

3.22

(a)

$$V = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, W = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

$$VW^T = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 2 & 6 \\ 2 & 1 & 3 \end{bmatrix}$$

Yes, the kernel VW^T is separable because 2-D kernel VW^T can be obtained by multiplying V and W^T

(b)

$$W = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 6 & 2 \end{bmatrix}, W = W_1 * W_2$$

$$\text{Let } E=1, C=\begin{bmatrix} 1 \\ 2 \end{bmatrix}, R=\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$$

$$\text{Then } \begin{cases} W_1 = C = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\ W_2 = \frac{R^T}{E} = \begin{bmatrix} 1 & 3 & 1 \end{bmatrix} \end{cases}$$

3.28

kernel 1 - size: 3×3 , $\sigma_1 = 1.5$

kernel 2 - size: 5×5 , $\sigma_2 = 2$

kernel 3 - size: 7×7 , $\sigma_3 = 4$

(a)

Yes, the product or convolution of two Gaussian are Gaussian function as well

(b)

$$\sigma' = (\sigma_1 * \sigma_2) * \sigma_3$$

$$= (\sqrt{1.5^2 + 2^2}) * \sigma_3$$

$$= \sqrt{6.25 + 4^2}$$

$$= 4.7169$$

(c)

new size: 13×13

$$(3+5-1) + 7-1 = 13$$

3.44

(a)

$$3.45(a) \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \text{ inseparable } \otimes$$

$$3.45(b) \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \text{ inseparable } \otimes$$

(b)

$$3.50(b) \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ inseparable } \otimes$$

$$3.50(c) \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \text{ inseparable } \otimes$$

(c)

$$3.50(d) \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, W = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes$$

$$3.50(e) \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, W = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \otimes$$

4.3

$$\mathcal{F}[f(t) * h(t)] = F(u)H(u)$$

$$\Rightarrow f(t) * h(t) = \mathcal{F}^{-1}[F(u)H(u)]$$

$$(a) \delta(t) * \delta(t-t_0) = \mathcal{F}^{-1}[1 \cdot e^{-(j2\pi u t_0)}] \\ = \mathcal{F}^{-1}[e^{-(j2\pi u t_0)}]$$

$$= \delta(t-t_0)$$

$$(b) \delta(t-t_0) * \delta(t+t_0) = \mathcal{F}^{-1}[e^{-(j2\pi u t_0)} \cdot e^{(j2\pi u t_0)}] \\ = \mathcal{F}^{-1}[1]$$

$$= \delta(t)$$

4.51

$$\nabla^2 f(t, z) = \frac{\partial^2 f(t, z)}{\partial t^2} + \frac{\partial^2 f(t, z)}{\partial z^2}$$

$$\mathcal{F}[\nabla^2 f(t, z)] = -4\pi^2(u^2 + v^2) F(u, v) = H(u, v) F(u, v)$$

$$\Rightarrow \underline{H(u, v) = -4\pi^2(u^2 + v^2)} \Rightarrow$$

Part 2:

此次作業目標是設計一介面，供使用者對輸入的 jpg 影像進行遮罩運算、邊緣檢測與區域強化等處理。

一、程式介紹

本次作業共有三個小題。第一小題為遮罩運算，由使用者選擇遮罩大小與遮罩係數，並將該遮罩與輸入影像進行捲積運算（圖一）。其中，進行捲積前，必須視遮罩大小對輸入影像進行相對應的 padding。本次作業使用 mirror-padding 對輸入影像進行預處理（圖二），使得影像邊界較貼近真實而非黑框。第二小題使用 Marr-Hildreth 與 Sobel 兩種邊緣檢測方法對輸入影像進行處理。其中，Marr-Hildreth 係先以 Gaussian filter 濾除輸入影像之高頻，再以 Laplacian mask 進行邊緣檢測，最後用 zero-crossing threshold 強化其邊緣線條。第三小題則事先算出輸入影像之平均值與標準差，再對輸入影像進行區域平均數與標準差之運算，最後比較所對應像素之區域平均數與標準差是否符合預設範圍，若符合則將該像素之像素值乘以一常數進行強化，反之則維持原像素值。此外，也加入了一些限制讓程式更為完善，例如輸入影像未以 Laplacian mask 進行邊緣檢測前，點按 zero-crossing 功能鍵將跳出“Please do the Laplacian of Gaussian (LoG) operators first” 警語，避免使用者因操作順序錯誤無法獲得預期結果。

```
int g = 0;
double weight = 0;

for(int j = 0; j < maskindex*2+1; j++)
{
    for(int i = 0; i < maskindex*2+1; i++)
    {
        weight += mask[i][j];
    }
}

outputimage = new QImage(512,512,QImage::Format_RGB32);

for(int j = maskindex; j < 512+maskindex; j++)
{
    for(int i = maskindex; i < 512+maskindex; i++)
    {
        for(int k = 0; k < maskindex*2+1; k++)
        {
            for(int l = 0; l < maskindex*2+1; l++)
            {
                g += round(mask[l][k]*array_g[i-maskindex+l][j-maskindex+k]/weight);
            }
        }

        if (g > 255)
            g = 255;
        if(g < 0)
            g = 0;

        outputimage->setPixel(i-maskindex, j-maskindex, qRgb(g,g,g));

        g = 0;
    }
}
```

