Example_Fiber_analysis

November 6, 2021

1 Example SEM Image analysis: porosity & fiber diameter

This notebook provides a step-by-step example of a single SEM image being analyzed using skimage and quanfima, with visualization provided by napari. First import needed modules. When running in python3, the `quanfima` module has some problems due to relatively old dependencies. A fork capable of running 2D image fiber analysis in python3 is available: https://github.com/psobolewskiPhD/quanfima

The workflow begisn with importing the needed modules.

[ ]:
```
import napari
import numpy as np
from quanfima import morphology as mrph
from skimage import filters, io, morphology
from skimage.color import rgb2gray
```

Next, we read in a representative image using the `skimage` function `imread`, ensure the image is grayscale, and set the scale information (micron per pixel).

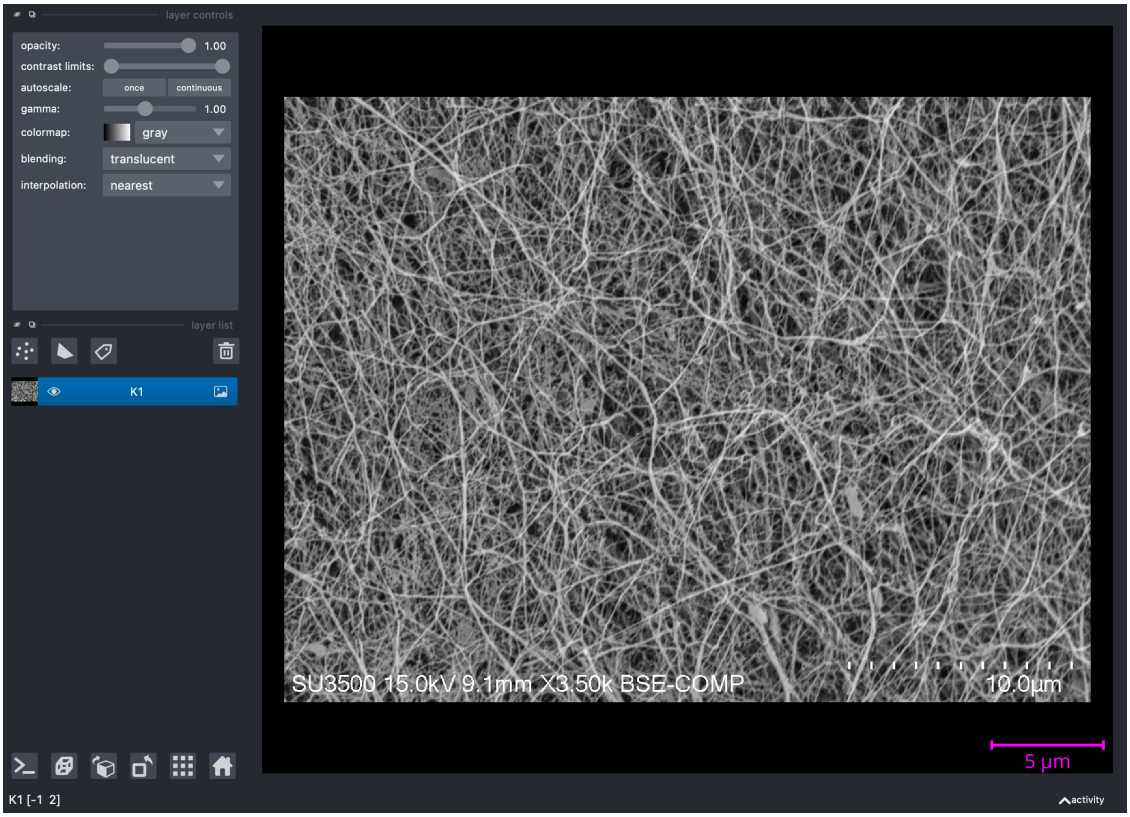
[ ]:
```
K1 = io.imread('fiber_data/K1.jpg')
scale = [1/35.5, 1/35.5]    #microns per pixel
K1 = rgb2gray(K1)
```

Next we will set up a napari viewer instance to visualize the image and workflow steps.

[ ]:
```
viewer = napari.Viewer()
viewer.add_image(K1, scale=scale)   #the scale keyword enables providing scale␣
↪information
viewer.scale_bar.unit = "um"    #set the scale bar unit
viewer.scale_bar.visible = True #show the scale bar
viewer.scale_bar.colored = True #ensure it has a color so it's more visible
```

[ ]:
```
napari.utils.nbscreenshot(viewer)
```

[ ]:



Next, we will segment the image using `skimage`. First, though, we need to crop out the bottom of the image where the SEM information is located. Then, for segmentation we will use the *Niblack* algorithm with the default parameters.

