Using edge detection algorithm to determine bio-film thickness

Introduction:

Edge detection is a fundamental image processing technique that involves identifying edges, curves, and discontinuities in the brightness of a digital image. It is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

In this project, the objective is to apply an edge detection algorithm to determine the average thickness of biofilm from a 2D-OCT scan.

Materials and Methodolgy:

This project will use **opencv-python** and **numpy** libraries which contains all the requird tools for this project, and before applying the edge detection algorithim some preprocessing and denosing was required to get the desired output from the edge detection algorithim.

Preprocessing :

Initially, the input scans were 2D-OCT scans in TIFF format with dimensions of 9064 pixels in height and 9722 pixels in width. To isolate the membrane and biofilm part from the scans, the images were cropped to a height of 4800 pixels using array slicing. Given that the pixel to µm ratio was 3.3324 (a scaler value obtained from ImageJ software), the images were resized to standardize the pixel to µm ratio using the cv2.resize method. The images were then rotated 180 degrees using the cv2.rotate method so that the biofilm was facing upwards. After this preprocessing, an example resulting image see **figure 1.**

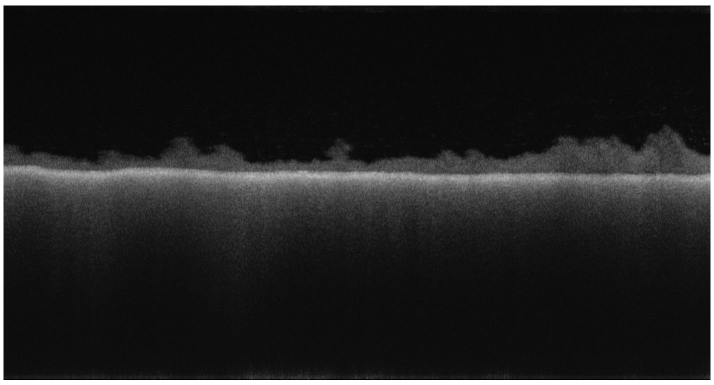

Figure 1

denoising :

Secondly, some filters had to be applied to the image to get rid of the noise in the images. For that, a non-local means denoising (**cv2.fastNlMeansDenoising**) method and Gaussian blurring (**cv2.GaussianBlur**) method were applied. Then, a range filter to fully distinguish the biofilm from the background. Here, I faced my first problem because at a range of (25-202), it was not suitable for some low-light images and at a range of (20-202), it was not suitable for some highly noised scans. So, I settled on a range of (22-202), which gave me the lowest error among these two options, after the denoising an example resulting image see **figure 2**.

Membrane detection:

Because the edge detection algorithm couldn't fully distinguish the membrane from the biofilm, I had to figure out another way to detect the membrane layer. For that, which is to iterate over the x-axis in blocks of 10 pixels. For each block, iterated over the y-axis in blocks of 10 pixels each to find the 10 blocks with the lowest values (the whiter pixels). Then, saved the coordinates for it in an array. To avoid errors, the outliers values were removed form that array and then passed it to the **np.polyfit** method that takes values and returns the m and b coefficients for the line corresponding to the values, assuming that the membrane layer should be either a straight or curved line. After that, all the pixels under that line were set to black. It also checks if the m value, which is the slope, is reasonable between (-1, 1). If not, the line will be a straight line on y = 75% percentile of the blocks array. an example from this phase is shown in **figure 3**.

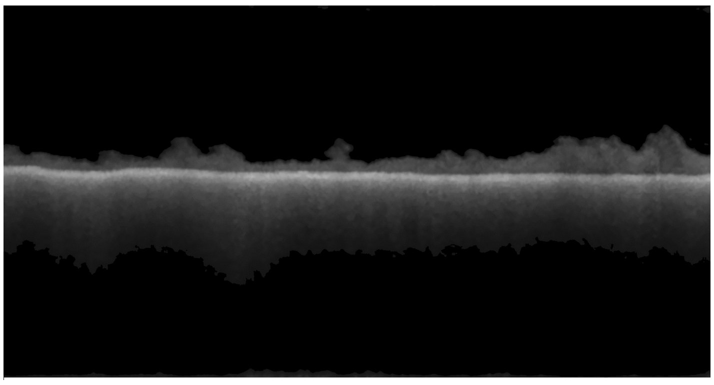
Figure 2

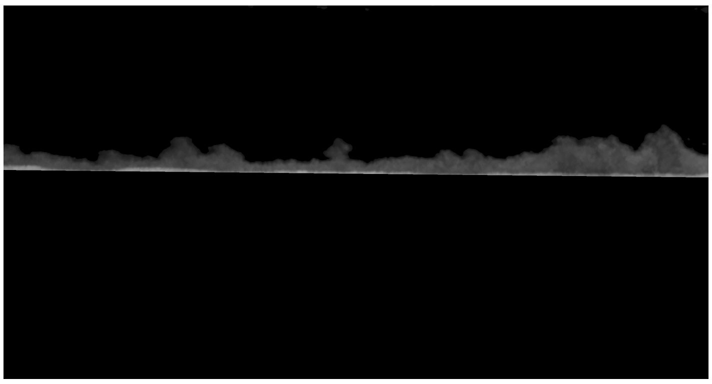
Figure 3

Edge detection :

Before applying the edge detection algorithm, all non-black pixels were set to white to produce cleaner edges. After trying multiple edge detection algorithms, the one that gave the best results was the Canny algorithm, using the cv2.Canny method. The parameters used for the function were threshold1 = 80, threshold2 = 50, and apertureSize = 3, which were chosen through trial and error. See **Figure 4** for an example.

