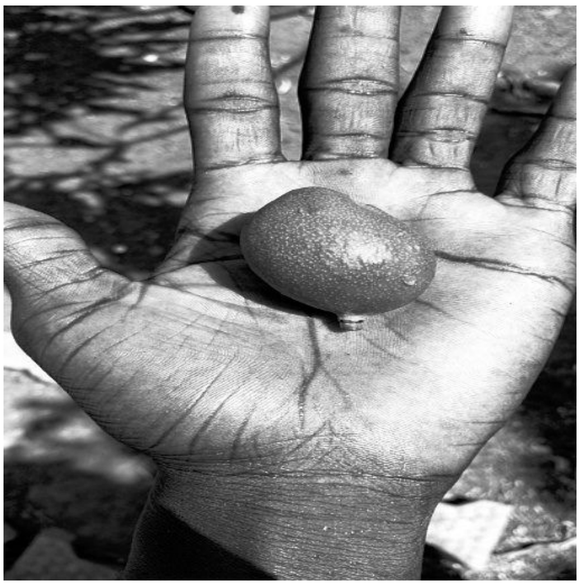
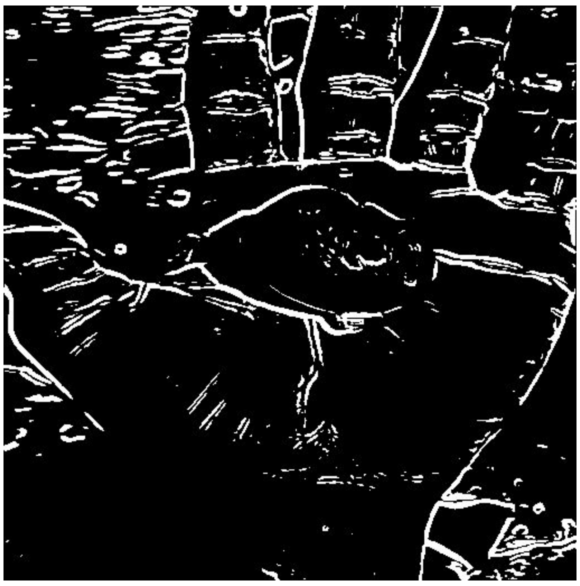
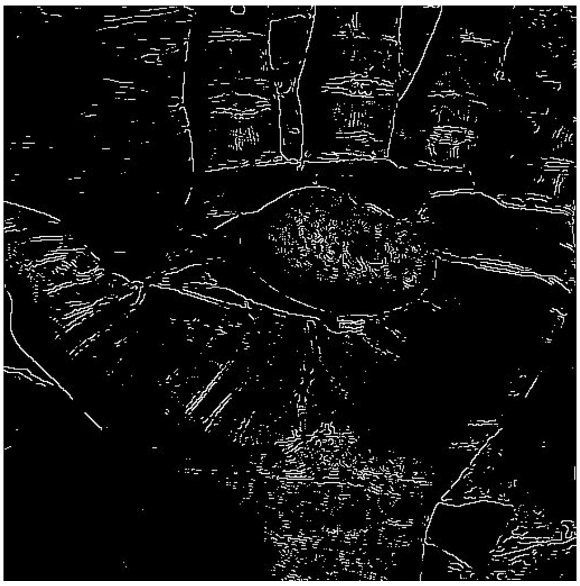
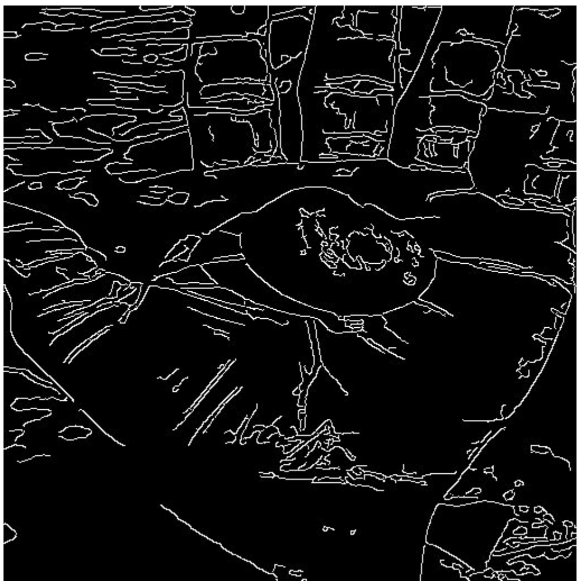
DIP HW2

