Optical Coherence Microscopy Image Processing and 3D Visualization

Matthew Caulfield

ECE-498: Electrical Engineering Capstone

Advisor: Professor Takashi Buma

November 20, 2018

Table of Contents

Table of Figures

1. Introduction

Traditional methods of microscopic imaging damages the specimen to prepare the sample for imaging. These methods of microscopic imaging also only create 2D images of a specific layer of the specimen. This project aims to address these problems by creating an imaging device that captures digital images at the microscopic level without harming the organism that is being imaged. While there are other forms of imaging that does not harm the specimen like MRI, CT, ultrasound, and x-ray, they are not designed for microscopic imaging. Optical coherence microscopes can capture images of a specimen at different depths without the need to slice or smear a specimen. Currently, Professor Takashi Buma is building an optical coherence microscope, and has yet to implement software to control the microscope and capture and process images from the microscope. The goal of this project is to create software that will capture images of an object at various depths and create a 3D visualization of the object or parts of the object.

This report will cover the following aspects of the project. Background on optical coherence microscopy, optical coherence tomography, and other 3D visualization methods. As well as ethical, usability, and maintainability considerations. It will then cover the design requirements of the project including image resolution, graphics processor unit (GPU) choice, render time, and quality of 3D visualization. The fourth section will cover the design approach of the microscope. The fifth section will detail the current design of the project. The sixth section will focus on current testing results of the project and compare them to expected results and design requirements. The seventh section will be a tentative winter implementation schedule. The eight section will be references. The final section will be an appendix containing code.

2. Background

2.1 Optical Coherence Tomography and Microscopy

Optical coherence tomography (OCT) was invented to conduct non-intrusive biological tissue imaging. OCT uses either low coherence light or laser pulses to create cross-sectional images of the internal structures of biological samples. By using light interference between the object being imaged and a reference signal the OCT is able image the object at various depths [2]. To capture en-face, XY, images the OCT scans across the object in the XY plane. An optical coherence microscope (OCM) is similar to an OCT but is able to capture en-face images of an object without scanning along the XY plane by using a CCD camera. Figure 1 shows a block diagram of the OCM. A beam splitter is used to split the light source to create a reference light and to light the sample.

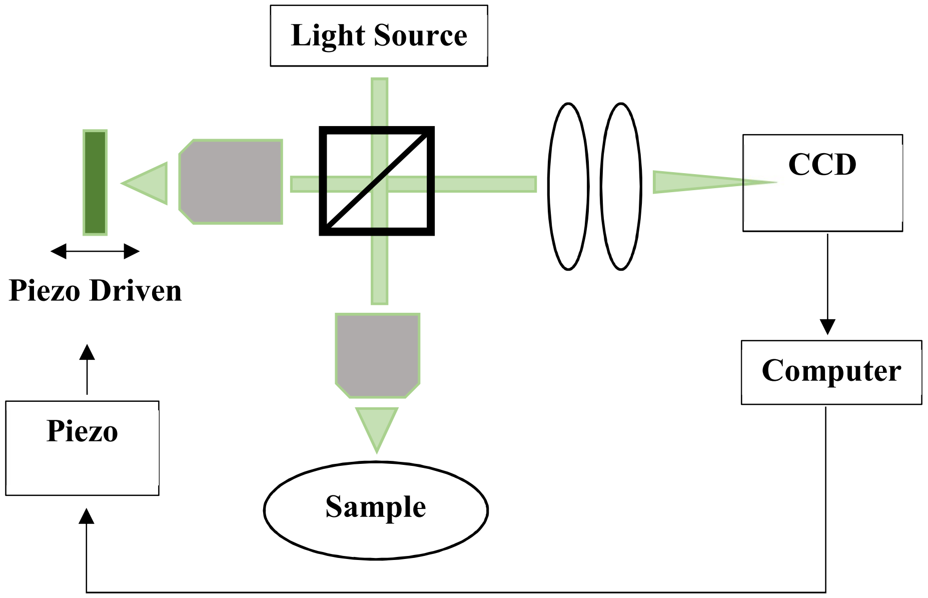

Figure 1: Diagram of an Optical Coherence Microscope

To use light interference to create an en-face image at a desired depth the OCM captures multiple images in succession of each other. However, the reference light comes from a mirror that is being oscillated by a piezo, so each image contains slightly different information. By combining these images, we get the en-face image at the desired depth. To create an image stack of an entire object the OCM takes images at different depths of the object [3]. Since the OCM is able to image the object at different depths, 3D visualization of the object is possible.

