3D Reconstruction of Collagen Fibers in Cornea

Zachary DeStefano

Due Date: June 13, 2014

1 Introduction

I work in the graphics group with Gopi and Aditi. The group was approached by the Department of Ophthalmology to see if we can reconstruct various parts of the eye using images they provided. I decided to work on reconstructing collagen fibers in the cornea. This mean taking cross-sectional images of them and using that to reconstruct them in a 3D virtual environment. The important aspects of the fibers include shape and size so I just had to locate them in each picture using brightness and not worry about a color value. The images are somewhat noisy and grainy and the fibers vary in shape and thickness so I had to use a variation of clustering and filtering in order to accurately locate them. Once located, I put the segmentation results into a volumetric data set and used a volume renderer to display it. Here are the steps at a glance that I will go into detail on:

1. Do Initial Smoothing of the Image
2. Do Clustering to get the fibers
3. Smooth the segmented image
4. Add Image to 3D data set
6. Repeat Steps 1-4 for all images in data set
7. Use Volume Renderer to show the images in a 3D environment

There was an attempted step here of trying to align the images together. I tried various methods to see what the best overlap would be between segmented images but in the end, it made sense to just overlay the images on top of each other without doing any transformations of them.

2 Initial Smoothing Step

For the initial smoothing, I tested out filters that would make the image look less grainy than the original image. I was aiming for a filter that would make the image look like it was a series of bands of black and white. That way whatever clustering algorithm I was using would have an easier time locating the fibers. I ended up trying Gaussian filters, median filters, and averaging filters.

Gaussian filters are convolution filters where the highest value is in the middle and the rest of the values are spread out using a Gaussian distribution. Median filters work by replacing each pixel with the median pixel in some neighborhood around the pixel. Averaging filters are convolution filters with uniform values, so that each pixel is replaced by a uniform average of the pixel values around it. For this step, the averaging filter worked the best. This is likely because averaging the pixels is more likely to create uniform bands across the image.

Below are pictures showing the results of different smoothing filters. In order to show this, here is first the original image to be smoothed.

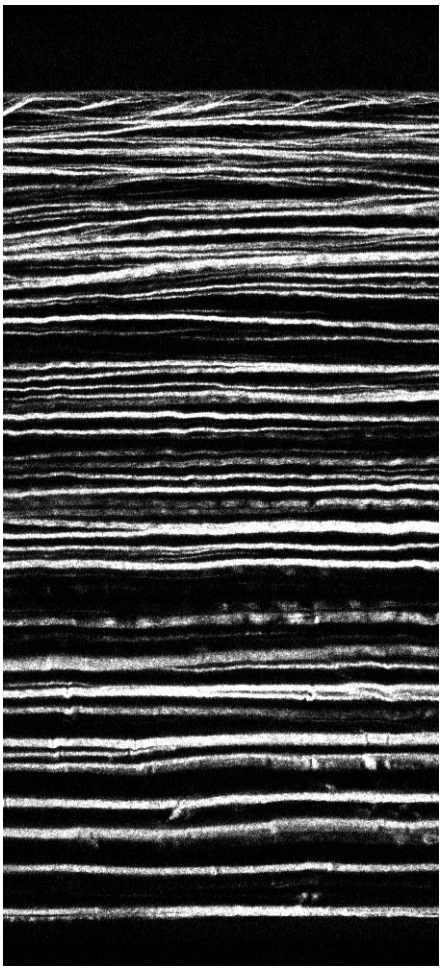


Figure 1: An example image to be smoothed

Here is a picture of the results of the three different filters.

