodf_mitoid_list)))
```

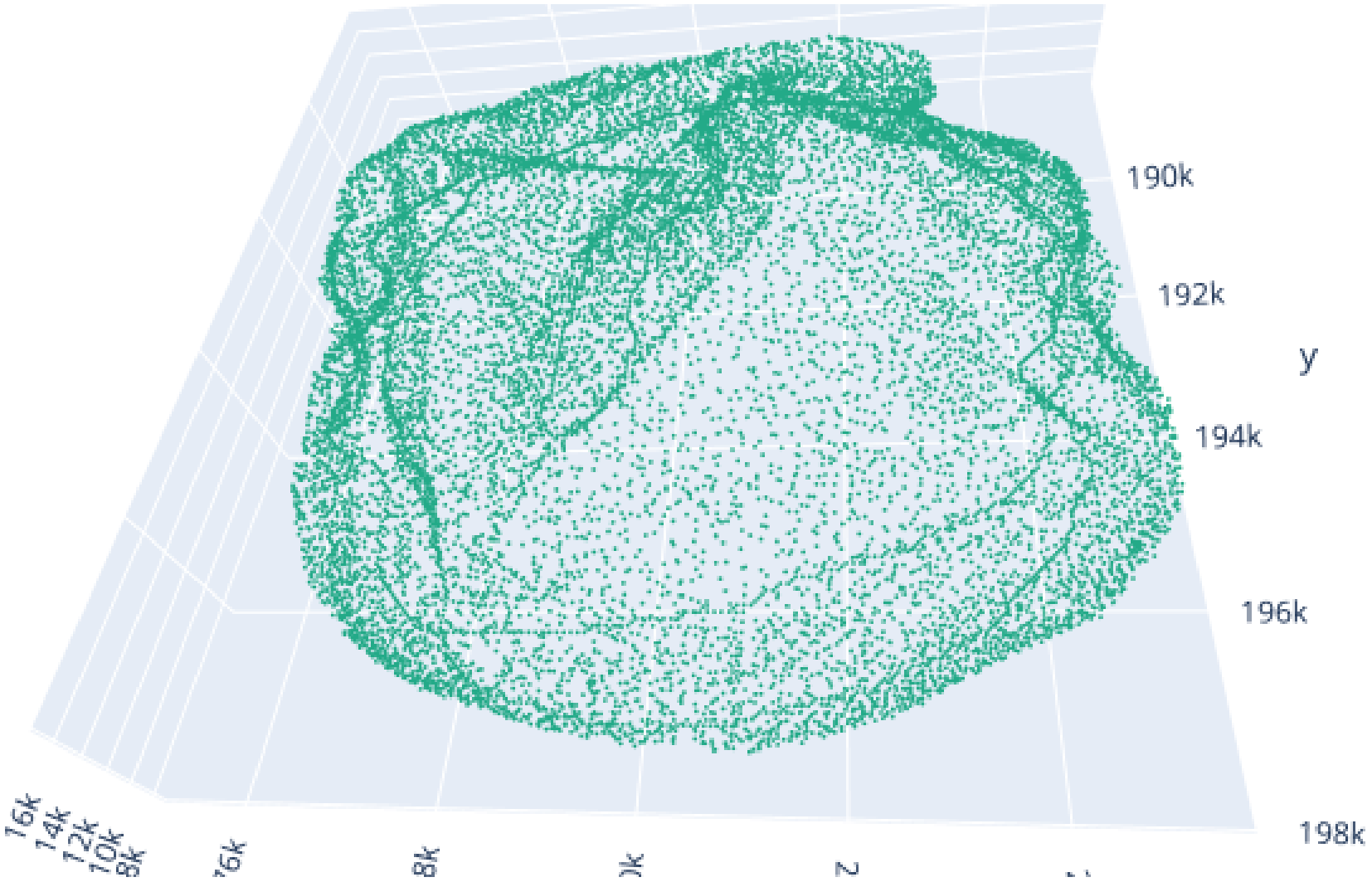
Mesh download for nuclei

```
nuc_mesh_dir = 'data/nuclei_meshes/'
nuc_source =
"precomputed://https://td.princeton.edu/sseung
-archive/pinky100-nuclei/seg"
nuc_mm =
trimesh_io.MeshMeta(cv_path=nuc_source,

disk_cache_path=nuc_mesh_dir)
```

Mesh download for nuclei

```
nuc_id = [5645]

for i in tqdm(range(len(nuc_id)),
desc="Downloading Meshes"):
downloadmesh = nuc_mm.mesh(seg_id=nuc_id[i],
remove_duplicate_vertices=True)
```

plotly visualization of a nucleus mesh