Calculating average:

Finally for counting the average and to avoid errors in some higly infected membranes, a two steps function that calculates the average were applied, firstly an 1000 randomly pixles samples will be taken to calculate the average height of the film by finding the first non black pixel above the membrane layer, after that iterated over the x-axis and found the first three non-black pixels if found and will take the closes one to the averaged height of the bio-film we calculated eariler then removed the outliers and calculated the average thcikness.

Figure 5 is an example of the final result after drawing a line over the average thikness of the bio-film.

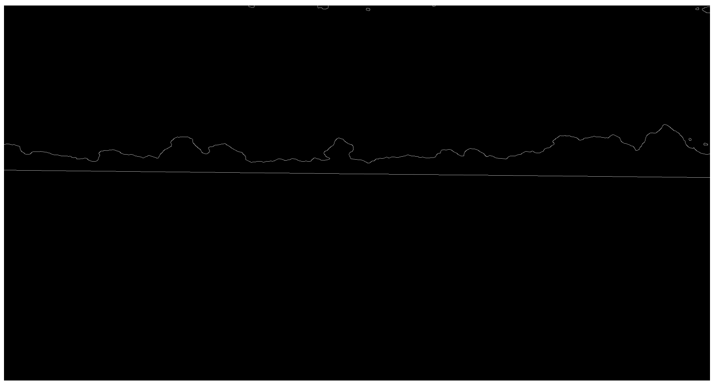
Figure 4

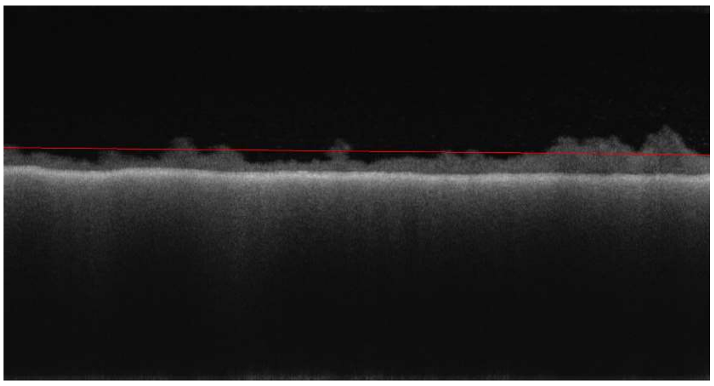
Figure 5

Results:

To test the error of the results I chose a randomly 10 scans and applied the algorithim on, then calculated it the average thikness of the bio-film manually using imagej software , the manual calculating was by appling almost 50 multi-points on the bio-film and 50 crossponded multi-points on the membrane, then substracted both and averaged the values here is the results for both

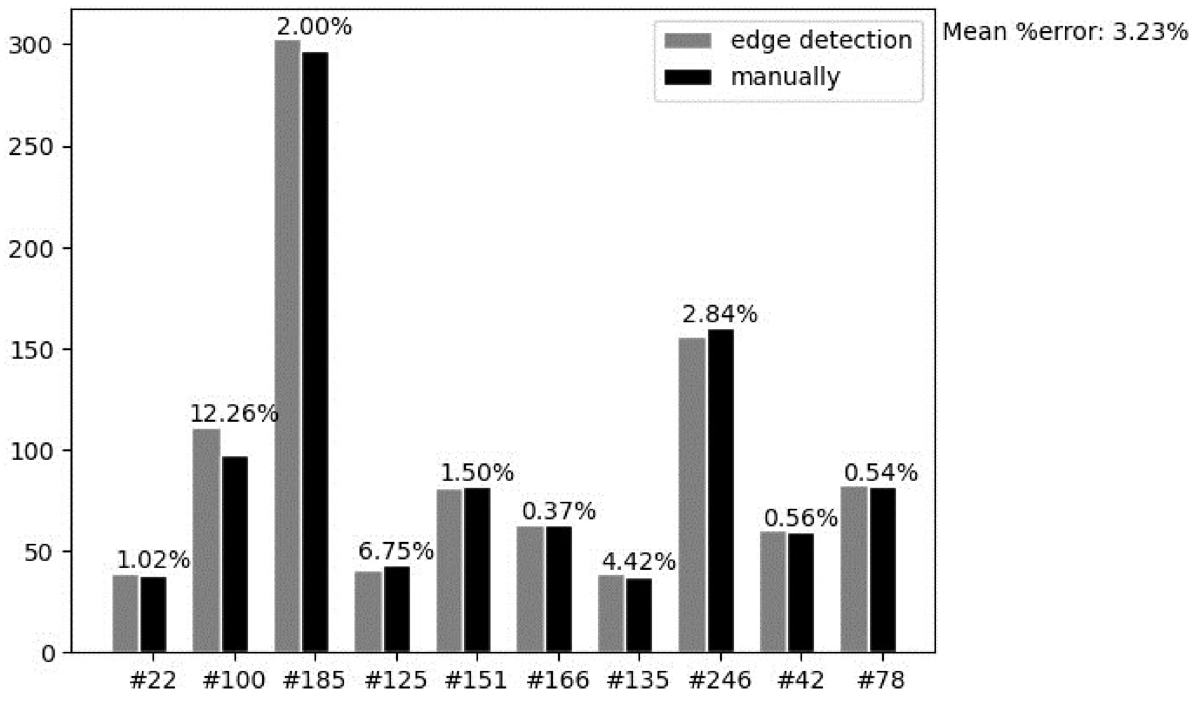

Figure 6

conclusion:

In conclusion, this project successfully used an edge detection algorithm to calculate the average thickness of biofilm from 2D-OCT scans after some preprocessing and denoising filters, it’s also important to note that this approach assumes that there are clear boundaries separating the biofilm from other elements in the scan, The average runtime for the code on a laptop with an Intel i5 4-core processor and 8GB of RAM is 10 seconds for a single image