2.2 3D Visualization Methods

The National Institute of Health created an open source software, ImageJ, that is used for image processing and more specifically biomedical image processing. Because it is open source it allows users to build their own plug ins. One such plug in is the 3D viewer. When a stack of images is uploaded to ImageJ, the plug in can turn the stack of images into three different 3D visualizations, the surface of the object, the volume of the object, and an orthoslice view of the object. For this project, the surface and volume visualization of the objects are the most relevant. ImageJ allows its users to perform manipulations to its surface and volume in the viewer window. For the surface, in ImageJ the user can change the thresholding value, use different type of interpolations, smooth the surface, and change the color and transparency of the surface. For the volume visualization of the object, ImageJ creates the volume by assigning transparency values to each voxel based on the brightness of the pixel in the corresponding image stack. ImageJ allows volume editing, and changes to voxel color and transparency in its 3D viewer [5]. For this project, Matlab will be used to implement similar features to ImageJ. Matlab is being used instead of an ImageJ macro, because the OCM will be run using Matlab. Instead of requiring the OCM user to use multiple programs and export data from Matlab in a format usable by ImageJ, it will be easier for the user to enter some information into Matlab, run the program and have not only the raw data from the OCM but a 3D visualization of the data as well. With

Matlab the user will also be able to use live data from the OCM to create surfaces and volumes as they image the object.

2.3 Maintainability and Usability

The goal of this project is to have an OCM that can be used by people not involved in building the OCM or writing the program used to run the OCM. Because it needs to be useable by people with different backgrounds, the operation of the microscope needs to be as simple as possible. Ideal use for the microscope would be to place a sample on the stage, hit run and after acquiring the images use scale or mouse to choose a desired region to make into surface visualization. The controls would also include a slider for choosing different pixel thresholds for the region. The code and microscope must also be easily maintained by those who have some background on the OCM but not necessarily know how every part of the microscope works. So, the code will be well commented and easily understandable, so bugs may be fixed as detected and new features added as necessary.

2.4 Ethical Considerations

There are ethical considerations when creating an imaging device that could be used on biological tissue. For instance, if the device is used to diagnose diseases it's accuracy may cause misdiagnosis, false positives, or false negatives. This could harm a doctor's ability to accurately treat and care for a patient and may cause life threatening results. A less severe consideration is when using any medical information there are ethical practices that need to be followed like hipaa to make sure the privacy of individuals is maintained. Even when testing the microscope if it were to be tested on someone in the lab with no known disease, the microscope my find something irregular. Because those in creating and testing the microscope are not medical doctors, they would no be able to diagnose the results from the images and would need to be careful with how they talk about and use the information.

3. Design Requirements

For the OCM to be considered successful it needs to meet certain design requirements and constraints. These requirements can be broken into three categories. Because Professor Takashi Buma is building the physical microscope, this project will focus on image and usability requirements. The three categories for this project are, imaging requirements, 3D visualization requirements, and usability. Imaging requirements focus on the individual en-face images that the OCM captures along with the frame rate of the camera. 3D visualization requirements focus on the processing speed and visualization quality. While usability focuses on the user interface of the system.

3.1 Imaging Requirements

Because this project is being completed for Professor Takashi Buma, he has specified certain requirements for the microscopes imaging capabilities. The requirements are as follows. The optical coherence microscope needs to be able to capture 30 frames per second (fps) of output images of the object in its stage. Because the final en-face images are compiled from multiple images to reach a frame rate of 30 fps the camera of OCM needs to capture at around 120 fps. The pixel resolution is also tied to the frame rate of the microscope, so a higher frame rate is also necessary [3]. Another reason for 30 fps is desired so if a living organism is placed under the OCM the microscope can capture organ movements.

The current optical coherence tomography imaging device, that Professor Buma uses, has a lateral resolution of approximately 6 microns, so he would like a lateral resolution of around 5 microns for the OCM. The lateral resolution will be tested using the full width half maximum. The depth resolution of the microscope needs to be 3 microns and will be able to differentiate between objects in the specimen that are at depths further than 3 microns away from each other. The image quality of each image will be judged spatial resolution and contrast-to-noise ratio.

The microscope needs to be able to image a specimen that is the size of at least a 2x2x2 mm cube. The microscope needs to be able to image and model various organisms like vegetables, insects, and fish.

