Homework 10　　資工所碩一 R05922068 彭宇劭

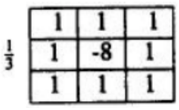
Write the following programs to detect edge
(Zero-crossing on the following four types of images to get edge images)

- Laplacian
- Minimum-variance Laplacian
- Laplacian of Gaussian
- Difference of Gaussian

Source code: hw10.py

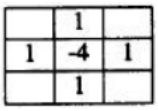
執行方式：python hw10.py

版本：Python 2.7.10

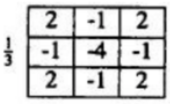


$\frac{1}{3}$

1	1	1
1	-8	1
1	1	1



	1	
1	-4	1
	1	



$\frac{1}{3}$

2	-1	2
-1	-4	-1
2	-1	2

Output(bmp folder)：

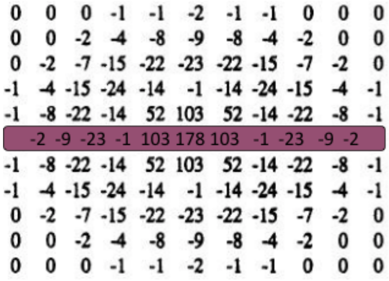
lena_laplacian_h.bmp —> Laplacian (kernel:上左 threshold: 15)

lena_laplacian_l.bmp —> Laplacian (kernel:上中 threshold: 15)

lena_min_var_laplacian.bmp —> minimum-variance digital Laplacian (kernel:上右 threshold: 20)

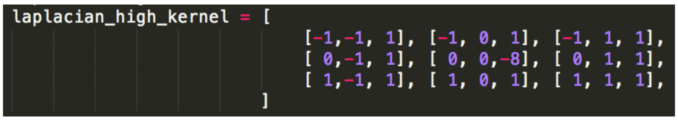
lena_gaussian_laplacian.bmp —> Laplacian of the Gaussian (kernel:下左 threshold: 3000)

lena_DoG.bmp —> Difference of the Gaussian (kernel:下右 threshold: 7000)



0	0	0	-1	-1	-2	-1	-1	0	0	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	-2	-7	-15	-22	-23	-22	-15	-7	-2	0
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-2	-9	-23	-1	103	178	103	-1	-23	-9	-2
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
0	-2	-7	-15	-22	-23	-22	-15	-7	-2	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	0	0	-1	-1	-2	-1	-1	0	0	0

$$\frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

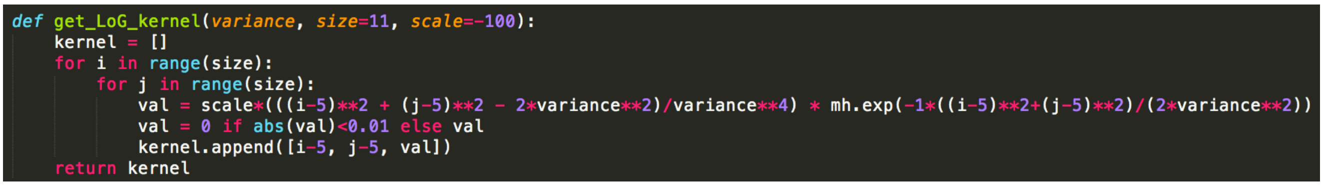


```
laplacian_high_kernel = [
                            [-1,-1, 1], [-1, 0, 1], [-1, 1, 1],
                            [ 0,-1, 1], [ 0, 0,-8], [ 0, 1, 1],
                            [ 1,-1, 1], [ 1, 0, 1], [ 1, 1, 1],
                        ]
```

簡述：

1. Define Kernels

在main function中定義 kernels偏移量、大小權重，以及Threshold
kernels中有三個元素 [x偏移量, y偏移量, 權重]
Laplacian, minimum-variance digital Laplacian, Laplacian of the Gaussian直接定義在main function中（如右上），而DoG的兩個kernel(var=1, var=3) 則由程式依據上述公式產生，產生後再做scale的動作，程式部分如下。