[ ]:
```
K1 = K1[0:890,]      #crop to top 890 px
thresh_NB = filters.threshold_niblack(K1)       #calculate the threshold using␣
↪Niblack
seg_NB = (K1 > thresh_NB).astype(np.uint8)      #segment the image, taking only␣
↪pixels above the threshold
```

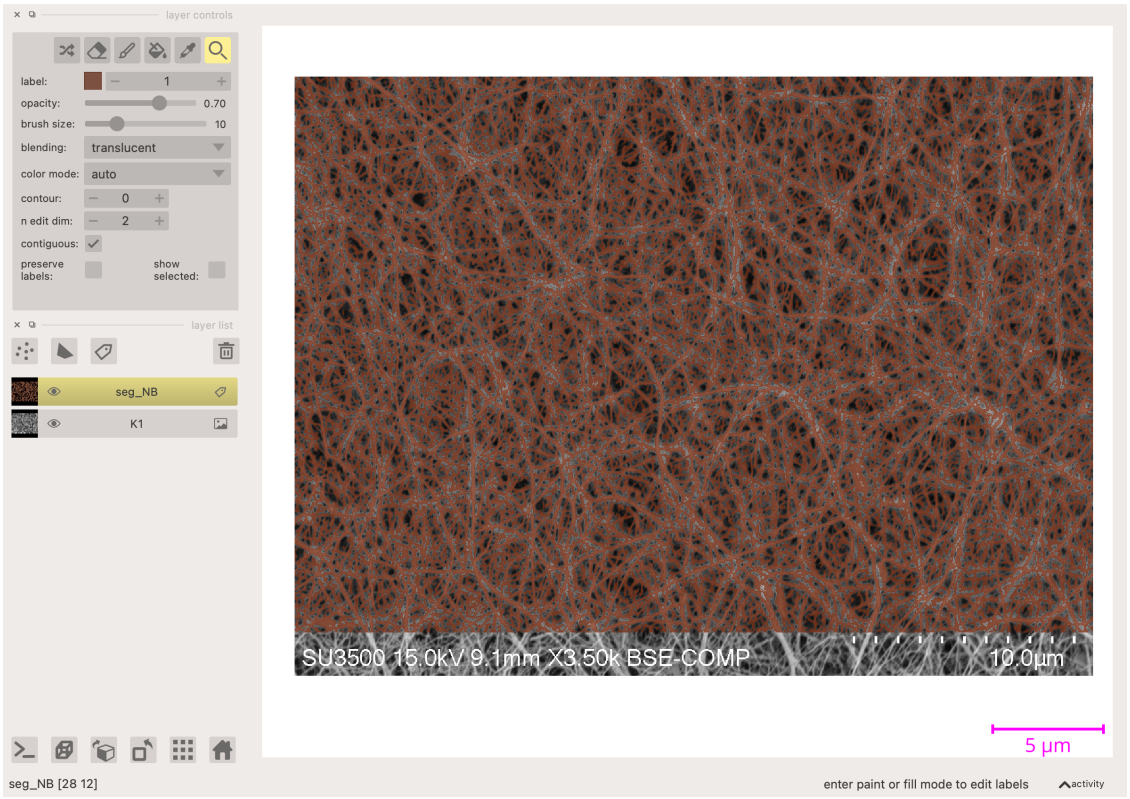
To visualize the segmentation, we will add it as a label layer in napari.