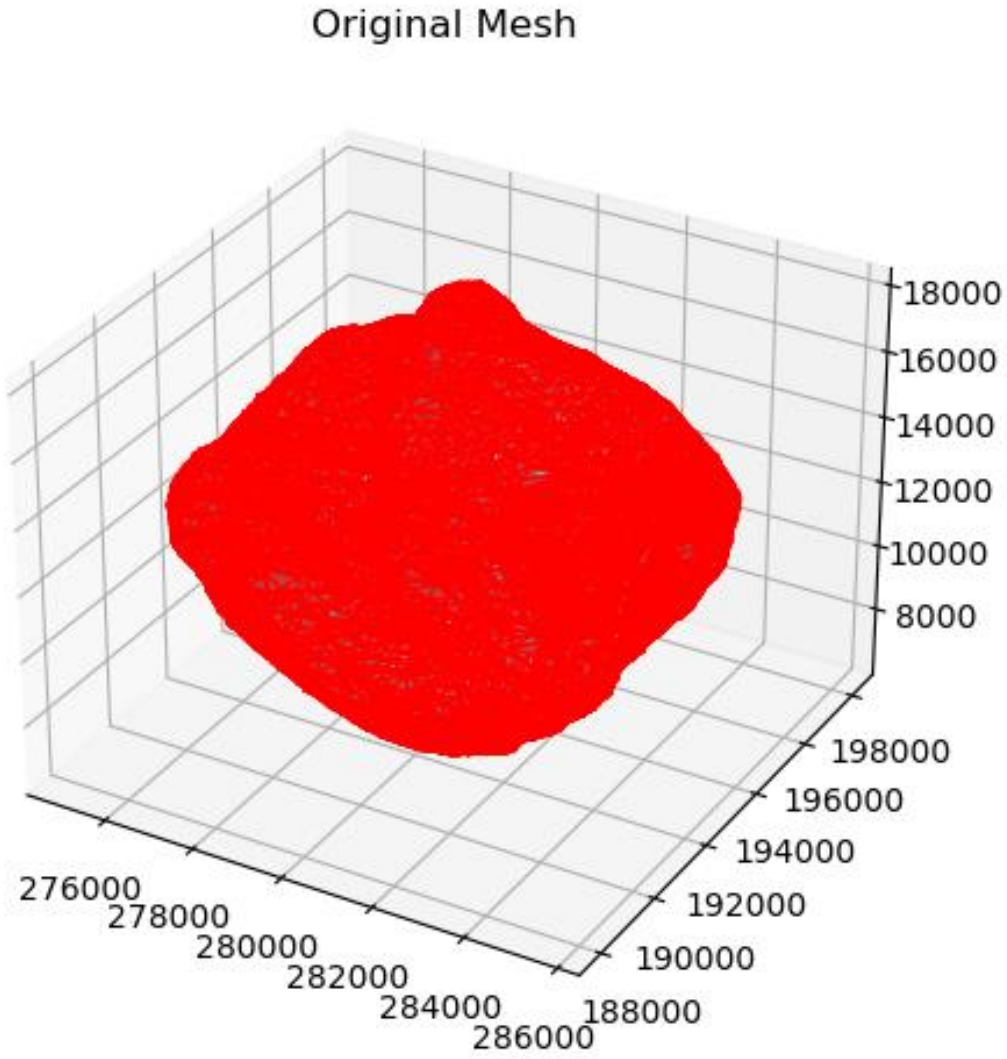
Layer 2/3: decimate meshes

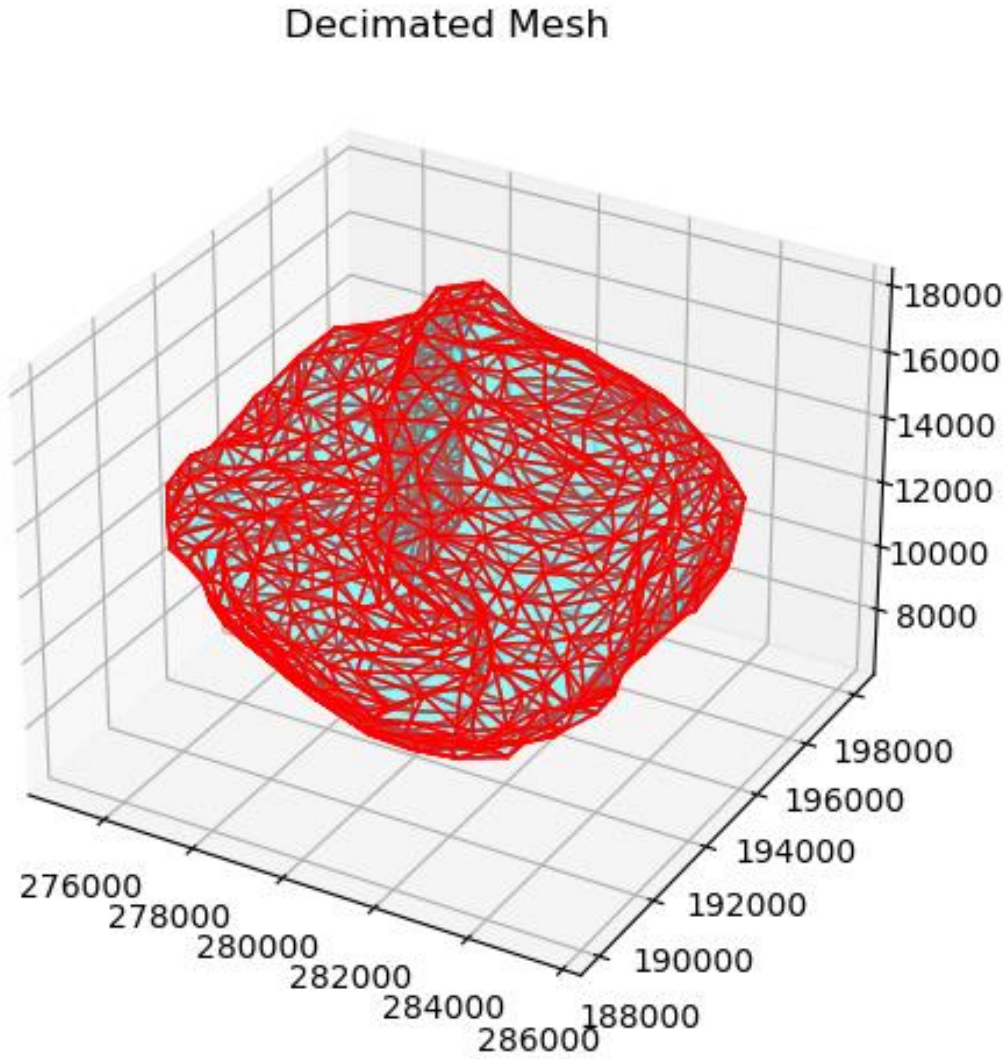
Visit the decimated_meshes folder on github

Use the respective notebooks to decimate the cell, mitochondria, and nuclei meshes

Options for single and batch processing meshes using lists

Mesh decimation: nucleus example

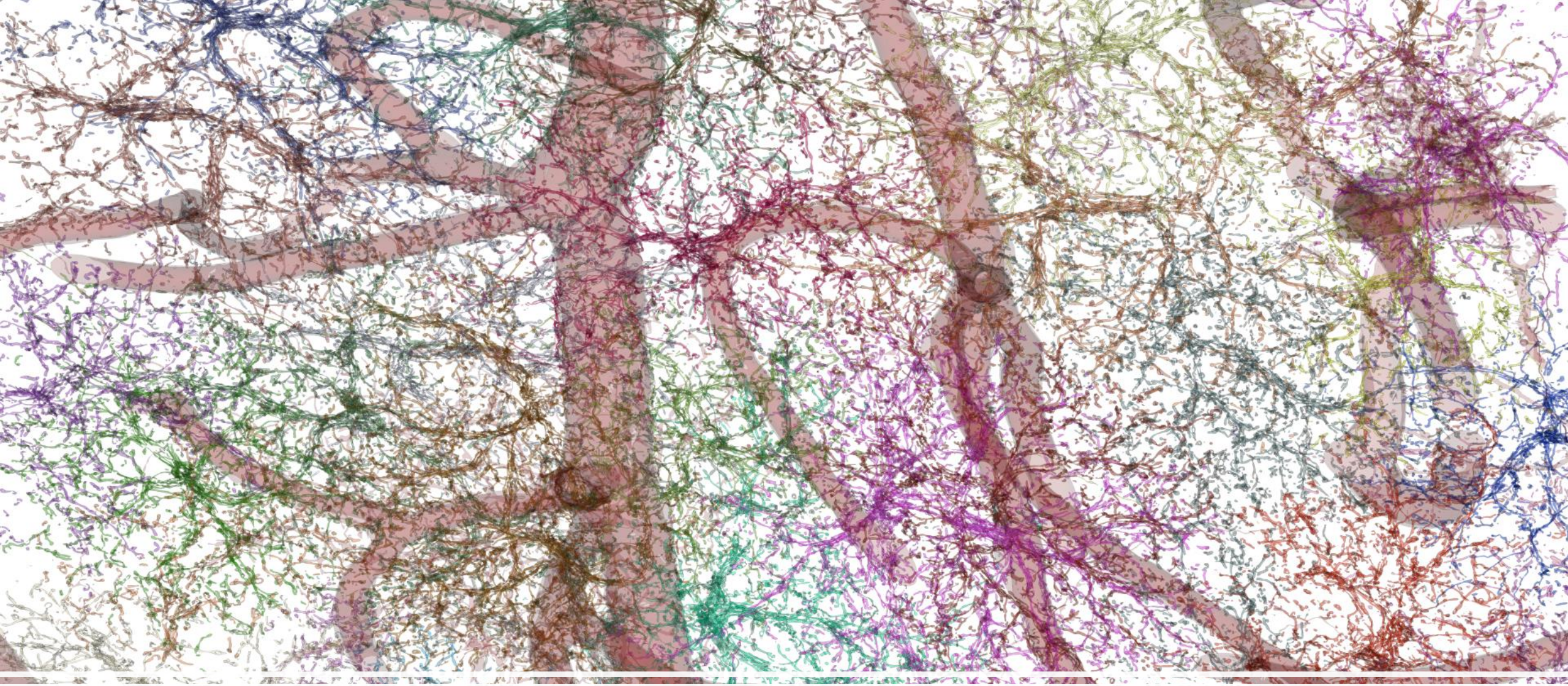




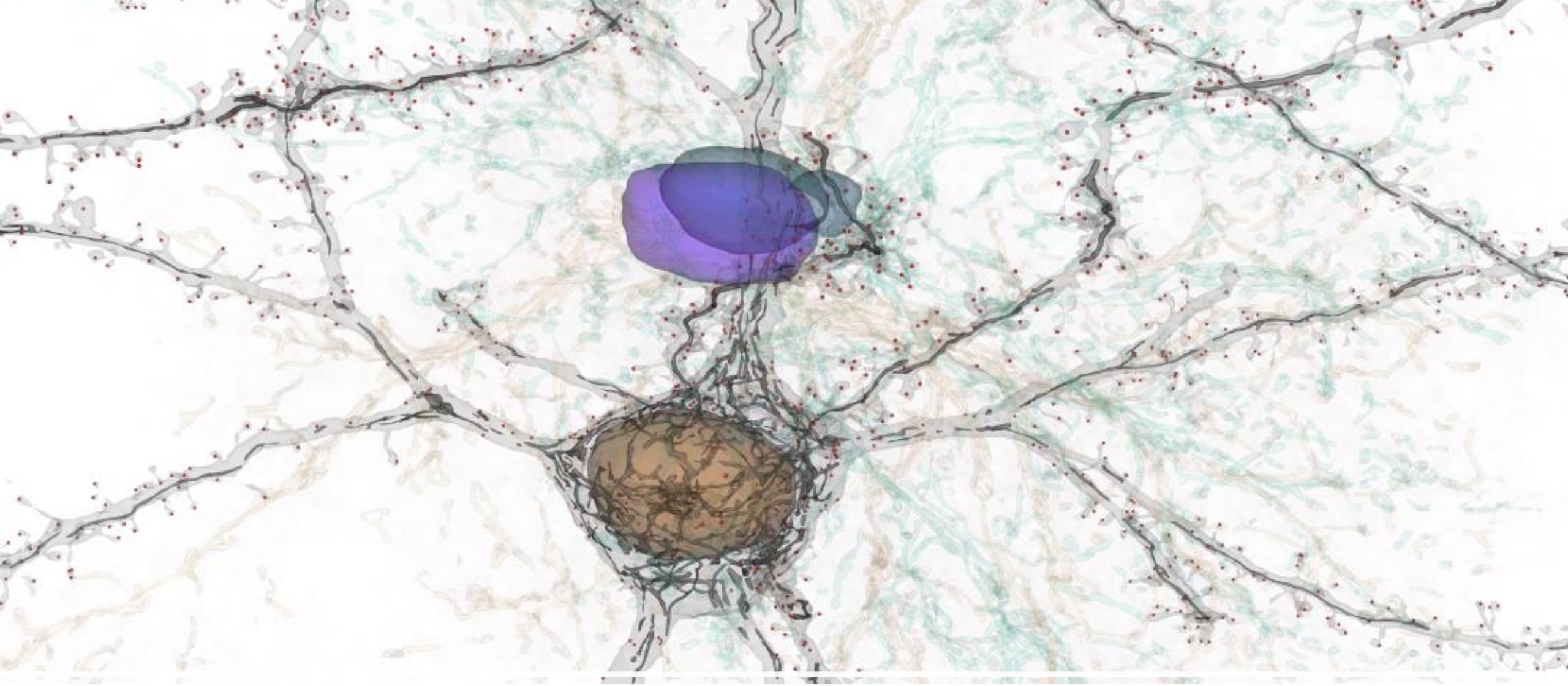
Layer 2/3 volume: visualize meshes

Visualize decimated meshes using a variety of methods with vtk with customized notebooks in the decimated_meshes directory

