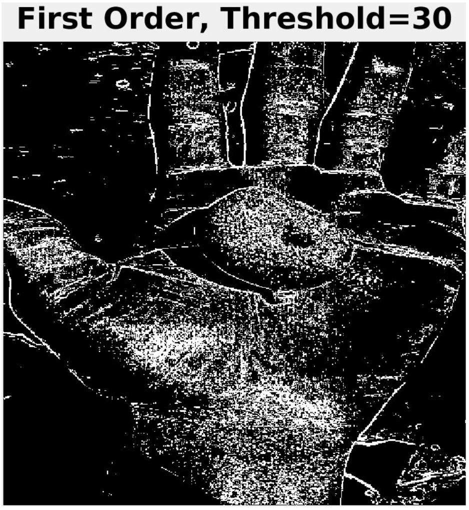
DIP HW2

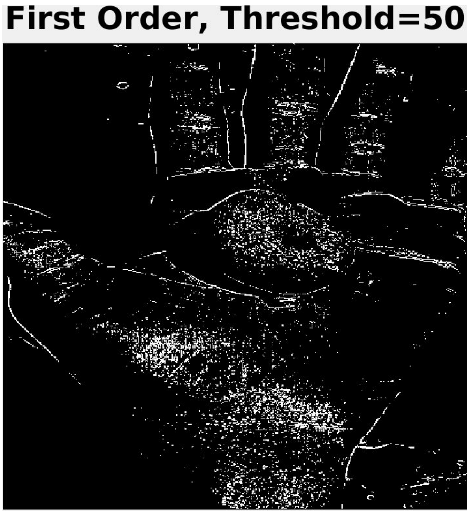
資工三 B04902105 戴培倫

Problem 1

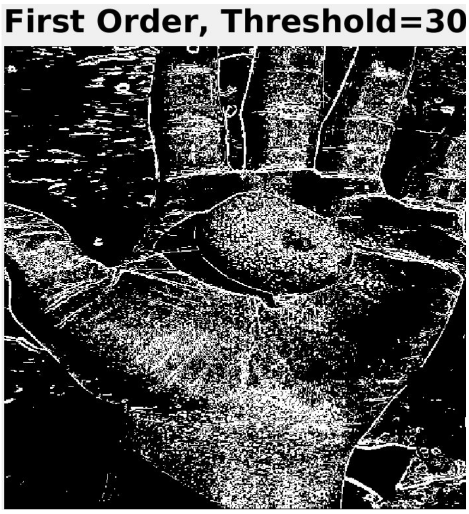
(a)

