COL780 Assignment 1 Report

Ananya Mathur : 2020CS50416

February 2024

1 Directory Structure

```
2020CS50416.zip
├── main.py
├── part1.py
├── part2.py
├── task2.py
├── task3.py
├── canny_edge_detector.py
├── report.pdf
└── data
    ├── img11.png
    ├── img12.png
    ├── img13.png
    ├── img14.png
    └── img15.png
```

2 Task 1 : Number of micro-sutures

First, I read the input images from their paths using Opencv imread function in grayscale.

2.1 Preprocessing

- **Dilation (Max Filter):** First, I dilate (max filter) the grayscale input image using a 1×13 kernel of 1s using dilate(image, kernel) function implemented by me. This expands the white areas in the image (not sutures) thereby allowing disintegration of joint sutures and also reduces any noise.
- **Thresholding:** Then, I perform thresholding using threshold_filter(image, threshold_value) function defined by me which makes pixels with intensity $\leq$ threshold_value equal to 0 and all others 255. During fine tuning, I chose threshold_value as 90, which gives best results for the given 10 images. Thresholding further removes noise and helps separate background from sutures.
- **Dilation (Max Filter):** After threholding, I again perform dilation with a smaller kernel (size 1×9). This helps in removing random noise in the background which remains even after thresholding.

2.2 Edge Detection

After preprocessing, I invoke the Canny Edge Detector to highlight the suture boundaries. After a lot of trial and error, the most suitable parameters (thin edges + distinguishing sutures) for the edge detector on the given 10 images turned out to be $\sigma = 6.5$, kernel size = 5, high threshold for non maximum suppression = 0.9 $\times$ maximum pixel intensity in image, low threshold for non maximum suppression = 0.9 $\times$ high threshold,

weak pixel for thresholding = 150, strong pixel = 255. High values of both low and high threshold resulted in very less noise and clean suture boundaries.

2.3 Filtering and Refinement

- **Dilation (Max Filter):** After edge detection, I perform dilation with a 1×120 size kernel. This is done to club any edges belonging to the same suture which might have separated in the horizontal direction.
- **Dilation (Max Filter):** Then, I perform another dilation with a 6×1 size kernel. This is done to combine connected components belonging to the same suture which separate vertically due to minor gaps after edge detection.

2.4 Suture Counting

Finally, after all sutures are identified as separate connected components in the best way possible, I use count_connected_components function to count the number of connected components in the resultant image and output the number of connected components as number of sutures.

3 Task 2 : Inter-Suture Spacing

The count_connected_components invokes dfs for each new connected component. I have implemented a modified dfs which also returns the size of the connected component it is traversing, sum of x coordinates of all points in it and sum of all y coordinates. In this way, the count_connected_components function is able to compute centroid of each new connected component which it stores in a dictionary called centroid (key : component number, value : centroid coordinates). Finally it returns centroid which is used to compute mean and variance of distances between consecutive suture centroids using inter_suture_distance function.

4 Task 3 : Angulation of the suture

The modified dfs also returns left most point coordinates of each connected component which is stored in a dictionary (key : component number, value : left most point coordinates) by count_connected_components function and returned. This dictionary along with centroid dictionary is used to compute mean and variance of angle the line segment joining centroid and left most point forms with x axis using find_angle_mean_var function.

5 Task 4 : Comparison of two micro-suturing outcomes

Simply compute mean and variance of inter suture distance and angulation of sutures for every given pair of images and compare them to output the better image wrt each feature.

6 Data

6.1 Given images

For the 10 images provided, following are the results obtained by my code for task 1:

Image number	Actual number of sutures	Detected number of sutures
1	9	11
2	5	5
3	9	10
4	10	10
5	10	10
6	13	13
7	8	8
8	17	16
9	16	16
10	14	15

For each of the 10 images in the given datset, following are the results after each step of processing. The 7 images for each input image are sequentially in the order:

- Input Image
- Max Filter 1 (Pre processing)
- Thresholding (Pre processing)
- Max Filter 2 (Pre processing)
- Canny Edge Detection
- Max Filter 1 (Post processing)
- Max Filter 2 (Final connected components)

