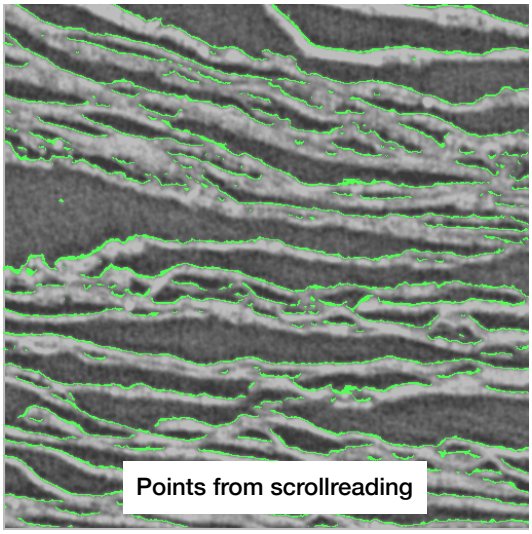
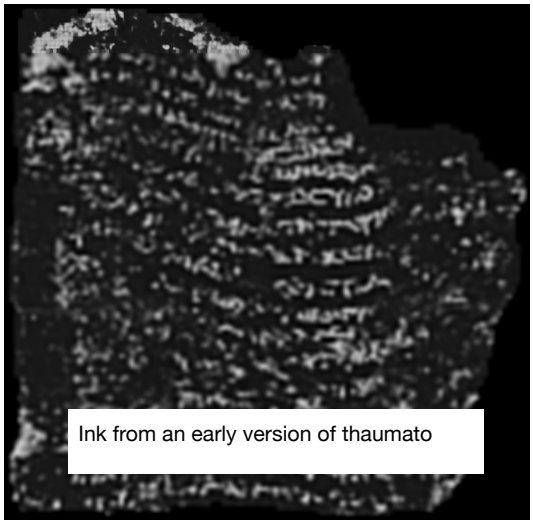
A brief introduction to auto segmentation on the Herculaneum scrolls, and a proposed "fiber following" solution

The Herculaneum scrolls, damaged during the eruption of mount Vesuvius, have many regions in which the verso and recto side have separated or split, the sheet itself is split, or both have occurred. Due to this loss of single sheet or single winding connectivity, creating large continuous segments is difficult. Local information is not enough to automatically segment regions larger than a few millimeters , and the global information required to properly match patches together is difficult to gather due these frequent losses of connectivity (or conversely too much connectivity when the sheets touch and the "correct" sheet is hard to properly gauge)



Points from scrollreading



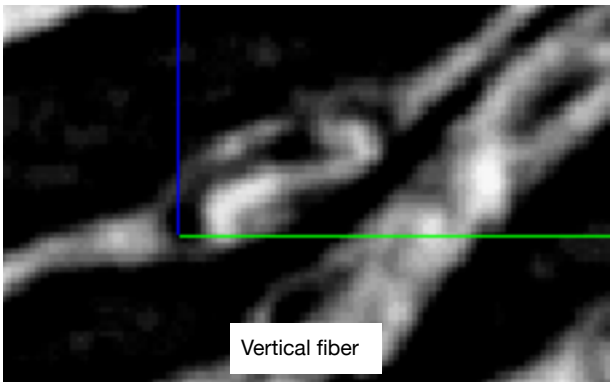
Ink from an early version of thaumato

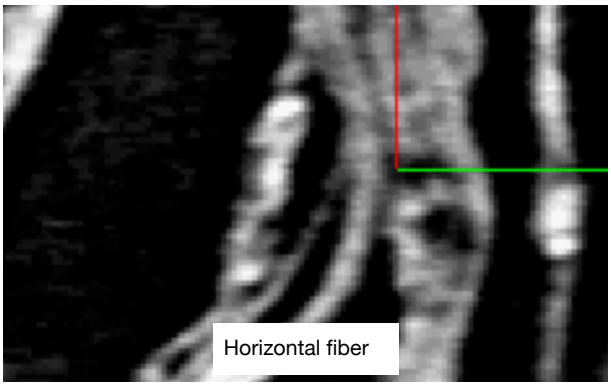
- The "inside" of the recto sheet is not distinguishable from the "true" verso when they are split — this is true even in other non-thaumato based segmentation pipelines like Will Stevens scroll reader (https://github.com/WillStevens/scrollreading):
- Once points are created along proposed verso , they have no global connectivity information, and so this must be created by either constructing a graph as is the case with ThaumatoAnakalyptor (https://github.com/schillij95/ThaumatoAnakalyptor/tree/main) , or another method
- Currently, Thaumato is prone to "sheet switching", due to lack of global information. This creates segments that have holes or become warped during the parameterization process, or are constructed of many patches of different sheets, making ink detection much more difficult
- Identifying which patches belong to which winding or sheet has been termed as "the graph problem", by Julian Schillenger (creator of thaumato). This is the greatest challenge that auto segmentation currently faces.
- Progress has been made from simplification of graph problem and improvement of point cloud generation by Julian schillinger and Georgio angoletti, but even with significant improvements to point cloud data and simplification of the graph problem the lack of global connectivity may prove too great a challenge to overcome , particularly in highly compressed or warped regions

