Principle and Applications of Digital Image Processing

Homework 3 Report 林東甫 R12631055

Part 1:

3.22

(a) Ans:

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

2nd row [4 2 2 6] = 2 × 1st row [2 1 1 3] = 2 × 3rd [2 1 1 3]

Yes, the kernel is separable.

(b) Ans: considering (a):

$w = \begin{bmatrix}1 & 3 & 1\\2 & 6 & 2\end{bmatrix}$ could be separated into

$W_1 = \begin{bmatrix}1\\2\end{bmatrix}$, $W_2 = \begin{bmatrix}1\\3\\1\end{bmatrix}^T$. $W = W_1 \star W_2$

3.27

(a) Ans: Every time use a 3×3 filter, for 4 times:

$4 = 2^2 \Rightarrow 3^n \times 3^n \quad n = 2 \quad \therefore \underline{9 \times 9}$ #

(b) Ans: Known standard deviation $\sigma = 1$ in size 3×3:

$\sigma' = \sqrt{n \cdot 1^2 \cdot n \cdot 1^2} = \sqrt{4} = \underline{2}$ #

3.38

Ans: If the Laplacian kernel is applied first, the nosie of whole ~~picture~~ image will become significant, however applying smoothing kernel first will make noise smoother, so the answer is <u>negative</u>, they are not equal. #

4.3

(a) Ans: $\delta(t) * \delta(t-t_0)$

$= \int_{-\infty}^{\infty} \delta(x)\delta(t-t_0-x)\,dx$ two impulses have value

when $x=0$, $t-t0$ $\therefore$ $\underline{\delta(t-t_0)}$. #

(b) Ans: $\delta(t-t_0) * \delta(t+t_0)$

$= \int_{-\infty}^{\infty} \delta(x+t_0)\delta(t-t_0-x)\,dx$ two impulses have value

when $x=-t_0, t-t_0$, thus the result is $\underline{\delta(t)}$ #

4.32

(a) Ans: $\{0, 0, a, b, c, c, b, 0, 0\}$ symmetric

as the same center "c" : coinciding.

(b) Ans: $\{0, 0, 0, -b, -c, 0, c, b, 0, 0\}$

symmetric

(c) Ans: $\{0, 0, a, b, c, d, c, b, 0, 0\}$

(d) Ans: $\{0, 0, 0, -b, -c, c, b, 0, 0\}$

Part 2:

總共使用了五種類型的mask，mask size可調整3x3、5x5和7x7，在此先介紹其中的Box filter和Gaussian filter，另外三種mask會在後續中提到。

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

1.Box filter: 單純的mask，將周遭數值全採計做平均數：

```
m0 = m0 + img.at<Vec3b>(i+k,j+l)[0]/p2;
m1 = m1 + img.at<Vec3b>(i+k,j+l)[1]/p2;
m2 = m2 + img.at<Vec3b>(i+k,j+l)[2]/p2;
```

從左至右分別是使用不同的filter size:3x3、5x5和7x7，可以看得出很明顯的模糊化效果，同時也可以看得出來filter size越大，整體影像也更為模糊平滑。隨著mask size增加，計算時間也略微提升，考慮到計算量增加，頗為合理。

2.Gaussian filter:

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

程式實作如下，先設sigma = 1：

```
void Filter3(double GKernel[][3])
{
    // initialising standard deviation to 1.0
    double sigma = 1.0;
    double r, s = 2.0 * sigma * sigma;

    // sum is for normalization
    double sum = 0.0;

    // generating 5x5 kernel
    for (int x = -1; x <= 1; x++) {
        for (int y = -1; y <= 1; y++) {
            r = sqrt(x * x + y * y);
            GKernel[x + 1][y + 1] = (exp(-(r * r) / s)) / (M_PI * s);
            sum += GKernel[x + 1][y + 1];
        }
    }

    // normalising the Kernel
    for (int i = 0; i < 3; ++i)
        for (int j = 0; j < 3; ++j)
            GKernel[i][j] /= sum;
}
```

模糊化的效果相較於box filter要自然得多，filter size的增加雖然也增加了模糊的效果，同時也略為增加計算量，但相較於Box filter的模糊效果並沒有那麼粗暴劇烈，也仍未喪失自然平滑的感覺。

