IIP CW report

There are three objectives of the CW of IIP, task1 for face region detection, task2 for filters implementation and task3 for vessel retina distraction. The following will report respectively on these three tasks and begin with task1.

Task1: Implement a face detector using HSV color space or HUV space.

Here I choose HSV space to detect the face region and for *face1.jpg*, I found a face picture from the internet to trained my algorithm to find proper HSV cluster set to detect the face region in *face1.jpg*. ***figure 1.1.1*** is the training picture and ***figure 1.1.2*** is the cluster set information generated by Matlab.

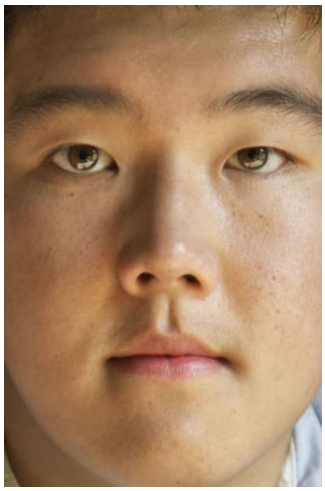


figure 1.1.1

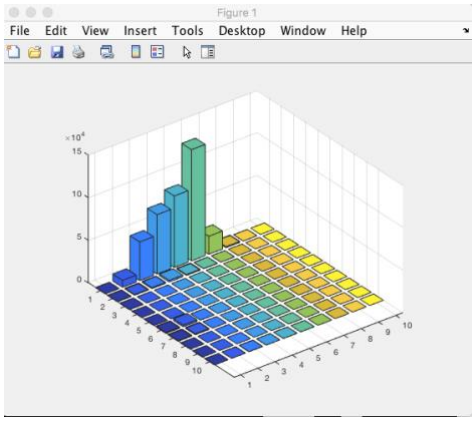


figure 1.1.2

When using this training picture, the best face-pixel cluster set I chose for later work is [0,0.1;0.4,0.6]. (match the format [H_start, H_end; S_start, S_end]). I also used other pictures from the internet and this one performed best, all the source image can be found in training_face_images file.

Then, change the RGB colorful *face1.jpg* to gray picture first by using rgb2gray() function and using the cluster set above to process the binary image, ***figure1.1.3*** is the origin picture of face1.jpg, ***figure1.1.4*** is the result of this operation.

