CSE554: Project Report

Sophia Dykstra & Olivia Guess
December 8, 2024

This project tracks the growth and shrinkage of a microtubule over time. A graphical user interface (GUI), implemented in Python as a Napari plugin, allows users to select a microtubule to track. The user-selected microtubule is segmented and fitted to a line to track its growth and shrinkage. Our interface reports the growth/shrinking events as length over time, displaying this information back to the user as a graph. It also creates a segmentation mask of the user-chosen microtubule.

Link to the Github repo with the source code:
https://github.com/OGuess10/napari-microtubules/tree/master

Required and Wishlist Features Accomplished

Required Features

- GUI Interface
    - Users can input video data in TIFF format
    - Users can select a microtubule in one frame of the uploaded video
    - Users can navigate the video frame-by-frame after segmentation and reselect the microtubule in each frame if it has been incorrectly segmented
    - Users can easily understand and use the GUI
    - Users receive the length data of their chosen microtubule after segmentation is performed
- Segmentation
    - The segmentation algorithm properly segments any microtubule in the video
        - The selected microtubule is mostly isolated from other microtubules, the background, and noise
        - The segmentation mostly encompass all parts of the microtubule
    - The segmentation is fairly robust and appropriately handles highly dynamic microtubules
        - For frames where the segmentation is off, users have the ability to re-select the microtubule to fix inconsistencies
    - The segmentation algorithm avoids exploiting user intractability

    - Manual user inputs are limited to resegmentation and merging segmentations
    - The algorithm effectively speeds up and improves upon the process of manual segmentation

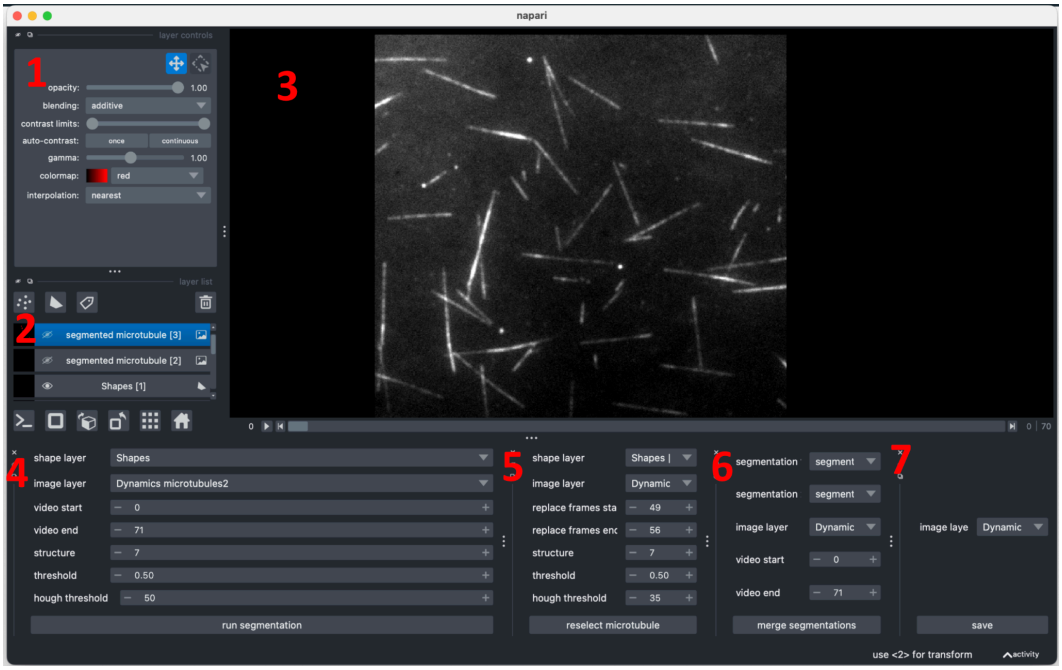
Wishlist Features

- The GUI interface is aesthetic for users
- The GUI displays an informative graph of the growth/shrinking events over time
- The GUI, however, is not able to track multiple microtubules at once and does not have the ability to compare/contrast their data

