Homework 9 資工所碩一 R05922068 彭宇劭

Write programs to generate the following gradient magnitude images and choose proper thresholds to get the binary edge images:

- Roberts operator (threshold: 12)
- Prewitt edge detector (threshold: 24)
- Sobel edge detector (threshold: 38)
- Frei and Chen gradient operator (threshold: 30)
- Kirsch compass operator (threshold: 135)
- Robinson compass operator (threshold: 43)
- Nevatia-Babu 5X5 operator (threshold: 12500)

Source code: hw9.py

執行方式：python hw9.py

版本：Python 2.7.10

Output(bmp folder)：

lena_robert.bmp	—>	Robert's Operator: 12
lena_prewitt.bmp	—>	Prewitt's Edge Detector: 24
lena_sobel.bmp	—>	Sobel's Edge Detector: 38
lena_frei_chen.bmp	—>	Frei and Chen's Gradient Operator: 30
lena_kirsch.bmp	—>	Kirsch's Compass Operator: 135
lena_robinson.bmp	—>	Robinson's Compass Operator: 43
lena_nevatia.bmp	—>	Nevatia-Babu 5x5 Operator: 12500

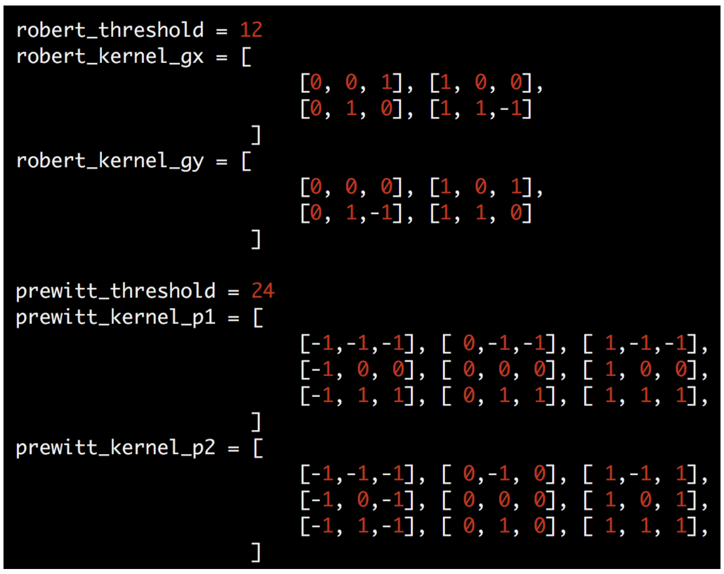
簡述：

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]

   因為有太多kernels，所以這邊拿前兩個示意圖舉例



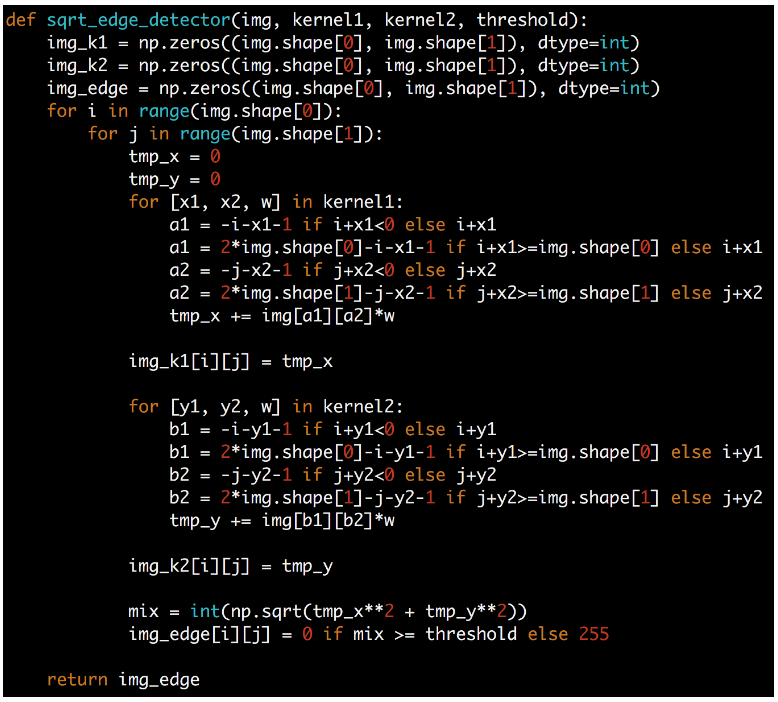
```
robert_threshold = 12
robert_kernel_gx = [
                        [0, 0, 1], [1, 0, 0],
                        [0, 1, 0], [1, 1,-1]
                    ]
robert_kernel_gy = [
                        [0, 0, 0], [1, 0, 1],
                        [0, 1,-1], [1, 1, 0]
                    ]

prewitt_threshold = 24
prewitt_kernel_p1 = [
                        [-1,-1,-1], [ 0,-1,-1], [ 1,-1,-1],
                        [-1, 0, 0], [ 0, 0, 0], [ 1, 0, 0],
                        [-1, 1, 1], [ 0, 1, 1], [ 1, 1, 1],
                    ]
prewitt_kernel_p2 = [
                        [-1,-1,-1], [ 0,-1, 0], [ 1,-1, 1],
                        [-1, 0,-1], [ 0, 0, 0], [ 1, 0, 1],
                        [-1, 1,-1], [ 0, 1, 0], [ 1, 1, 1],
                    ]
```

