Clathrin Light Chain with Green Fluorescent Protein Image Segmentation

William Plucinsky, wplucinsky@drexel.edu
Department of Electrical and Computer Engineering, Drexel University, Philadelphia, Pennsylvania 19104

***Abstract* – Matlab's Image Processing Toolbox will be utilized to perform a segmentation on a 600 layer stack of clathrin light chain microscope images.**

I. Introduction

This project aims to segment clathrin light chains that have been tagged with green fluorescent proteins using Matlab [1]. The images have been provided by the Laboratory for Optical and Computational Instrumentation (LOCI) [2], and will be segmented using a variety of Matlab image filtering and identification functions within Matlab's Image Processing Toolbox [3] along with techniques learnt in Drexel University's Cell and Tissue Image Analysis class [4].

Some challenges of this segmentation are the small, black and white, 131px by 122px images with varying intensities between frames. There is also a need to segment both the large and small clathrin light chains which are highlighted in Figure 1 in cyan. Care must be taken to ensure the pre-filtering does not remove the small chains in an attempt to brighten the larger chains.

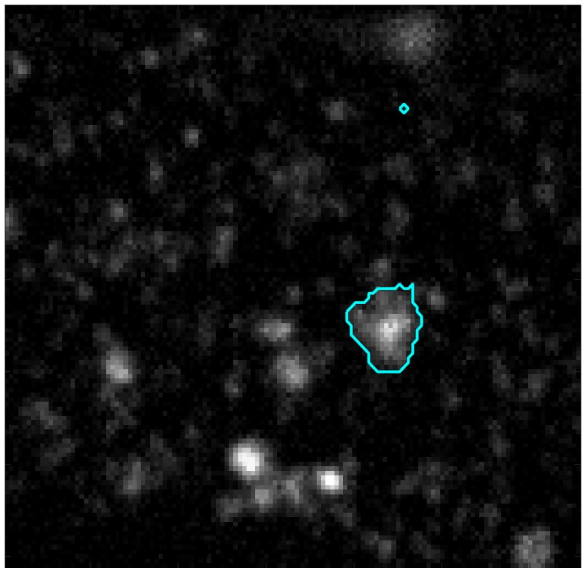

Figure 1. Large and Small Clathrin Light Chain Segmentation

II. Process

To perform the segmentation the 600 layer Tagged Image File Format (TIFF) image stack was first imported into Matlab and viewed using the `imagesc` [5] function. This was performed to more easily view the intensities which can be seen in Figure 3, when compared to Figure 2, in order to prepare a segmentation plan.

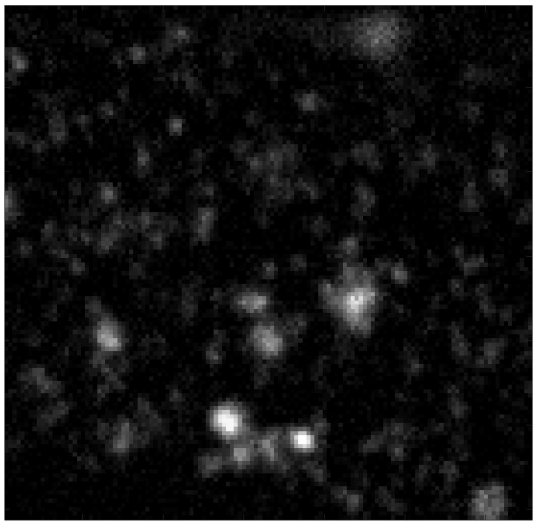

Figure 2. Supplied Image

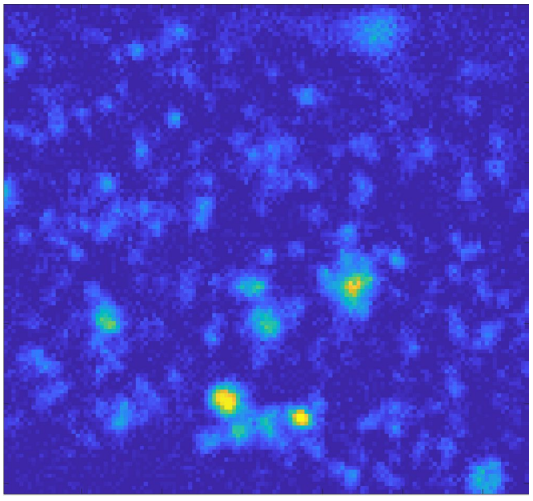

Figure 3. Supplied Image Viewed Using `imagesc()`

The contrast and sharpness of these images were raised using `imsharpen()` [6] with empirically chosen values for the radius and amount parameters. This brought up a lot of salt and pepper noise for which Matlab's `medfilt2` [7] function is designed to reduce. This initial processing can be seen in Figure 4 and Figure 5, respectively.