Core Algorithms, Coding Components, and GUI

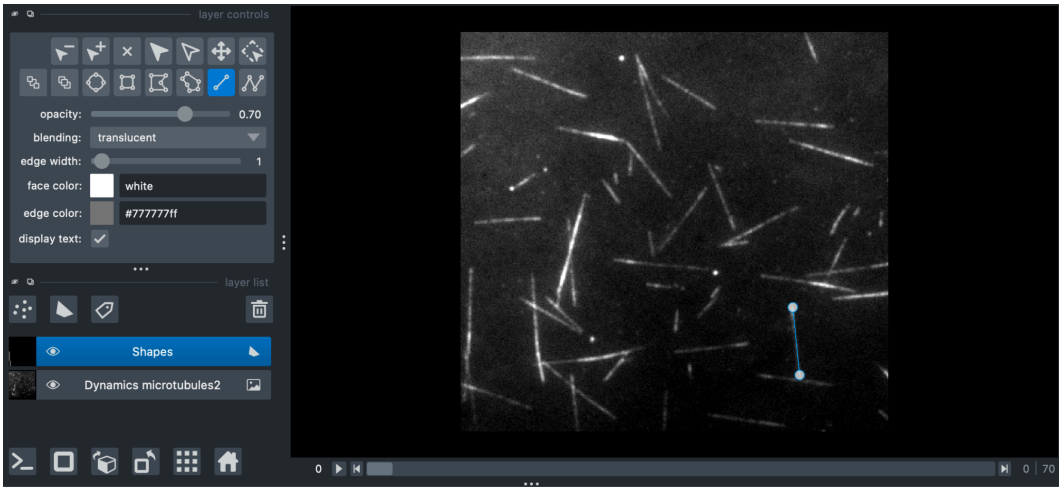
This project is a Napari plugin based in Python. The GUI consists of several elements:

1. **Layer Settings Panel**
    a. Users can tailor layer settings such as contrast, blending, gamma, etc.
2. **Layer Panel**
    a. The list of layers with options to create a new layer and hide/show layers
3. **Frame Viewer**
    a. Displays all layers
4. **Run Segmentation Panel**
    a. Users can choose the shape layer the line is drawn on, the image layer, the starting and ending frame for segmentation, structure and threshold for image processing, and hough threshold to find the best fit line
5. **Reselect Microtubule Panel**
    a. Users can select a shape layer with a line, the image layer, the starting and ending frames for resegmenting, and redo structure, threshold, and hough threshold
6. **Merge Segmentations Panel**
    a. Users can select two different segmentation layers and combine these into one layer
7. **Save Panel**
    a. Users can select a segmentation layer and save this to their computer