[ ]:
```
viewer.add_labels(seg_NB, scale=scale)
```

[ ]:
```
<Labels layer 'seg_NB' at 0x29731c370>
```

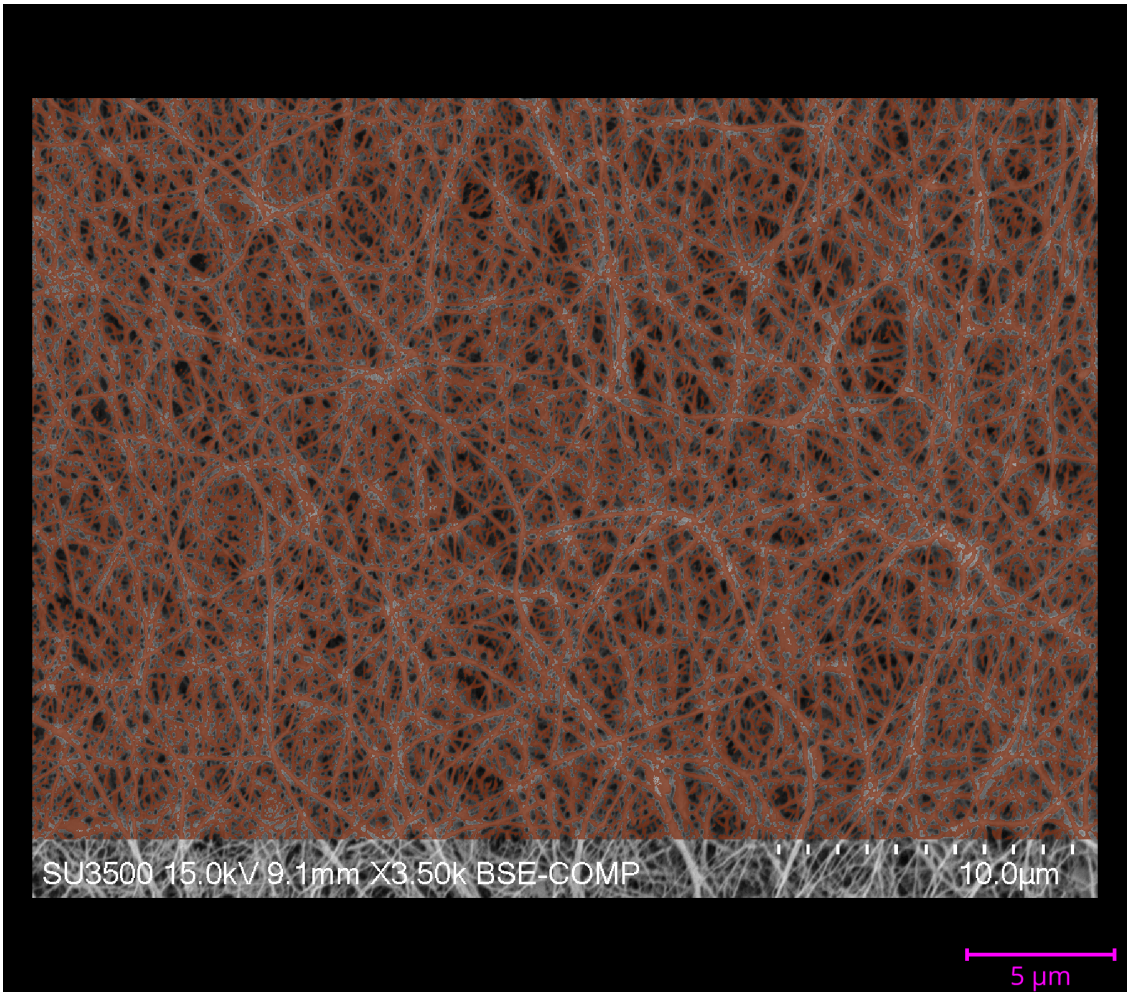
[ ]:
```
napari.utils.nbscreenshot(viewer)
```

[ ]:



[ ]:
```
napari.utils.nbscreenshot(viewer, canvas_only=True)      #a zoomed in view, with␣
↪label opacity reduced to 0.50
```

[ ]:



To proceed with the workflow, the segmented fibers need to be skeletonized, again using `skimage`. The default algorithm will be used (Zhang's), as it is appropriate for 2D analysis.