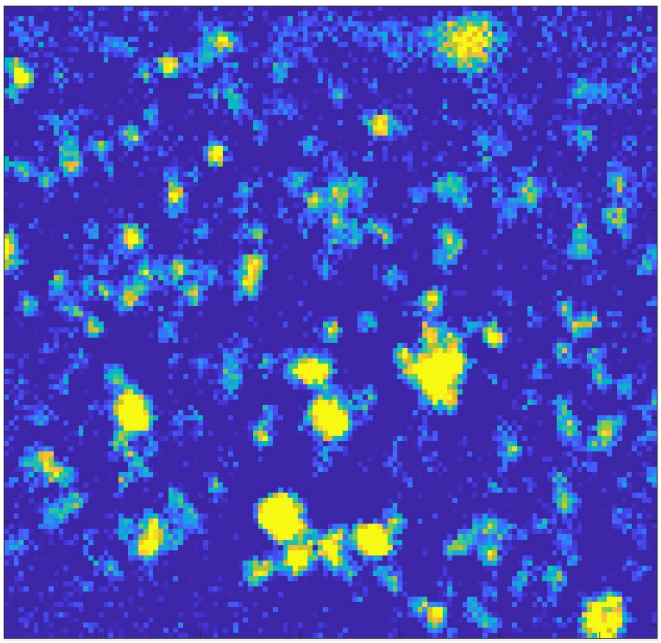

Figure 4. Process Step 1: `imsharpen()`

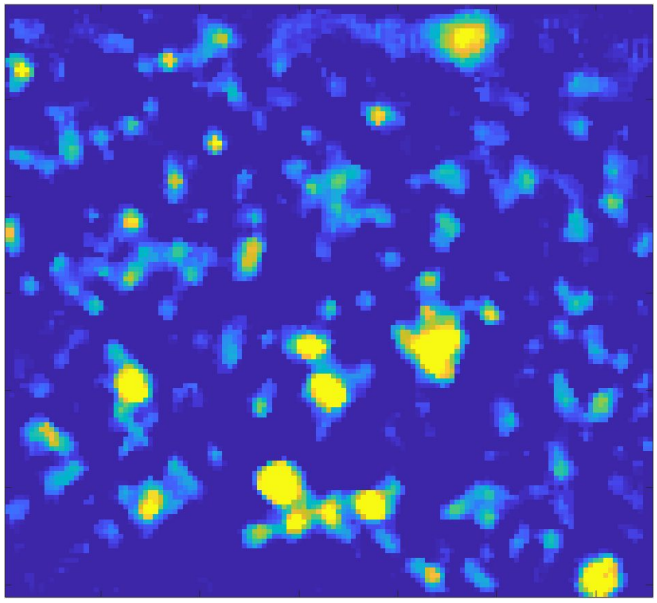

Figure 5. Process Step 2: `medfilt2()`

Although the median filter removed all of the salt and pepper noise, it also eroded a lot of the clathrin cell data. To restore this information the `imdilate` [8] function was used with a supplied disk structuring element set to a radius of one to not over dilate the cells. This made it clear there were some elements in the image that were not in the originally supplied image, so a simple thresholding function was used to eliminate this data. These processing steps can be seen in Figure 6 and Figure 7, respectively. Although minimal, the thresholding can be seen most when comparing the bottom left between Figure 6 and Figure 7. Once the noise of the image was cleaned up and the number of pronounced cells had been raised significantly, it was time to start the segmentation process.