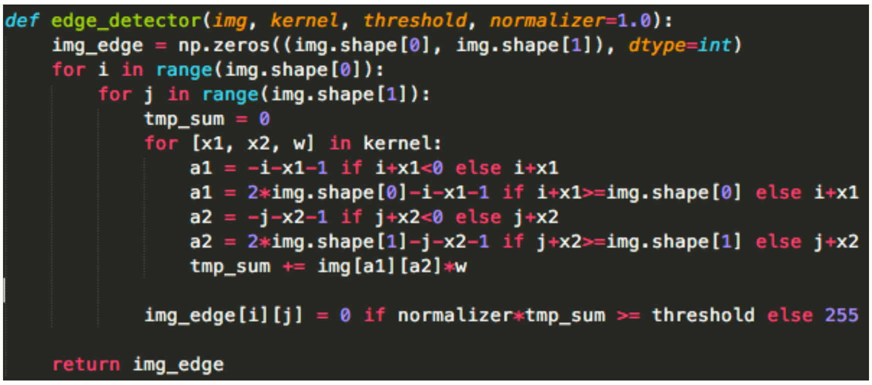
```
def get_LoG_kernel(variance, size=11, scale=-100):
    kernel = []
    for i in range(size):
        for j in range(size):
            val = scale*(((i-5)**2 + (j-5)**2 - 2*variance**2)/variance**4) * mh.exp(-1*((i-5)**2+(j-5)**2)/(2*variance**2))
            val = 0 if abs(val)<0.01 else val
            kernel.append([i-5, j-5, val])
    return kernel
```

2. Edge Detector

接著這邊寫兩個不同方法的Edge Detector，分別為：

edge_detector(img, kernel, threshold, normalizer)

這個function除了img外，需要kernel、threshold及normalizer，直接將此kernel滾過整張img，超出邊界的部分用鏡像表示該值，經過kernel及該點的灰階值相乘相加後產生一數值帶表該pixel的點，若此值超過threshold則此點設為0 (edge)，反之，設為255



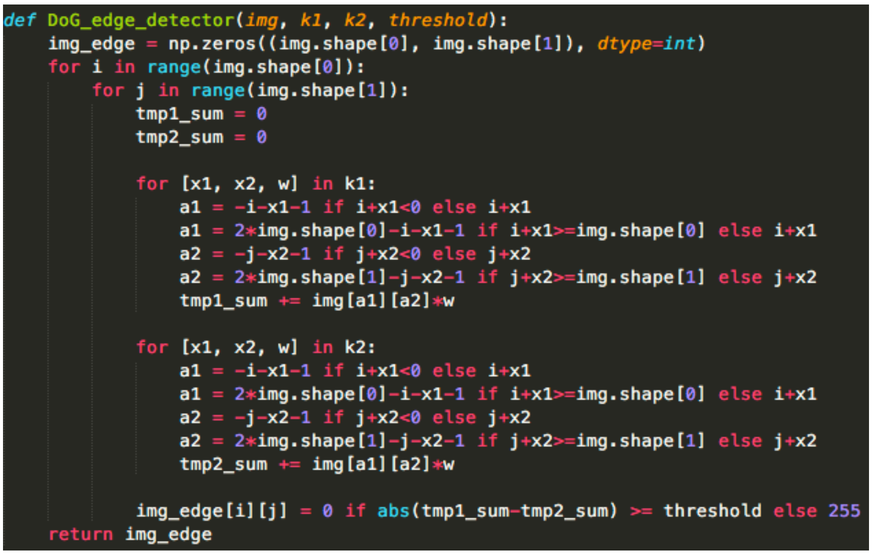
```python
def edge_detector(img, kernel, threshold, normalizer=1.0):
    img_edge = np.zeros((img.shape[0], img.shape[1]), dtype=int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            tmp_sum = 0
            for [x1, x2, w] in kernel:
                a1 = -i-x1-1 if i+x1<0 else i+x1
                a1 = 2*img.shape[0]-i-x1-1 if i+x1>=img.shape[0] else i+x1
                a2 = -j-x2-1 if j+x2<0 else j+x2
                a2 = 2*img.shape[1]-j-x2-1 if j+x2>=img.shape[1] else j+x2
                tmp_sum += img[a1][a2]*w

            img_edge[i][j] = 0 if normalizer*tmp_sum >= threshold else 255

    return img_edge
```