3.2 3D Visualization Requirements

For 3D visualization the design requirements come from using ImageJ as a standard for 3D visualization. This software for this project needs to be able to be ran in Matlab as further detailed in the user interface requirements, so the microscope cannot use ImageJ for its visualization. However, since ImageJ is a considered a standard for medical image processing, this project aims to recreate some of the 3D visualization abilities of ImageJ in Matlab. ImageJ has two 3D visualization techniques that would be acceptable for this project, surface and volume visualization. While implementing both types of visualization in Matlab would be ideal, it will only be necessary to implement one of the visualization techniques. To measure the result of the visualization we will compare the results of the Matlab technique to how ImageJ visualizes the same data set. Comparisons will be done with either the Hausdorff distance or the second mesh structural distortion measure [6]. The Matlab visualization needs to be within five percent of the ImageJ visualization according to the proposed comparisons to be considered adequate.

Along with the types of 3D visualization being consistent with those from ImageJ, this project would also like to maintain some of the functionality of ImageJ as well. For surface visualization, this means allowing the user to choose thresholding levels, how much to smoothing to apply to the surface, and change the transparency of the surfaces. For volume visualization of the object the Matlab code needs to allow the users to edit the volume itself and colors of the volume once it is made.

When computing 3D visualizations, time must also be taken into consideration. The overall computation time for the 3D visualization needs to be below 7 minutes. This includes time for transforming the data and rendering the data into 3D. While computation time is heavily based on the object being visualized, 7 minutes is a worst-case time for an object with a lot of surfaces and noise. The ideal time for a typical object being visualized will be between two and three minutes.

3.3 User Interface Requirements

Because the OCM will be used by those who did not build it or program it, the interface must be simple enough for them to use. For example, the user should not have to change lines of code to get their desired results. There must be a way for the user to adjust settings using either an input box or sliding scales on the computer. For ease, the whole program needs to be able to be run in a Matlab. Ideally the system would work by first adjusting some settings in Matlab and hitting run to collect the data from the OCM. Then after viewing the data adjusting a few settings for the 3D visualization, run a Matlab script and the 3D visualization of the desired object would be made. The only time the user should have to change lines of code is if they want to implement new methods for the software. The software should also be well commented and come with a guide so if necessary, the user can change the code or implement new features.

4. Design Alternatives

Because the physical microscope is being built by Professor Takashi Buma, this project focuses on controlling the microscope with a computer and processing the images from the microscope. The software for the OCM is primarily programmed in Matlab. The first thing that the code needs to do is move the microscope stage so that the microscope can image an object at different depths. In order to match the required depth resolution of 3 microns, the software will need to adjust the stage with a precision of at least 3 microns. There are two possible ways to do this. The first is to use the Thorlab ActiveX software that the microscope stage supports. While the software is not written in Matlab it has a graphical user interface in Matlab that provides control of the stage [7]. The other option is to use a motor attached to the stage control and use Matlab to control the motor. While the first option requires using commands that are not directly made for Matlab, which may cause some problems, it will give greater and more precise control of the stage. It will also not require physical construction which is another pro. The second step is to capture images using the charge coupled device camera. As one of the design requirements is for the software to be primarily in Matlab and Matlab can be used to control and collect the data from the camera, Matlab will be used for interfacing with the camera. The next step is to combine multiple images in succession of each other to create an image of the specimen at the desired depth. Matlab will be used to create the en-face image. After capturing an image at an individual depth, the OCM needs to be able to capture images at all the desired depths and create an image stack with the desired depth. Once an image stack is created the software will need to turn the stack into a 3D visualization.

The software will create a surface visualization of the object instead of a volume visualization. The surface visualization is preferable to a volume visualization because it conveys similar information while being easier to implement. There are four steps to creating the volume visualization: scaling the data, applying a low pass filter to the data, binarizing each image, creating the visualization. The data is first logarithmically scaled using log based 10 and then linearly scaled between 0 and 255. The data needs to be logarithmically scaled because the OCM will return values from zero into the millions. We then apply a linear scale to the data to bring it into a standard pixel scale. After scaling a low pass filter is applied to the data to smooth the data and lower the processing time of the final surface visualization. Three types of 3D filters were considered, a center weighted filter, a gaussian filter, and a cross. Based on preliminary testing the center weighted filter had the fastest processing time for the final surface visualization with similar 3D image quality. After filtering the data, it needs to be binarized. There were three possible methods for thresholding the data. Thresholding at a pixel value of 123 approximately half of the range of pixel values. Using the Matlab's imbinerize function which uses the Otsu method for thresholding. Otsu's method takes into consideration surrounding pixels when binarizing the image. The final method is to use an upper and lower threshold set by the user. The half point threshold and Matlab's imbinerize worked similarly and both require no user input. However, both had the same con, it only took into consideration a lower threshold. When using both an upper and lower threshold set by the user, preliminary results showed that the computation time of the surface visualization took less time depending on the thresholds chosen. The one draw back is that the user is required to enter thresholds. If the user is unfamiliar with the object, they may not choose the correct thresholds. However, it also allows greater control over what is in the final visualization.