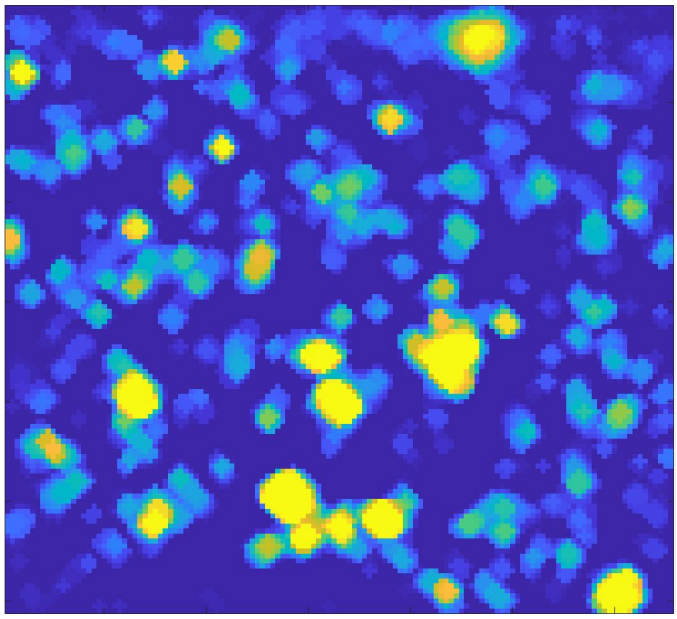

Figure 6. Process Step 3: `imdilate()`

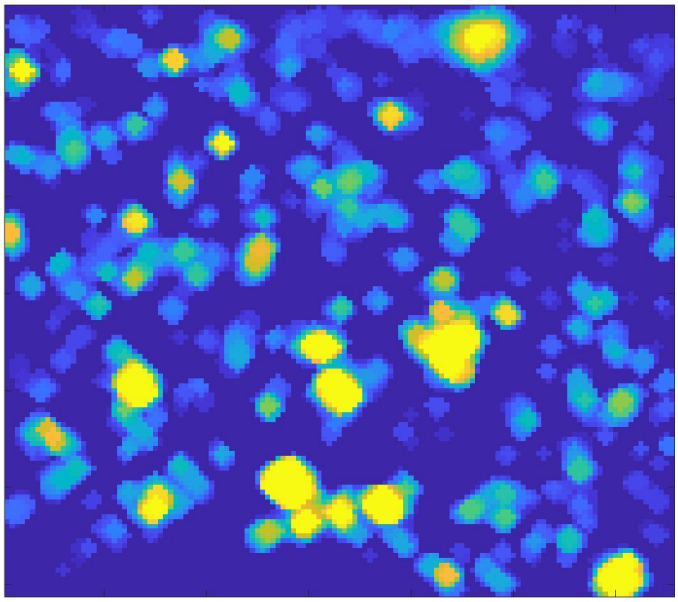

Figure 7. Process Step 4: `find()` Threshold

First, the euclidean distance transform calculation was performed using Matlab's built in `bwdist` [9] function. Then the watershed transform, `watershed()` [10], was selected as the clathrin cells were in contact with each other and needed to be separated. This data was then easily passed to the `bwboundaries` [11] function where the cell boundaries could be drawn on both the filtered image and the original image, as seen in Figure 8 and Figure 9, respectively. Code can be seen in Appendix C.