DoG_edge_detector(img, k1, k2, threshold)

大部分操作跟上面的function差不多，不過這邊需要兩個kernel分別計算兩個kernel對每個pixel所產生的值，將這兩個值相減取絕對值（代表兩個不同Gaussian 產生的kernel計算出值得差），若這個差大於等於Threshold則此點設為0 (edge)，反之，設為255。



```python
def DoG_edge_detector(img, k1, k2, threshold):
    img_edge = np.zeros((img.shape[0], img.shape[1]), dtype=int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            tmp1_sum = 0
            tmp2_sum = 0

            for [x1, x2, w] in k1:
                a1 = -i-x1-1 if i+x1<0 else i+x1
                a1 = 2*img.shape[0]-i-x1-1 if i+x1>=img.shape[0] else i+x1
                a2 = -j-x2-1 if j+x2<0 else j+x2
                a2 = 2*img.shape[1]-j-x2-1 if j+x2>=img.shape[1] else j+x2
                tmp1_sum += img[a1][a2]*w

            for [x1, x2, w] in k2:
                a1 = -i-x1-1 if i+x1<0 else i+x1
                a1 = 2*img.shape[0]-i-x1-1 if i+x1>=img.shape[0] else i+x1
                a2 = -j-x2-1 if j+x2<0 else j+x2
                a2 = 2*img.shape[1]-j-x2-1 if j+x2>=img.shape[1] else j+x2
                tmp2_sum += img[a1][a2]*w

            img_edge[i][j] = 0 if abs(tmp1_sum-tmp2_sum) >= threshold else 255
    return img_edge
```

3. Gaussian Kernel 產生方法

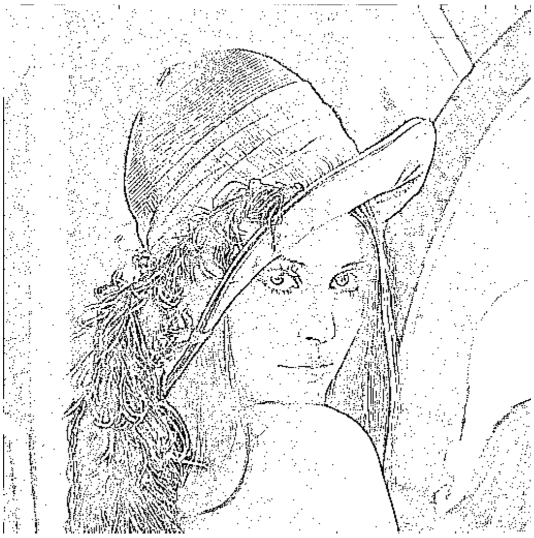
根據投影片及網路上找到的推導方式，可以知道Gaussian Kernel與row, col有以下近似關係

$$LoG \triangleq \Delta G_\sigma(x,y) = \frac{\partial^2}{\partial x^2}G_\sigma(x,y)+\frac{\partial^2}{\partial y^2}G_\sigma(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