Additional options include adding pre- and post-synaptic sites, filtered for pre-synaptic sites for a neuron of interest, highlighted cells, highlighted mitochondria by size, and more

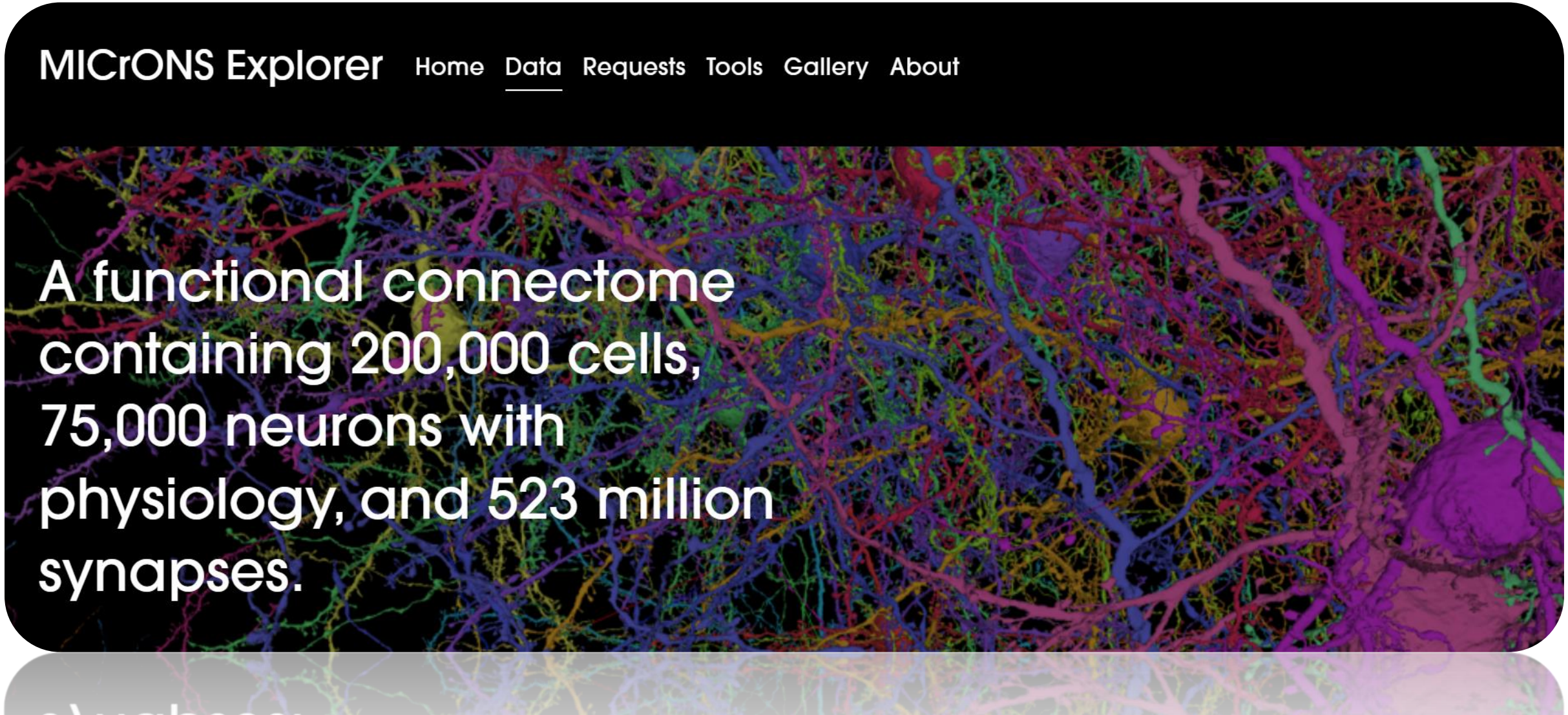


All mitochondria in 44 astrocytes in the Layer 2/3 volume with vasculature shown in red



Two astrocyte cell bodies distort the apical dendrite of a Layer 2/3 pyramidal neuron

MICrONS 1 mm^3 (minnie) volume



minnie volume: download meshes

Visit the Jupyter Notebook folder on github