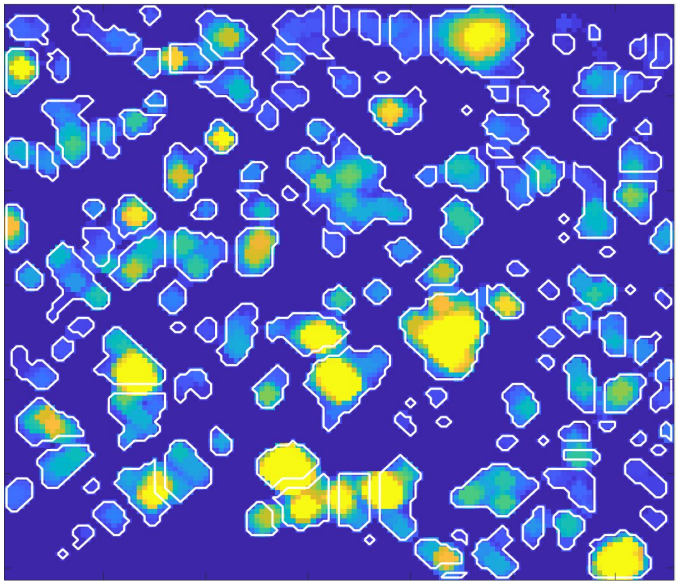

Figure 8. Process Step 5: Segmentation

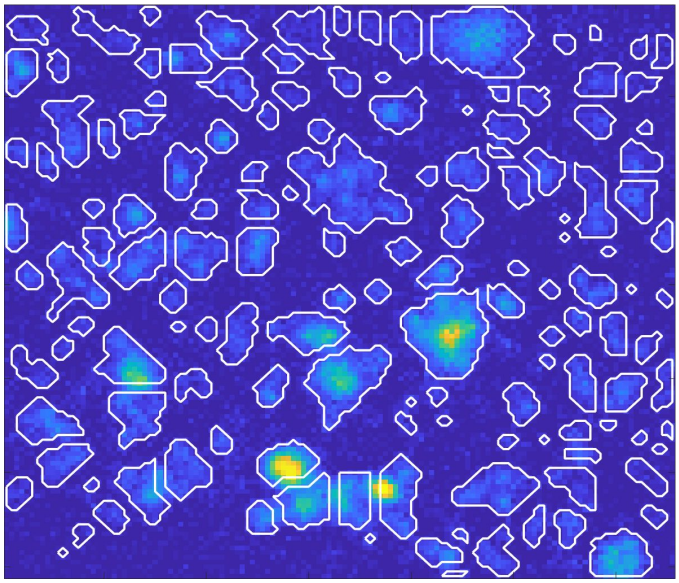

Figure 9. Segmentation on Original Image

III. Analysis

At this point, the first image of the stack was properly segmented, but the algorithm had to hold up to the intensity variations in the remaining images of the 600 layer stack. Figure 10 shows the first encountered issue, outlined in pink, where the algorithm created a segment with a large perimeter but less than a two square pixel area.

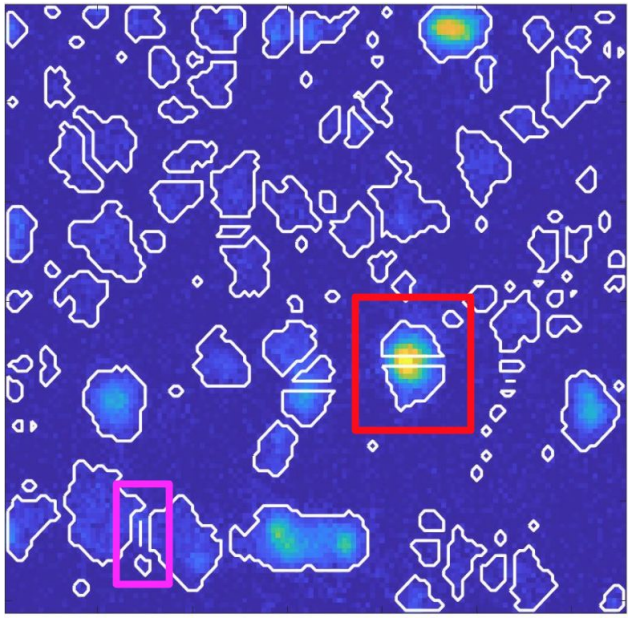

Figure 10. Segmentation Issues

This issue was found and solved by getting the area using Matlab's `polyarea` [12] function of the boundary box then adding a check to only plot the boundary if the area was over two square pixels. The updated segmentation can be seen in Figure 11.