B03902082 資工四 江懿友 big2632@gmail.com

Problem 1: Edge Detection

(a) 1st-order, 2nd-order, Canny

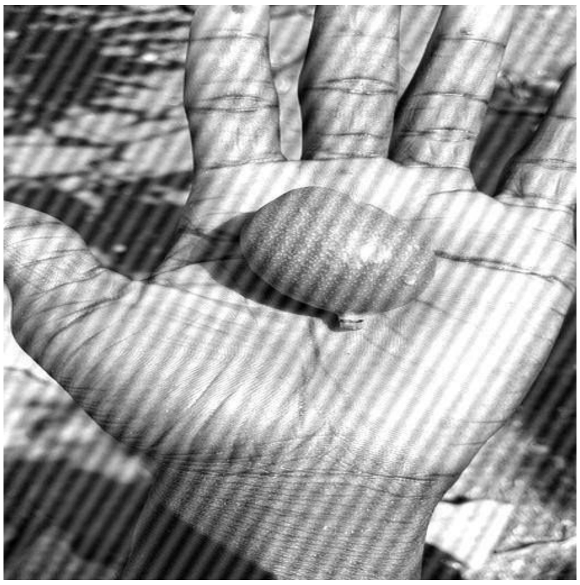
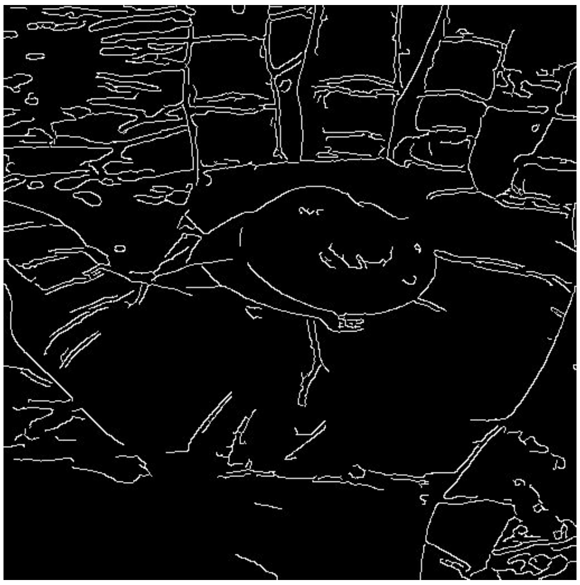
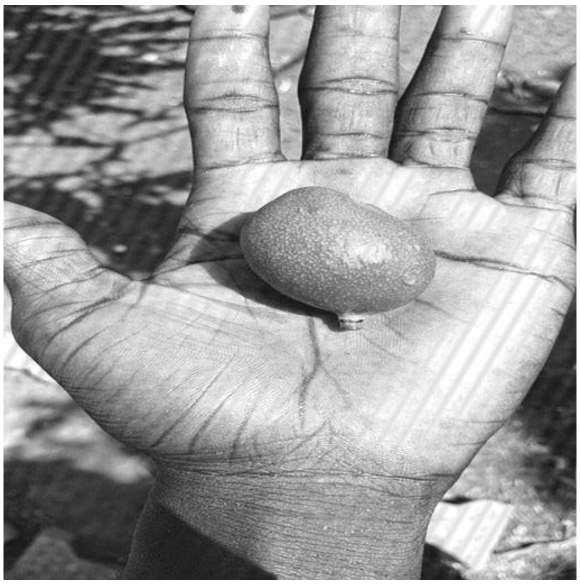
input	1st-order
	