Use minnie_plotly_and_vtk_visualizer.ipynb to download and view cell meshes (list-based approach)

Mesh download

```
# setup the mesh meta to handle downloads and caching
mesh_dir = 'data/cell_meshes'
seg_source = "precomputed://gs://iarpa_microns/minnie/minnie65/seg"
mm = trimesh_io.MeshMeta(cv_path=seg_source,
                         disk_cache_path=mesh_dir,
                         cache_size=20)
```

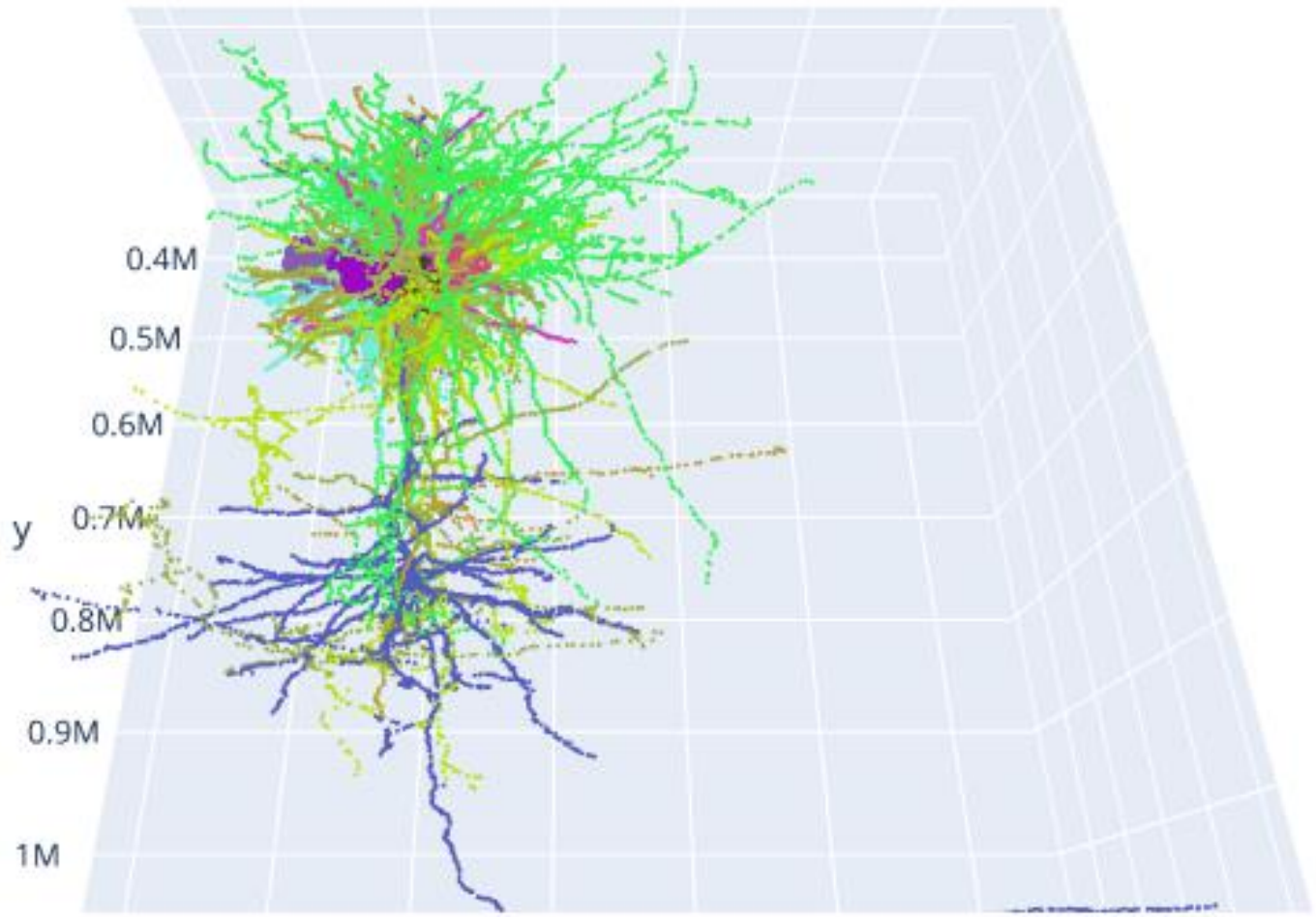
Mesh download

```
cell_id_list = [8646911349644462 39,
8646911350125247 90, 8646911352561386 71,
8646911352699132 53, 8646911353378513 66,