(i) First Order: 3 points

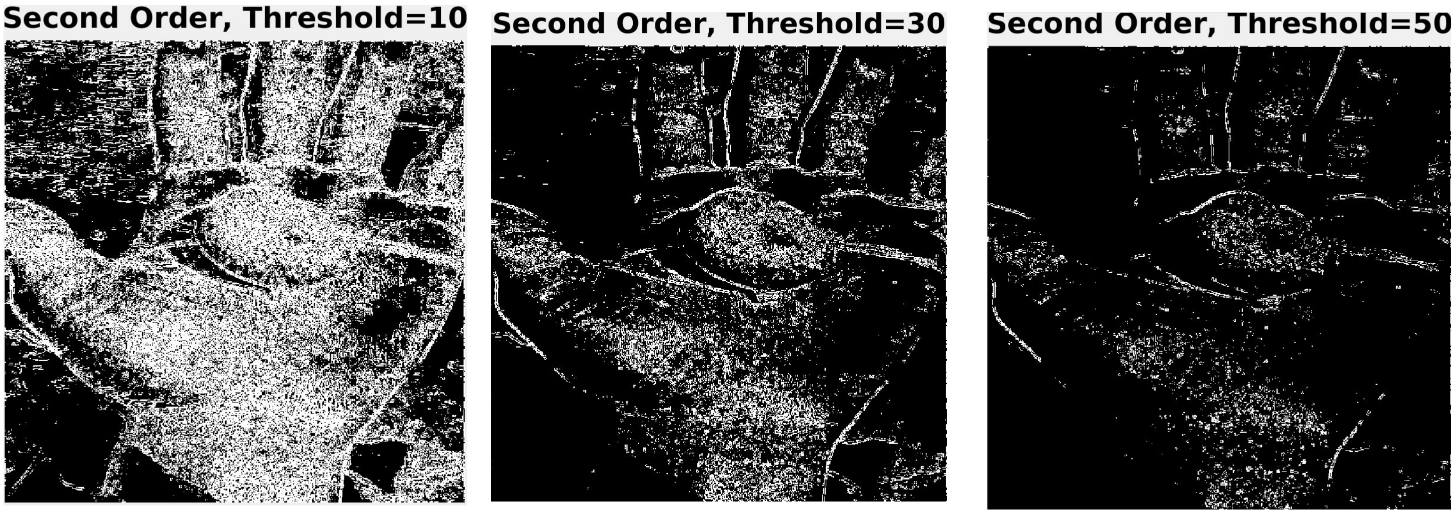




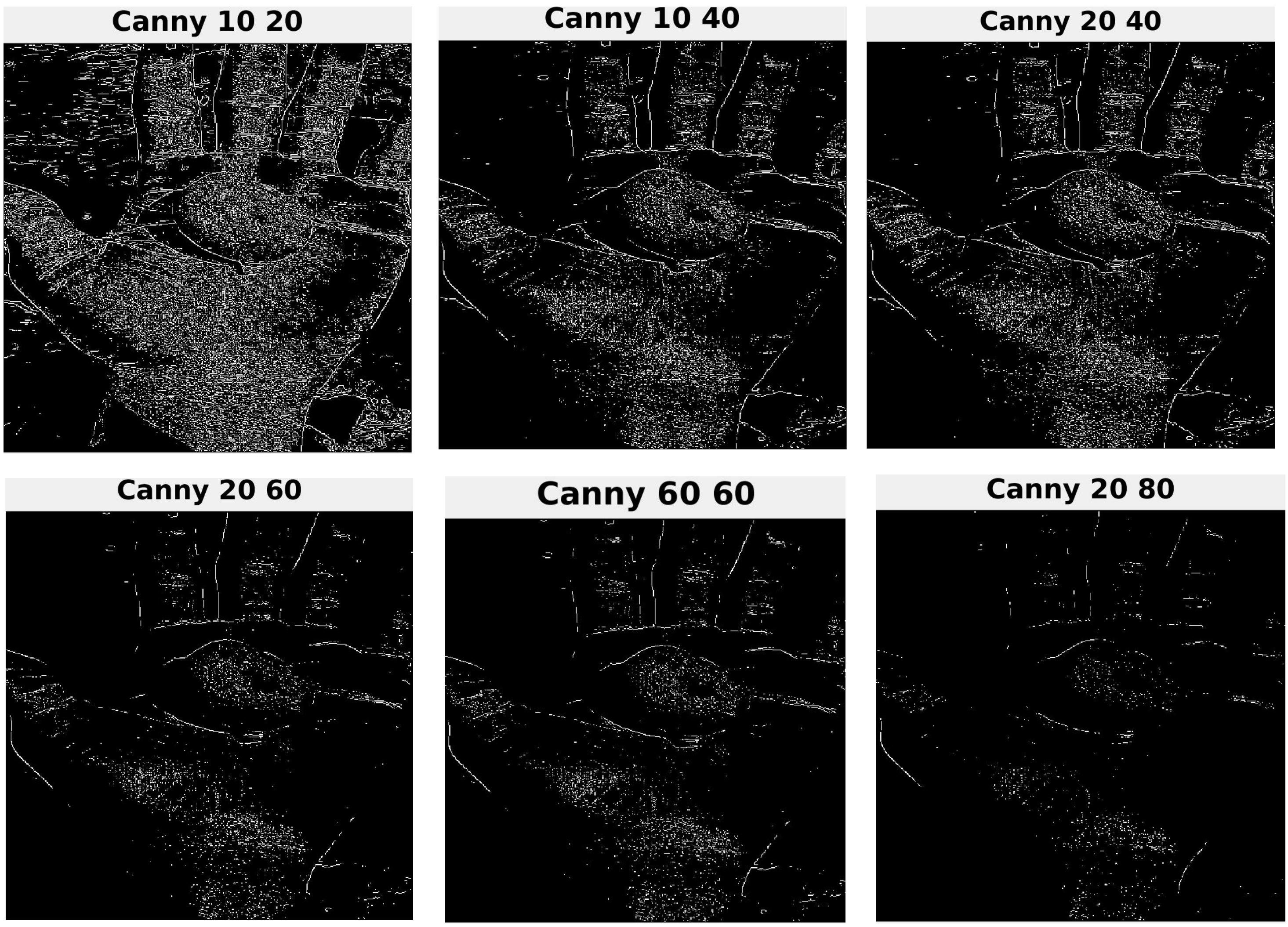
(ii) First Order: Roberts cross differentiation