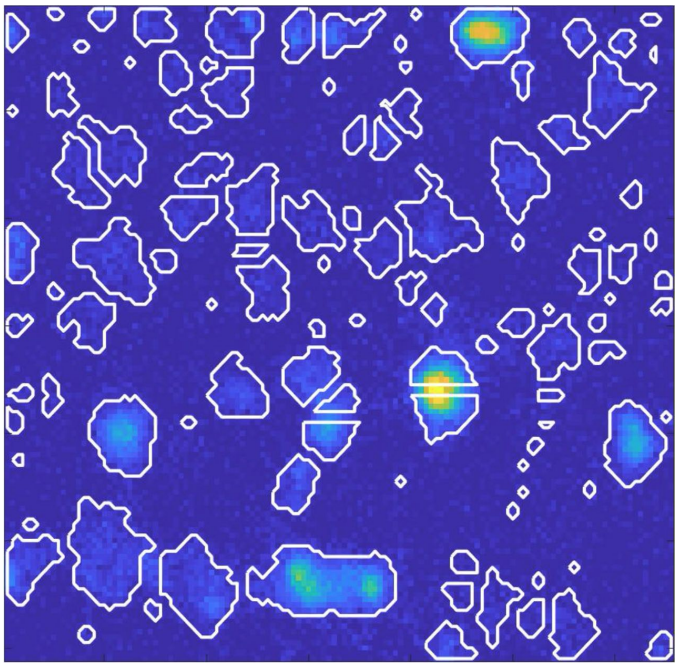

Figure 11. Fixed Segmentation Issue One

The red box seen in Figure 10, where a cell is mistakenly divided into two cells, was seen in only 3% of segmented images and fixing it caused issues that negatively affected the entire image stack, so the fix was not implemented.

To provide a comparison between the dataset as a whole, the area, perimeter, and number of segmentations were plotted in Figure 12 through Figure 15. One major takeaway from the area histogram is that the small clathrin cells greatly outnumber the large and medium cells, and that there is nearly a 450px difference between the largest and smallest cells. The perimeter histogram shows that the perimeter is not as widely distributed as the area, and there is a much smaller spread of the data, just 180px. With the number of segmentations histogram, we see a slightly right skewed distribution with the maximum being exactly double the minimum. The scatter plot in Figure 12 shows that the data is very much clumped with very few outliers. The linear regression model can be seen in Appendix A.