5. Design

The design for this project is focused on software design and implementation because Professor Takashi Buma is building the physical OCM. The software design can be broken into two sections microscope control and 3D visualization. Microscope control encompasses stage control and image capturing. 3D visualization design can be broken into, 3D visualization type, implementation, and graphics processing unit choice.

5.1 Microscope Control Design

The microscope control design can be broken into four steps detailed in the flow diagram in figure 2.

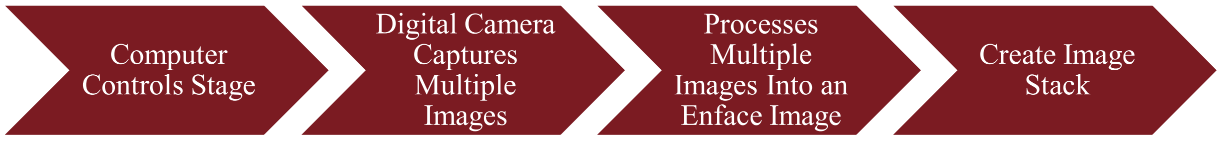
Computer Controls Stage
Digital Camera Captures Multiple Images
Processes Multiple Images Into an Enface Image
Create Image Stack

Figure 2 Process steps for en-face image stack

The first step for controlling the microscope is to control the stage of the microscope with the computer. The stage control will be done in Matlab using Thorlabs APT Active X controls. With the Active X controls Matlab the microscopes user will be able to control the stage with either a graphic user interface provided by active x or with commands that will be programmed for the OCM [7]. The programmed commands will allow the user to input a range of depths they would like imaged and Matlab will move the stage to the desired image depths. Control of the CCD camera will be done in Matlab as the camera being used has a Matlab interface. Image processing to combine multiple images into a single en-face image using light interference will also be done in Matlab as Matlab has an image processing suite with the tools needed for this.

The image stack that will be created of the object will be a three-dimension matrix of pixel values instead of an array of images. This is to make the image processing of the stack easier and in Matlab each individual image will still be viewable. Storing it as a matrix of pixels also allows image slices of the x-z and y-z plane to be viewed instead of just en-face images. The image processing will be easier because instead of having to iterate through each layer of the matrix, functions can be applied to the whole matrix.

5.2 3D Visualization Design

Surface visualization was chosen for the type of 3D visualization. It was chosen because visually it is similar to volume the visualization, but it is easier to implement in Matlab and should take less processing time than volume visualization.

The implementation of surface visualization can be broken into five steps shown in the figure below. Each step is performed on the image stack created by the OCM in succession.

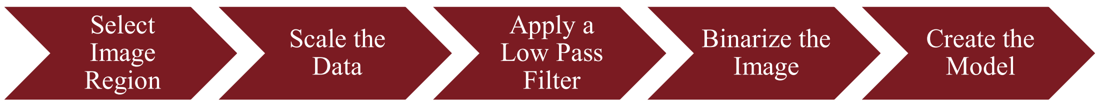

Figure 3 Process steps for surface visualization

The first step is for the user to choose a region of the image stack to turn into the model. This is necessary if the user wants to only visualize part of the object and not the whole object. To do this the user will be given an en-face image from the stack at a desired depth and will then input the desired region into an input box. To input the region the user will give the starting and ending x, y, and z coordinates of the region. The data is scaled two times. The first time it is scaled logarithmically because the range of the data is between zero and over a million. Twenty times the log based ten was used to scale the data because it brought the data down to values between approximately 0 and 100. After the data is scaled logarithmically it is then scaled linearly between 0 and 255 because these are standard pixel values. To filter data the data a three-dimension center weighted low pass filter was chosen. The figure below shows the low pass filter.

$$\begin{bmatrix} \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \\ \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \\ \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \end{bmatrix} \begin{bmatrix} \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \\ \frac{1}{52} & \frac{26}{52} & \frac{1}{52} \\ \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \end{bmatrix} \begin{bmatrix} \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \\ \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \\ \frac{1}{52} & \frac{1}{52} & \frac{1}{52} \end{bmatrix}$$