[ ]:
```
skeleton = morphology.skeletonize(seg_NB)
```

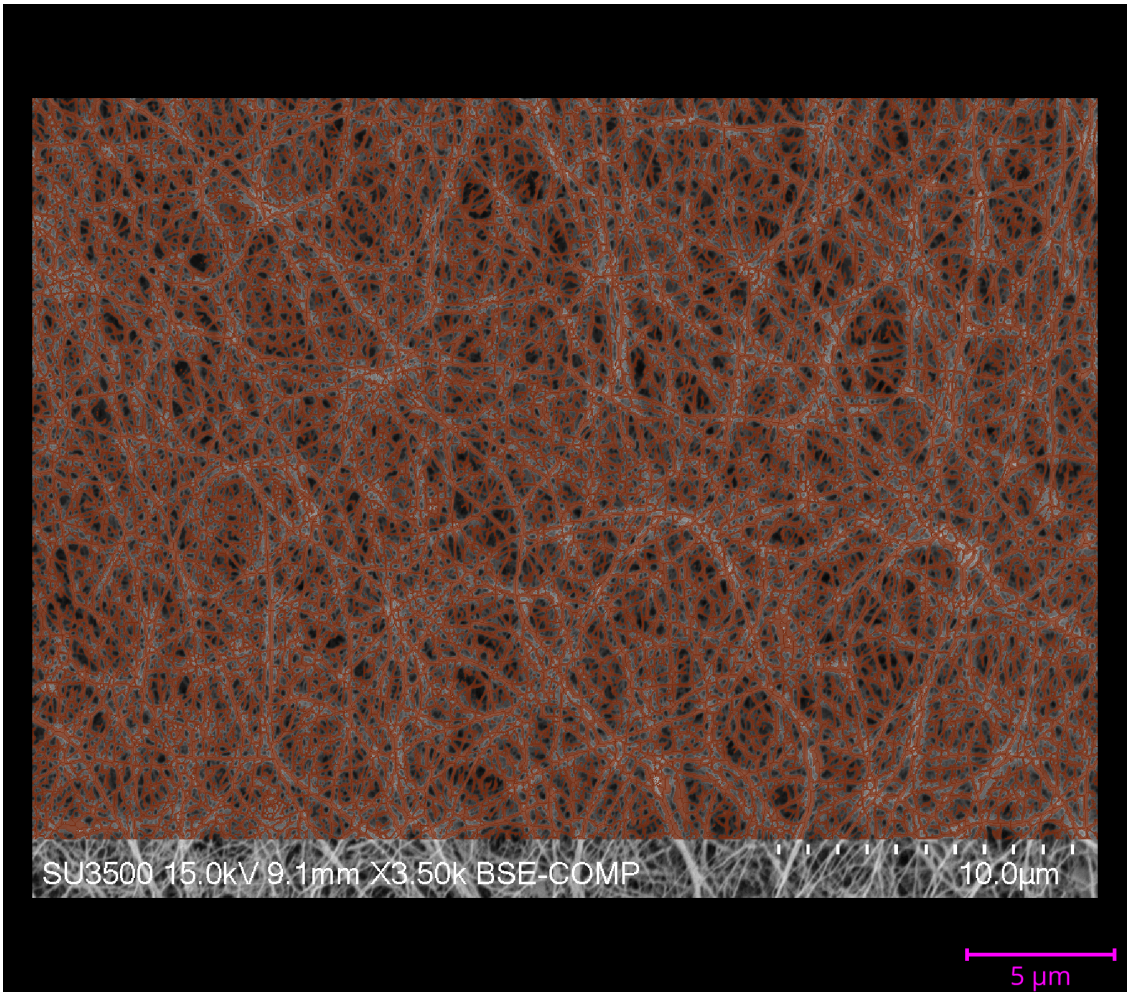
Again, we will visualize the skeleton segments using a napari labels layer.

[ ]:
```
viewer.add_labels(skeleton, scale=scale, name="Skeleton")
```

[ ]:
```
<Labels layer 'Skeleton' at 0x17a8a7850>
```

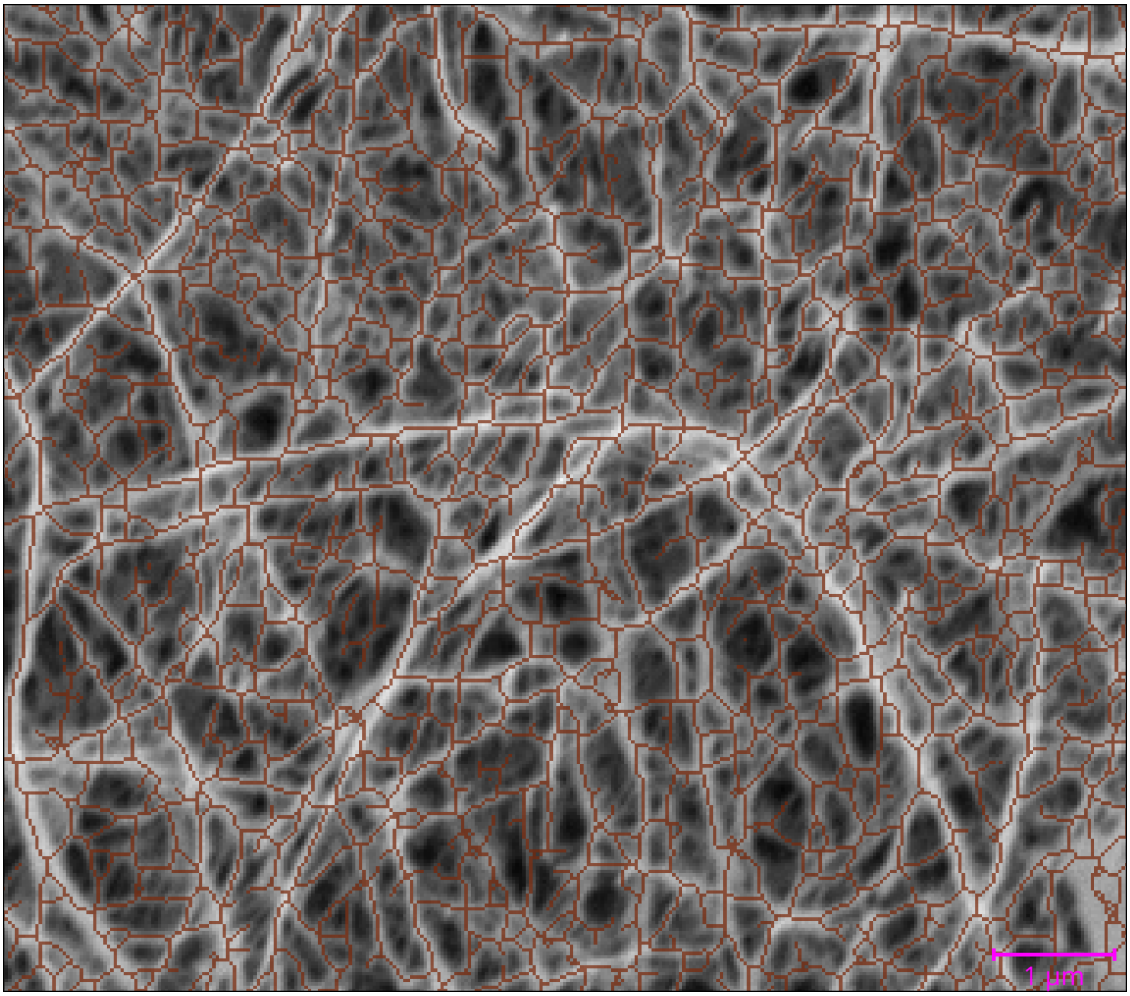
[ ]:
```
napari.utils.nbscreenshot(viewer, canvas_only=True)      #a zoomed in view, with␣
↪segmentation and skeleton
```