2nd-order	Canny
	

For 1st-order edge detector, first I used a 5x5 gaussian filter to smooth out any noise in the image. Next I used the Sobel kernel to calculate row & column gradients. I also tried using three-point method, Robert's cross and Prewitt filter to calculate the gradients, and all gave similar results. Then I calculate the CDF of gradient magnitude and set the threshold at 90%, so 10% of the pixels are marked as edges. 90% is just an empirical value.
For 2nd-order edge detector, first I used the same 5x5 gaussian filter to smooth out the image. Next I used the eight-neighbor laplacian approximation to calculate the laplacian. Again I tried using four-neighbor, eight-neighbor and separable eight-neighbor and all gave similar results. Then I calculate the CDF of laplacian magnitude, and set the threshold at 80%, so 80% of the pixels are marked as 0 and the remaining 20% are marked as +1 or -1. Then I mark all zero crossing pixels as edge.
For the Canny edge detector, first I used the same 5x5 gaussian filter to smooth out the image. Next I used the Sobel filter to calculate row & column gradients and used them to calculate the gradient magnitude and orientation. Then I perform non-maximal suppression on gradient magnitude. Then I perform hysteretic thresholding, with the high & low thresholds be 0.11 & 0.05. These values are also empirical. Last I mark candidate pixels as edge pixels if they are connected to any edge pixel.

The 1st-order detector edge map's lines are too wide, in other word it violates the "single response constraint".
The 2nd-order detector gives thin lines so it produce better results than 1st-order detector. However it is very sensitive to noise. Even if the input image is smoothed with a gaussian kernel, the output edge map is still very noisy.
Canny detector produces clean thin lines without noise. Canny has 2nd-order detector's advantage but not it's disadvantage.

(b) Edge detection on periodic noise

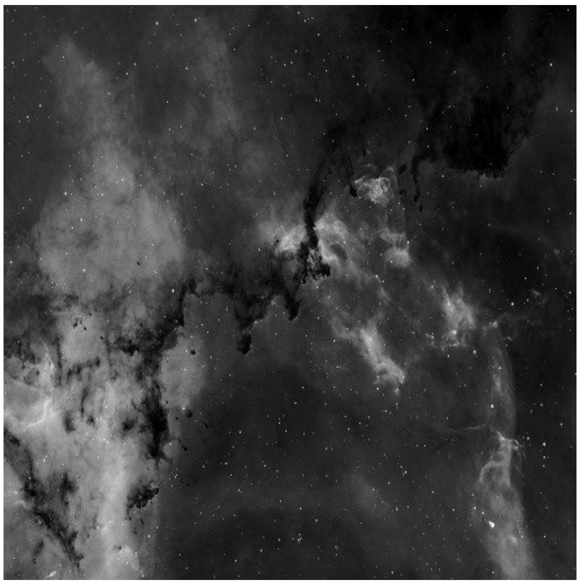
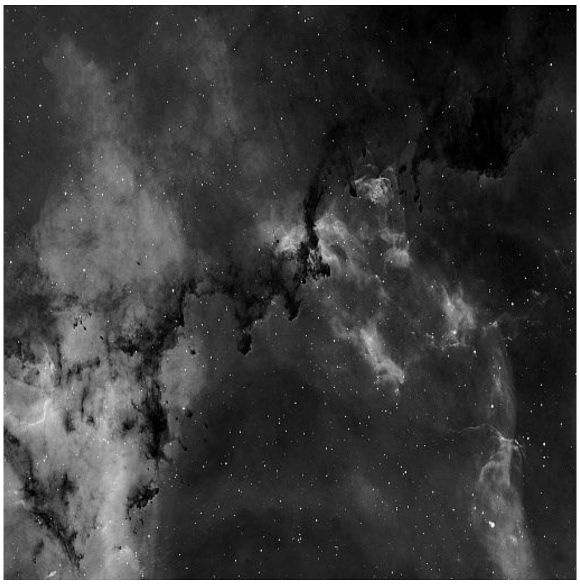
input	output edge map

	
input denoise	
	

I observe that the input image have some periodic noise added to it, so I try to model the noise and subtract the noise from the input.
I model the noise to be a rotated sine signal and I determined the parameters of the noise empirically, by trying different parameters and tune the parameters by hand. The slant of the sine wave is 31 degrees, magnitude is 25/255 and the wavelength is 16 pixels. After subtracting the sine signal from the input image I get the denoised image, which looks a lot less noisy. Then I perform Canny edge detector on the denoised image and get the output edge map.