figure 1.1.3

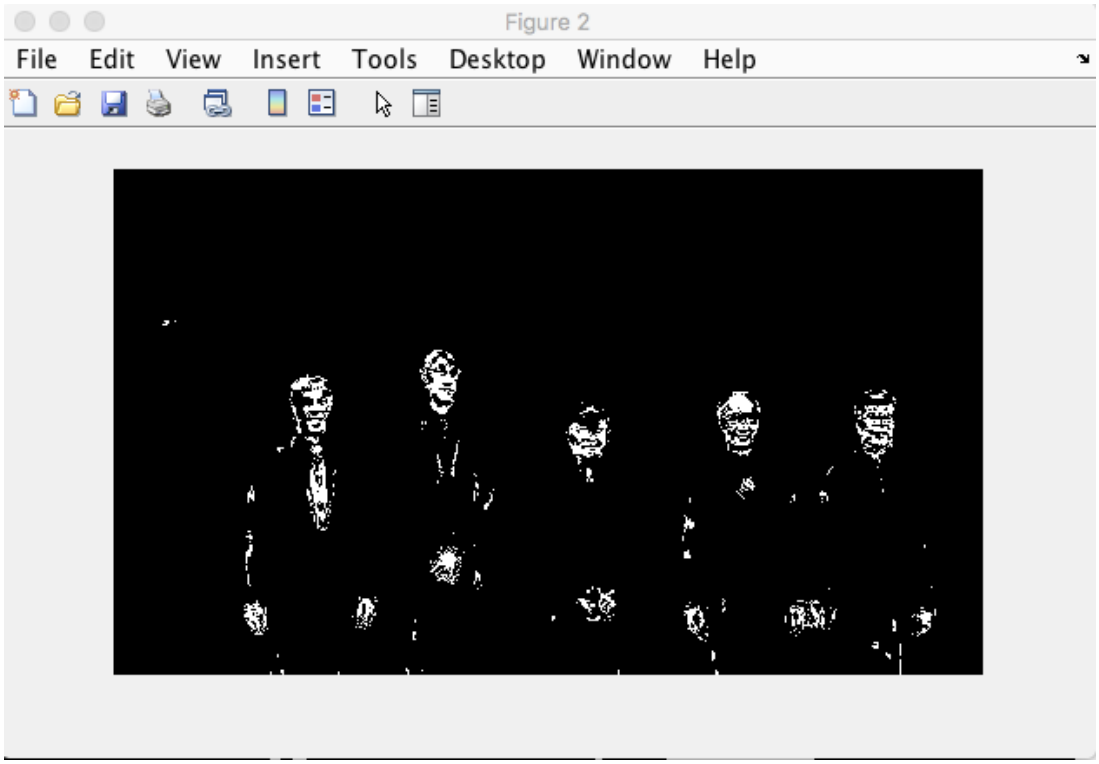


figure 1.1.4

While it looks that there is much noise on the binary image, so last step is to reduce the noise to highlight the face region, *figure 1.1.5* show the final detection result for face1.jpg.

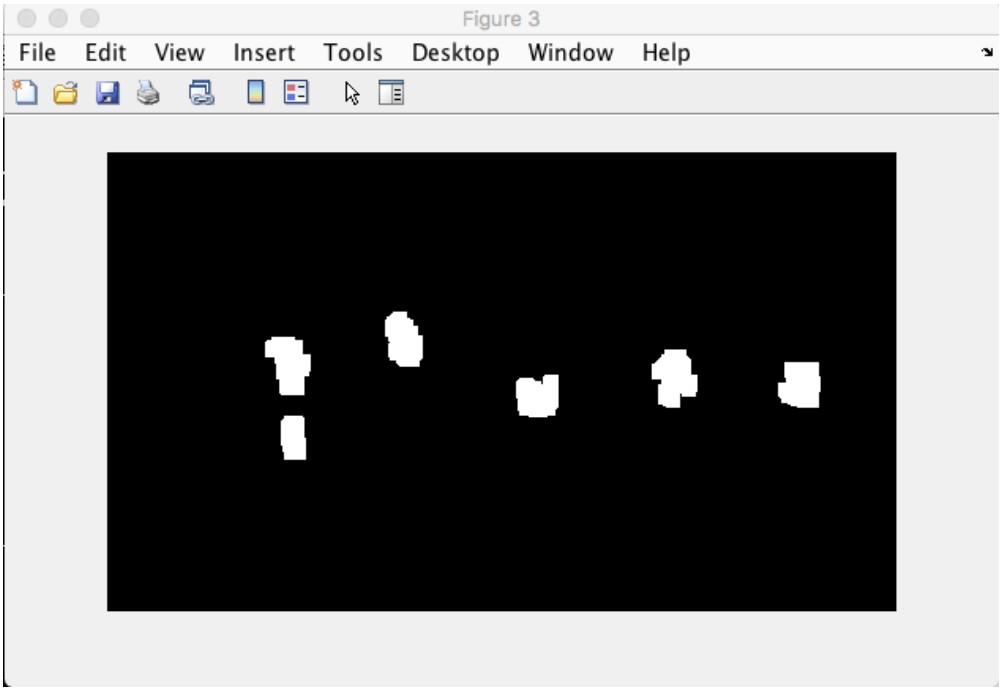


figure 1.1.5

For face2.jpg, similar steps as face1.jpg, figure 1.2.1, training picture, figure 1.2.2, cluster set information generated by Matlab.

figure 1.2.1

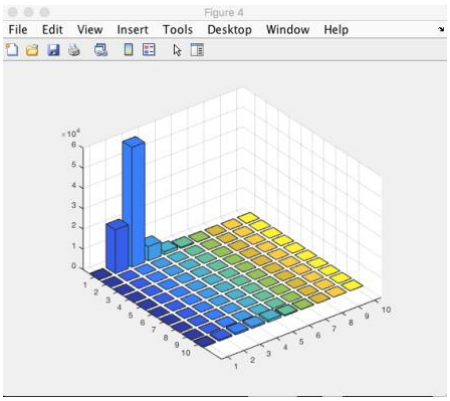


figure 1.2.2

When using this training picture, the best face-pixel cluster set I chose for later work is [0,0.1;0.1,0.3]. (match the format [H_start, H_end; S_start, S_end]). I also used other pictures from the internet and this one performed best, all the source image can be found in training_face_images file.