Clearly, a method of maintaining global connectivity between surface points must be found

Some background on fibers

- To create papyrus the reed of the plant is cut into strips, stacked along a plane, and then compressed. It appears that this compression is not as great in the center of the fiber, where there is more material. Visible on a micrometer scale ct scan as a rectangular or “figure 8” structure in a “capsule”. Due to the damaged nature of the scrolls, they are not always perfectly encapsulated like this





- Although carbonization and compression have left the scrolls severely damaged, many parts of the scrolls, such as these “fibers” have maintained their structure well. These thicker “fibers”, or what may actually be the vascular bundles of the plant it self, are particularly resilient to the type of damage present in most other scroll structures. It’s not uncommon to find vertical fibers that survive intact for many centimeters (as far as I’ve been able to follow them by eye and as far as I have ran predictions on for model discussed next)

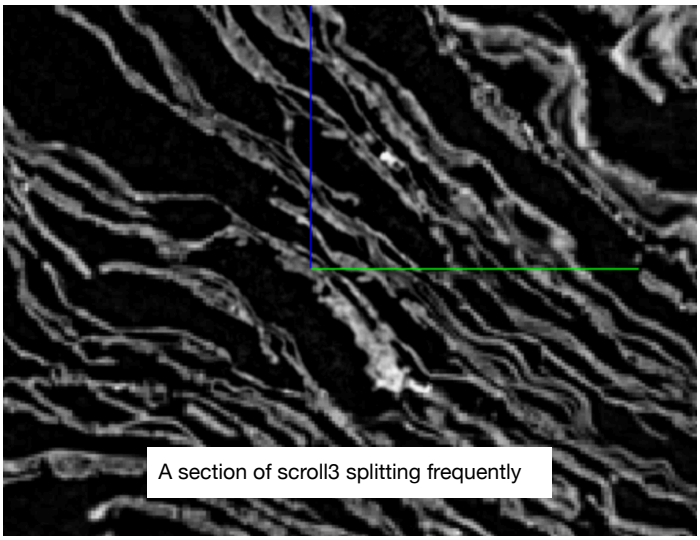
Fiber focused segmentation of the Herculaneum scrolls

I noticed during manual segmentation that when panning through the z slices of the 3d volume, the horizontally oriented fibers appeared as “streaks” or “flashes” along the winding direction of the scroll, and could be utilized as “guides” to allow a segmentor to follow a single sheet through compressed or highly damaged regions of the scroll volume

- Similarly, vertical fibers are able to be followed through these regions due to the fact that they are typically intact for large portions of the scroll , allowing a segmentor to verify correct winding number and segmentation line accuracy

This method of “fiber following” is understood to be one of the most accurate ways to achieve a true surface segmentation of a “wrap”, and is utilized by the Vesuvius challenge segmentation team members, and members of the competition. Iin some scrolls (p.herc 0332, p.herc Paris 3) , due to either the severe splitting of the verso and recto sheet for the entirety of the scroll, or substantial compression and warping it is the only way to effectively manually segment.