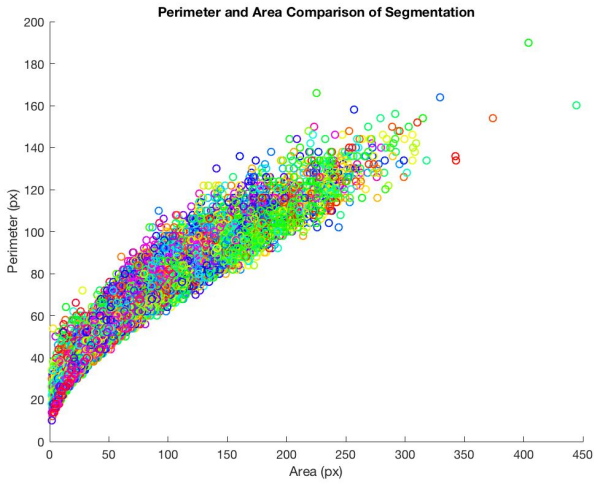

Figure 12. Perimeter vs. Area Scatter Comparison

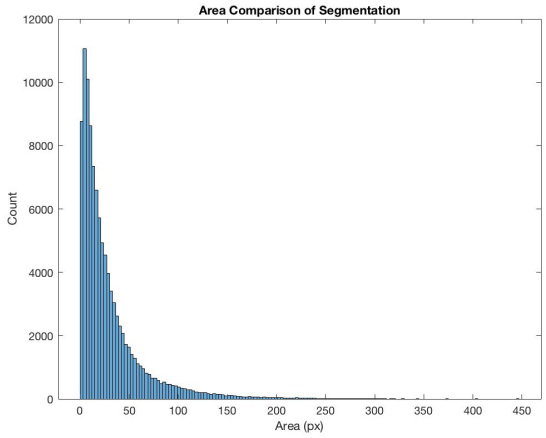

Figure 13. Area Comparison Histogram

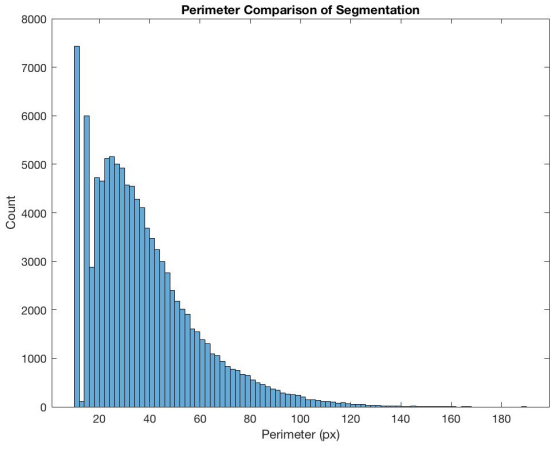

Figure 14. Perimeter Comparison Histogram

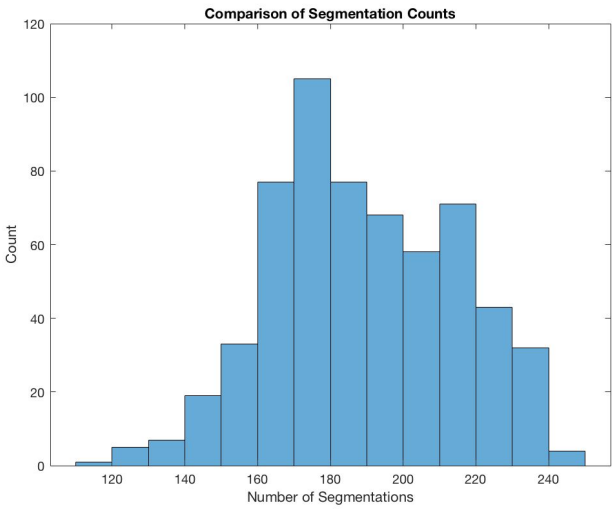

Figure 15. Number of Segmentation Histogram

Creating the segmentation algorithm from just the first image in the stack worked for this analysis, simply because that image fit with the global structure of the stack. This can be seen when comparing Figure 16 through Figure 18 with Figure 12 through Figure 14. The linear regression model for Figure 16 can be seen in Appendix B.

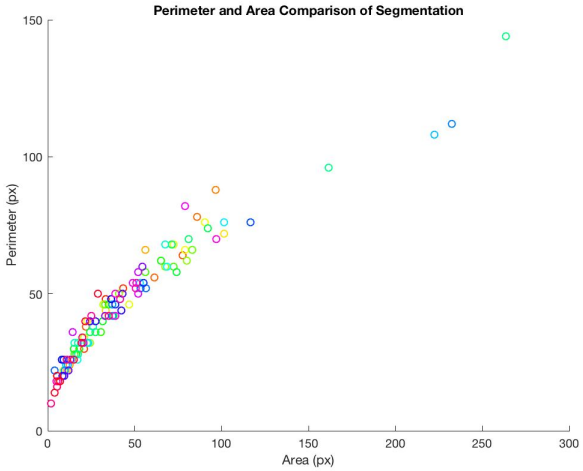

Figure 16. Perimeter vs. Area Comparison, Image One

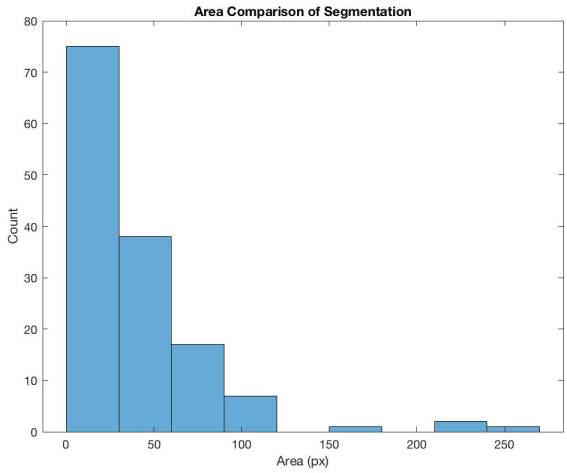

FIgure 17. Area Histogram, Image One

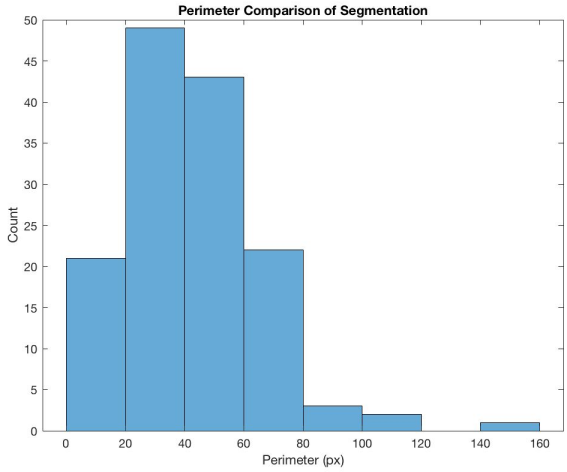

Figure 18. Perimeter Histogram, Image One

IV. Conclusion

The first image of the stack accurately represented the midpoint of all the variations in all the image layers, making changes after the initial segmentation minimal. In future segmentations, a process of finding the average of all the images could be helpful in identifying an image to start with. This could be done by user selection or an automated script described in [13].