[ ]:



[ ]:
```
napari.utils.nbscreenshot(viewer, canvas_only=True)      #a zoomed in view, with␣
↪just image and skeleton
```

[ ]:



Next, we analyze the segmentation and skeleton using `quanfima` morphological tools. First, we will compute the porosity, the ratio of segmented fibers to empty (air) space.

[ ]:
```
porosity = mrph.calc_porosity(seg_NB)    #calculate the porosity, based on the␣
↪segmented image
print(porosity)
```

```
{'Material 1': 0.4232435042134831}
```

Next, we run the fiber properties analysis using `quanfima`. You can see more information regarding the function in the `morphology.py` code of the original codebase. Briefly, we need to input the segmented pixels and the calculated skeleton and the outputs `dmap` and `dvals` provide fiber diameter maps and the fiber diameter values, while `ovals` provides the fiber orientations (in radians). The Warning: this can take ~1 min per image.

[ ]:
```
cskel, fskel, omap, dmap, ovals, dvals = mrph.estimate_fiber_properties(seg_NB,␣
↪skeleton)
```

```
/Users/piotrsobolewski/Dev/quanfima/quanfima/morphology.py:195: FutureWarning:
```

```
Until version 0.16, threshold_rel was set to 0.1 by default. Starting from
version 0.16, the default value is set to None. Until version 0.18, a None value
corresponds to a threshold value of 0.1. The default behavior will match
skimage.feature.peak_local_max. To avoid this warning, set threshold_rel=0.
  corner_points = feature.corner_peaks(method, min_distance=3)
```