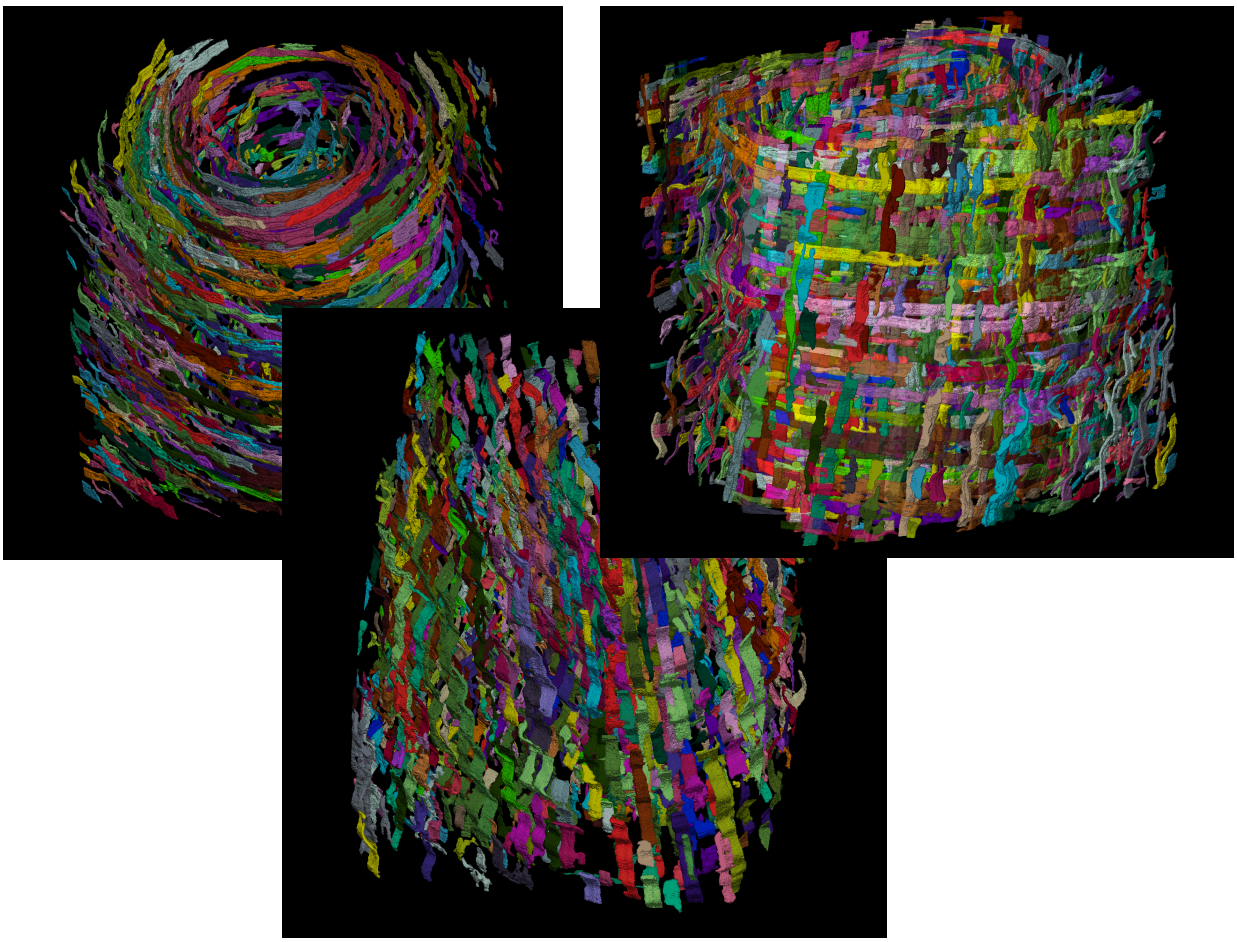
Given the success of manual “fiber following” , it seemed beneficial to train a neural network to semantically segment the fibers into two separate classes, respective to their orientation. I trained a 3d unet leveraging nnUNetv2, in the ‘3d_fullres’ configuration and using 5 annotated .tif volumes (https://github.com/MIC-DKFZ/nnUNet) that I annotated using Dragonfly3d (https://dragonfly.comet.tech/) these volumes are in total around 750x750x500 — a tiny fraction of the scroll.



A section of scroll3 splitting frequently

Results

Given the small dataset, the model has exceeded expectations. Running a simple connected-components-3d after thresholding for size with cc3d.dust, vertical and horizontal fibers are well separated in most areas, with minimal component merging , and most vertical fibers are quite long. The following images show this process on a 1,000 tif volume.