圖一、捲積運算

```

//上
for(int j = 0; j < maskindex; j++)
{
    for(int i = 0; i < 512; i++)
    {
        QRgb paddedimagergb = propimage.pixel(i,j);
        paddedimage.setPixel(maskindex+i, maskindex-1-j, paddedimagergb);
    }
}

//下
for(int j = 512-maskindex; j < 512; j++)
{
    for(int i = 0; i < 512; i++)
    {
        QRgb paddedimagergb = propimage.pixel(i,j);
        paddedimage.setPixel(maskindex+i, 512+2*maskindex-1-i, paddedimagergb);
    }
}

//左
for(int j = 0; j < 512+2*maskindex; j++)
{
    int k=1;
    for(int i = maskindex; i <= maskindex*2-1 ; i++)
    {
        QRgb paddedimagergb = paddedimage.pixel(i,j);
        paddedimage.setPixel(maskindex - k, j, paddedimagergb);
        k++;
    }
}

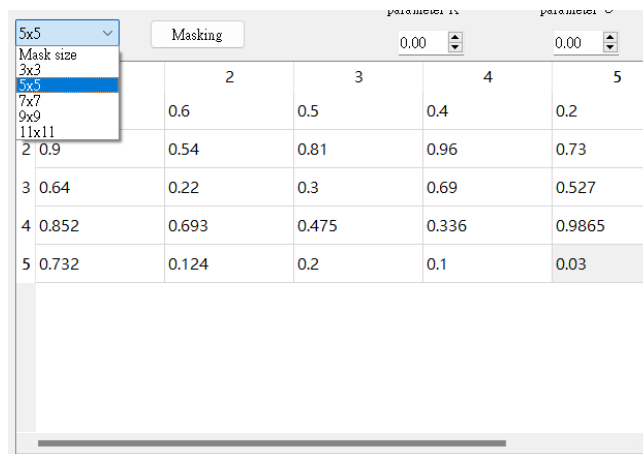
//右
for(int j = 0; j < 512+2*maskindex; j++)
{
    int m=0;
    for(int i = 1079; i <= 512+maskindex-1 ; i++)
    {
        QRgb paddedimagergb = paddedimage.pixel(i,j);
        paddedimage.setPixel(512+maskindex*2-1-m, j, paddedimagergb);
        m++;
    }
}

```

圖二、mirror-padding

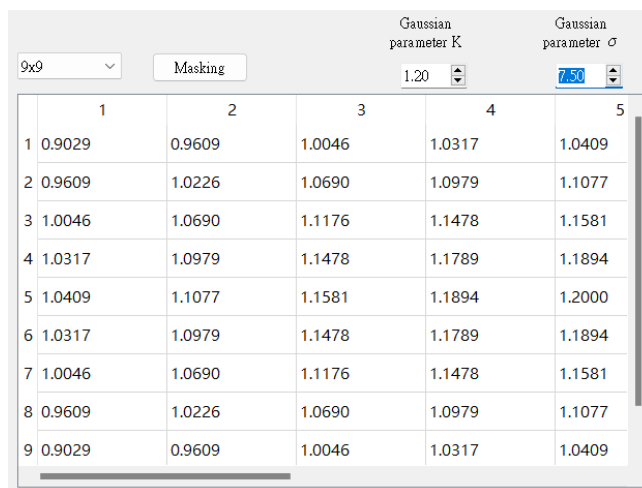
二、UI 介紹

1. 選擇輸入影像後，可於介面右上方選擇欲使用之遮罩大小。選擇後下方顯示框將會產出表格，於表格內輸入遮罩係數（未輸入之儲存格將預設係數值為 0）後按下“Masking”按鈕即可對輸入影像進行遮罩運算，並將輸出影像顯示於介面左下角（圖三）。