Problem 2

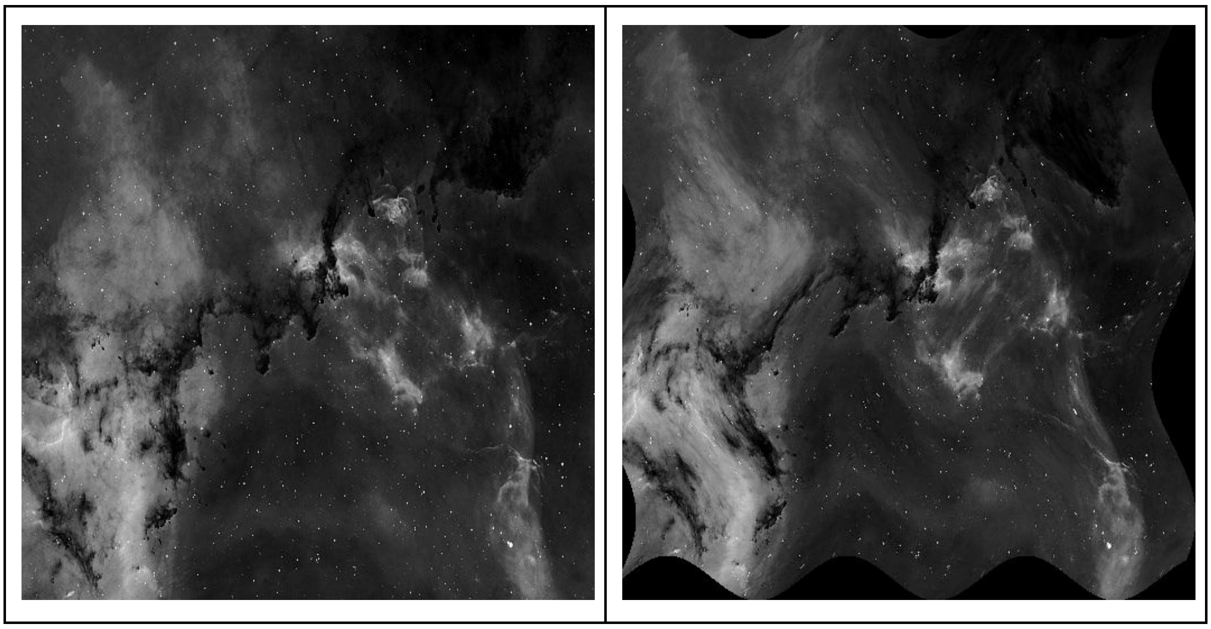
(a) Edge crispening

input	output
	

I used unsharp-masking to filter the image. For the low-pass filter I tried both 3x3 and 5x5 gaussian kernels. Both kernels yield similar results, so I choose the 3x3 kernel. The parameter "c" I choose is 0.7. A "c" value lower than 0.7 will make the output looks not "sharp" enough.

(b) Warping

input (output of 2(a))	output



I used wave shaped distortion along the x and y axis to warp the image.
The wave distortion is composed of sine function and scaling. The distortion function (mapping function) is specified as:

$$u(x,\ y) = 1.022 \times (8 + x + 20\sin(\tfrac{2\pi}{160}y))$$
$$v(x,\ y) = 1.022 \times (8 + y + 20\sin(\tfrac{2\pi}{250}x))$$

where (x, y) is the output image's coordinate and (u, v) is input image's coordinate. So to render the pixel (x, y) of the output, I have to look for the pixel located at (u, v) of the input. If either of u or v is out of boundary, then the pixel (x, y) is black. Otherwise a pixel value is calculated by bilinear interpolation of the four pixels nearest to (u, v) as either u or v may not be integer values.