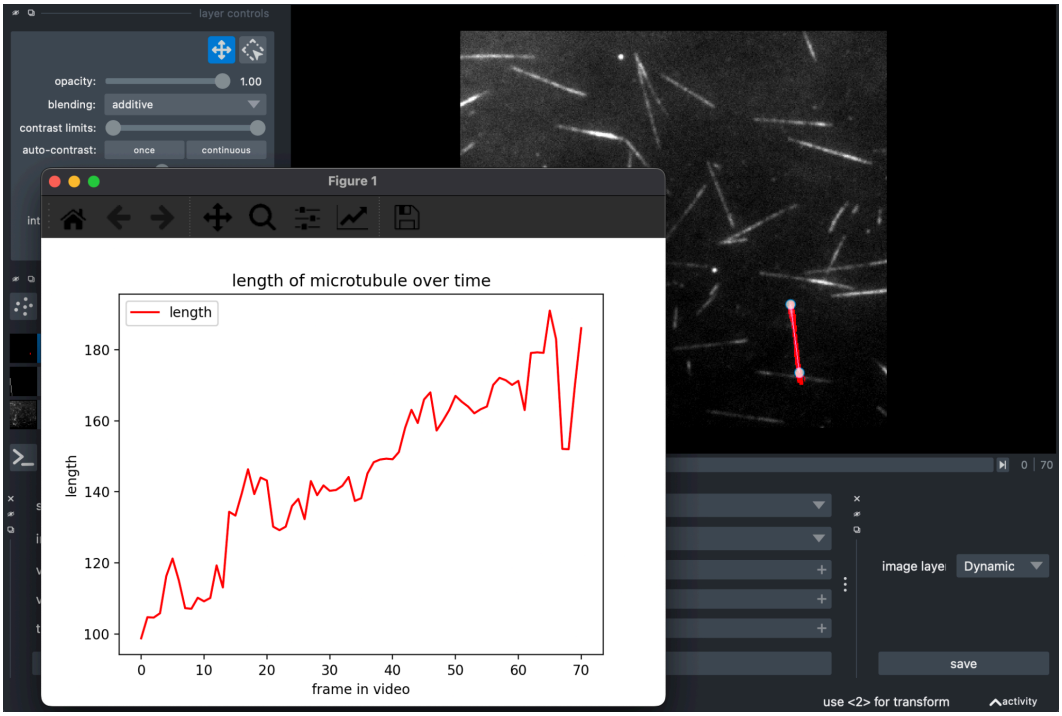


Our Napari plug-in, with panels labelled by numbers corresponding to the list above

The core algorithms used are gaussian blur, Otsu's Method, connected components, smoothing pipelines, Hough transform, and loss calculations. The user first selects a microtubule to segment, the clicks "run segmentation" to start the tool.

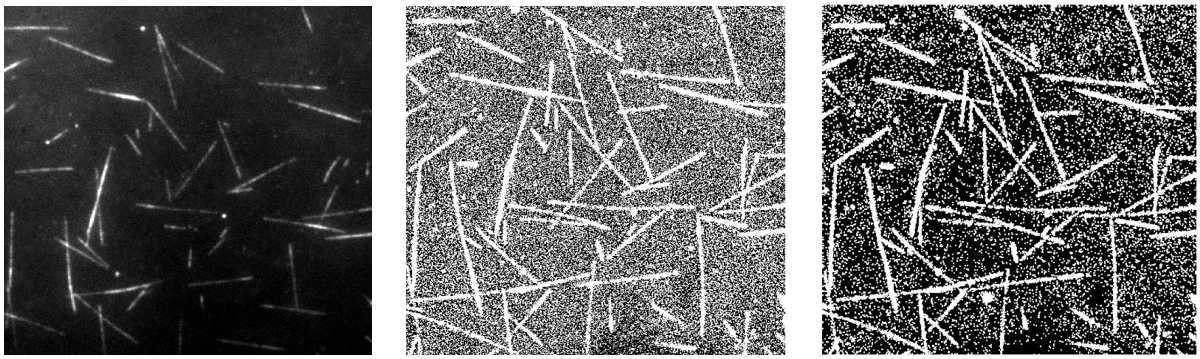


Napari dashboard after the user selects a microtubule to segment



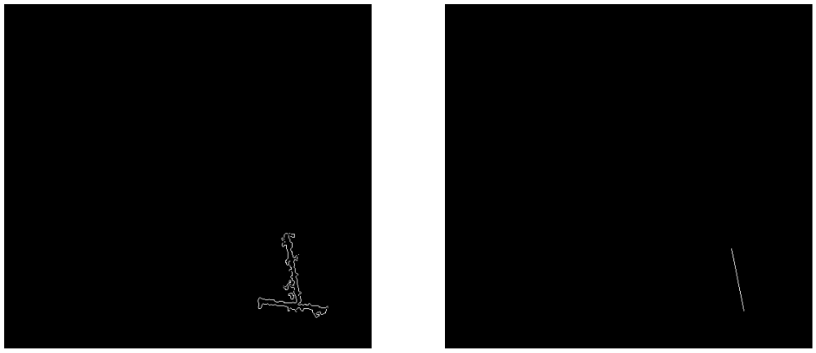
Napari dashboard after segmentation runs, includes a pop-up graph with the microtubule length

After a user draws a line over their desired microtubule on the starting frame, the program applies gaussian blur and Otsu's method to get an image binary. At this step, the Otsu thresholding calculates the adaptive threshold to maximizes between-class variance, which is useful for the frames that have higher intensity variance. From the binary image, the program finds the closest connected component and smooths the mask. Hough transform then finds lines on the image mask. Loss calculations on the user drawn line determines which line is the closest match. The component/pixels that best align with the closest match becomes the segmented mask. The length of the best-fit line is also recorded.