8646911353414217 45, 8646911353482398 31,
8646911353863632 65, 8646911354977502 91,
8646911356577831 70, 8646911358094460 92,
8646911358371828 67, 8646911358804052 61,
8646911360239332 41, 8646911361940080 22,
8646911363303940 07]
```

Mesh download

```
for i in tqdm(range(len(cell_id_list)),
desc="Downloading Meshes"):
    downloadmesh = mm.mesh(seg_id=cell_id_list[i],
    remove_duplicate_vertices=True)
```

Plotly visualization of mesh objects



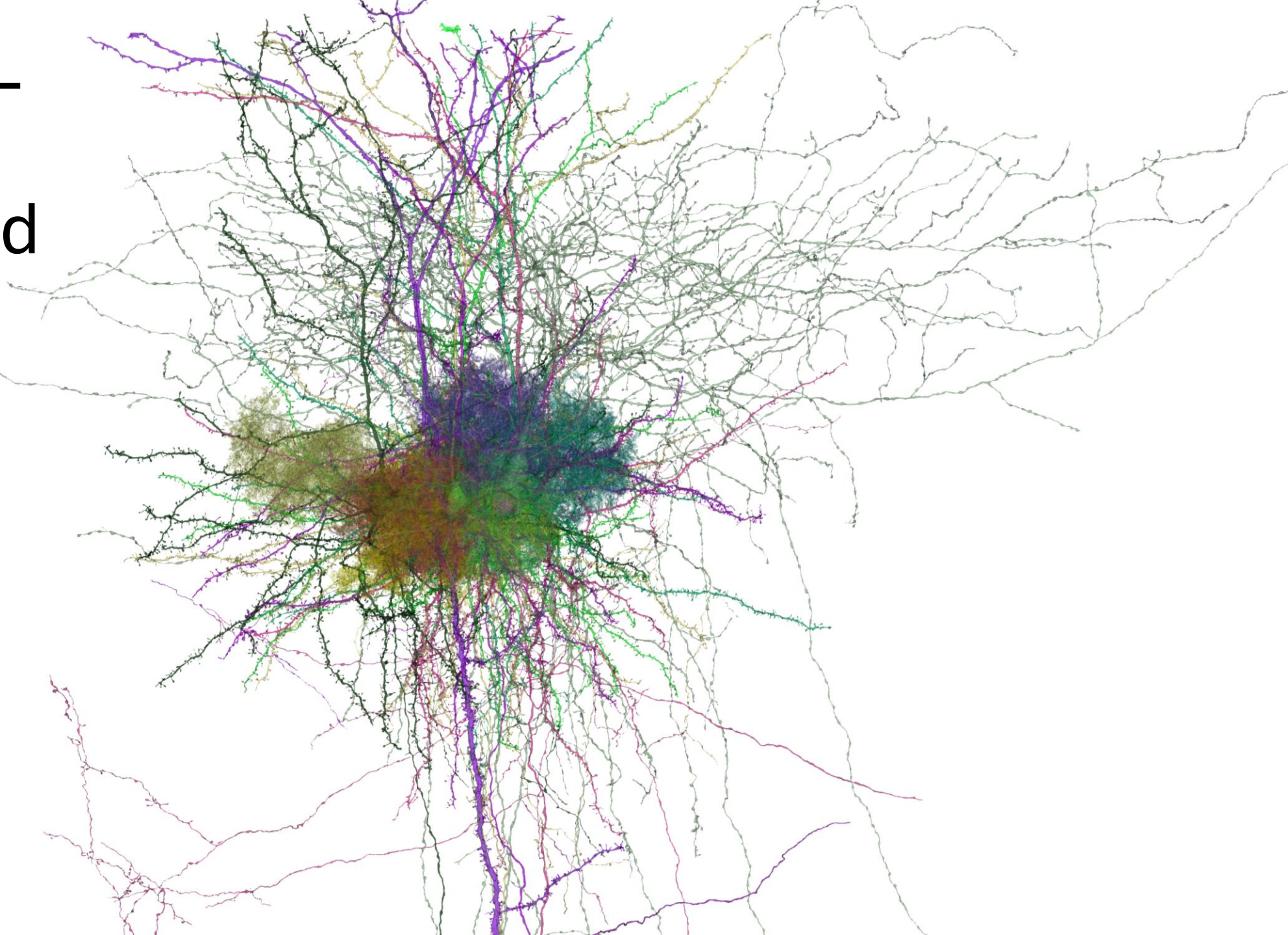
minnie volume: decimate meshes