In order to verify the segmentation results, which look valid for these images, data would need to be collected on the true sizes of clathrin light chains. This could be helpful to determine the likelihood of an area over 300px and possibly trigger a second segmentation on that area.

The same could be done for the small areas to possibly exclude segmentations that are actually just noise.

To continue with this segmentation, the error seen in the red box of Figure 10 could be remedied. Ideally the issue would be identified and corrected by a software fix instead of a user needing to intervene part way. Most likely the entire segmentation process would need to be altered. With this complete, tracking cells between layers could be implemented, but challenges would arise as most of the small cells disappear between frames.

In summation, clathrin light chains coated with green fluorescent protein are relatively straightforward to segment once sufficient and targeted preprocessing has been completed. This segmentation method could be applied to different cell types, but the empirically chosen values would need to be altered accordingly.

V. References

[1] Matlab Corp. (2017, March 9). MATLAB. Retrieved February 12, 2018, from https://www.mathworks.com/products/matlab.html

[2] Laboratory for Optical and Computational Instrumentation. (n.d.). Retrieved February 12, 2018, from https://loci.wisc.edu/

[3] Matlab Corp. (2017, March 9). MATLAB. Retrieved February 12, 2018, from https://www.mathworks.com/products/image.html

[4] Drexel University. (n.d.). Electrical & Computer Engineering - Systems. Retrieved March 12, 2018, from http://catalog.drexel.edu/coursedescriptions/quarter/undergrad/eces/

[5] Matlab Corp. (2018, Jan 1). Display Image with Scaled Colors - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/matlab/ref/imagesc.html

[6] Matlab Corp. (2018, Jan 1). Sharpen Image Using Unsharp Masking - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/images/ref/imsharpen.html

[7] Matlab Corp. (2018, Jan 1). 2-D Median Filtering - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/images/ref/medfilt2.html

[8] Matlab Corp. (2018, Jan 1). Dilate Image - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/images/ref/imdilate.html

[9] Matlab Corp. (2018, Jan 1). Distance Transform of Binary Image - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/images/ref/bwdist.html

[10] Matlab Corp. (2018, Jan 1). Watershed Transform - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/images/ref/watershed.html

[11] Matlab Corp. (2018, Jan 1). Trace Region Boundaries in Binary Image - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/images/ref/bwboundaries.html

[12] Matlab Corp. (2018, Jan 1). Area of Polygon - MATLAB. Retrieved March 12, 2018, from https://www.mathworks.com/help/matlab/ref/polyarea.html

[13] Matlab Corp, A. Taylor. (2014, Jan 14). Compare Images and Find the Most Similar - MATLAB Answer. Retrieved March 19, 2018, from https://www.mathworks.com/matlabcentral/answers/112388-compare-images-and-find-the-most-similar#answer_120892

VI. Appendix

A. Figure 12. Linear Regression Model

```
Linear regression model:
    y ~ 1 + x1

Estimated Coefficients:
                   Estimate      SE       tStat       pValue
                   ________    ________    ______    ___________

    (Intercept)     19.177       0.578     33.179     1.3863e-82
    x1             0.62509     0.01386     45.101    8.9084e-106


Number of observations: 199, Error degrees of freedom: 197
Root Mean Squared Error: 6.18
R-squared: 0.912,  Adjusted R-Squared 0.911
F-statistic vs. constant model: 2.03e+03, p-value = 8.91e-106
```

B. Figure 16. Linear Regression Model

```
Linear regression model:
    y ~ 1 + x1

Estimated Coefficients:
                   Estimate      SE       tStat       pValue
                   ________    ________    ______    ___________

    (Intercept)     21.69        0.86294    25.135    1.4791e-53
    x1              0.5121      0.015261    33.556    1.5774e-68


Number of observations: 141, Error degrees of freedom: 139
Root Mean Squared Error: 7.51
R-squared: 0.89,  Adjusted R-Squared 0.889
F-statistic vs. constant model: 1.13e+03, p-value = 1.58e-68
```

C. MATLAB Segmentation Code.