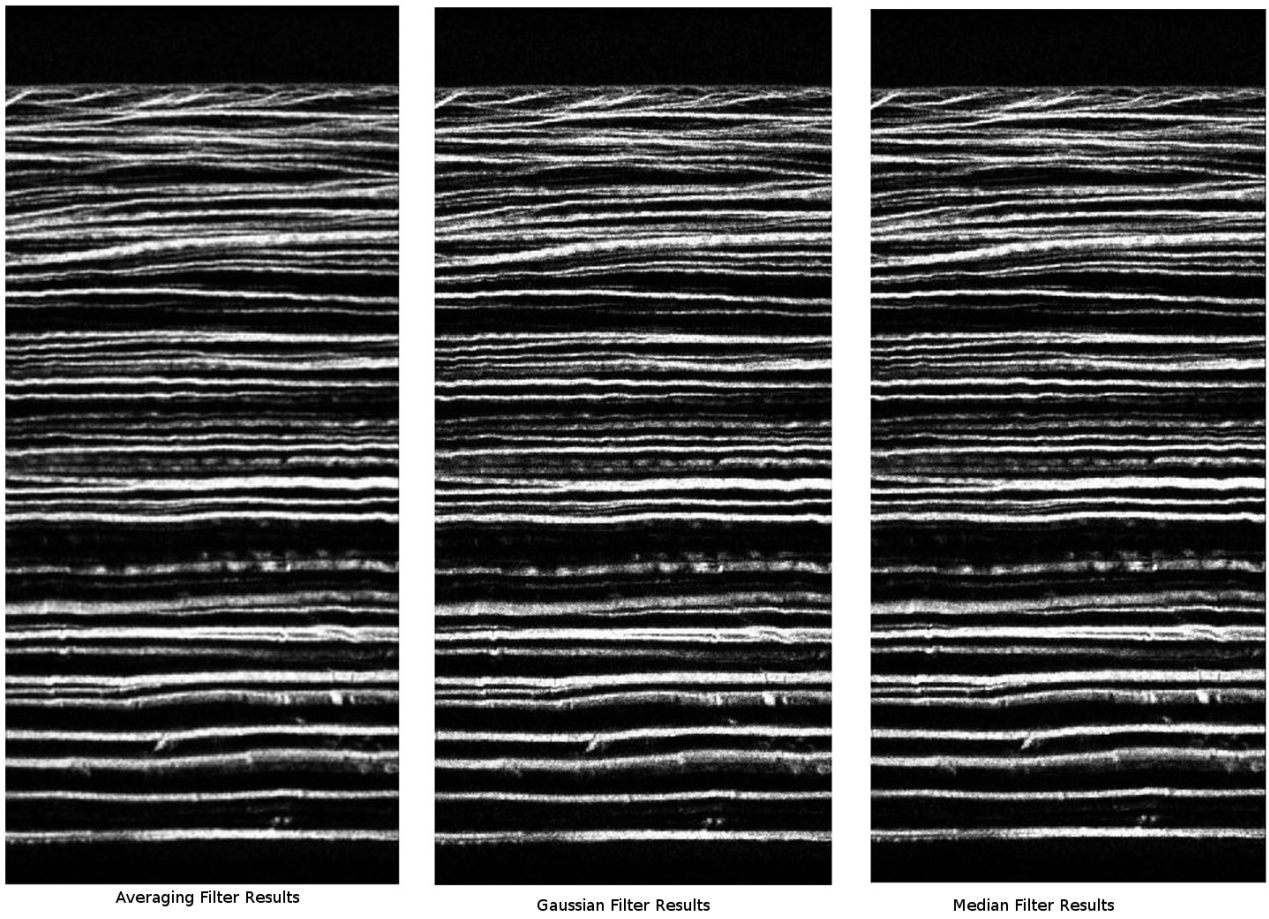


Figure 2: The three different initial smoothings done

3 Clustering Step

This was the most difficult and most important part of the project. Locating the fibers is all about clustering together the different regions of the image where the brightness is high. This was especially difficult for this problem due to the variation in the fibers. The thin fibers at the top were especially easy to miss. With this in mind, I tried the EM algorithm, the min-cut algorithm for MRF, k-Means, simple thresholding, as well as object correlation, and in the end k-Means performed the best.

3.1 Simple Thresholding

I first tried simple thresholding, where any brightness value over a certain amount was classified as a fiber. While this produces some good results, it was not taking spatial proximity into account in the clustering,

which is important for this problem. After trying thresholding though, it turned out that 0.3 was a good classification boundary and I ended up using that when classifying clusters generated by other methods.

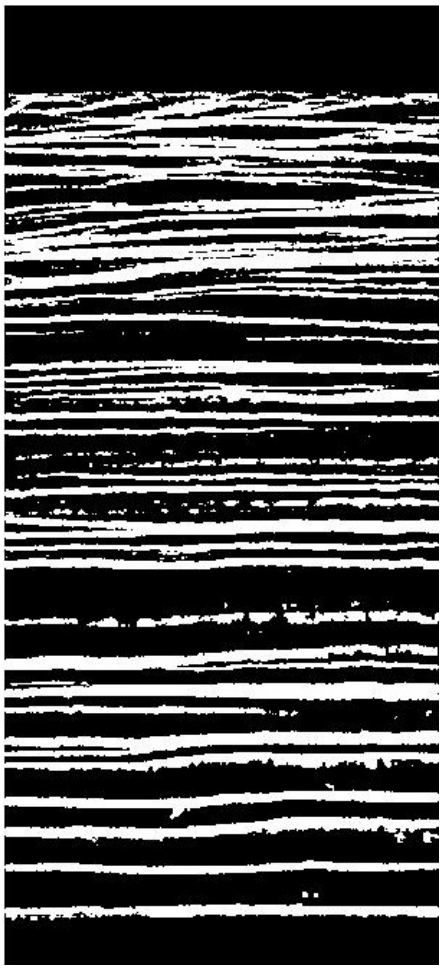


Figure 3: The segmentation result with simple thresholding

3.2 Simple k-Means

I tried k-Means with just the brightness values. The k-Means algorithm is one that iteratively groups points into clusters by whichever cluster center they are nearest. It then uses the mean values of the existing clusters to redefine the cluster center. I used $k = 2$ so that it would group it into one cluster that is not a fiber and one that is a fiber. The result was good but there was a lot of noise in the clustering likely due to the fact that just like simple thresholding, it was not taking into account spatial proximity. Below is the result of the initial k-Means clustering where the greater brightness cluster is colored white and the lower brightness cluster is colored black.