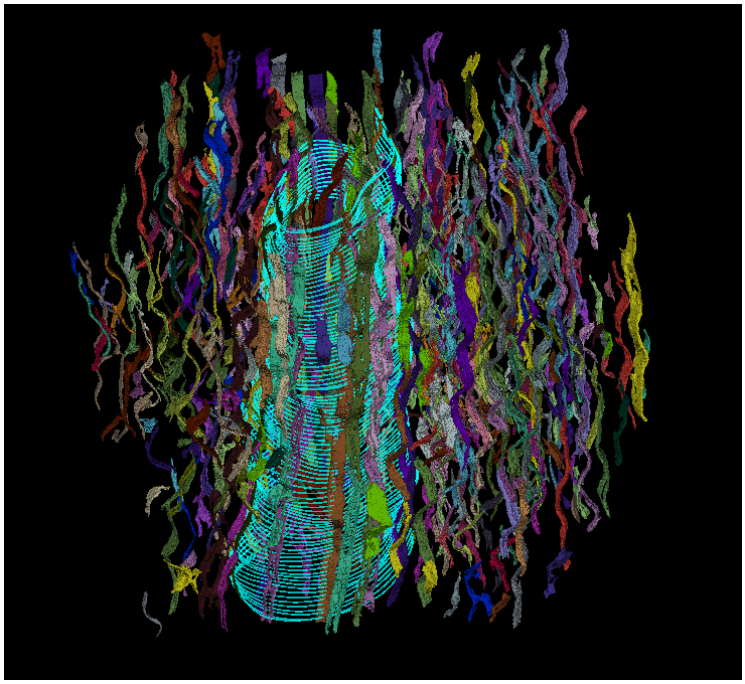


From these 3d connected component volumes, I explored a few potential meshing options, which I'll explain in more detail here. Note that these are very simply proof of concepts that I have not fully fleshed out, there are tons of cheap and easy optimizations to be made (primarily increased labeling, as this takes an exceptionally long time to do correctly).

Method 1 — Using Vertical Fibers as Guides:

I tried this method as a somewhat simple test to check if the idea that given an algorithm that could properly select the verticals from the same sheet, could we create a basic mesh that is accurate by simply following the verticals down.

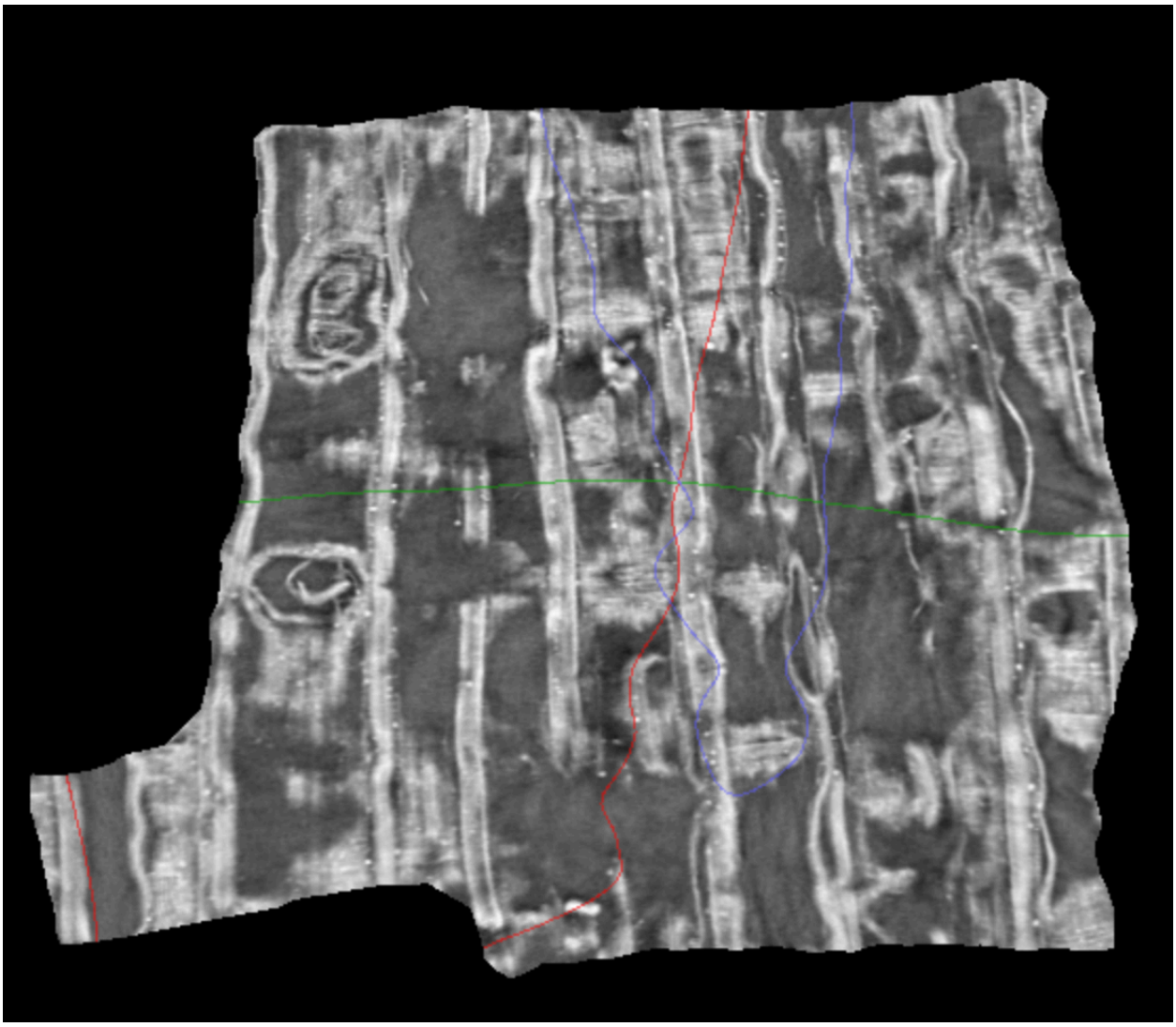
I. Through the volume of vertical fibers to find which label IDs corresponded to the same sheet, and inserted these into a list
II. For each of these labels, I specify a frequency , and sample the center of mass using ndimage.measure — I then create a "node" here, using the points layer in napari
III. Then for each z layer in the frequency (for example, if there I create the nodes every 10 slices, I'll also do this step every 10 slices), I interpolate a curve along these nodes.