Decimate meshes using a list of cellids of interest with pyvista_decimate_cell_bodies_minnie.ipynb

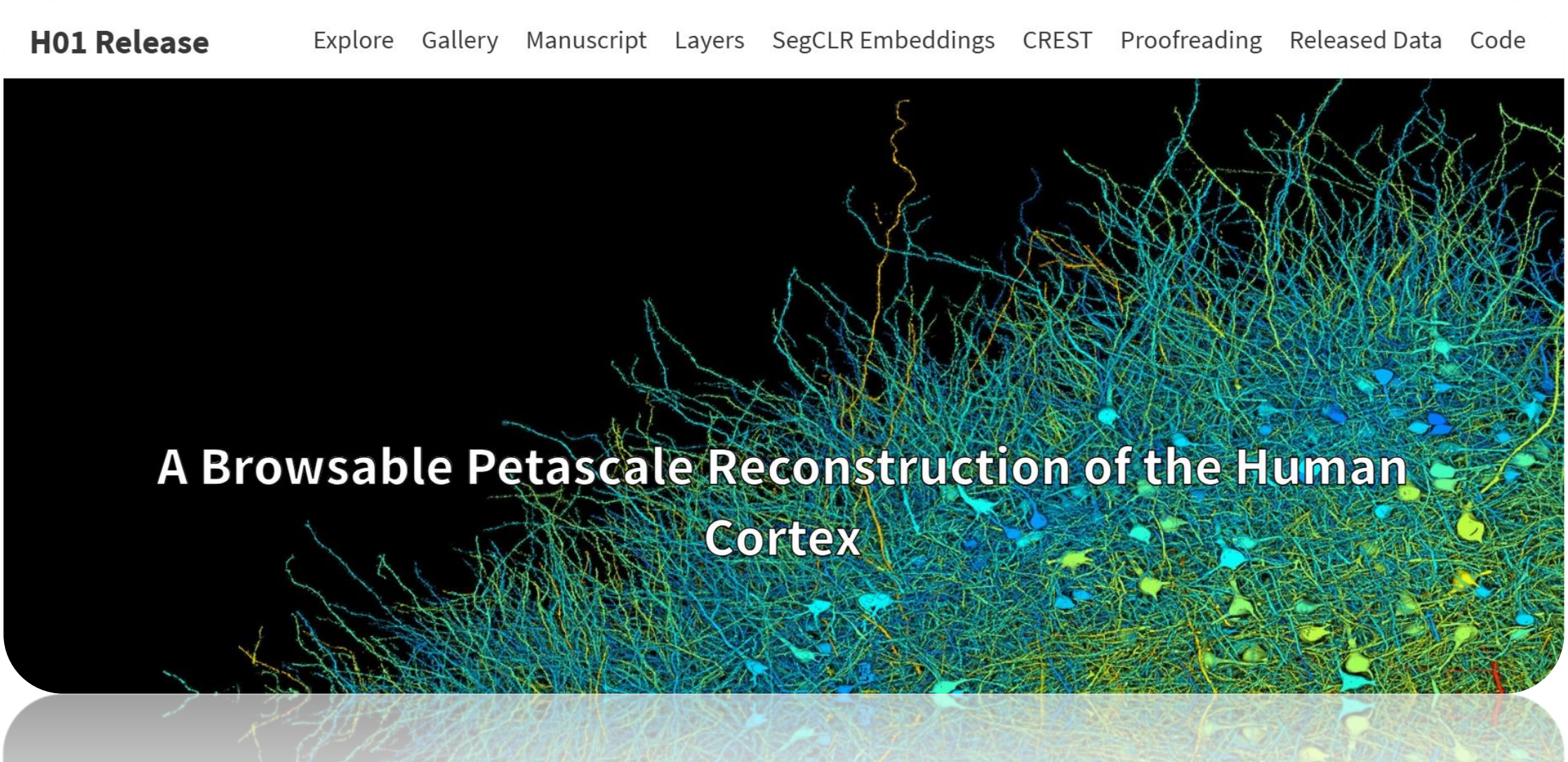
minnie volume: visualize meshes

Visualize cell meshes by cell type (e.g., neurons, astrocytes, vasculature) with vtk

vtk OpenGL rendered neurons and astrocytes



Harvard/Google H01 volume



H01 volume: download meshes

Visit the Jupyter Notebook folder on github

Start with h01_plotly_visualizer.ipynb to download and view cell meshes

See h01_cell_types folder for a list of cellids of interest in the H01 volume (e.g., astrocytes, pyramidal neurons, etc.)