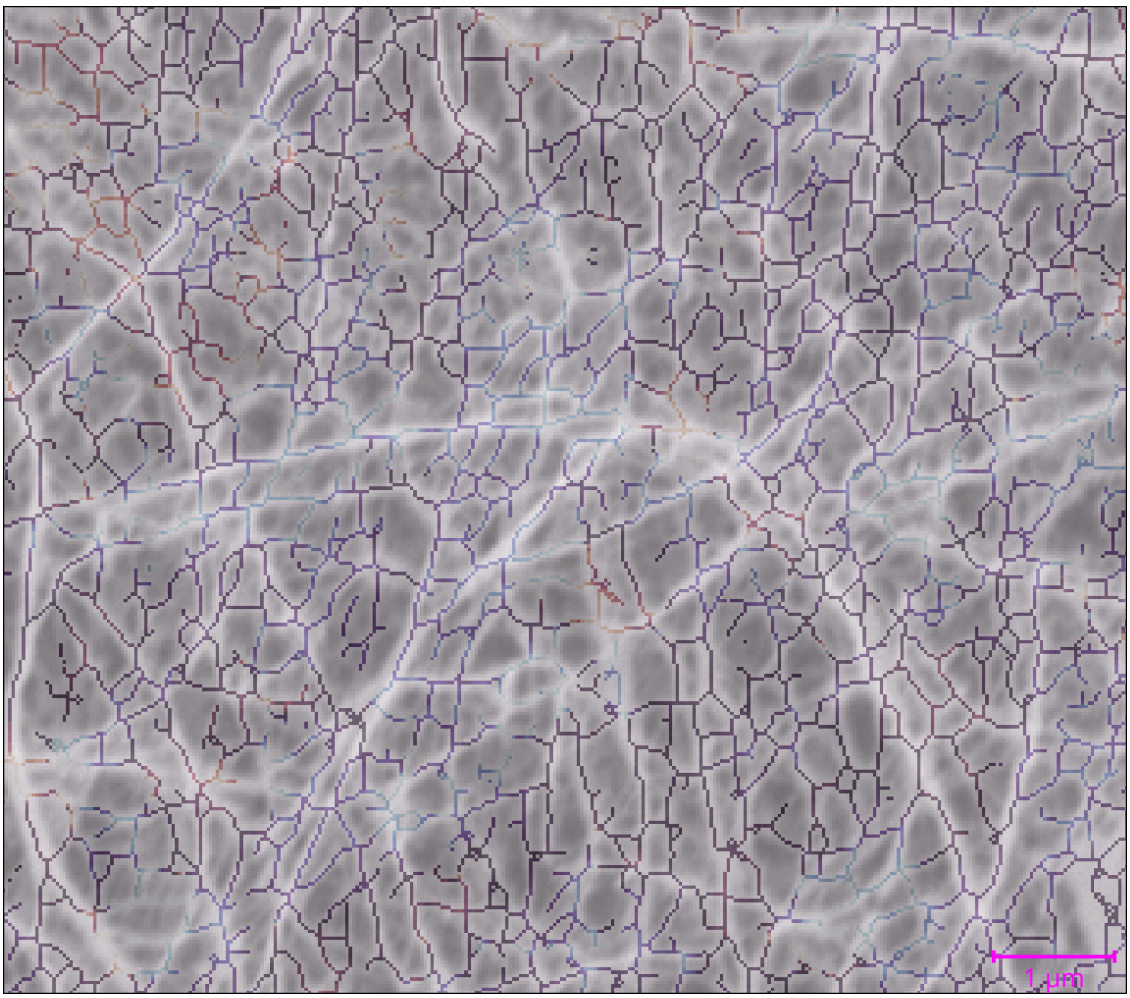
Now the orientation map (`omap`) can be visualized with napari and simple summary statistics can be computed. The colormap (LUT) `twilight` will be used to fibers near verticle orientation will be dark, while those that are near horizontal will be light.

[ ]:
```
viewer.add_image(omap, scale=scale, name="OrientationMap", colormap="twilight",␣
↪opacity=0.5)
```

[ ]:
```
<Image layer 'OrientationMap' at 0x2ae2923a0>
```

[ ]:
```
napari.utils.nbscreenshot(viewer, canvas_only=True)      #a zoomed in view of␣
↪the orientation map overlayed on the image
```

[ ]:



Important: It's crucial to inspect the orientation map and ensure that fiber orientations are correctly estimated!
Here, as can be noticed, a large number of clearly horizontal segments are marked with dark colors, indicating they are estimated to have a vertical orientation. The diameter calculation strongly relies on proper orientation estimation. In this case, the default `window_radius` was used: `12` (~350 nm). This is the local radius used to compute orientations. In this case, due to a combination of small diameter and high density of short branches, the default value does not yield accurate orientations. The value needs to be *reduced*, closer to the value of the fiber radii, for example `3` (~85 nm).

[ ]:
```
cskel, fskel, omap, dmap, ovals, dvals = mrph.estimate_fiber_properties(seg_NB,␣
↪skeleton, window_radius=3)
```