(iii) Second Order

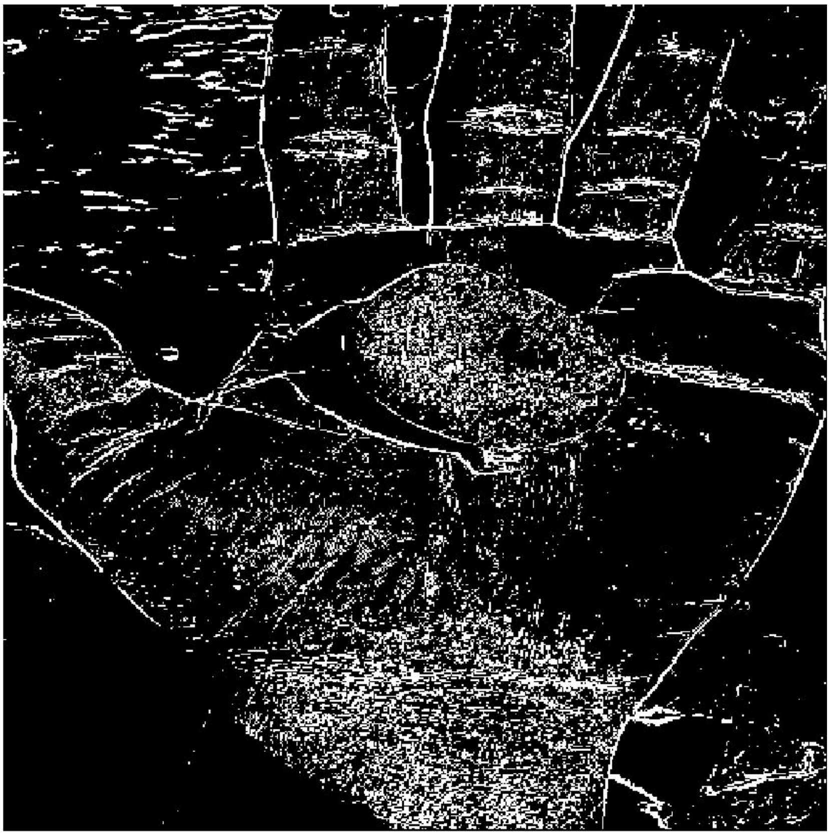


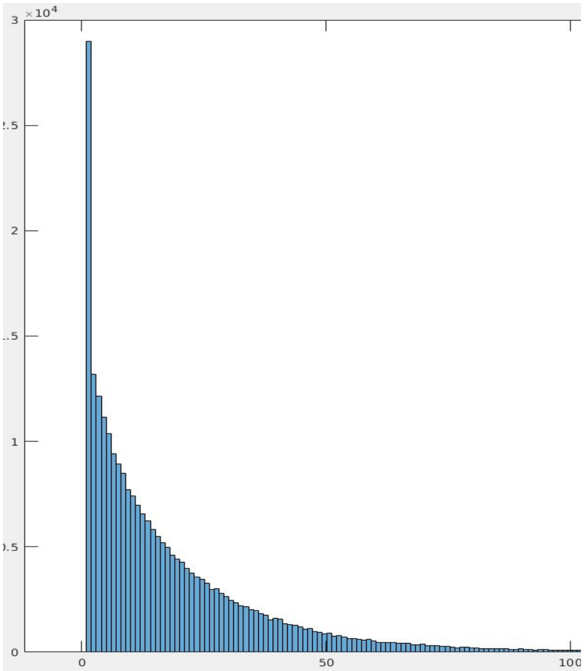
(iv) Canny

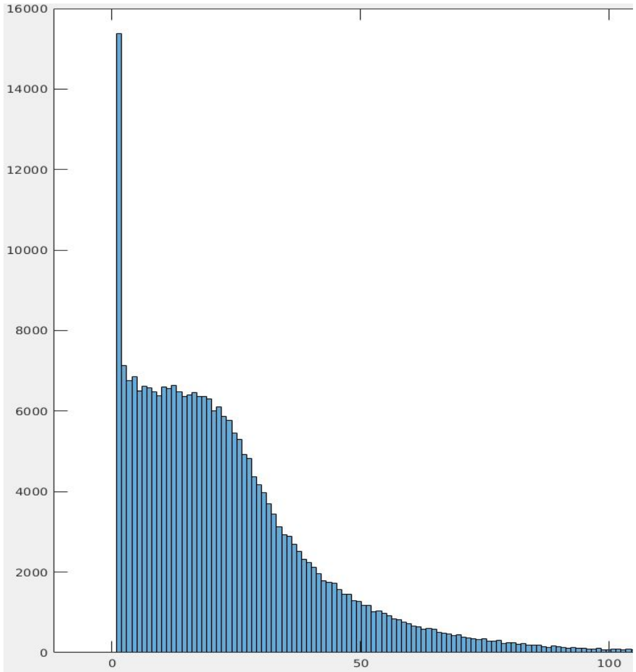


I tried two differentiation method for first order edge detection: 3-point differentiation and Roberts cross differentiation. It’s clear that Roberts outperforms on higher threshold(30, 50). However, when threshold is low(10), 3-point method performs better. Roberts method cannot clearly find the edge of the fingers and the background.
Second order detection performs better than first order detection (threshold=10,30) since it filters out a lot of pixels that are considered edge in first order detection. However, when threshold=50, it filters out too many pixels that we can’t clearly see the shape of the hand.
The two numbers after the Canny’s image title are TL, TH respectively. From my experiment, TL doesn’t matter much to this image, as we can see from “Canny 20 60” and “Canny 60 60”.
Overall, I think Canny edge detection performs the best since it can filter out many non-edge pixels while still keeping most important edge pixels.