The same translation as operation on *face1.jpg*, and ***figure 1.2.3*** is the origin picture of *face2.jpg*, ***figure 1.2.4*** is the results of processing the picture using HSV cluster sets offered above, ***figure 1.2.5*** is the result for detecting the face region of *face2.jpg*.

figure1.2.3

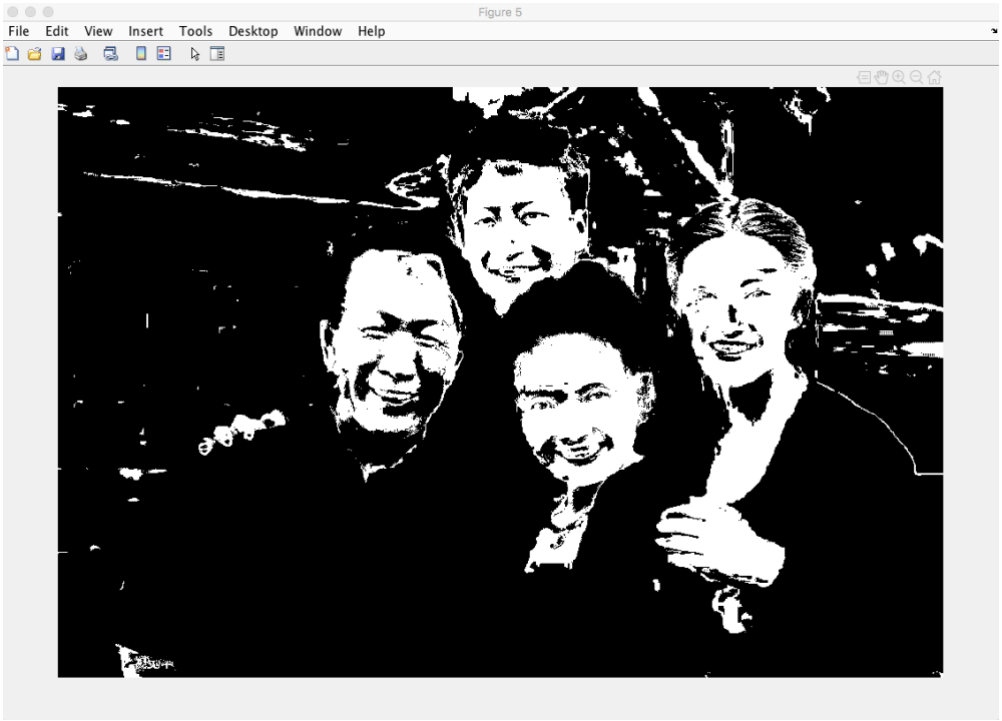


figure 1.2.4

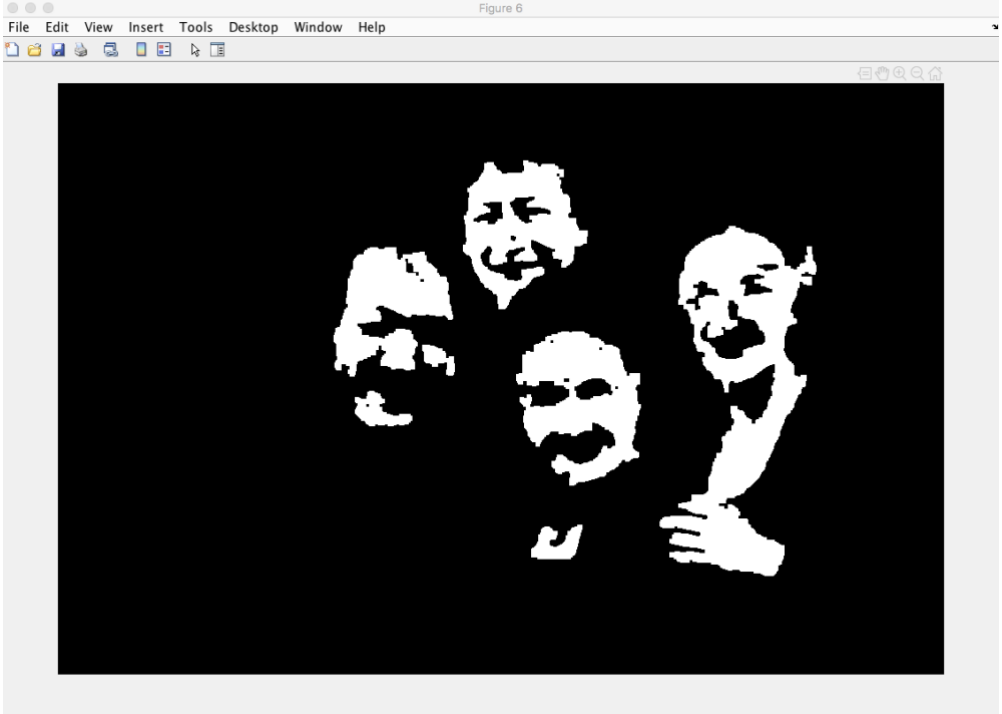


figure 1.2.5

In conclusion, in the process of detecting face region in face1.jpg, the tie of the first man from left to the right is also be detected as a face region, this is because in the training image, the color of the mouth of the boy seems to have similar color with the tie, so it is mismatched; second image detection performed well, all the white region represents the human skins, as well as the face.

File descriptions. 8 file in task1; dilation.m for expanding the white pixels, and erosion.m for eroding some certain regions; connection_comp.m is to eliminate some white regions; getFaceModel.m generates HSV face model for dace detection, used in the both image detection; getBinarizedFaceImage.m is used to translate the gray image into binary image; getMorphFace1.m is used to reduce the noise in face1.jpg image and getMorphFace2.m is used to reduce the noise in face2.jpg image. Finally, all the relative results will display when call for the script file task1main.m.