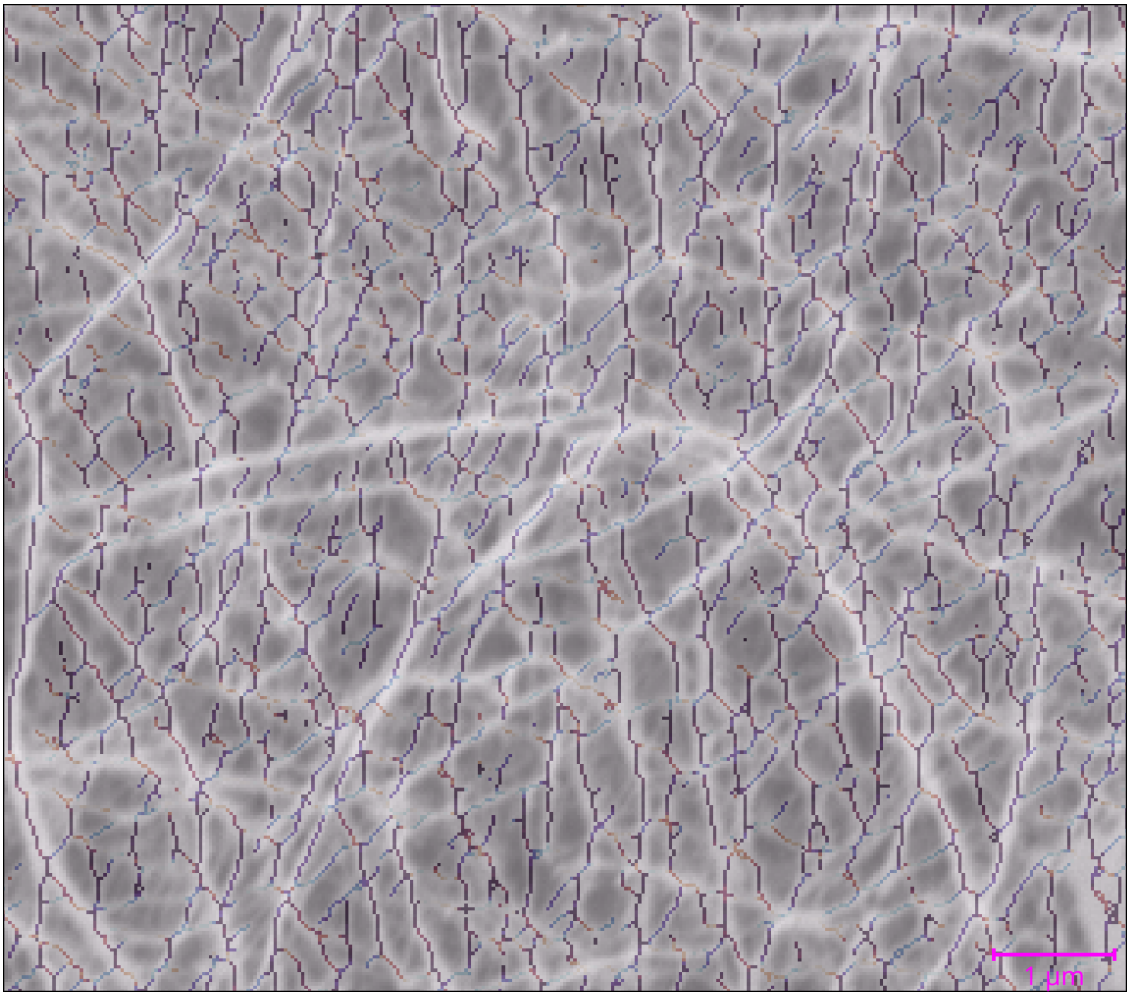
```
/Users/piotrsobolewski/Dev/quanfima/quanfima/morphology.py:195: FutureWarning:
Until version 0.16, threshold_rel was set to 0.1 by default. Starting from
version 0.16, the default value is set to None. Until version 0.18, a None value
corresponds to a threshold value of 0.1. The default behavior will match
skimage.feature.peak_local_max. To avoid this warning, set threshold_rel=0.
  corner_points = feature.corner_peaks(method, min_distance=3)
```

[ ]:
```
viewer.add_image(omap, scale=scale, name="OrientationMap3",␣
↪colormap="twilight", opacity=0.5)
```

[ ]:
```
<Image layer 'OrientationMap3' at 0x2ae23a8b0>
```

[ ]:
```
napari.utils.nbscreenshot(viewer, canvas_only=True)     #a zoomed in view of␣
↪the orientation map overlayed on the image
```

[ ]:



As can be noticed, the horizontal fibers are now properly light in color, with the dark colors marking primarily vertical fibers.