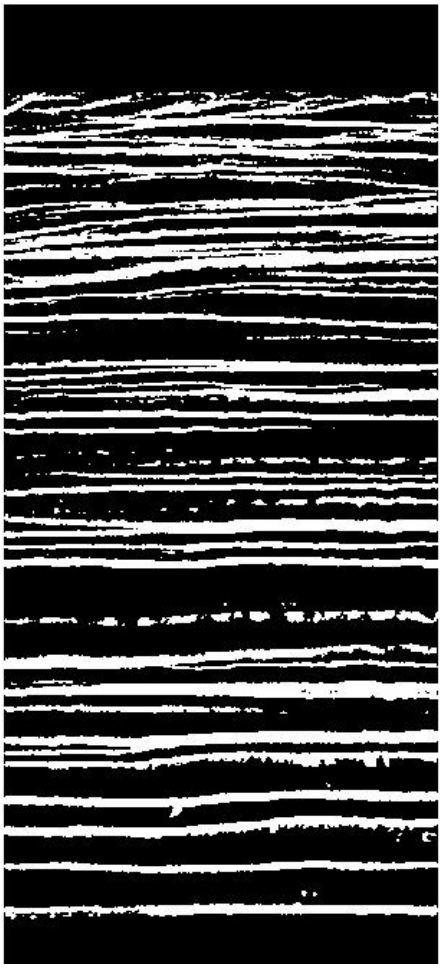


Figure 4: k-Means result when k=2

3.3 k-Means on spatial and brightness data

When I tried k-Means but with $k = 30$ and points which consisted of a normalized x and y value as well as brightness, I ended up with very nice looking fibers. This method was still able to detect the thin fibers at the top. From my experience with thresholding, I used 0.3 as the classification boundary between clusters classified as representing regions that were fibers and ones that were not fibers. Below is the result of running k-Means in this manner with the raw clusters displayed on the left and the segmented clusters displayed on the right.

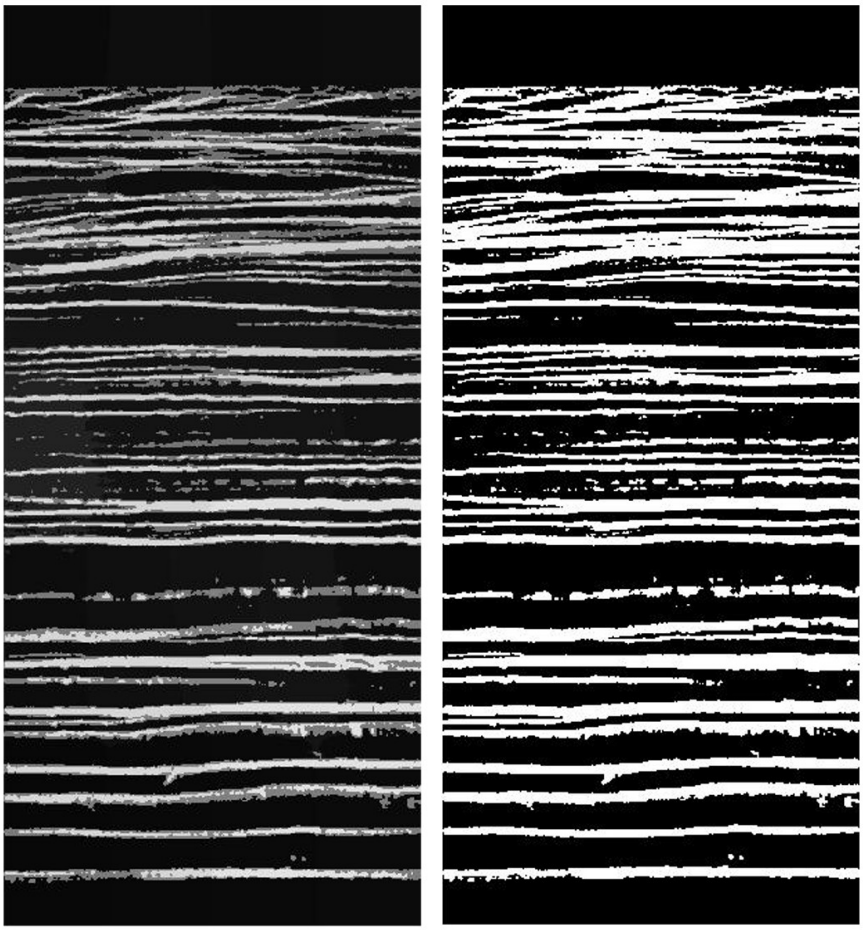


Figure 5: k-Means result when k=30 and (x,y,brightness) are the points

3.4 EM Algorithm for Gaussian Mixture Models

I then decided to try and use the EM algorithm to see if it would improve the clustering. This particular EM algorithm works by assuming that the clusters are a series of Gaussian distributions, also called a Gaussian Mixture Model. It iteratively updates the cluster information by maximizing the likelihood of particular cluster centers with our data. I used the same initial procedure as the more complex k-Means where I made points that consisted of a normalized x and y as well as brightness value. Instead of calling k-Means to do the clustering, I called the EM algorithm. The code randomly initialized 20 cluster centers. I tried having it use 30 cluster centers because that is what worked well with k-Means, but unfortunately it did not converge when I did that. I used external code for calling the EM algorithm. In the file *emScript.m* in the associated zip file of code is a link to the webpage on the Matlab file exchange where I obtained the code.

