

Part 1

3.22

$$\begin{aligned}
 (a) \quad V &= [1 \ 2 \ 1]^T, W^T = [2 \ 1 \ 1 \ 3] \quad (b) \quad W = W_1 \star W_2 = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 6 & 2 \end{bmatrix} \\
 VW^T &= \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [2 \ 1 \ 1 \ 3] \quad \text{rank}=1 \quad \text{rank}=1 \\
 W &= \begin{bmatrix} 2 & 1 & 1 & 3 \\ 4 & 2 & 2 & 6 \\ 2 & 1 & 1 & 3 \end{bmatrix} \quad \Rightarrow W_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\
 & \quad \quad \quad W_2 = [1 \ 3 \ 1]
 \end{aligned}$$

3.27

(a) convolve a 3×3 kernel four times

$$\Rightarrow (3+3+3+3) \times (3+3+3+3) \Rightarrow 12 \times 12$$

Size of the single Gaussian kernel is 12×12 .

(b)

standard deviation (σ):

$$\sigma = \sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_4^2} = \sqrt{4} = 2$$

the standard deviation of the single Gaussian kernel is 2.

3.38

Smoothing kernel and Laplacian are linear operations,
so it makes no difference.

4.3

$$\begin{aligned}
 (a) \quad \delta(t) \star \delta(t-t_0) & \quad (b) \quad \delta(t-t_0) \star \delta(t+t_0) \\
 = F^{-1}(F(\delta(t)) \star \delta(t-t)) & \quad = F^{-1}(e^{-j^2 \pi \omega} \cdot (e^{j^2 \pi \omega})) \\
 = F^{-1}(1 \times e^{-j^2 \pi \omega}) = \delta(t-t_0) & \quad = F^{-1}(1) = \delta(t)
 \end{aligned}$$

4.32

(a) $\{a, b, c, c, b\}$

1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0

1	2	3	4	5
a	b	c	c	b

\Rightarrow embedding

1	2	3	4	5	6	7	8	9
0	0	a	b	c	c	b	a	0

The array is symmetric as the original array.

(b)

1	2	3	4	5	6	7	8	9	10
0	0	0	-b	-c	0	c	b	0	0

For this case also (3,4) of original array coincide with (5,6) of new array to make it symmetric.

(c)

1	2	3	4	5	6	7	8	9	10
0	0	-a	b	c	d	c	b	0	0

(3,4) coincide with (5,6) to make it symmetric.

(d)