(b)





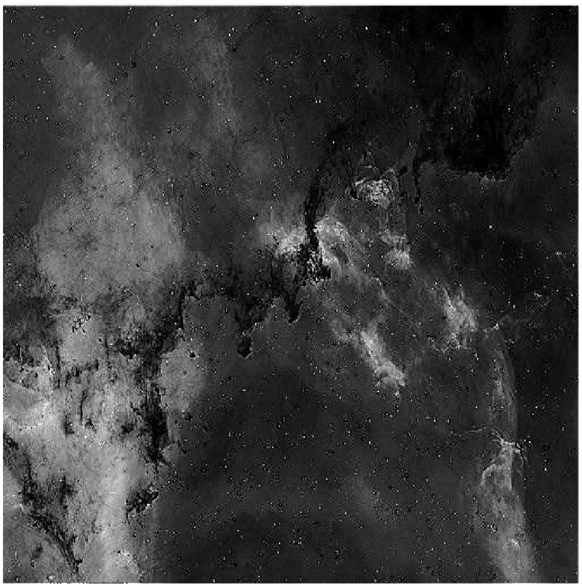


I used a mask to detect the 45° gradient, from the histogram of first order value of sample1.raw and sample2.raw below. We can see that the gradient of the noise lands around 20~35.
Therefore, I set different threshold for different direction of gradients. 40 for 45°, and 30 for other three directions.

Problem 2

(a)High-pass filtering

C1



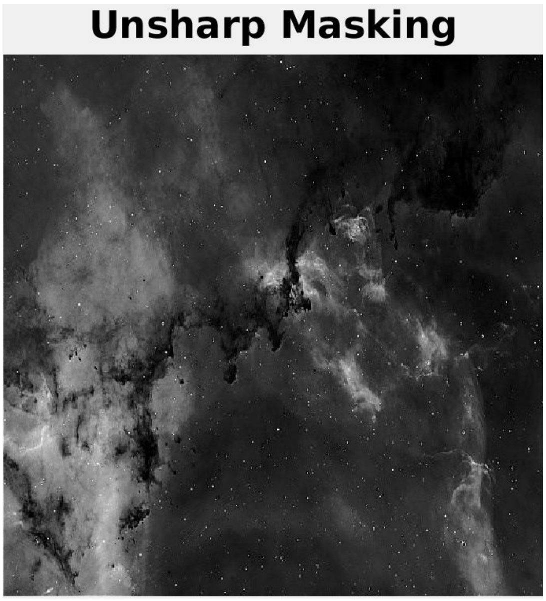
$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

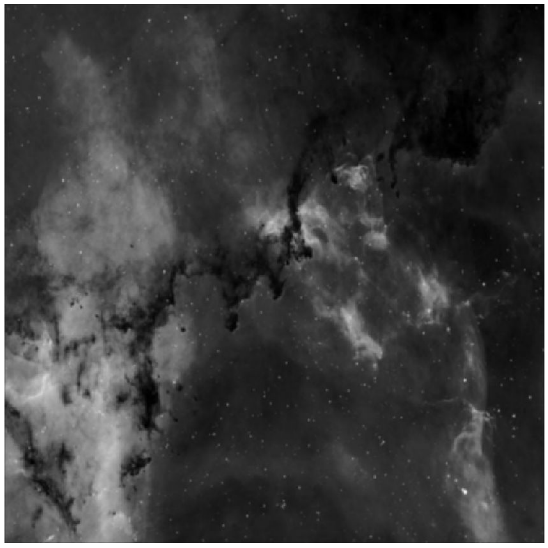
C2

$$H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Unsharp Masking (b = 2, c=2/3)

(b = 2, c = 5/6)





I think that in this image, lower c leads to better output is reasonable since lower c means more approximate to high pass filtering. Therefore, we can see clearer stars in the image on the left.

$$H = \frac{1}{(b+2)^2} \begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix}$$

(b)Warping Image

As for warping the Image, I used a sin function to implement it. (shifting the original pixels to new position)

For different warping on 4 edges, I divide the Image into 4 parts. Each division applies to its own sin function with different amplitude, frequency and shifting.