Task2: Implement and compare spatial filters

This task is to implement 5 filters for processing images discussed in the lecture, and they are 3 x 3 Mean Filter, 5 x 5 Gaussian Filter with sigma = 1, 3 x 3 Meidian Filter, 3 x 3 Anisotropic Filter with similarity function of (D-d)/D, 5 x 5 Bilateral Filter with space Gaussian sigma = 1 and range Gaussian sigma = 10.

When implemented these filters, next step is to add some noise to the image, firstly, add Gaussian noise with sigma = 20 on RGB channels of lena.jpg. Secondly, add another type of noise on the image, salt and pepper noise with noise rate of 10%.

The follow showed the results of after using different filters to process the image.

Origin picture (figure 2.1), Gaussian noise picture (figure 2.2), Salt & Pepper noise picture (figure 2.3), picture convoluted by 7 x 7 Gaussian filter (figure 2.4)

figure 2.1

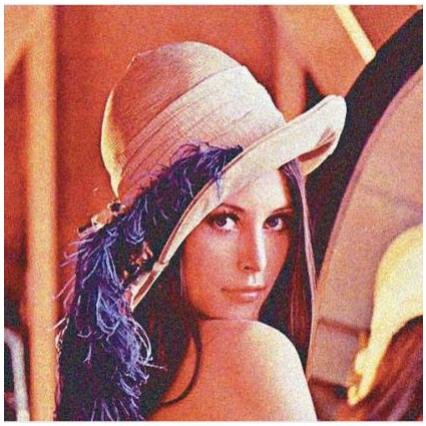


figure 2.2

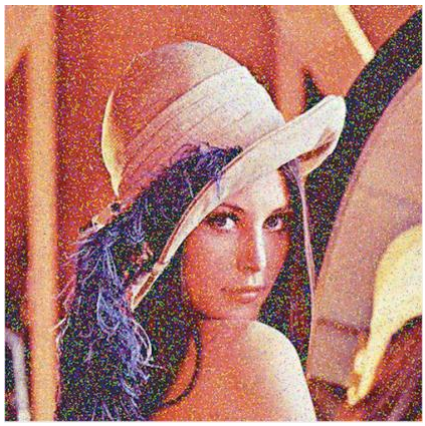


figure 2.3

figure 2.4

7 x 7 Gaussain filter processed distorted image processed by the filter I implemented