Unfortunately, the EM algorithm did not perform better than k-means. It did not bring out the thin fibers in

the top very well and parts of other fibers that should have appeared did not seem to appear. Additionally, the unsegmented image was quite noisy and did not have nice looking clusters like what appeared with k-Means. It could be that the EM algorithm is meant to be used when the data follows a Gaussian model, whereas this data set may not really be following that model. There are also often problems that can occur with using the EM algorithm due to the fact that the Gaussians could overlap significantly. Since the fibers are so close together in this case, that is definitely a potential issue. Another problem could have been that the initialization was random and not done with k-Means. Due to the problem of our data not really being Gaussian, the algorithm would likely still have tended toward improper clusters. Due to all those problems and the poor intial results, I decided not to try the route of improving the EM algorithm results.

Below is the result of using *imagesc* to display the resultant clusters from running the EM algorithm. As can be observed, the result seems noisier than k-Means.

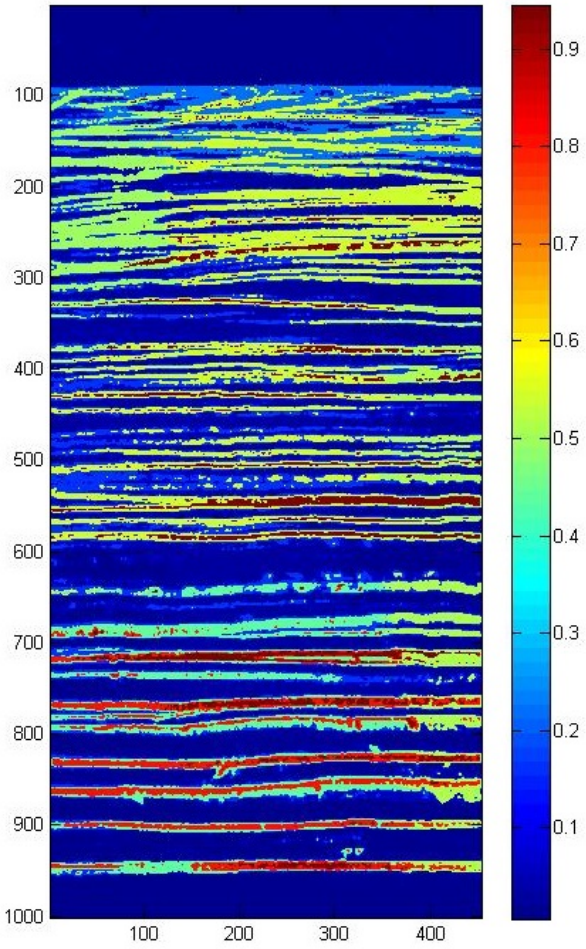


Figure 6: The resultant clusters from running EM

I did make sure to segment the clusters that resulted from running EM and see how that looked. I used

a classification boundary of 0.25, obtained through trial and error, on the result and any cluster above that was given a white color while clusters below that were given a black color value. Below is the result.

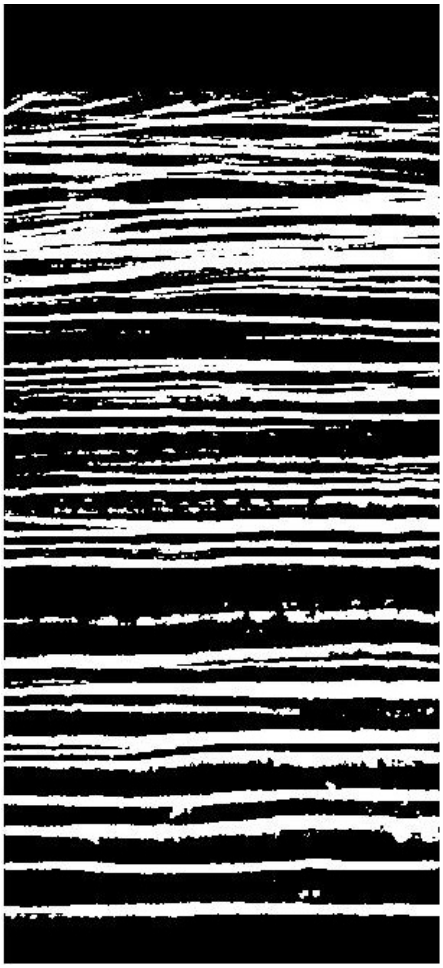


Figure 7: The resultant clusters from running EM segmented

3.5 Min-cut algorithm on Markov Random Fields

I tried using the min-cut algorithm on the MRF model to separate out the fibers. This works by assuming we have a hidden variable influencing the pixels and drawing a large undirected graph with that in mind. We then find the min-cut on that graph which separates the pixels into foreground and background. I used the min-cut code from Homework 3 to run this algorithm. It performed well at selecting certain fibers but it did not select all the fibers that I wanted it to. This could have happened because the min cut algorithm is optimized for selecting foreground and background connected regions, due to the graphical structure of the model which favors connected pixels. The fibers do not quite have the image properties of a foreground in that they are not all in one region of the image, so using min cut is not as effective. Perhaps if I had done the algorithm but with multiple cuts and a seed for each region, the results would have been better.