Figure 4 3D center weighted low pass filter used for smoothing

The low pass filter above is broken into three matrices each matric represents a consecutive layer of the filter in the x-y plane. This filter was chosen after some preliminary testing because the final surface visualization of the center weighted filter was of similar quality to the other tested filters, but the computation time of the visualization was much better for the center weighted filter. For the same thresholding, the visualization computation time after the center weight filter was 13 minutes while visualization computation time was 15 and 16 minutes for the other filters respectively. The next step to visualize the surface of the object is to binarize the image stack, so the image stack consists of pixels that are either a part of the object or part of the background. To threshold the images three techniques were considered. The final choice of thresholding was to allow the user to set a lower and upper threshold so only pixels between the two values were considered part of the object. It was chosen because it had similar computation times as the other thresholding choices but allowed the user a greater flexibility in what to consider part of the imaged object. For instance, an air bubble may cause a bright patch in the image which if only using lower thresholding would be considered part of the object, but when upper thresholding is removed. To create the surface of the object, the Matlab isosurface function is used. It is used because by using Matlabs parallel computing tool box isosurface can be optimized to run on multiple cores of the computers GPU to decrease the computation time.

Part of the visualization design is choosing a new GPU for the computer running the software for the OCM. The current GPU is not fast enough for the desired computation times so a new GPU is necessary. The new GPU needed to be a Nvidia GPU so that it is runs CUDA, Nvidia’s parallel computing language, and is supported by Matlabs parallel computing toolbox. The GPU was chosen off a list provided by Matlab and Nvidia with the budget in mind. Ultimately the Nvidia EVGA GeForce GTX 1060 GPU was chosen [8]. A student research grant will provide 189 dollars while the electrical engineering department will provide 89 dollars.

6. Preliminary Testing Results

Because the OCM has not been built yet and will be built over winter break the preliminary tests completed during the fall term were based around surface visualization of OCT data. Because the output the optical coherence tomography is similar to the data from an optical coherence microscope the OCT data makes a suitable replacement for designing a prototype system to create surface visualizations of the data. The OCT provided data from a chicken embryo and a nylon screw.

To find the best filter to use as a low pass filter and smooth the data with, the time it took for the computer to create the surface visualization was used. Figure 5 below shows the time it took for the computer to create a surface visualization for each filter. The Gaussian filter had a standard deviation of 0.5 and the threshold value of the binarization used for each filter was 0.5 of the maximum pixel value. The OCT data from the chicken embryo was used for these tests.

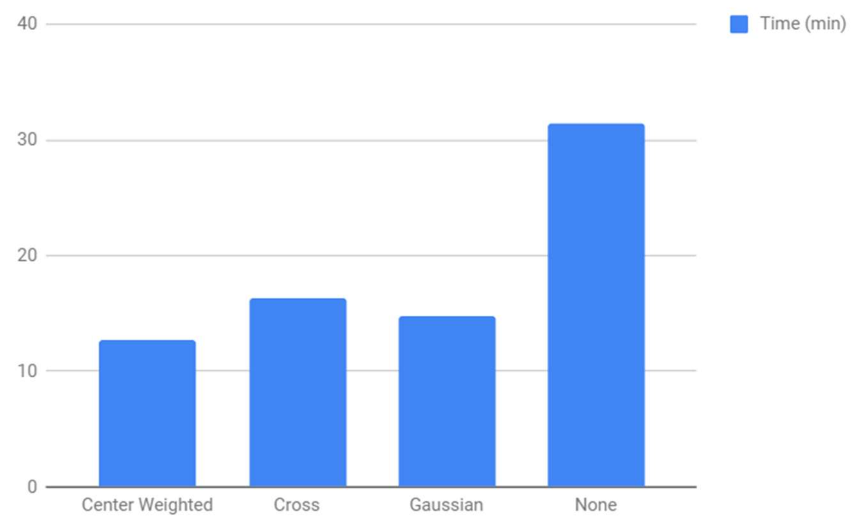

Figure 5 Graph of filters and their computation time in minutes

Based on the results from the experiment the center weighted filter was the best filter to use due to its lower computation time. After finding the best filter to use, different thresholds were tested to see how it affected the output visualization and computation time. Figure 6 shows the how the surface visualization computation time changes as the binarization threshold changes. A center weighted filter was used as the low pass filter for each test and only images in the image stack that were between 1 and 55 in the z direction were used. The OCT data from the chicken embryo was used for these tests.