1	2	3	4	5	6	7	8	9
0	0	0	-b	-c	c	b	0	0

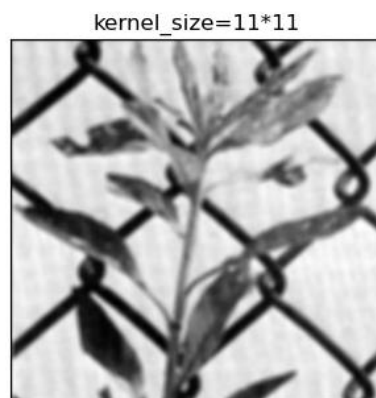
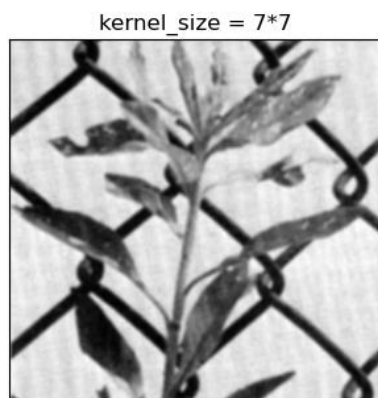
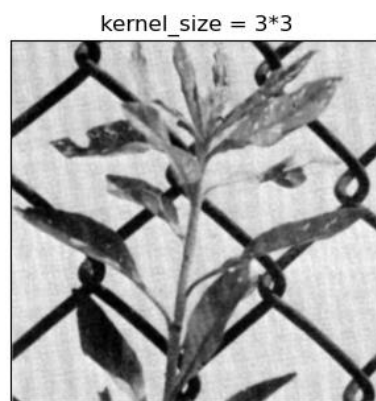
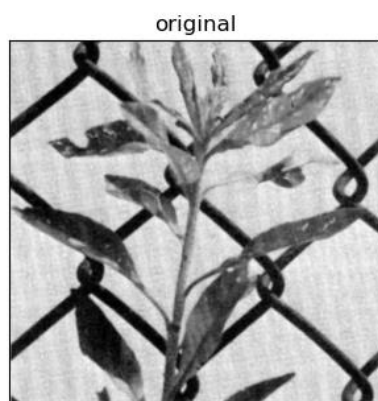
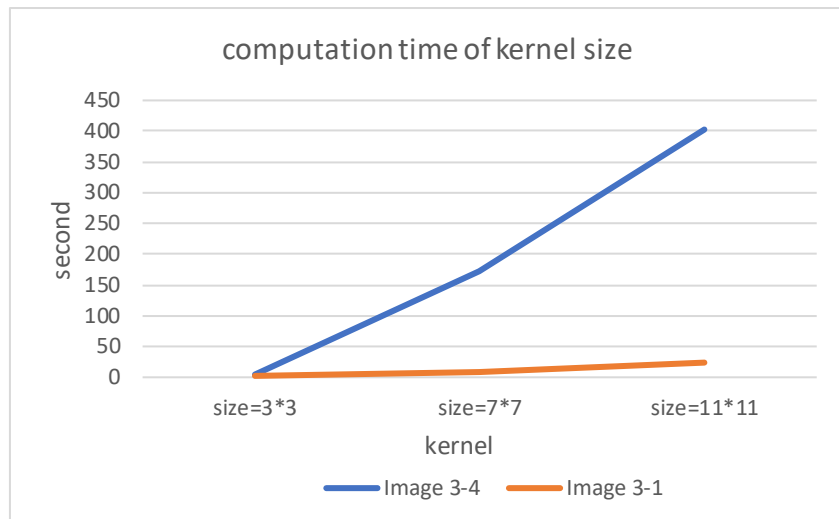
(3) coincide with (5) to make the new array symmetric.

Part 2

Design a computer program for spatial filtering operations using various types of masks. Test your program with several images and report your results. Discuss the effect of mask size on the processed images and the computation time.

`gray_operation()` 為灰階圖像卷積運算之函式，卷積的運算會先用 `kernel_rotate()` 翻轉遮罩，再利用 `np.pad()` 對圖像進行 zero padding，而彩色影像的卷積運算函式為 `RGB_operation()`。

運算時間隨著遮罩的大小上升，尤彩色影像的運算時間又較灰階圖來的長，因其有 3 個通道所致。



kernel_size = 7*7



kernel_size=11*11



Part 3

Test your program with at least 4 images and compare the results with those processed with the Sobel operator. Discuss the effect of zero-crossing threshold on the Marr-Hildreth edge detection method.

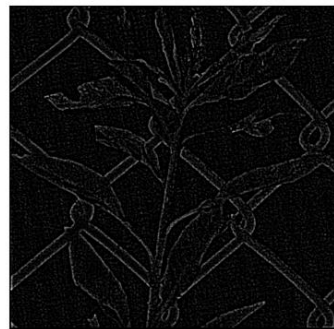
零交叉閾值是在拉普拉斯算子（LoG）來檢測邊緣之後，決定哪些點被認為是邊緣的閾值。LoG 操作可用於檢測圖像中的邊緣，因為它對邊緣上的像素產生零交叉，也就是從一個邊緣區域到另一個區域時像素值會發生變化。

Marr-Hildreth 和 Sobel 運算子都是用於圖像處理和邊緣檢測的技術，而 Marr-Hildreth 邊緣檢測對於大範圍的邊緣和細節有較好的表現，但對於對噪聲也較敏感；Sobel 運算子對於檢測較小、銳利的邊緣具有較好的敏感度，但對於細節檢測較不佳。