Unfortunately, the number of fiber regions is not always known. Since there is a large number of images in each data set, having to figure out the number of regions is impractical. Thus, it would have not been worth trying to continue making the MRF model work well with my data set. Here is the result of using it which shows that some of the fibers have been located but not all of them.

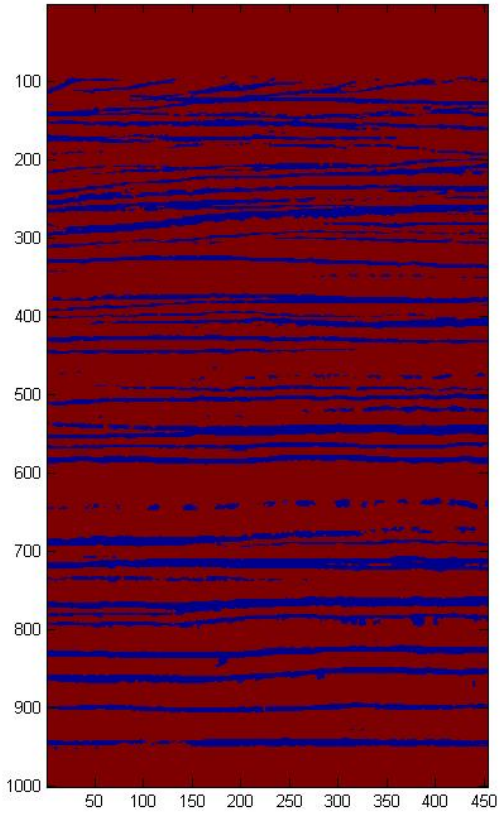


Figure 8: The resultant segmentation after using min-cut

3.6 Object Correlation

I finally tried object correlation. This works by taking a small template which represents the object to detect and performing the correlation function on the image and template. The correlation function gives us a number for the similarity in pixel values between two image patches. I tried a few different templates. I used one template that was just a little square where a fiber was located. I also tried a larger middle square that had a few fibers. Unfortunately, when I did the correlation and then displayed the results as an image, it just looked like a blurrier version of the old image in both cases and did not bring out the fibers well at all.

I did also think about using correlation for alignment. I would correlate two images with the same template and I could see where the highest correlation was in both images and match up those points. Unfortunately, the results were not reliable enough to be useful in that the two points did not match up well enough.

Correlation likely failed due to differences in the fibers in size, shape, and thickness, making them hard to detect with a single template. In addition to the variation in fibers themselves, many of the fiber objects will diverge into two separate fibers or there will be fibers that will converge into one fiber. I did not try to use a HOG approach but it is not likely to produce great results due to the variations mentioned above.

Because of all of this, object detection using correlation does not seem like a practical route to pursue to improve the clustering. Below is an illustration of the correlation results when using the small template. The results are seen for the first two images and it can be observed that the fibers are not brought out too well. Additionally, the points with the highest correlation do not correspond too well to be used for alignment.