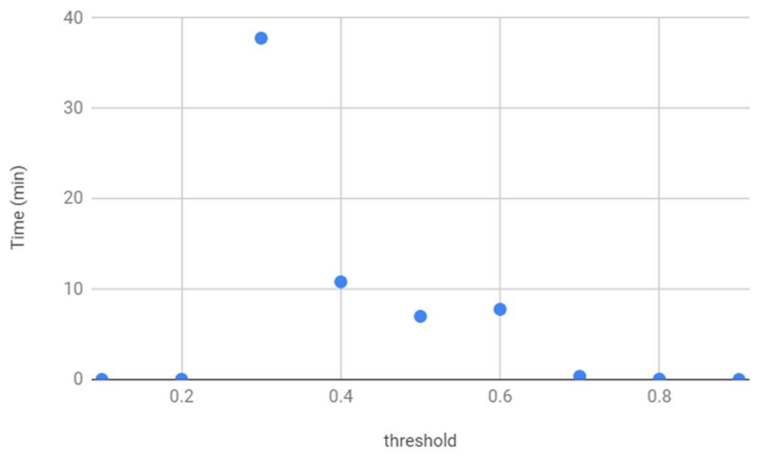

Figure 6 graph of threshold versus computation time for a center weighted filter

From the tests we found that thresholding with 0.4, 0.5, and 0.6 had the best final surface visualization clarity based on how they looked. Between the three that contained the best surface visualization a threshold of 0.5 took the least amount of time to create.

After deciding on threshold levels and filter type, the surface visualization of the OCT chicken embryo data was created. Figure 7 shows an image of the chicken embryo after being logarithmically then linearly scaled and figure 8 shows the embryo after being filtered with the center weighted low pass filter. Figure 9 shows an image of the embryo after a threshold of 0.5 was applied and the image matrix was binarize. Figure 10 shows the surface visualization of the embryo.

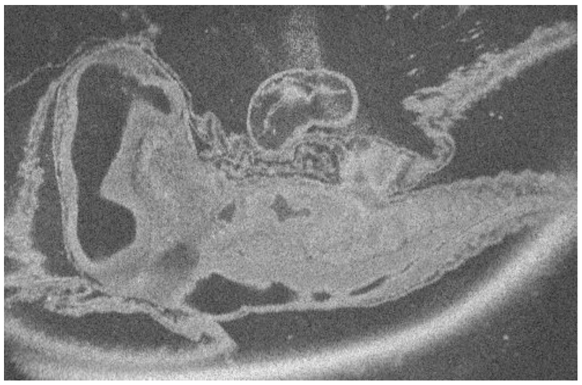
Figure 7: Image of chicken embryo after scaling

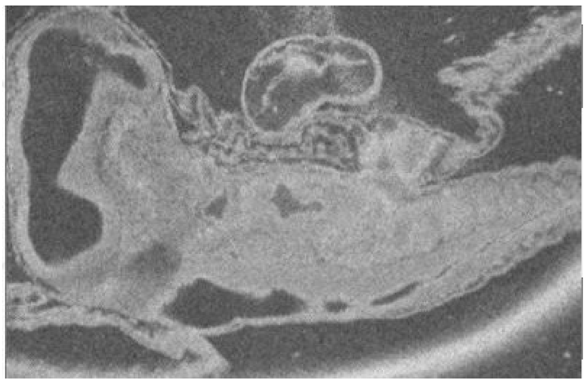
Figure 8: Image of chicken embryo after filtering

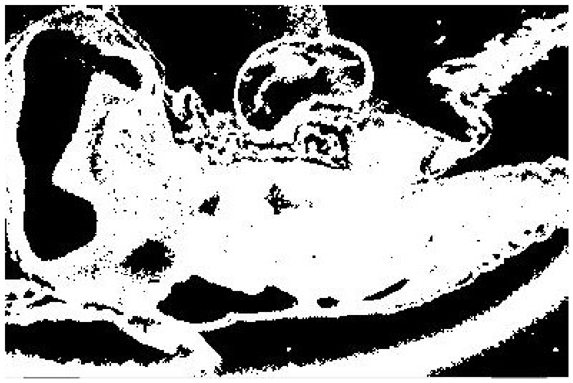
Figure 9: Binarized image of chicken embryo