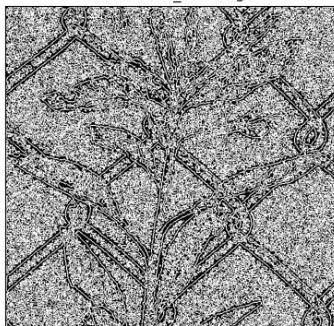
original



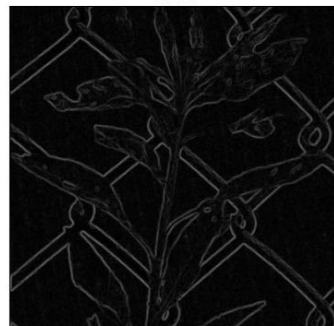
LoG



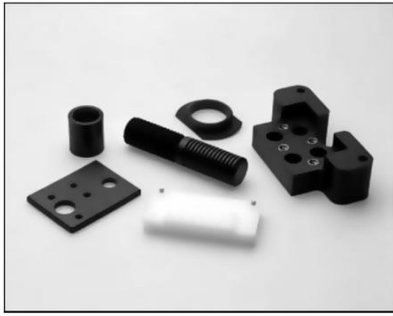
LoG + zero_crossings



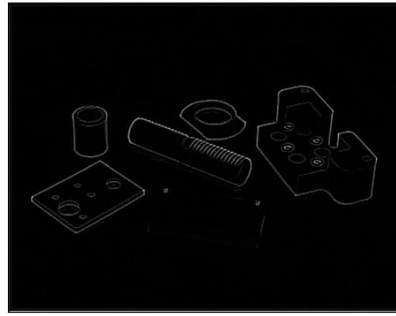
sobel



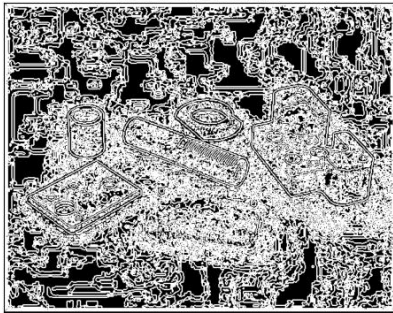
original



LoG



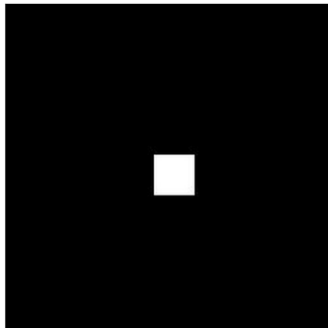
LoG + zero_crossings



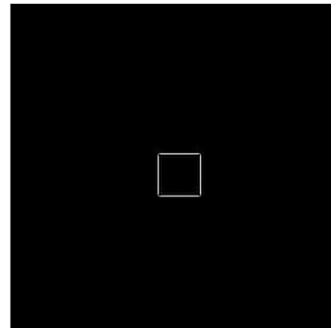
sobel



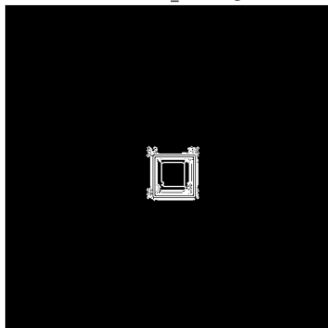
original



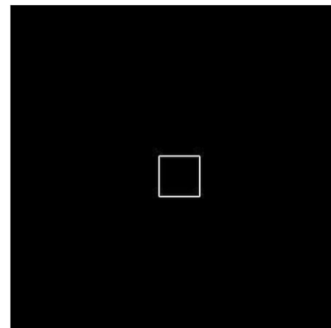
LoG



LoG + zero_crossings



sobel





Part 4

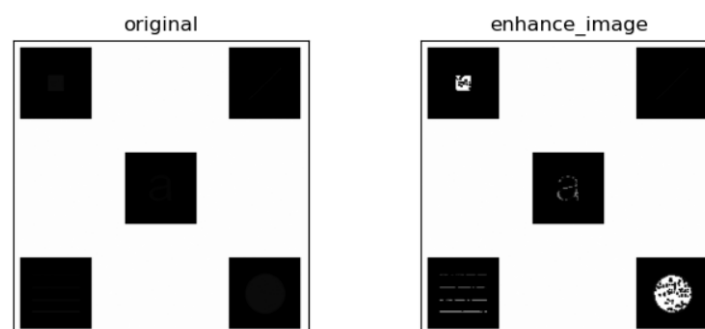
Reproduce the enhanced image of Figure 3.27b using the original image provided.