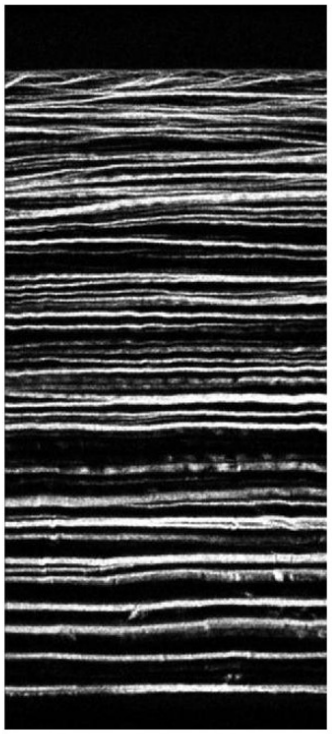


Image 1 Averaged

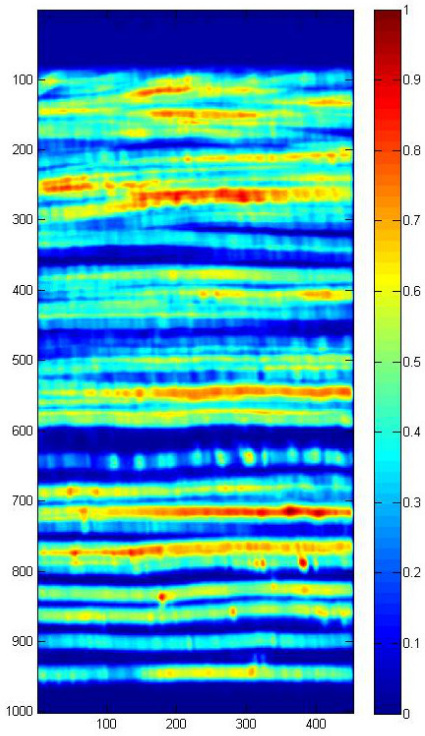


Image 1 Averaged and
Template Correlation Results

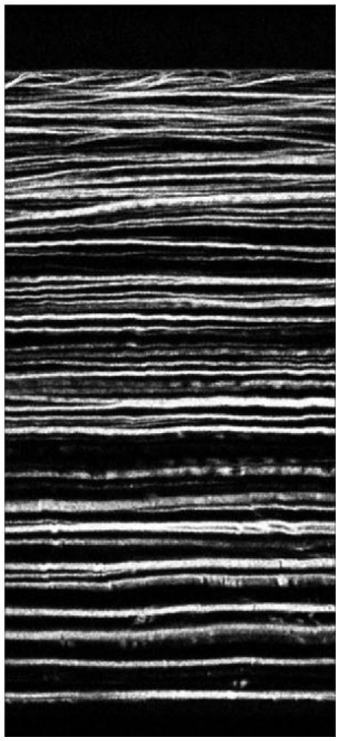


Image 2 Averaged

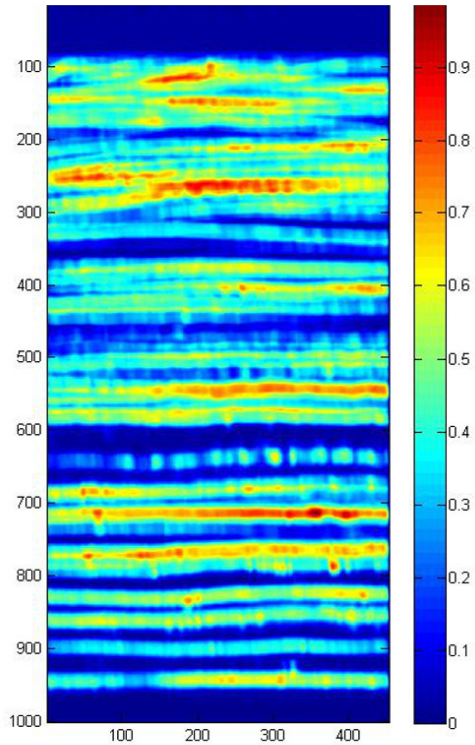
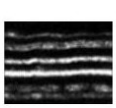


Image 2 Averaged and
Template Correlation Results

Figure 9: The correlation results on image1 and image2 using template 1

I was hoping that the results would be better with a bigger template, but as can be observed below, they were not.