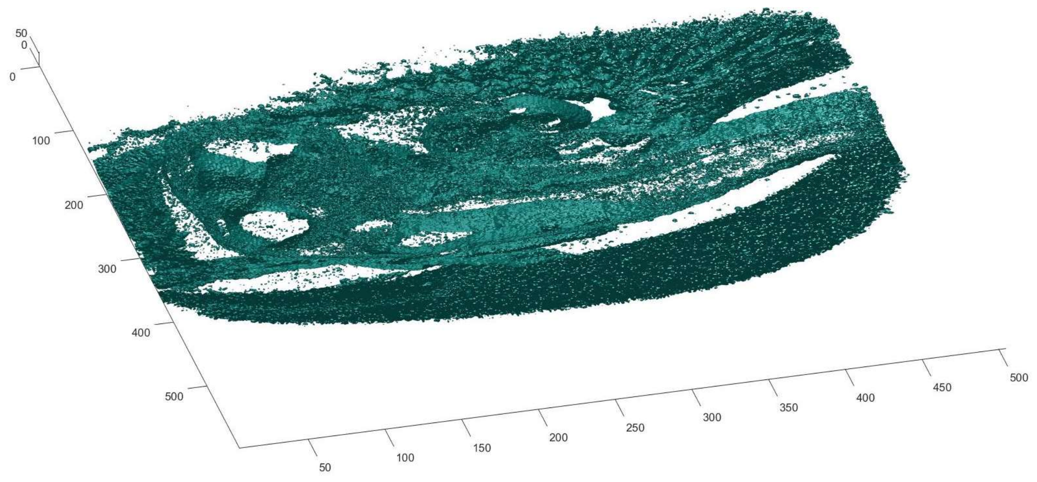

Figure 10: 3D visualization of chicken embryo

The OCT chicken was a hard data set to use as a test for the surface visualization because it was not a known object. So for the next surface visualization test a nylon screw was imaged by the OCT and the surface visualization was made in both ImageJ and with the prototype software.

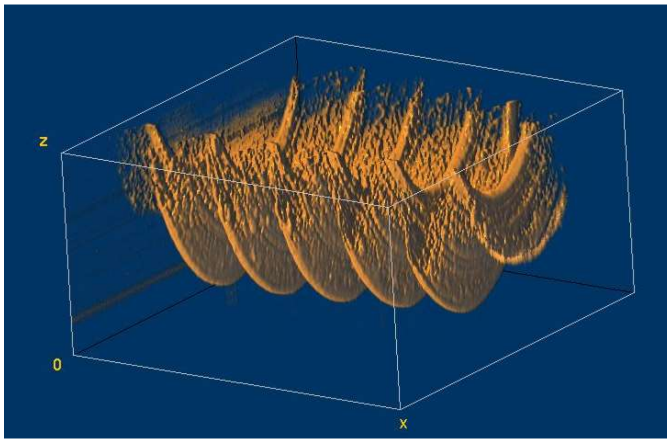

Figure 11 Nylon screw surface visualization using ImageJ

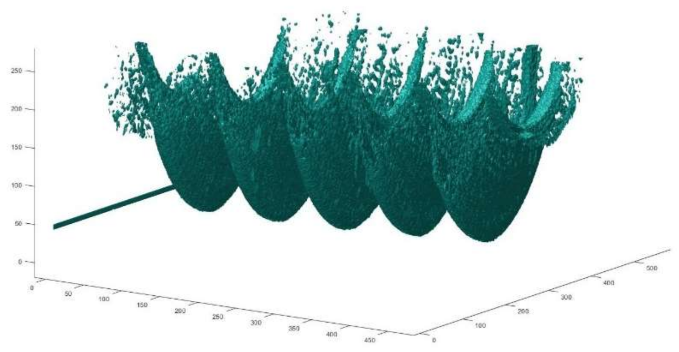

Figure 12 Nylon screw surface visualization using Matlab

Figure 11 shows the surface visualization of the nylon screw with ImageJ and figure 12 shows the surface visualization of the same data set with the Matlab prototype. There was not enough time to apply the Hausdorff distance to the two surfaces but based on how they look they are similar.

7. Implementation Schedule

By the first week in winter term hopefully the physical OCM will be built. If it has been built the goal for that week is to implement the software required to control the stage of the microscope and capture images using the microscope. For the second week the goal is to fix any bugs found in week one and be able to combine multiple images of a certain depth into an individual en-face image of a desired depth using interference. By week three the microscope should be able to create an image cube of an object, so during week three the OCM data will be tested to the standards outlined in the design requirements. By week 4 the microscope should be able to image an object so in the following weeks the project will be focused on 3D visualization. During week 4 hopefully the graphics card will have arrived, and the 3D visualization code can be implemented in parallel. For week 5 continue working on 3D visualization and fixing any bugs in it along with beginning to develop the code required to compare the Matlab visualization to the ImageJ visualization. The goal for week 6, is to implement user friendly interfaces to the Matlab software. Weeks 7, and 8 will be testing the final image quality, 3D visualizations, and user interface, along with fixing any bugs that come up. Weeks 9 and 10 will focus on final presentation and the final write up of the capstone project.