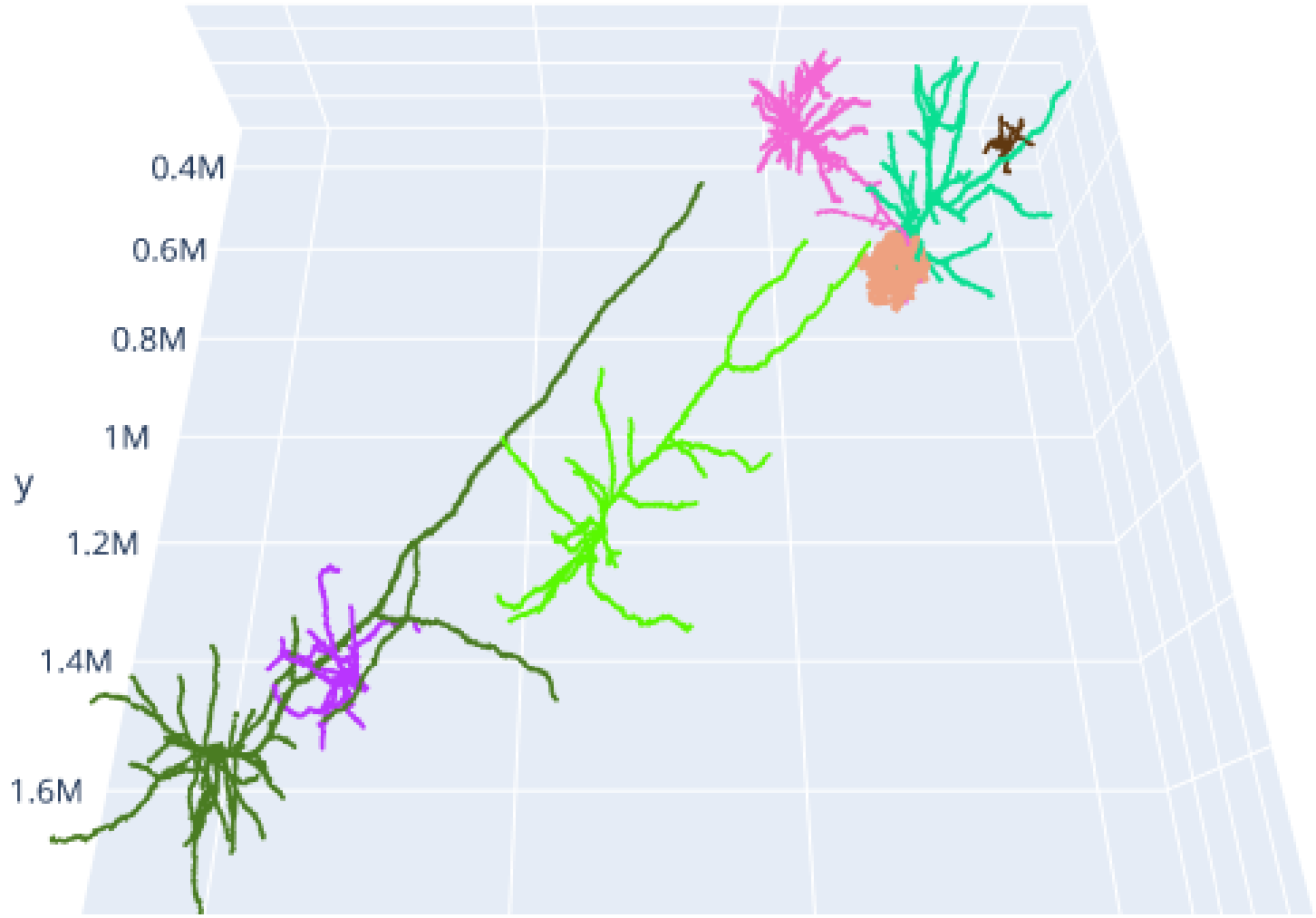
Mesh download

```
# setup the mesh meta to handle downloads and caching
mesh_dir = 'data/c3_cell_meshes/' # or change to your desired folder
seg_source = "precomputed://gs://h01-release/data/20210601/c3"
mm = trimesh_io.MeshMeta(cv_path=seg_source,
                         disk_cache_path=mesh_dir,
                         cache_size=20)
```

Mesh download

```
cell_id_list = [28541451958,27843438291,
4970736617]


for i in tqdm(range(len(cell_id_list)),
desc="Downloading Meshes"):
    downloadmesh = mm.mesh(seg_id=cell_id_list[i],
remove_duplicate_vertices=True)
```

Plotly visualization of mesh objects



H01 volume: decimate meshes

Decimate astrocyte meshes of interest as a list
(can be used for other cell types too)

Complex meshes, such as astrocytes and vasculature, may take hours (~12 hours per astrocytes) or run out of memory on a typical computer