Template

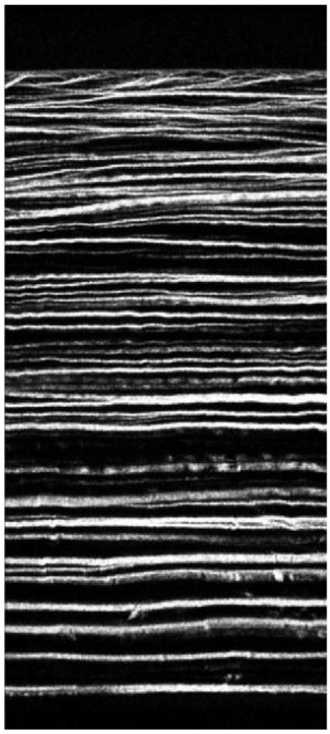


Image 1 Averaged

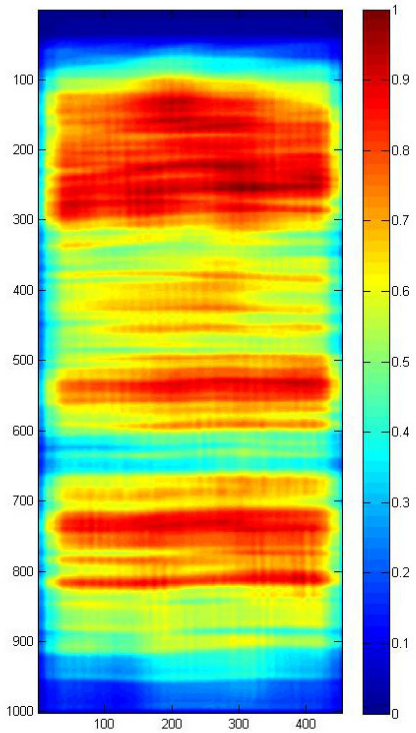


Image 1 Averaged and
Template Correlation Results

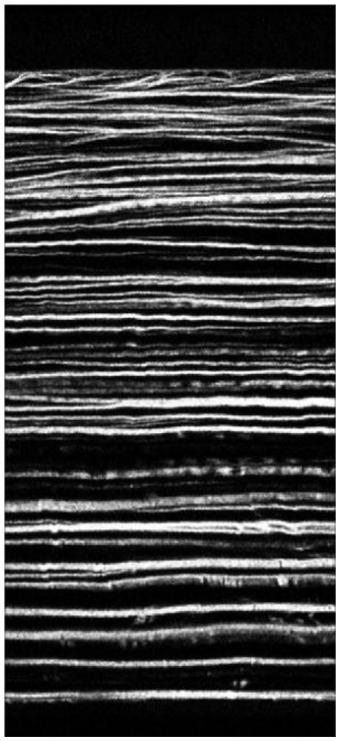


Image 2 Averaged

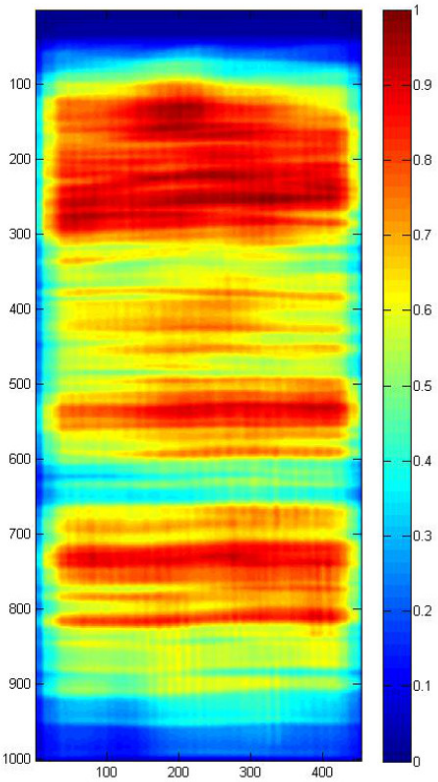


Image 2 Averaged and
Template Correlation Results

Figure 10: The correlation results on image1 and image2 using template 1

4 Smoothing of Segmented Images Step

After deciding that k-Means would segment the images, I wanted to smooth out the segmented images to make the fibers look better. I tried a few different filters and in the end, a median filter followed by a bilateral filter proved to be the best option. I liked the result of using a median filter as it takes out noisy pixels and does not use them in any computation. Bilateral filters are another averaging filter where the weights depend on both spatial proximity and brightness value. I used that filter because it is both noise reducing and edge preserving. In this way, it was really good at bringing out the fibers. When I implemented bilateral filtering, I used Matlab code from the Matlab file exchange. The file *getSegImage.m* in the associated zip file of code contains the web link where I downloaded it from.

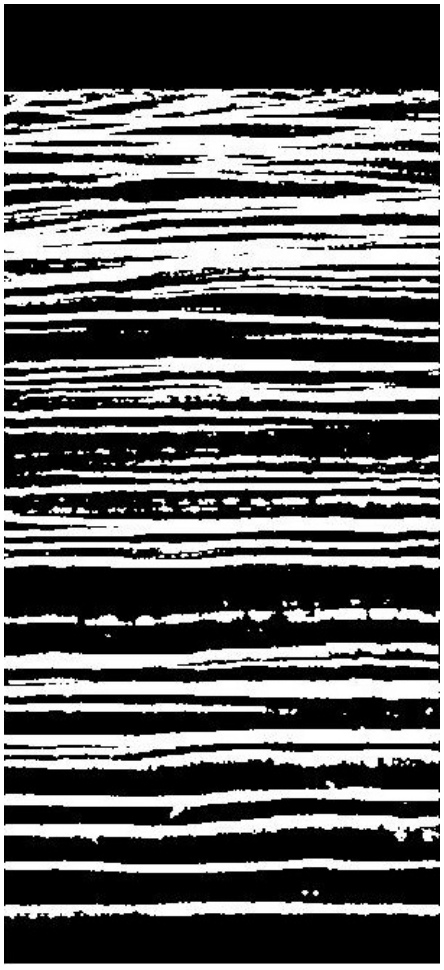


Figure 11: Result of applying median filter to segmented image

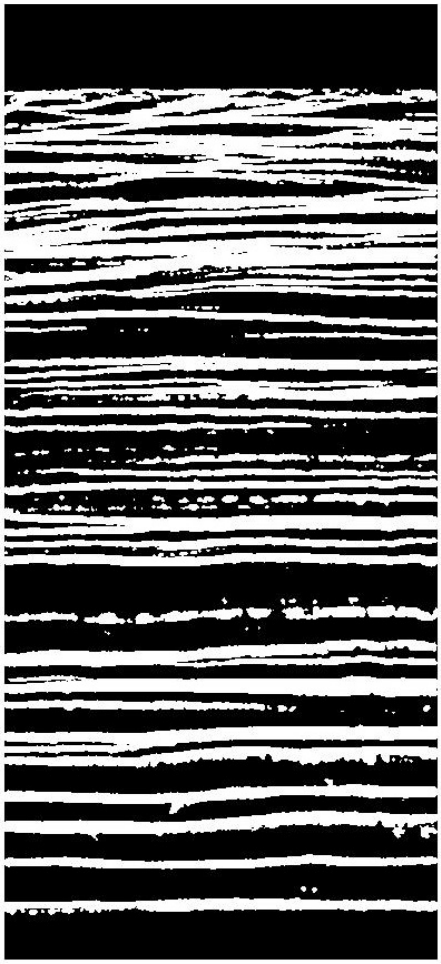


Figure 12: Result of applying bilateral filter to the median filtered image

5 Making the Data Set

This section describes what was done once the images were segmented and smoothed.