1 Noise picture convoluted by 3 x 3 Mean filter (figure 2.5.1)

2 Noise picture convoluted by 5 x 5 Gaussian filter with sigma = 1 (figure 2.5.2)

3 Noise picture convoluted by 3 x 3 Meidian filter (figure 2.5.3)

4 Noise picture convoluted by 3 x 3 Anisotropic filter (similarity fcn: (D-d)/d) (figure 2.5.4)

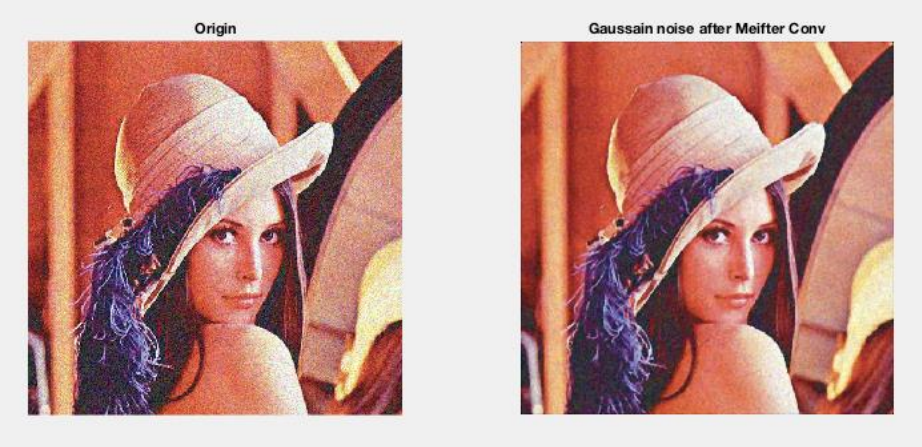
5 Noise picture convoluted by 5 x 5 Bilateral filter with space Gaussian sigma = 1 and range Gaussian sigma = 10. (figure 2.5.5)

Gaussian noise picture processed by the filter I implemented

1 Noise picture convoluted by 3 x 3 Mean filter (figure 2.6.1)

2 Noise picture convoluted by 5 x 5 Gaussian filter with sigma = 1 (figure 2.6.2)

3 Noise picture convoluted by 3 x 3 Meidian filter (figure 2.6.3)



4 Noise picture convoluted by 3 x 3 Anisotropic filter (similarity fcn: (D-d)/d) (figure 2.6.4)

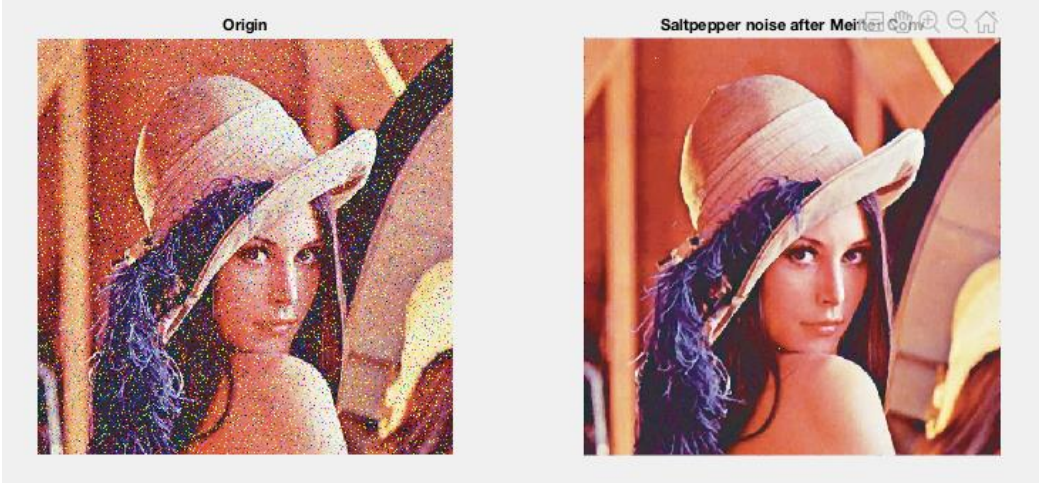
5 Noise picture convoluted by 5 x 5 Bilateral filter with space Gaussian sigma = 1 and range Gaussian sigma = 10. (figure 2.6.5)