圖三

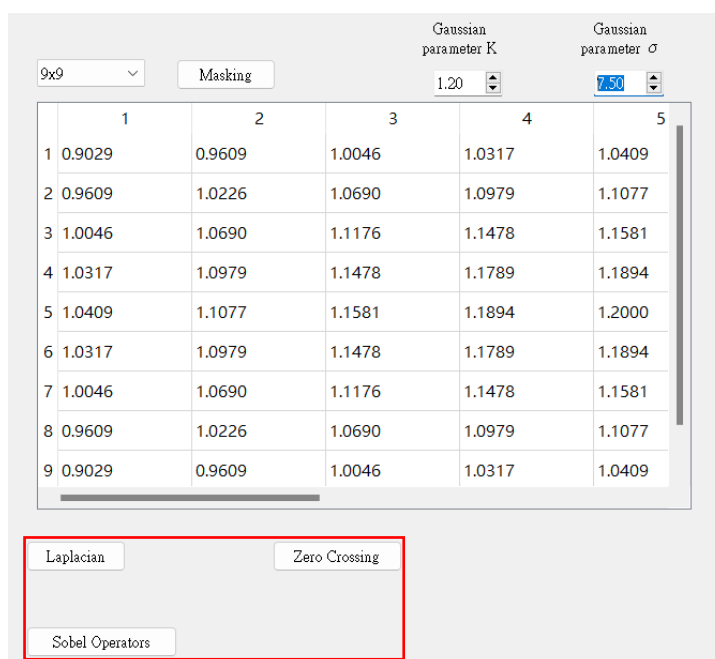
- 選定遮罩大小後，於“Masking”按鈕旁調整左右兩個格子中的數值可分別對 Gaussian filter 的 K 值與標準差 σ 進行設定，其範圍分別為-2~2 以及 0~10。下方之顯示框會同步顯示當前參數下的遮罩係數。按下“Masking”按鈕將可對輸入影像進行 Gaussian filter 捲積運算，並將輸出影像顯示於介面左下角（圖四）。



	1	2	3	4	5
1	0.9029	0.9609	1.0046	1.0317	1.0409
2	0.9609	1.0226	1.0690	1.0979	1.1077
3	1.0046	1.0690	1.1176	1.1478	1.1581
4	1.0317	1.0979	1.1478	1.1789	1.1894
5	1.0409	1.1077	1.1581	1.1894	1.2000
6	1.0317	1.0979	1.1478	1.1789	1.1894
7	1.0046	1.0690	1.1176	1.1478	1.1581
8	0.9609	1.0226	1.0690	1.0979	1.1077
9	0.9029	0.9609	1.0046	1.0317	1.0409

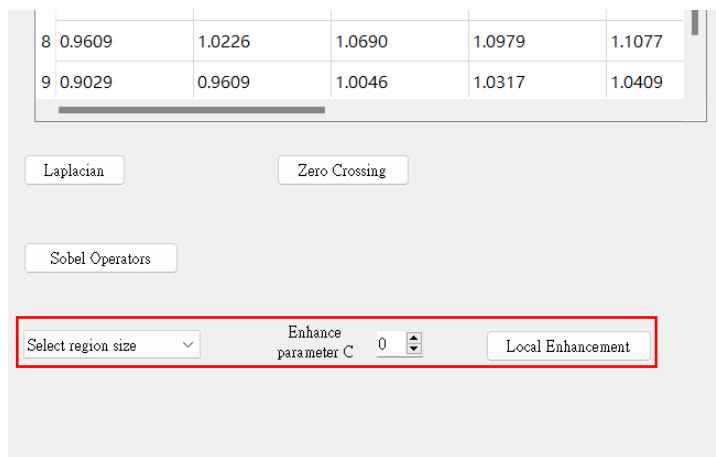
圖四

- 點按介面右下方的“Laplacian”按鈕可對輸出影像（若尚無輸出影像則對輸入影像）進行 Laplacian 遮罩運算。點按“Zero Crossing”按鈕對經 Laplacian 運算後的輸出影像進行 zero-crossing threshold 處理。點按“Sobel Operators”則對輸入影像進行 Sobel 邊緣檢測（圖五）。



圖五

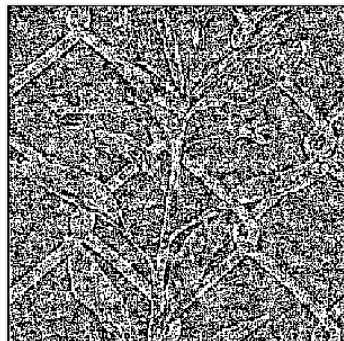
- 在介面下方“Select region size”下拉選單可選擇區域強化（local enhancement）的區域大小，調整一旁的儲存格“Enhance parameter C”可設定強化強度（-50 ~ 50），最後按下“Local Enhancement”按鈕即可對輸入影像進行區域強化（圖六）。