5.1 Alignment

I had images taken at different depths but I was not told that that they should be overlaid on top of each other directly so I tried to see if it would make sense if the images should be shifted slightly as I make the 3D data set.

I took an image and put it on top of another image and then shifted the top image slightly. I then took the overlap region for both images and considered a pixel a fiber if both of them had fibers there. I then

counted the number of fibers in that overlap region. Each of the overlap regions were different sizes so I made sure to normalize the number of matches by the total number of pixels. Even after doing this, the most overlap occurred when you did not do any shifting.

As mentioned above, I thought about using correlation and matching up the top points between two images as an alignment method but my results were not reliable enough for that to work well. I thought about whether I could use another method to find points and then match them up but due to the variations in the fibers, there does not seem to be a reliable way of doing that. Thus I finally decided not to do any special alignment and just overlay the images on top of each other.

5.2 Displaying the Data Set

After determining that doing a normal overlay was the best route, I decided to bring out the fibers even more in the final data set by adding images between each pair that consisted of pixels where both images had fibers. I then took all the images generated and made them all different slices in a 3D data set. I used the volume renderer VolView (http://www.kitware.com/opensource/volview.html) to view the 3D data set that I generated from the segmented images. Here is a screenshot of the volume rendering:

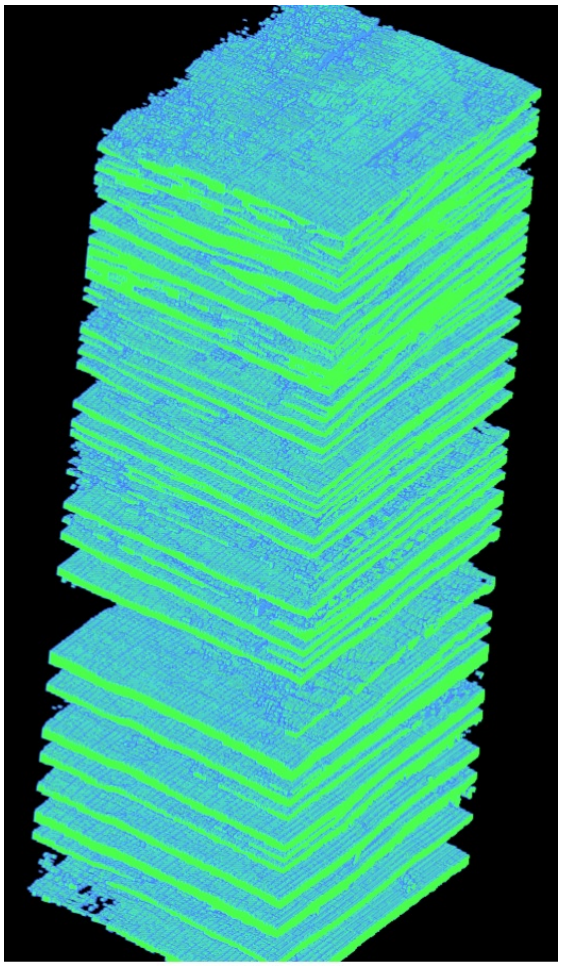


Figure 13: Volume Rendering screenshot

6 Results and Future Work

After all was said and done, I ended up with a nice looking rendering of the collagen fibers in a rabbit cornea. I showed it to the Ophthalmologists and they thought it was a good first step in using 3D visualization to analyze the fibers. I was especially pleased that the thin fibers at the top were brought out very well. Using techniques from this course allowed me to accomplish all of that. The volumetric rendering would have been hazy and uninformative had I not done any segmentation and smoothing on the images. While I am very happy with these initial results, there is still future work to be done.

6.1 Applying the techniques to other data sets

The pictures above show a data set of collagen fibers for the cornea of a rabbit. There are other data sets that we were given. I ran the above algorithms on them but the segmentation was not as successful. The above

tools turned out to be the ideal choice and their parameters were tuned for the rabbit data set. Since the other data sets have images that look different though, a different set of tools and parameters with those tools are required.

6.2 Work on improving segmentation

Qualitatively, my results seemed good and the ophthalmologists seemed to think they looked nice, but there is definitely room to improve the segmentation. When evaluating a clustering, I ended up relying on my subjective comparison between the clustered image and the original image to decide how well the fibers were brought out. It would be great if I could get a few preliminarily segmented images to compare my results with so that I can have a quantitative evaluation of how well the clustering performed. I could then compare the error rates that the various methods produce and use whichever method has the least error overall.

6.3 Work on 3D reconstruction

The Ophthalmology department really liked seeing the fibers in the data set and their concern lies with analyzing these fibers. They want to know how many fibers there are and how they branch off each other as well as know their shape and size. In order to enable this type of analysis, the next step then will be to take the 3D data set and generate meshes for the isosurface that corresponds to a voxel value of 1. The goal will be a mesh for each of the fibers that we can then move around, analyze, and run simulations on.