Salt & Pepper noise picture processed by the filter I implemented

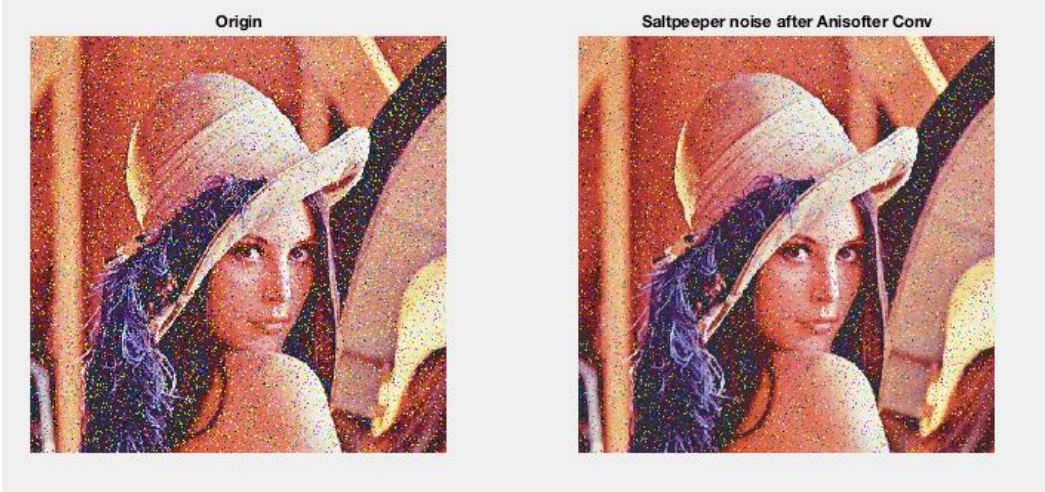
1 Noise picture convoluted by 3 x 3 Mean filter (figure 2.7.1)

2 Noise picture convoluted by 5 x 5 Gaussian filter with sigma = 1 (figure 2.7.2)

3 Noise picture convoluted by 3 x 3 Meidian filter (figure 2.7.3)



4 Noise picture convoluted by 3 x 3 Anisotropic filter (similarity fcn: (D-d)/d) (figure 2.7.4)



5 Noise picture convoluted by 5 x 5 Bilateral filter with space Gaussian sigma = 1 and range Gaussian sigma = 10. (figure 2.7.5)

Reflective conclusions

According to the results showed in these pictures, it can be known that when deal with Gaussian noise, Gaussian filter with sigma = 1 and size 5 x 5 perform better than other filters, while the worst one is 5 x 5 Bilateral filter with space Gaussian sigma = 1 and range Gaussian sigma = 10, few changes between the noise image and the processed image; Mean filter and Meidian filter, Anisotropic filter performed similar result then. For salt and pepper noise, Meidian filter seemed to perform better than others, while Mean filter and Gaussian filter performed also good and the result is more blurred than that of Mean filter. And Anisotropic filter and Bilateral filter performed less well than the other three filters which there are still some obvious noise pixels in the processed image, especially in the result of the Bilateral filter. And for the image processed by 7 x 7 Gaussian filter, the following 5 filter may have few changes on it.

Task3: Segment the retina blood vessel.

This task is asked to segment the vessel region from the retina image. Several filters have been used for this task and the following image show the relative results. And I choose the FrangiFilter to get the result. The implementation of Frangi filter was referred to the internet source which I wrote in the code file where it came from.

Segmentation steps using frangi filter.

Step1 : use Frangifilter process the gray image of the retina (figure3.1.1 is the origin image, figure3.1.2 is the result of frangi filter process)

figure 3.1.1

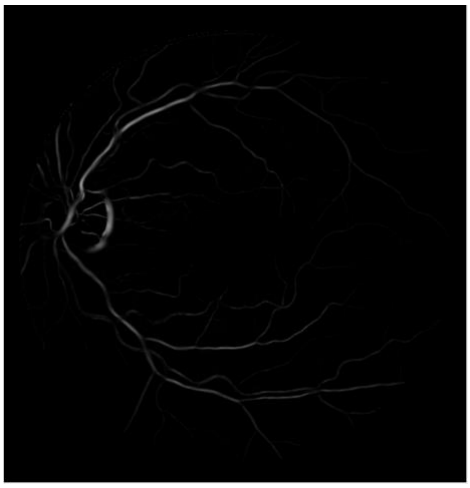


figure 3.1.2

Step2 : use logarithm translation to increase the difference between the bright region and the dark region twice. (figure3.1.3 is the first time using logarithm translation, figure3.1.4 is the second time.) so compare to figure3.1.2, the vessel is much clearer then.