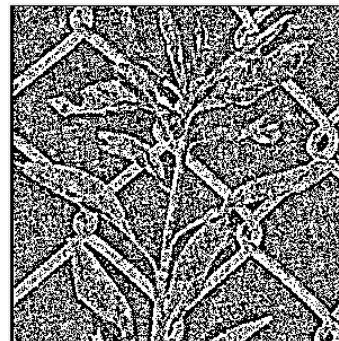
圖六

三、結果與討論

- 若輸入影像未先以 Gaussian filter 濾除高頻，即使用 Laplacian mask 進行邊緣檢測，則輸出影像之雜訊較多，邊緣檢測較差（圖七）。



未使用 Gaussian filter 濾除高頻



先使用 Gaussian filter 濾除高頻

圖七

- Gaussian filter 在相同參數（ K 、 σ ）下，遮罩大小愈大，則輸出影像愈模糊（圖八）



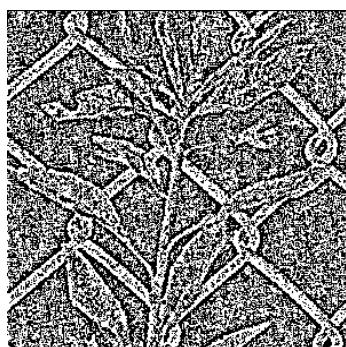
Mask size: 3x3 ; $K=9.00$; $\sigma=1.2$



Mask size: 11x11 ; $K=9.00$; $\sigma=1.2$

圖八

3. Marr-Hildreth 邊緣檢測法需視輸入影像狀況，調整適當參數（Gaussian filter 之 mask size、 K 、 σ 等），方能有效濾除輸入影像之噪點，進行檢測。然而，Sobel 邊緣檢測法先分別對輸入影像之橫向與縱向進行邊界檢測，再將二者之結果結合，能有效檢測出輸入影像之連續線條（圖九）。



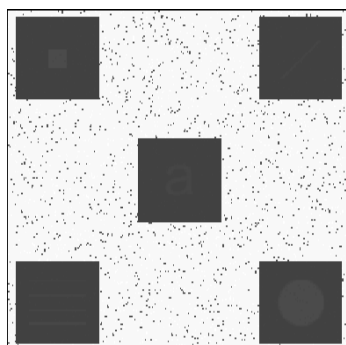
Marr-Hildreth 邊緣檢測法



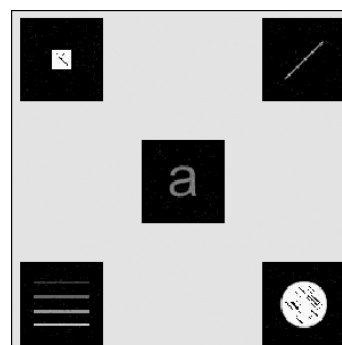
Sobel 邊緣檢測法

圖九

4. zero-crossing threshold 在 Marr-Hildreth 邊緣檢測法中的作用係將檢測出的邊界部分進行強化以利觀察。
5. 雖然使用 histogram equalization 亦可顯示出輸入影像中低對比度區域之圖案，但同時也會改變影像其他區域的灰階值，顯示強度也未優於 local enhancement（圖十）。



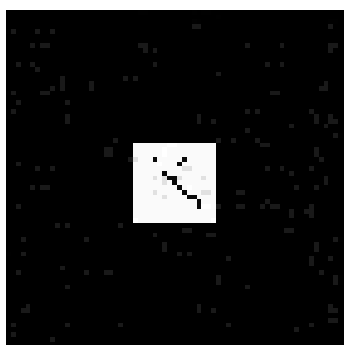
Histogram equalization



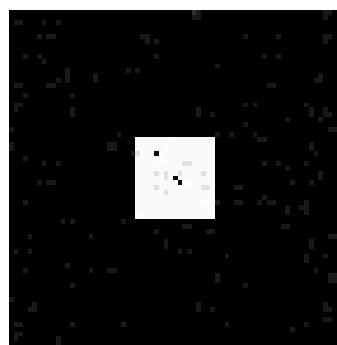
Local enhancement

圖十

6. 區域強化的範圍（region）愈小，顯示低對比度區域圖案之效過愈好（圖十一）。



Region size: 11x11



Region size: 3x3

圖十一