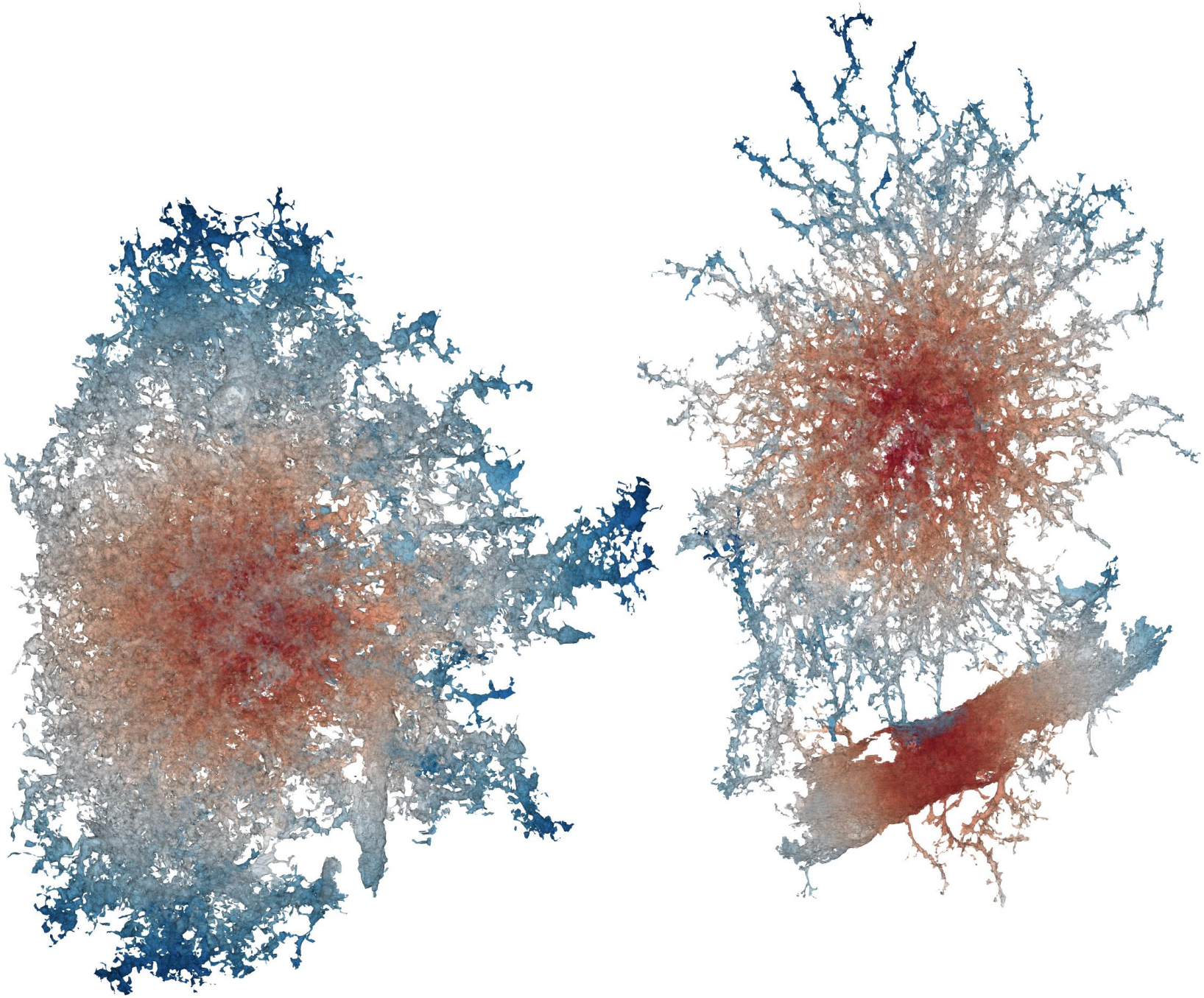
Use Google Colab with High-RAM (subscription to Google Colab Pro required) to allocate enough memory and reduce processing time (<10 min)

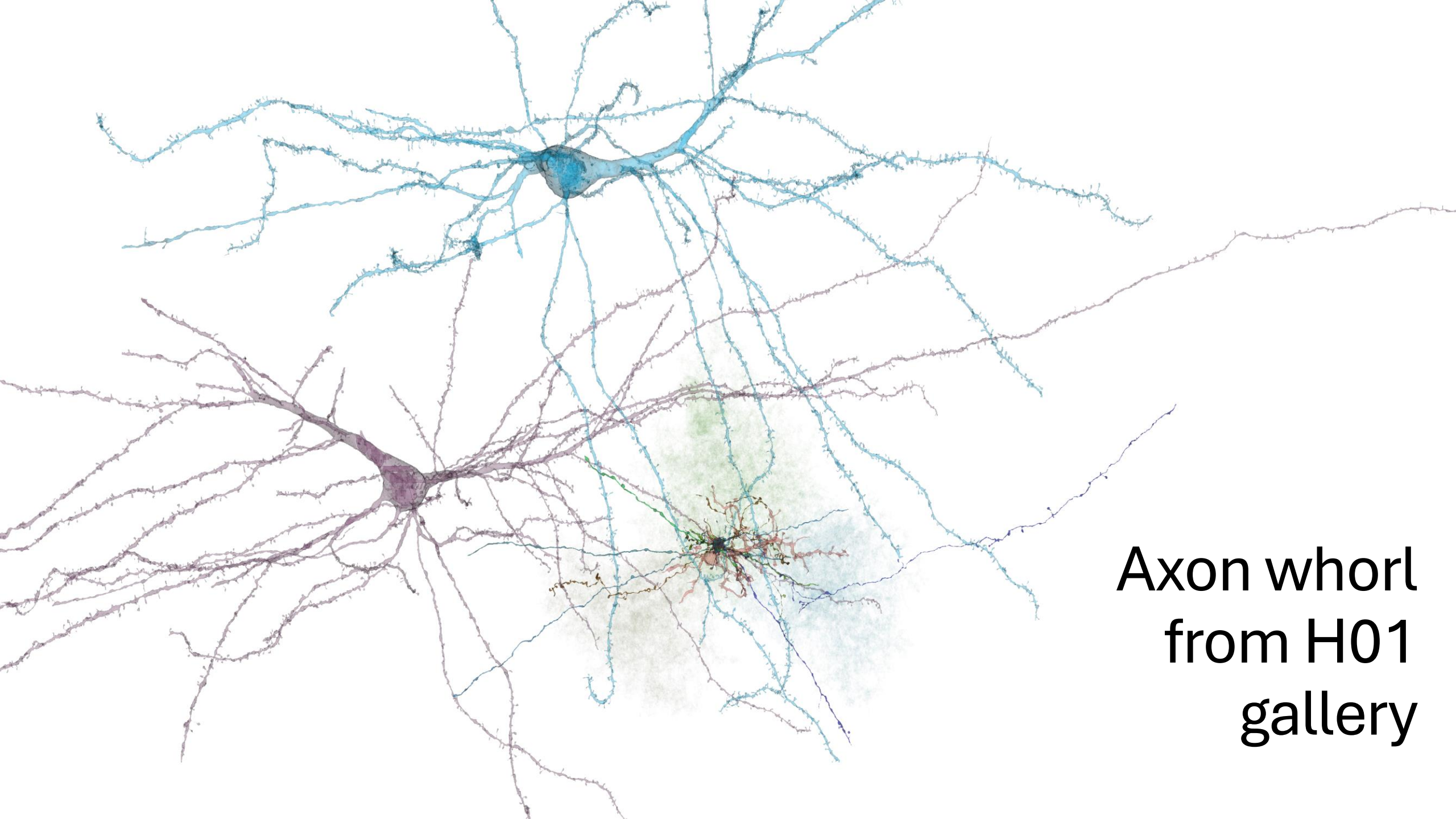
H01 volume: visualize meshes

Visualize decimated astrocyte meshes using a gradient coloring method with vtk

Visualize decimated meshes by four cell categories (neurons, astros, microglia, and unidentified segments) with vtk

Gradient coloring of astrocytes





Axon whorl from H01 gallery

Acknowledgements: MICrONS

Read the original research papers on the Citation page at Allen Institute

Read the Terms and Conditions page

Acknowledgements: H01

Link to the original research paper on the Manuscript page of H01 Release website

Read the license on the Released Data page

Code Availability

Visit my github repositories for each volume:

- layer23-volume
- minnie-volume
- h01-volume