Image 1

Image 2

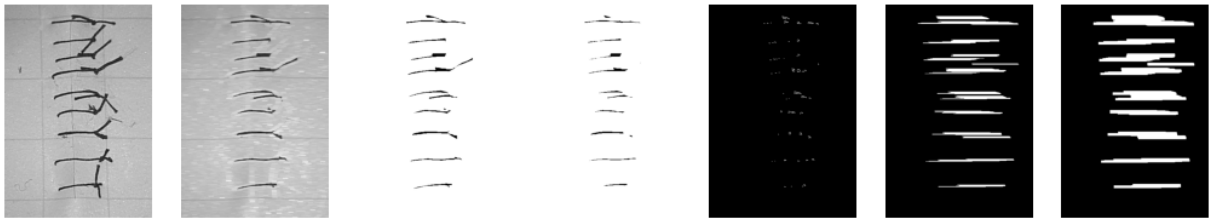

Image 3

Image 4

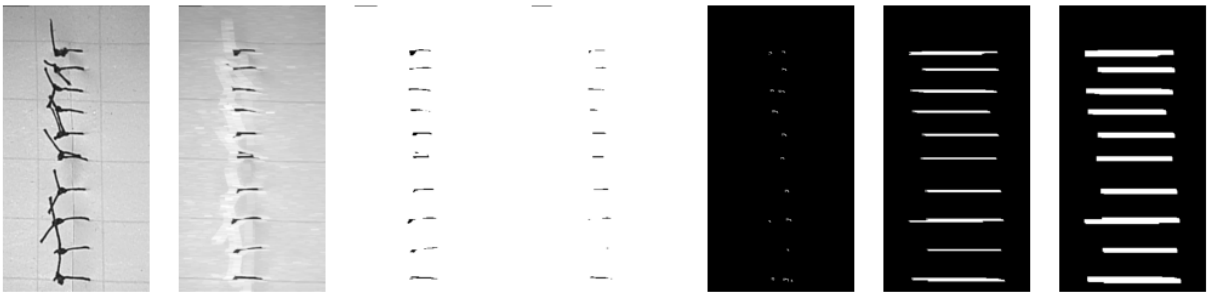

Image 5

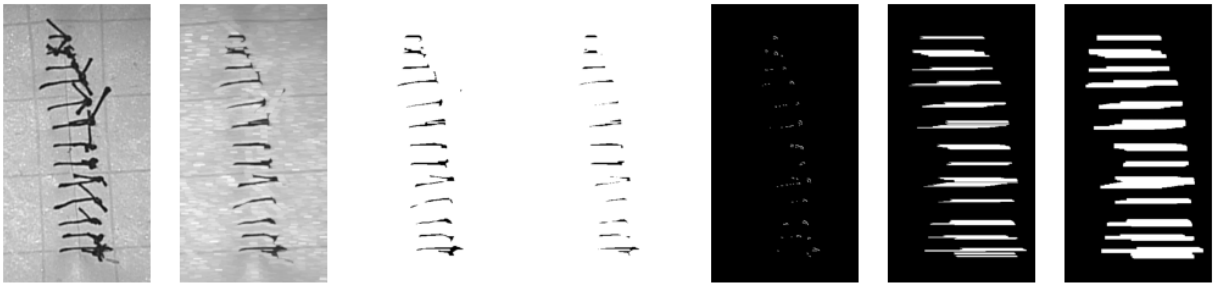

Image 6

Image 7

Image 8

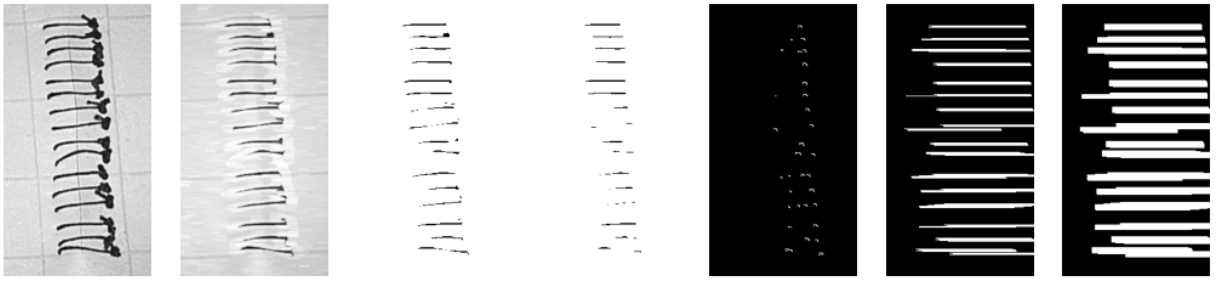

Image 9

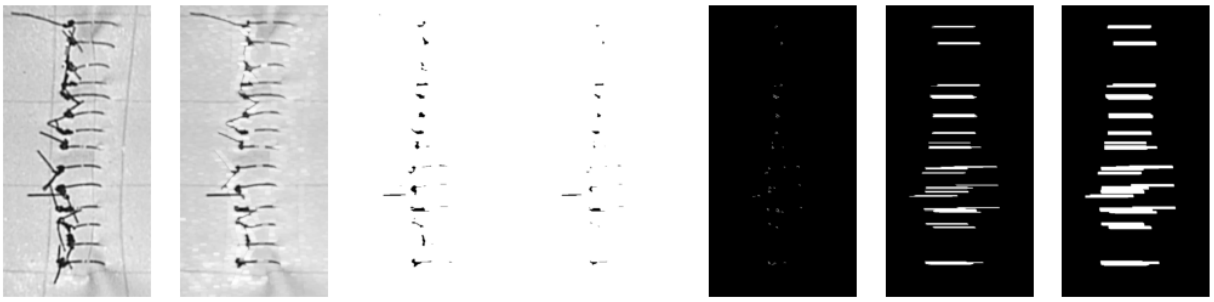

Image 10

6.2 Extra images for testing

I picked up 5 random pictures from the dataset and tested my code on them. I have included these pictures in the data directory. Following are the results:

Image number	Actual number of sutures	Detected number of sutures
11	13	13
12	13	14
13	10	10
14	15	16
15	9	10

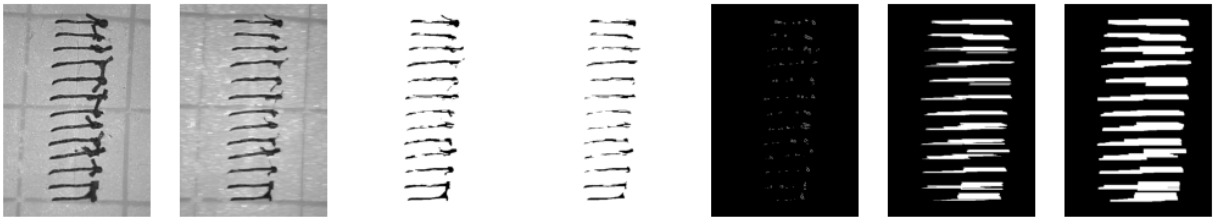

Image 11

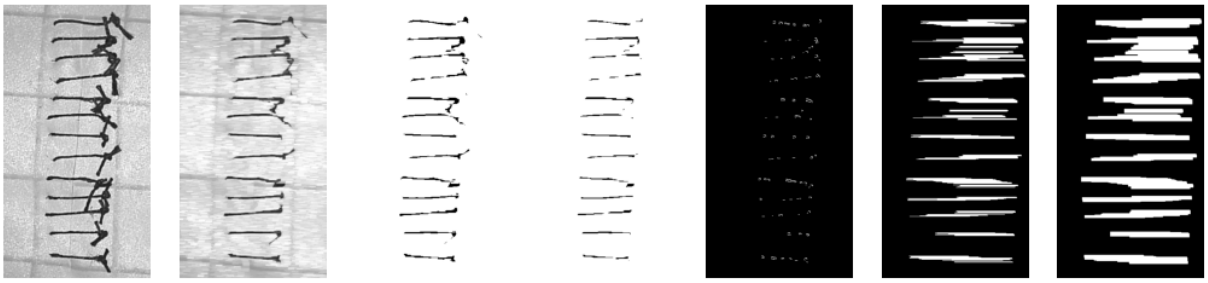

Image 12

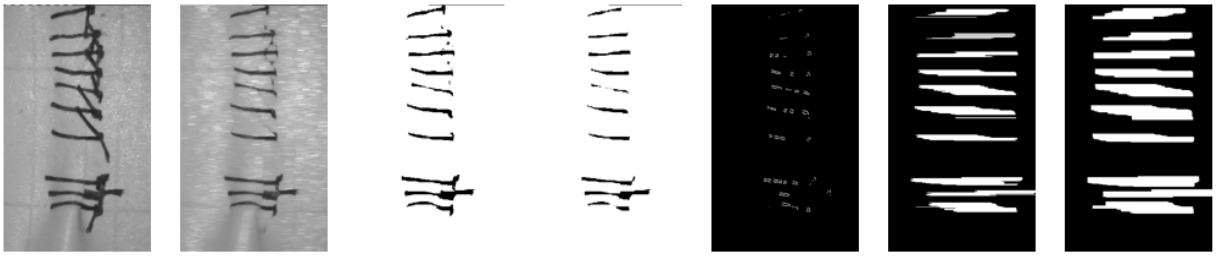

Image 13

Image 14

Image 15

7 References

- I referred to `https://github.com/FienSoP/canny_edge_detector` for the implementation of Canny Edge Detector.
- For some other implementations, I also referred to `https://github.com/ebeyabraham/Computer-Vision-in-Numpy`

.