8. References

[1] Huang, D., Swanson, E., Lin, C., Schuman, J., Stinson, W., Chang, W., . . . Fujimoto, J. (1991). Optical Coherence Tomography. Science, 254(5035), 1178-1181. Retrieved from http://www.jstor.org/stable/2879328
[2] Podoleanu, A. G. (2012). Optical coherence tomography. Journal of Microscopy, 247(3), 209-219. doi:10.1111/j.1365-2818.2012.03619.x
[3] Dubois, A., Vabre, L., Boccara, A., & Beaurepaire, E. (2002). High-resolution full-field optical coherence tomography with a Linnik microscope. Applied Optics, 41(4), 805. doi:10.1364/ao.41.000805
[4] Schneider, C. A., Rasband, W. S., & Eliceiri, K. W. (2012). NIH Image to ImageJ: 25 years of image analysis. Nature methods, 9(7), 671-5.
[5] ImageJ 3D Viewer User FAQs, http://132.187.25.13/ij3d/?page=User_FAQs&category=Documentation.
[6] Lavoué, G. (2011). A Multiscale Metric for 3D Mesh Visual Quality Assessment. Computer Graphics Forum, 30(5), 1427-1437. doi:10.1111/j.1467-8659.2011.02017.x
[7] Wang, Wei. (2011). Using Thorlabs APT ActiveX Control in MATLAB. Thorlabs.
[8] Nvidia, CUDA GPUs, https://developer.nvidia.com/cuda-gpus.
[9] Schmid, B., Schindelin, J., Cardona, A., Longair, M., & Heisenberg, M. (2010). A high-level 3D visualization API for Java and ImageJ. BMC Bioinformatics, 11(1), 274. doi:10.1186/1471-2105-11-274
[10] Al-Damegh, S. A. (2005). Emerging issues in medical imaging. Indian Journal of Medical Ethics, (4). doi:10.20529/ijme.2005.061
[11] Aguirre, A. D., Zhou, C., Lee, H., Ahsen, O. O., & Fujimoto, J. G. (2015). Optical Coherence Microscopy. Optical Coherence Tomography, 865-911. doi:10.1007/978-3-319-06419-2_29
[12] Boudier, Thomas. “3D Processing and Analysis with ImageJ,” Universit´e Pierre et Marie Curie.

9. Appendix

9.1 Prototype code for surface visualization

```
function filtered_model(filter)
        %loads the desired OCT/OCM data set
        load ('aug06_embryo5_cmodeCube1.mat');
        %crops the image cube to the desired region
        cmode_cube_cropped = cmode_cube(:,:,1:55);
        %gets teh dimensions of the data set
        imSize = size(cmode_cube_cropped);
        %creates a matrix to hold the binary images
        binary_images = zeros(imSize(1),imSize(2), imSize(3));
        %the folllwing line applies a log and linear scale to the cube
        cube_scaled = rescale(20*log10(cmode_cube_cropped), 0, 255);

        % applies the filter to the scaled cube
        cube_filtered = convn(cube_scaled, filter, 'same');

        % binarizes each layer of the cube
        for plane = 1:imSize(3)
                plane_scaled  = (cube_filtered(:,:,plane));
                images(:, :, plane) = plane_scaled;
                bin_im=imbinarize(plane_scaled/255, 0.5);
                binary_images(:, :, plane) = bin_im;
        end

        % creates the surface visualization and times how long it takes
        figure;
        tic
        isosurface(binary_images)
        toc
end
```

9.2 Prototype code for upper and lower thresholding

```
%takes the image cube, a lower threshold, and upper threshold and
%binarizes the data
function bin_im_cube = binarize_im_cube(im_cube, low_thresh, upper_thresh)

imSize = size(im_cube);
bin_cube = zeros(imSize(1),imSize(2), imSize(3));
for x = 1:imSize(1)
   for y = 1:imSize(2)
         for z = 1: imSize(3)
                        if low_thresh < im_cube(x, y, z) && im_cube(x,y,z) < upper_thresh
                              bin_cube(x, y, z) = 1;
                        else
                              bin_cube(x, y, z) = 0;
                        end
                  end
         end
end
bin_im_cube = bin_cube;
end
```