先利用 `mirror padding` 方法填補邊緣的缺值，接著利用 `np.mean()` 和 `np.std()` 獲取整張影像的平均值和標準差。再對影像作 `convolution`，並計算 `kernel` 範圍內的平均值和標準差並依據公式

$$g(x,y) = \begin{cases} C \cdot f(x,y), & \text{if } k_0 m_G \leq m_{s_{xy}} \leq k_1 m_G \text{ AND } k_2 \sigma_G \leq \sigma_{s_{xy}} \leq k_1 m \sigma_G \\ f(x,y) & , \text{otherwise} \end{cases}$$

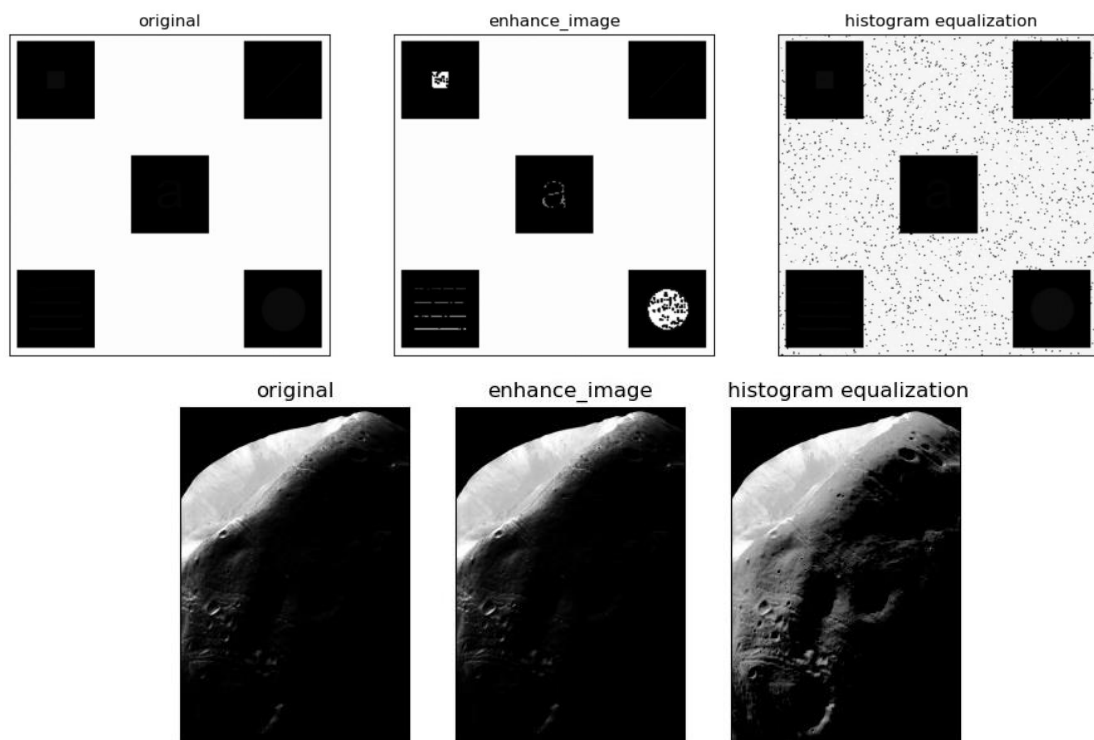
將對像素做運算並儲存於新的陣列中。

image 4-1.jpg 中 5 個黑色區塊中都有肉眼難以辨識的圖案，對其做 `local enhancement(kernel size =3)` 後可以發現 4 個區塊都浮現出明顯的圖案，唯有右上角的方框效果較不佳，可見 `local enhancement` 可以有效的增強圖像中特定區域或特徵。