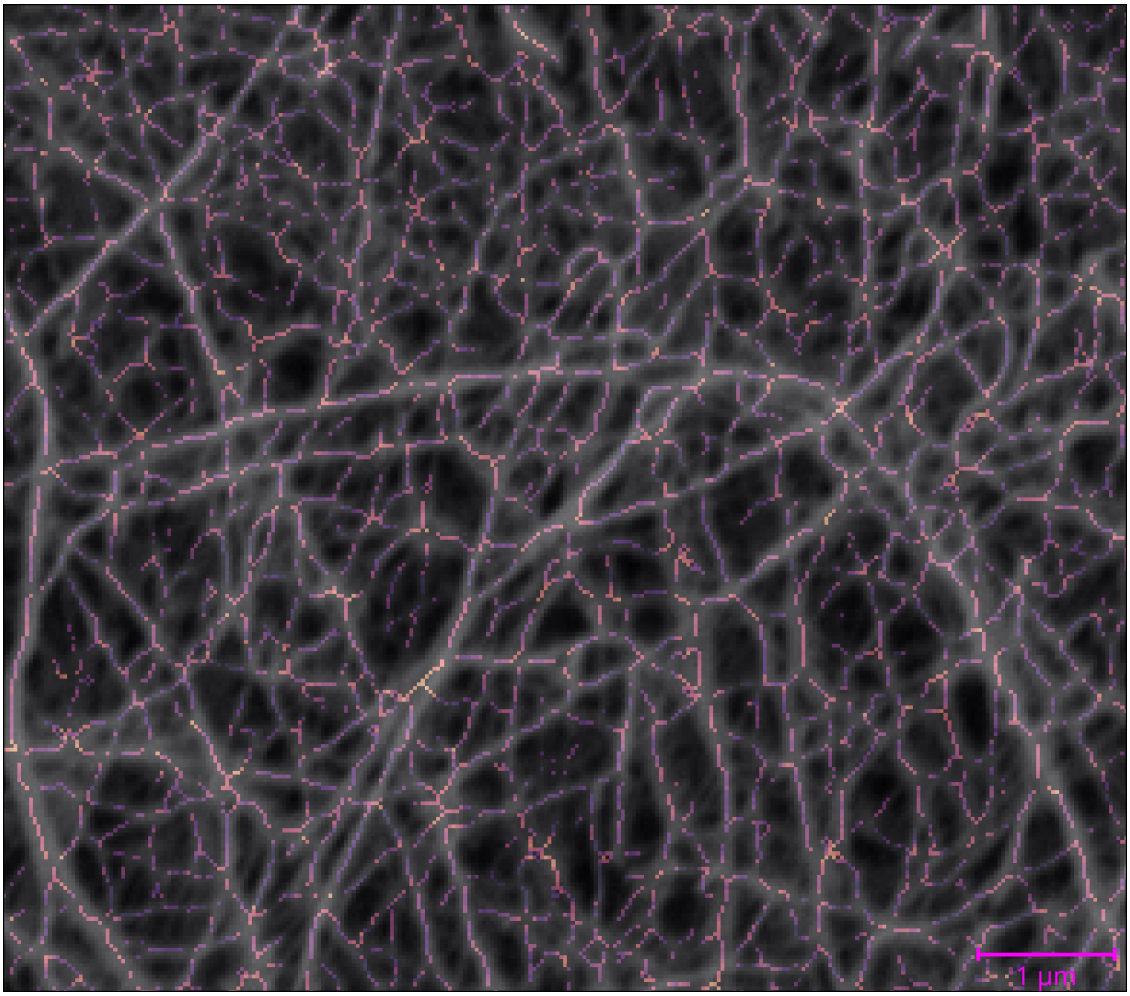
Now the diameter map (`dmap`) can be visualized with napari and simple summary statistics can be computed. The colormap (LUT) `magma` will be used and the contrast limits will be set to 1–10, which represents the diameters values (in pixels) to be represented by colors. Dark colors (black, dark purple) represent thinner fibers, progressing through reds, and to yellow, which indicates diameters of 10 px and higher (>280 nm).

[ ]:
```
viewer.add_image(dmap, scale=scale, name="DiameterMap", colormap="magma",␣
↪opacity=0.5, contrast_limits=[1,10])
```

[ ]:
```
<Image layer 'DiameterMap' at 0x1781c4a30>
```

[ ]:
```
napari.utils.nbscreenshot(viewer, canvas_only=True)      #a zoomed in view of␣
↪the diameter map overlayed on the image
```

[ ]:



Summary statistics calculated for `ovals` and `dvals`. Note, these returned values are of type `list` so we convert them into numpy arrays for the calculations.

[ ]:
```
mean_ovals = np.mean(np.asarray(ovals)*180/np.pi)                #mean converted␣
↪from radians to degrees
stdev_ovals = np.std(np.asarray(ovals)*180/np.pi)                #standard␣
↪deviation
mean_dvals = np.mean(np.asarray(dvals)*scale[0]*1000)           #mean converted␣
↪from px to nm
stdev_dvals = np.std(np.asarray(dvals)*scale[0]*1000)           #standard deviation
```

[ ]:
```
print("Fiber diameter: %3.0f nm ± %2.0f nm" % (mean_dvals, stdev_dvals))
print("Fiber orientation: %2.0f° ± %2.0f°" % (mean_ovals, stdev_ovals))
```

```
Fiber diameter: 122 nm ± 43 nm
Fiber orientation: 85° ± 47°
```