Original (left), binary after adaptive thresholding (middle), binary after thresholding pipeline (right)

After Gaussian Blur in Otsu's Method

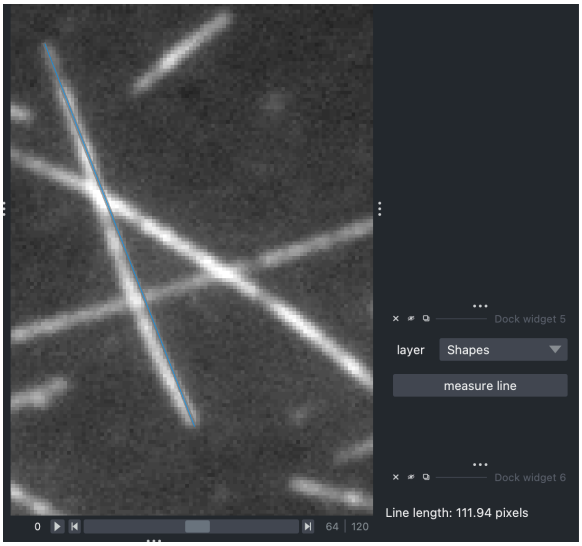


Microtubule edge detection (left) and best-fit Hough line (right)

On subsequent frames, the best-fit line from the previous frame is used in the loss calculation to determine the next best fit line. This way, users can scroll through each frame of the TIFF and get the segmented mask for all the frames.

Development and Quantitative Analysis

To calculate the error of the microtubule, another panel was added to the application to measure the length of user drawn lines. We ran the segmentation algorithm for six different microtubules, varying in length, overlap, dynamism, and video. We recorded the calculated length of the microtubule outputted from the tool. Then, for each frame, we drew a line that represented the length of the microtubule and recorded its length. The results below are after one segmentation run, based solely off the initial user line — frames were not resegmented. The calculated and measured lengths are in pixels, and error is the relative error as a percentage.



Manually measuring the length of the microtubule

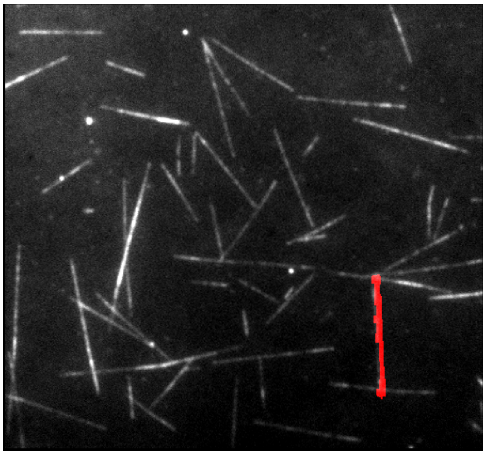
Microtubule 1

Chosen for a single overlap on the bottom edge of the microtubule.

Source

Video: Dynamic microtubules2.tif
Structure: 7
Threshold: 0.50
Hough Threshold: 50

First Frame of Segmentation:



Results

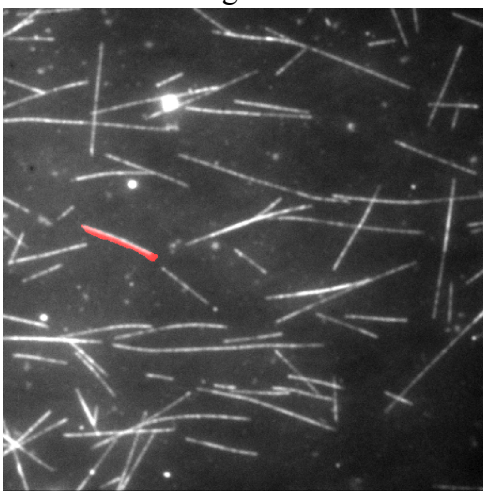
Frame	Calculated	Measured	Error
20	129.189783	125.28	3.12083563
21	128.471787	123	4.44860714
22	132.185476	124.92	5.81610291
23	130.188325	126.56	2.86688142
24	129.189783	126.14	2.41777618
25	136.180028	127.71	6.63223546
26	134.182711	128.62	4.32491934
27	137.524543	128.96	6.64124012
28	139.359248	128.73	8.25700924
29	138.361844	132.65	4.30595134
		Average:	4.88315588