IV. Using this curve and the nodes, I create an OBJ with only vertices
V. I open the obj in mesh lab, and apply ball pivoting mesh reconstruction, and laplacian smoothing

Opening the mesh in khartes (https://github.com/KhartesViewer/khartes) , it appears it has followed the vertical fibers well. Keep in mind that doing this method creates a segmentation line along the vertical fibers, so the image shown here is more representative of about 15-20.tif

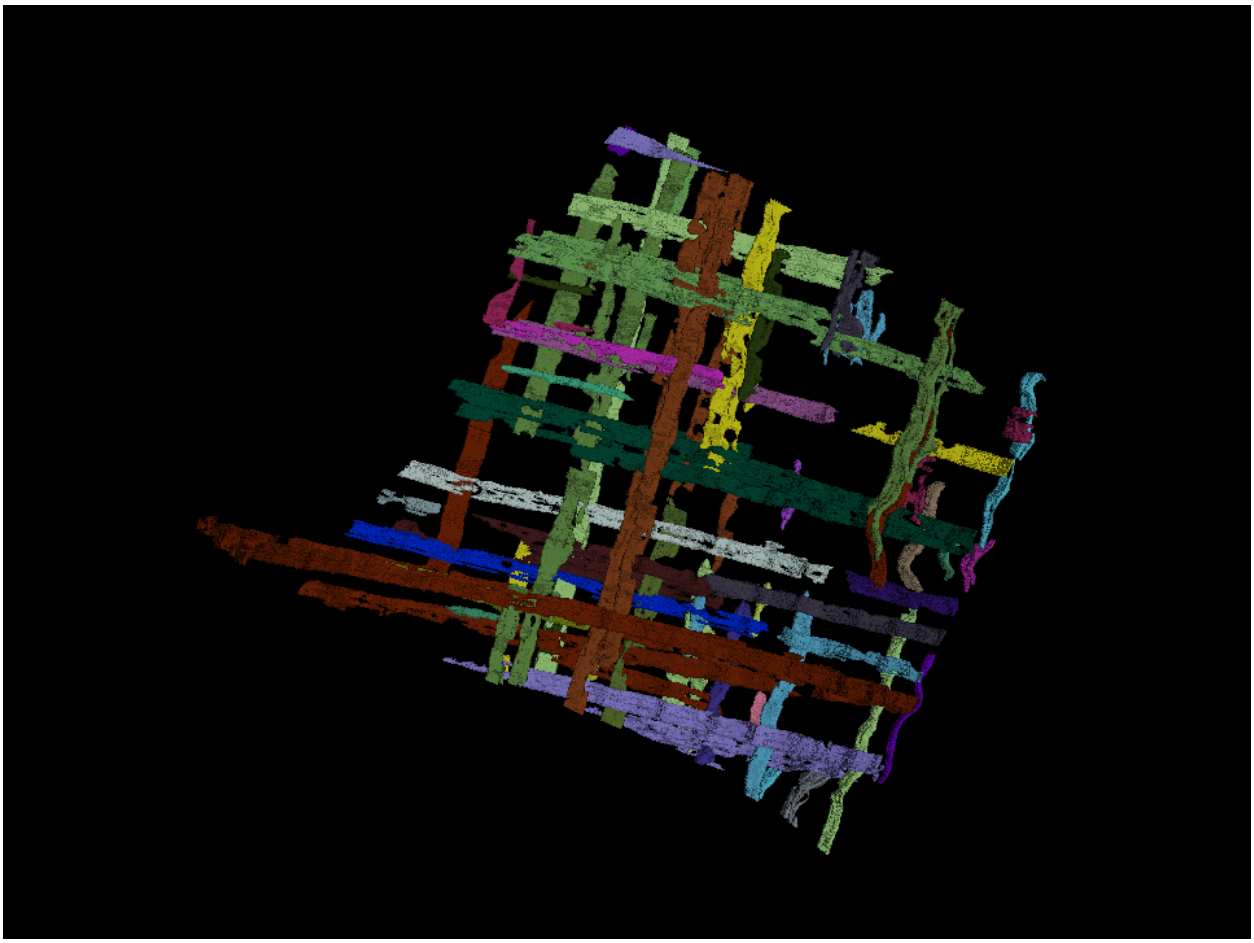
rather than the true surface, traditionally 32.tif — theoretically this should not introduce problems with ink detection or other downstream tasks as it just requires thinking of a different number as the “true surface”