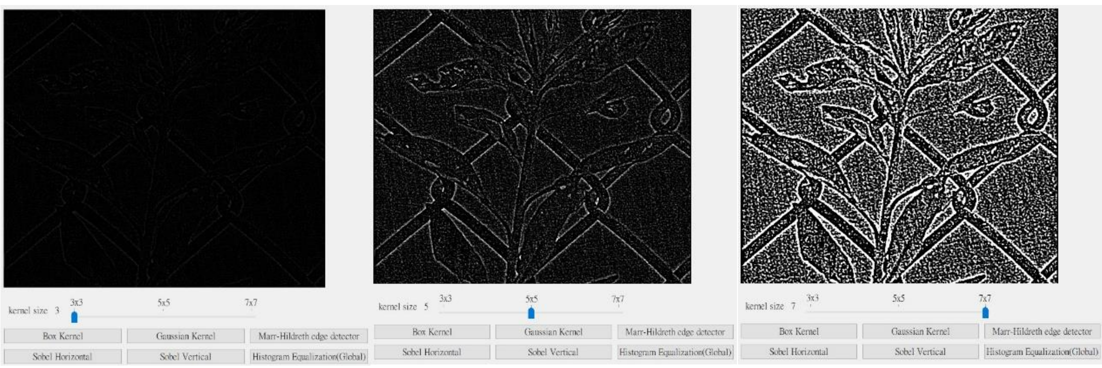
因為sigma值設定在1，因此整體的模糊化效果也比較弱，若將sigma調整較大，則整體輸出影像也會較為模糊，效果有如透過一半透明螢幕來觀察影像。

Part 3

Laplacian of Gaussian operators 程式實作如下：

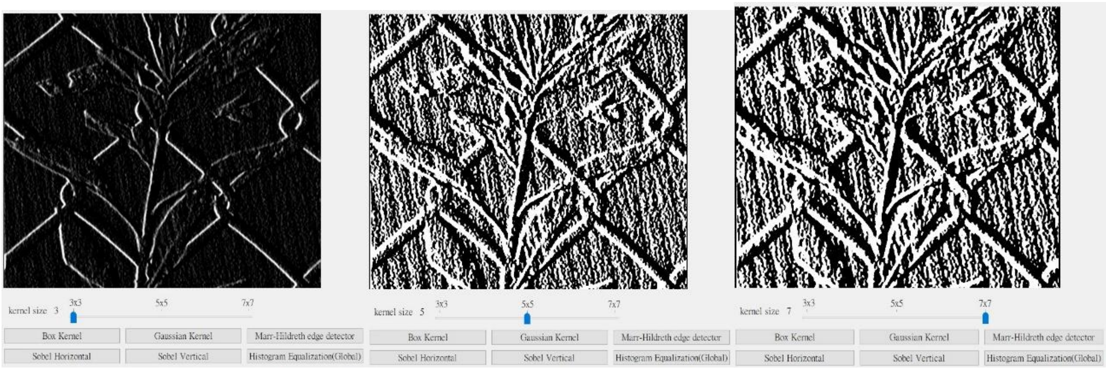
```
void LOGFilter3(double GKernel[][3])
{
    // initialising standard deviation to 1.0
    double sigma = 1.0;
    double r, s = 2.0 * sigma * sigma;

    // sum is for normalization
    double sum = 0.0;

    // generating 5x5 kernel
    for (int x = -1; x <= 1; x++) {
        for (int y = -1; y <= 1; y++) {
            r = sqrt(x * x + y * y);
            GKernel[x + 1][y + 1] = (1 - ((r * r) / s)) * (exp(-(r * r) / s)) * (1/((-M_PI) * pow(sigma, 4)));
            sum += GKernel[x + 1][y + 1];
        }
    }

    // normalising the Kernel
    for (int i = 0; i < 3; ++i)
        for (int j = 0; j < 3; ++j)
            GKernel[i][j] /= sum;
}
```

通常在色塊區域內的灰階值是相近的，但在不同兩個區域交界處的灰階值會出現很大的變化，因此邊緣地帶在一次微分後的會有波峰的特性，在二次微分時就會出現Zero-crossing(二次微分值在邊緣交界處與0相交)，利用這個特性就可以做為threshold的設定來做邊緣偵測，因此Marr-Hildreth使用了Laplacian mask，只需找Laplacian operators一正一負時的差超過設定的threshold(>0)，就能知道該處是否為邊緣。以下與sobel mask進行比較：