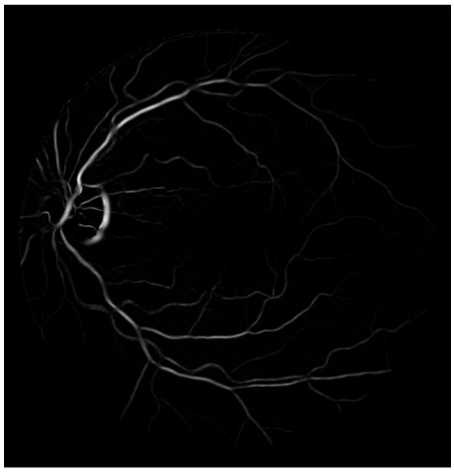


figure 3.1.3

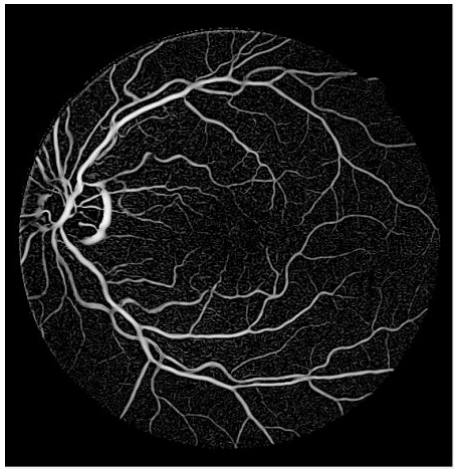


figure 3.1.4

Step3 : use Otsu thresholding algorithm to translate the gray image to binary image.

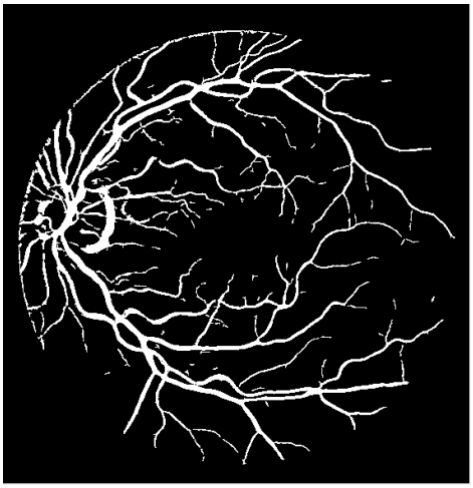


figure 3.1.5

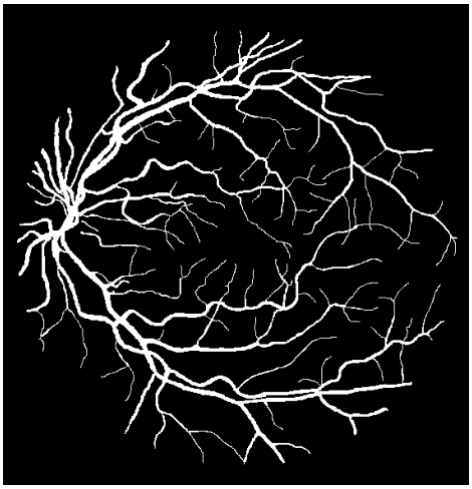


figure 3.1.6

Comparing to the label image, the result accuracy calculated gave as follows. (figure3.1.5 is the result got by using otsu thresholding, figure3.1.6 is the label image.)

The percentage of blood vessel pixels that is being correctly classified as blood vessel. Denoted as P.

$$P = 81.2\%$$

The percentage of background pixels (only consider the region in the mask.) that is being correctly classified as background. Denoted as N.

$$N = 93.3\%$$

The percentages of pixels are being correctly classified. (only consider the region in the mask). Denoted as T

$$T = 91.8\%$$

During the whole process and according to the result, it seemed that the tiny vessel in the image is hard to segment, many of them segmented as a white hole and may be regarded as the noise in the image, so more works need done further for more accurate result.

For the code files, there are 9 code files in total in this task, which are, FrangiFilter2D.m, which referred to another two code files, eig2image.m and Hessian.m, all the three files are from the internet, the source URL has been provided in the end of these three files. The files are main implementation of FrangiFilter; Sobel.m, the sobel operation; loG.m, lapacian of Gaussian filter; DoG, differentiate of Gaussian filter; Canny.m, Canny operation, laplacian.m, laplacian filter; and all the result will be showed when call task3main.m.