This method could use some refinement. The interpolation is just the first thing I saw on scipy.interpolate — there could be some interesting work with utilizing structure tensors (one idea given by @khartes_chuck would be to use the algorithm in st.py within the khartes repository to follow the nearest true surface). There also exists the obvious issue of me having chosen the verticals by just manually selecting numbers, perhaps a faster ui or using segmentation lines from VC or Khartes could allow one to place a segmentation line and follow the verticals down for a few thousand tifs. I believe this could be done rather effectively.

Method 2: Use the connectivity information to connect the windings

I. create connected-components for all the verticals and horizontals and store them as separate arrays.
II. Given a “seed” label (I’d typically go with a vertical as the starter here), calculate the surface voxels by dilating the seed label, and then removing the original. (This is just to reduce the amount of rays we end up casting). Then, create a vector in the direction of the umbilicus, along this vector cast a ray a specified distance, checking each voxel for intersection with the opposing class. If one is found, stop and add it to a list of “found labels” of the class you have found.
    1. For vertical fibers casting rays to target horizontals, you want to cast rays “in”, and from the “front”. Horizontal fibers are always in “front” of the vertical ones
    2. For horizontals casting rays to target verticals, you cast rays “out” and from the “back”
III. With this list of all horizontals that “belong” to the selected vertical, cast rays “out” from these new horizontals, to find all the verticals that belong to them,
IV. Cast rays from your new set of verticals to find new horizontals
V. Repeat this process until no more are found



I have yet to mesh the output from this, but it should be easy to skeletonize this and then use this as vertices for an obj. Both of these methods are unrefined and quite buggy. I have a Napari script I will include in a google drive (and later ash2txt) upload. I'm working on an upgraded version.

I believe both of these methods show extreme promise, along with graph construction methods that utilize this essentially greatly simplified graph representation of the scroll. The verticals in particular are rarely merged with each other, and resilient enough that its easy to find very long ones even with this poorly trained model. With some additional labels for training data, and some refinement in the process of selecting fibers I'm confident that accurate and repeatable auto segmentation can be achieved in the Herculaneum scrolls. Each of the scrolls provided for this competition have similar looking vertical and horizontal fibers, and so the method should work similarly for them as well.

Things I think could be "easily" implemented and immediately impactful:

1. Allowing VC or Khartes to use the verticals in a predicted volume as guides, so one could create a segmentation line and then follow the verticals down for many thousand slices.
    1. Perhaps Philip Algiers "point cloud magnets" can already be adapted for this
2. A better napari implementation than I have provided here. I am working on this but will be very busy over the next few weeks — I think even being able to manually click labels to add to a list would be helpful.
3. Better methods for finding nearby fibers — ray casting is not likely the best way, perhaps a kdtree or other method would work better and faster.

I wrote these scripts in a very short amount of time and am not a very strong programmer. I am certain someone in this discord can use the information gained from this model to solve segmentation, and I'm not certain when I will have the time necessary to gain the requisite skills to do so. I'm making it available in the interest of advancing the challenge towards this goal and hope someone here is able to take it the rest of the way.