Microtubule 2

Source

Video: 1380 nM SPR1GF WT - 1…00063595_z0_T0000_C1.tiff
Structure: 7
Threshold: 0.50
Hough Threshold: 35

First Frame of Segmentation:



Results

Frame	Calculated	Measured	Error
70	85.6154192	80.36	6.53984466
71	86.9252552	80.76	7.63404563
72	81.9390017	81.45	0.60037041
73	78.9240141	80.58	2.05508304
74	82.07923	80.95	1.39497222
75	84.3089556	80.17	5.16272375
76	81.9328994	80.63	1.61589901
77	82.6377638	78.79	4.88356868
78	83.7735042	80.03	4.6776261
79	81.0246876	81.1	0.09286363
		Average:	3.46569971

Microtubule 3

Chosen because it overlaps a few other microtubules.

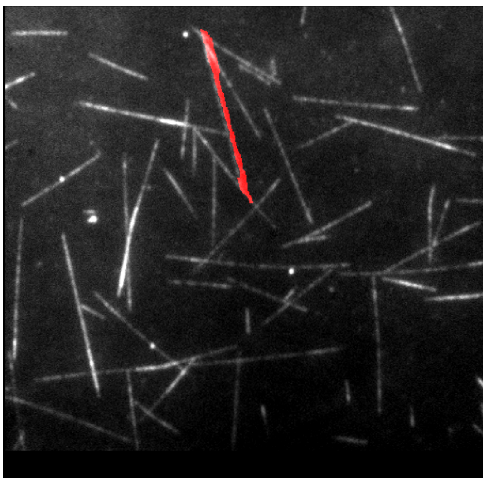
Source

Video: Dynamic microtubules2.tif
Structure: 7
Threshold: 0.50
Hough Threshold: 50

First Frame of Segmentation:

Results

Frame	Calculated	Measured	Error
40	181.201545	168.82	7.33416968
41	182.386951	173.49	5.12822139
42	188.215302	169.9	10.7800484



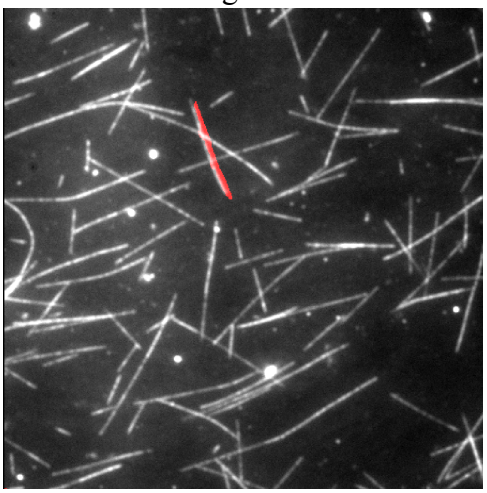
43	178.538511	173.41	2.95744839
44	182.167505	174.03	4.67592101
45	184.358347	172.56	6.83724311
46	189.282857	172.71	9.59577159
47	184.358347	173.33	6.36263007
48	186.123615	171.15	8.7488255
49	187.256509	172.41	8.61116441
		Average:	7.10314436

Microtubule 4

Source

Video: 1380 nM SPR1GFP WT-10_XY1500060270_Z0_T000_C1.tiff
Structure: 7
Threshold: 0.50
Hough Threshold: 50

First Frame of Segmentation:



Results

Frame	Calculated	Measured	Error
60	106.625513	108.32	1.56433448
61	106.976633	109.38	2.1972637
62	103.46497	109.17	5.22582219
63	108.268186	111.85	3.20233747
64	106.042444	111.94	5.26849711
65	114.162165	115.28	0.96966919
66	117.324337	112.8	4.01093686
67	116.456	115.09	1.18689745
68	117.745488	115.75	1.72396392
69	117.987287	118.71	0.60880511
		Average:	2.59585275

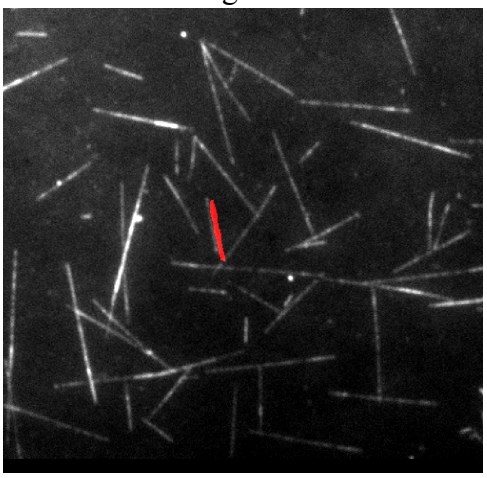
Microtubule 5

Chosen because it grows and shrinks rapidly.

Source

Video: Dynamic microtubules2.tif
Structure: 7
Threshold: 0.50
Hough Threshold: 35

First Frame of Segmentation:



Results

Frame	Calculated	Measured	Error
22	59.6657356	61.11	2.36338478
23	62.8171951	61.44	2.24152848
24	80.0499844	69.29	15.5289138
25	79.906195	64.7	23.5026198
26	89.5600357	74.88	19.6047486
27	82.2982381	70.92	16.043765
28	66	69.67	5.26769054
29	68	81.22	16.2767791
30	68	80.98	16.028649
31	81.0061726	80.25	0.94227116
		Average:	11.780035

Microtubule 6

Chosen because it grows and shrinks rapidly.

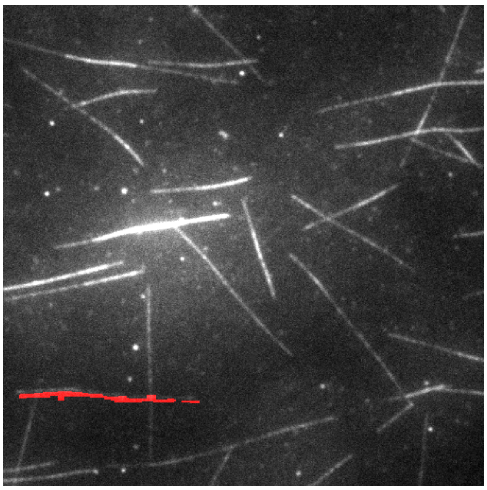
Source

Video: MT dynamics example image stack.tif
Structure: 7
Threshold: 0.50
Hough Threshold: 50

First Frame of Segmentation:

Results

Frame	Calculated	Measured	Error
80	189.129585	190.06	0.48953741
81	156.38734	165.13	5.29441068
82	162.077142	153.45	5.62211934
83	108.018517	144.74	25.3706529



84	102.078401	117.41	13.0581712
85	96.1301201	99.93	3.80254163
86	88.0511215	89.46	1.57486976
87	81.0061726	79.01	2.52648096
88	80	65.66	21.8397807
89	76.2364742	62.39	22.1934192
		Average:	10.1771984

The average error overall is 7.02438604.

Future Work

For future work, it would be useful to implement the last feature on our wish list: tracking multiple microtubules at once and capabilities to compare/contrast their data.

Custom morphological functions, like opening and closing, take a long time to run on TIFF files with a large amount of frames. This can be fixed using open-cv, but exploring *why* the custom functions increase runtime could be valuable.

The tool also struggles with extremely curved microtubules. Rather than having user's re-segment the microtubule, streamlining the process to automatically handle extremely curved microtubules would be an improvement. Additionally, the tool sometimes shifts the segmented mask slightly off of the chosen microtubule. The tool is also prone to holes in segments and jagged edges. Fixing these bugs would be part of future work.