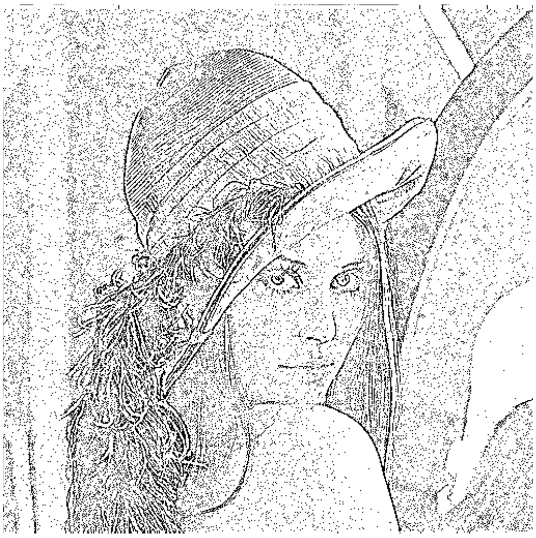
Ref: http://fourier.eng.hmc.edu/e161/lectures/gradient/node8.html

進而利用此公式及縮放產生kernel，程式部分在簡述1有提到。

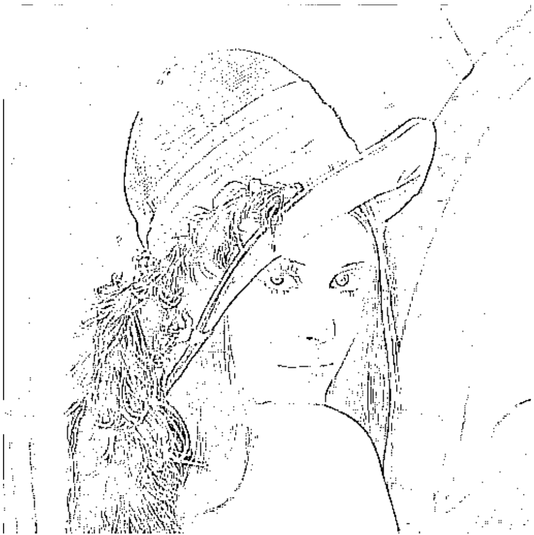
結果：



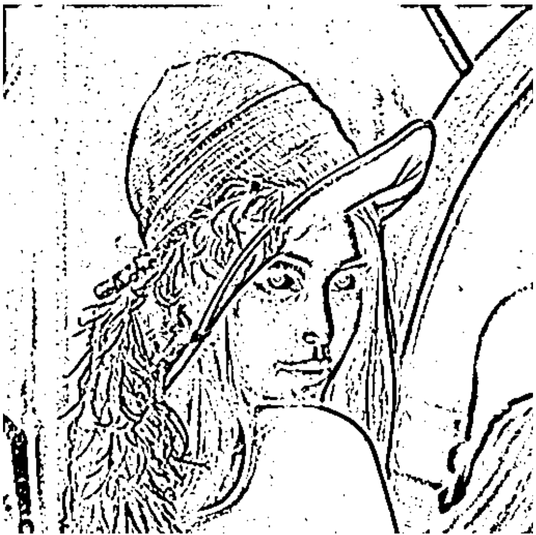
lena_laplacian_h.bmp



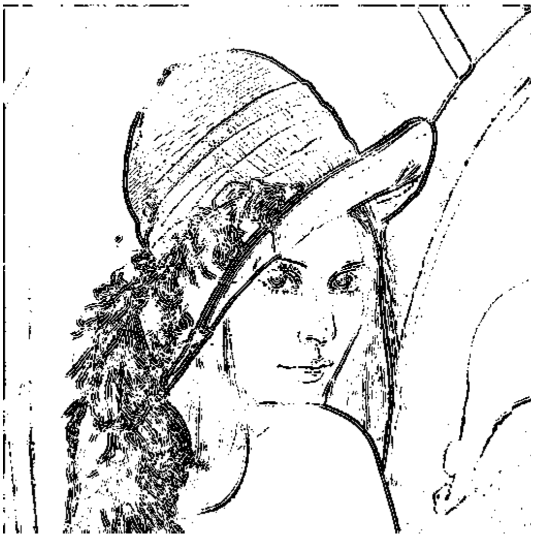
lena_laplacian_l.bmp



lena_min_var_laplacian.bmp



lena_gaussian_laplacian.bmp



lena_DoG.bmp