Some test I have done in the following

1 Sobel filter (figure3.2.1, the origin image; figure3.2.2, image processed by sobel operation)

figure 3.2.1

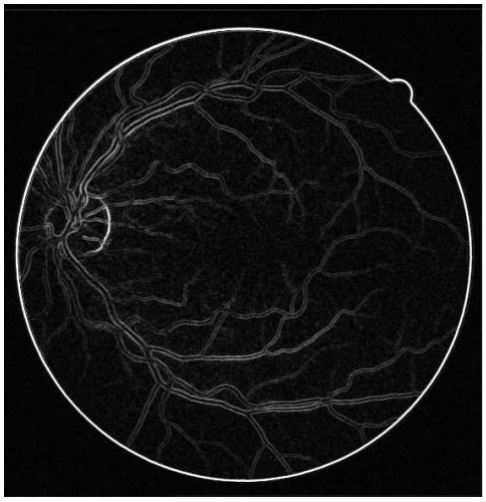


figure 3.2.2

2 loG filter (figure3.3.1, the origin image; figure3.3.2, images processed by loG filter with sigma = 1,2,3,4)

figure 3.3.1

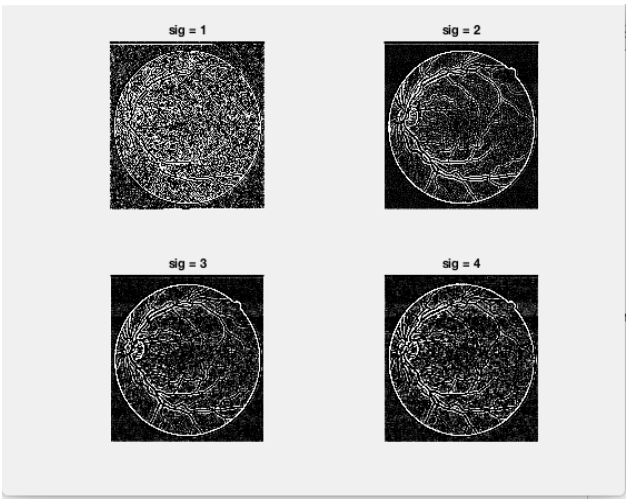


figure 3.3.2

3 DoG filter (figure3.4.1, the origin image; figure3.4.2, image processed by DoG operation)

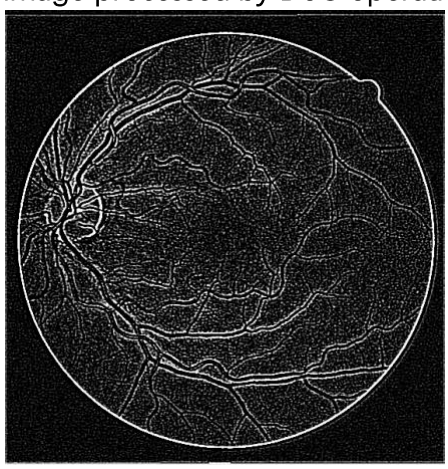


figure 3.4.1 *figure 3.4.2*

4 Canny filter (figure3.5.1, the origin image; figure3.5.2, image processed by Canny operation)

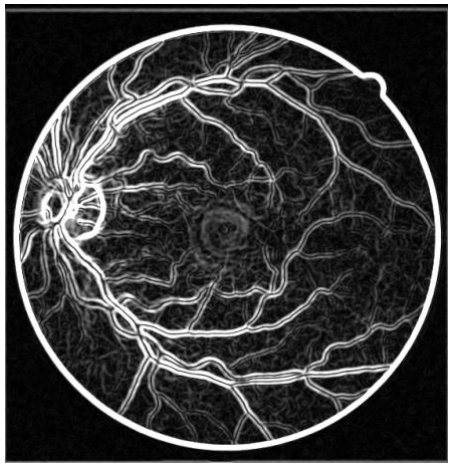


figure 3.5.1 *figure 3.5.2*

In conclusion, several filters can use to segment the vessel region while I did not successfully use them to get the expected result, more works will be done further. Here will talked some reflections on these operations. Firstly, is the **frangifilter**, which get a reasonable result and gave good answer for task3; then, for **sobel** operation, it can detect the frontier of the vessel precisely but it seems that this operation will be such sensitive to the salt and pepper noise, so before using this operation, reduce the salt pepper noise is necessary; for laplacian of Gaussian filter, seen in the picture that the vessel frontier can be detected clearer with the increasing of sigma; for **DoG**, also showed at the picture that the vessel is segmented but there are mush noise in the image; **Canny** operation is to depict the profile of the object, here the canny I implemented is not a completed one, so I just provide what I have done at present. In future, if I got any chance to continue working on this, I will try some machine learning methods on this task, such as CNN.