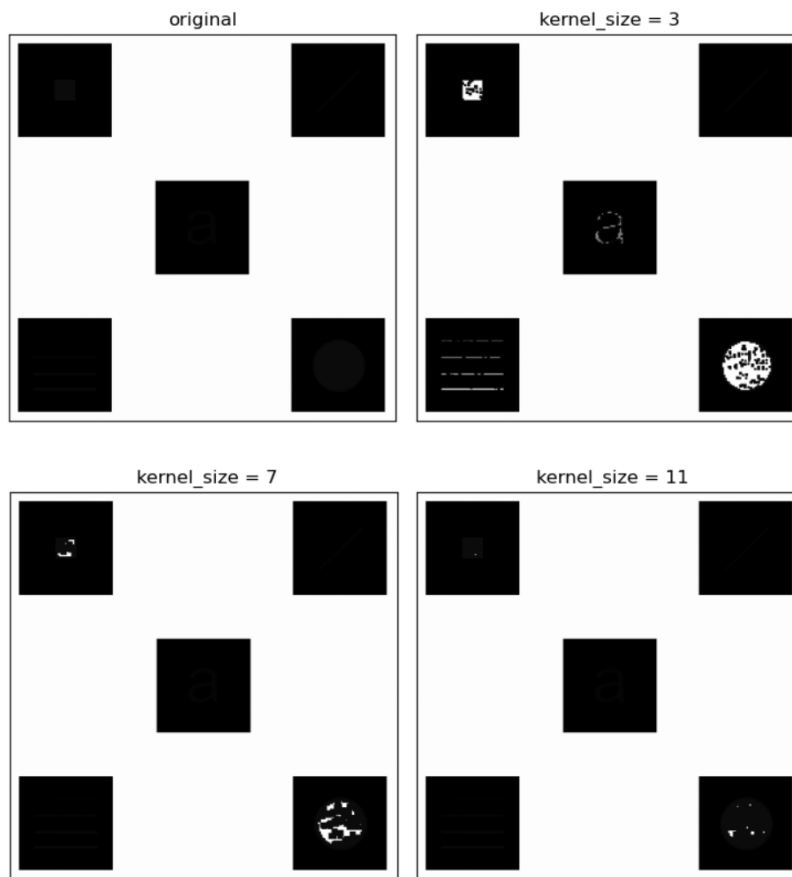
Process the original images (image 4-1.jpg, image 4-2.jpg) with the local enhancement method and compare the results with the processed images using histogram equalization method.

Image 4-1 是想增強影像中特定區域(圖中的黑色區塊)，因此利用 Local Enhancement 效果會較好，而 Image 4-2 是因為整體的像素值未均勻分佈在整個動態範圍(整體像素值偏低)，因此利用 Histogram Equalization 可以將其動態範圍拉大，讓圖片有更大的對比度以及提升圖片細節。總而言之，當需要增強圖像中特定區域或特徵時，使用區域增強;當整個圖像的對比度不佳或像素分佈不均勻，並且希望改善整體可見性和對比度時，使用直方圖均衡會是最佳的選擇。



Discuss the effect of neighborhood region size, S_{xy} , on the local enhancement image processing results.

根據下圖發現當 $S_{xy} = 3 \times 3$ 時有最佳的表現，因較小的 S_{xy} 能夠考慮較小尺度的細節與特徵，因此對於增強局部紋理和細微結構有較好的表現。而利用 $S_{xy} = 7$ 會發現相較於 $S_{xy} = 3$ 的結果，無法還原左下角方框的細直線紋路，顯示較大的 S_{xy} 對於小特徵的處理會較差。因此對於不同的影像會需要不同的 S_{xy} ，有一些局部增強技術，如自適應直方圖均衡 (AHE) 和有限對比度的自適應直方圖均衡 (CLAHE)，使用自適應遮罩大小。能夠基於局部圖像特性變化 S_{xy} ，對於有不同紋理和光照的圖像非常有用。



● UI 部分說明

Part2 :

Part2 Part3 Part4 **kernel size較大和使用彩色影像需要較長之運算時間，請稍等謝謝**

input image

box kernel size = * generate

填入 kernel 大小，ex : 3 UI 自動填入 kernel 大

填入係數
3x3 kernel coefficient:
 generate

kernel: Lowpass Gaussian Filter Kern generate

original result

Part4 :

Part2 Part3 Part4 **kernel size較大和使用彩色影像需要較長之運算時間，請稍等謝謝**

input image region size, Sxy * generate

填入區域大小，ex : 3 UI 自動填入 kernel 大