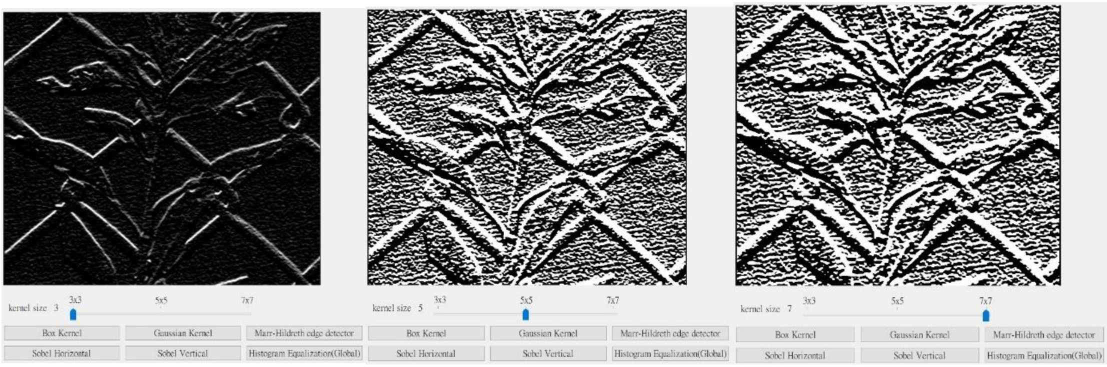


以上分別是3x3、5x5和7x7的處理效果。

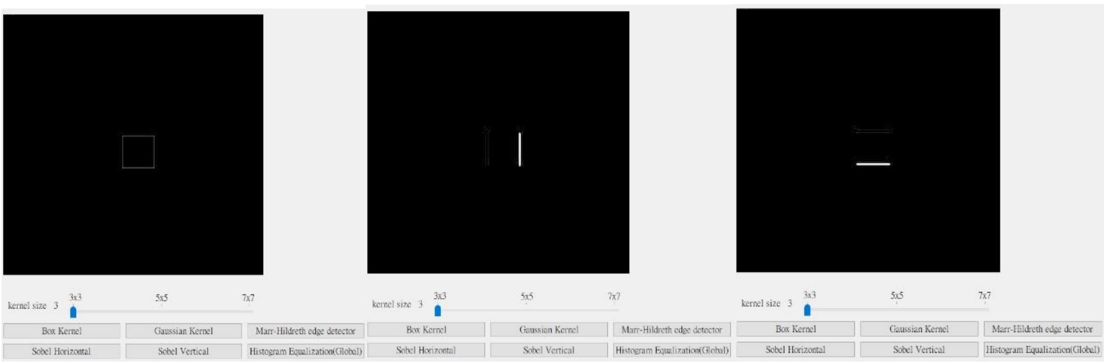
作為part2的補充與邊緣偵測的比較，這邊實作了Sobel filter:

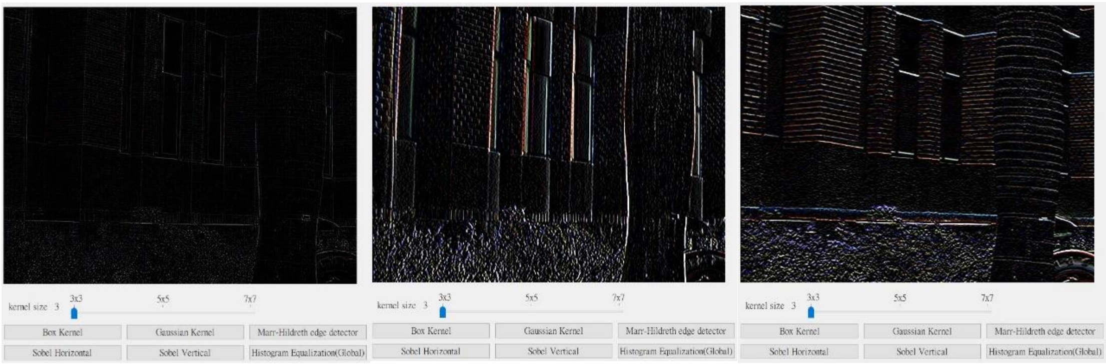


上面是Sobel Horizontal，而下方則是Sobel Vertical，filter size:3x3,5x5,7x7



接著是在不同影像上的比較，由左至右分別是Marr-Hildreth,Sobel.H,Sobel.V:





Part4:

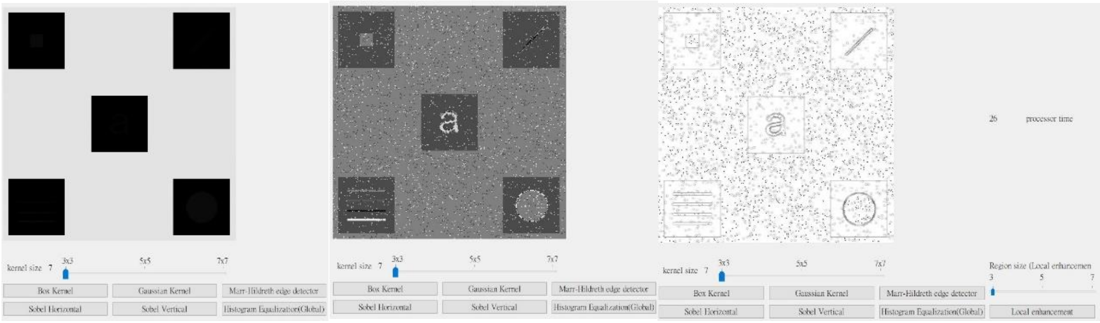
使用上次作業(HW2)所使用的亮度與對比調整(a=100,b=30)便可將影像強化。

左)原始影像　　右)增強後的影像

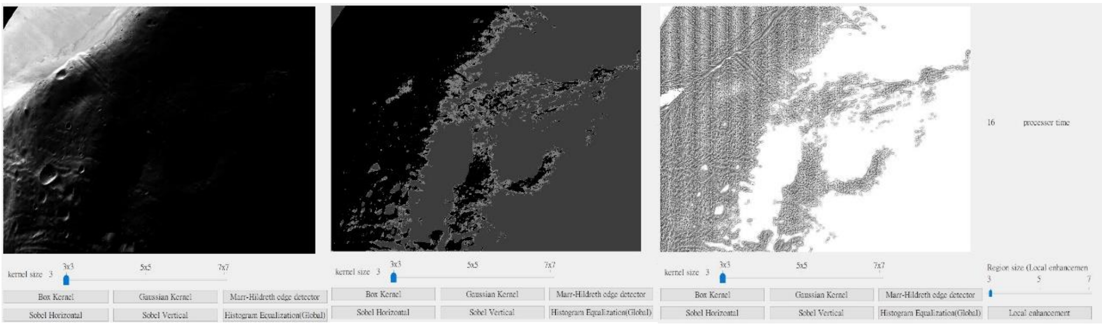
(HW2.exe將一併附於code資料夾中)

接著實作local enhancement的程式如下:

```
index_hist = data_in[v * w + u];
indexs_hist.push_back(index_hist);
for(y = v - halfsize;y <= v + halfsize;y++)
    for(x = u - halfsize;x <= u + halfsize;x++)
    {
        index_hist = data_in[y * w + x];
        size_index = indexs_hist.size();

        if(index_hist == indexs_hist[size_index - 1] || index_hist==indexs_hist[0])
            ;
        else if(index_hist > indexs_hist[size_index - 1])
            indexs_hist.push_back(index_hist);
        else if(index_hist < indexs_hist[0])
            indexs_hist.insert(begin(indexs_hist), index_hist);
        else
        {
            for(i = 1; i < size_index; i++)
                if(index_hist == indexs_hist[i])
                    break;
                else if(index_hist < indexs_hist[i] && index_hist > indexs_hist[i-1])
                {
                    indexs_hist.insert(begin(indexs_hist)+i, index_hist);
                    break;
                }
        }
        hist[index_hist]++;
    }

for(i=0; i < (int)indexs_hist.size();i++)
    for(j=0; j <= i;j++)
```



由左至右分別是原始影像、histogram equalization以及local enhancement



可以看得出histogram equalization這樣的方法雖對於提升整張影像的色階對比與清晰有很大的效果，但是某些局部影像並沒有得到太多重視，特別是有些圖片可能比較大的範圍是較高或較低的色階，然而局部卻有很重要的訊息或者局部的細節豐富因為色階相近而難以分辨，而做全影像的均值化又會被大面積的色塊平均掉，因此值得特別去針對某些區域的細節去做對比強化或者均化。

而使用的region size也是有差異,由左至右的region size分別是3 , 5, 7:

可以看出，較小的region在細微處有很多鋸齒狀使得許多細小的結構特徵跟紋理細節相當突出，與之相反，較大的Region影像則看似更為平滑柔順一些。