2. Edge Detector

接著這邊寫兩個不同方法的Edge Detector，分別為：

sqrt_edge_detector(img, kernel1, kernel2, threshold)

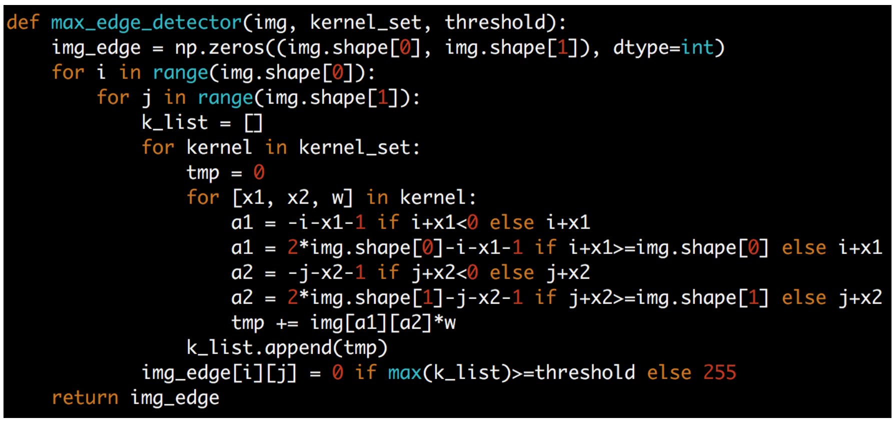
這個function除了img外，需要兩個kernel及threshold，由兩個kernel可以產生出兩個值，再將此兩個值平方相加開根號，可以產生gradient magnitude，若此值超過threshold則此點設為0 (edge)，反之，設為255



```
def sqrt_edge_detector(img, kernel1, kernel2, threshold):
    img_k1 = np.zeros((img.shape[0], img.shape[1]), dtype=int)
    img_k2 = np.zeros((img.shape[0], img.shape[1]), dtype=int)
    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_x = 0
            tmp_y = 0
            for [x1, x2, w] in kernel1:
                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_x += img[a1][a2]*w

            img_k1[i][j] = tmp_x

            for [y1, y2, w] in kernel2:
                b1 = -i-y1-1 if i+y1<0 else i+y1
                b1 = 2*img.shape[0]-i-y1-1 if i+y1>=img.shape[0] else i+y1
                b2 = -j-y2-1 if j+y2<0 else j+y2
                b2 = 2*img.shape[1]-j-y2-1 if j+y2>=img.shape[1] else j+y2
                tmp_y += img[b1][b2]*w

            img_k2[i][j] = tmp_y

            mix = int(np.sqrt(tmp_x**2 + tmp_y**2))
            img_edge[i][j] = 0 if mix >= threshold else 255

    return img_edge
```

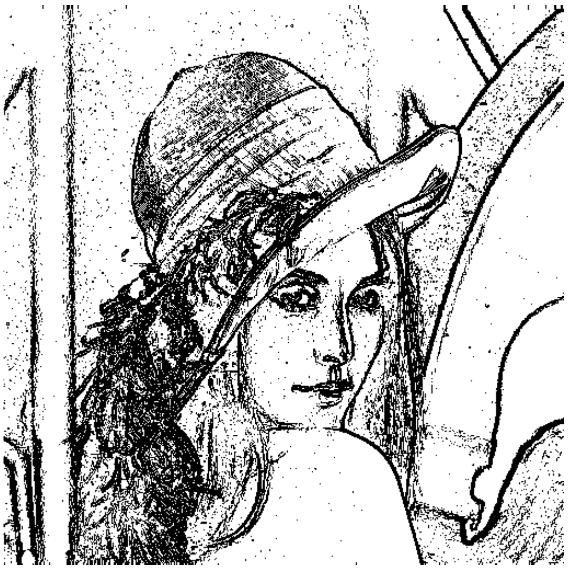
max_edge_detector(img, kernel_set, threshold)

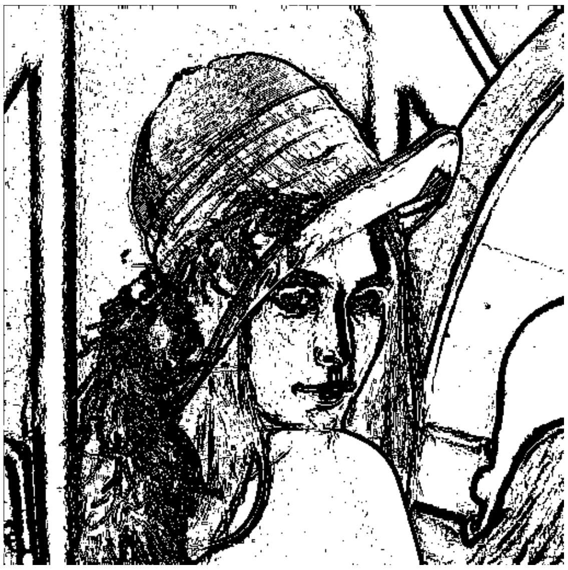
大部分操作跟上面的function差不多，不過這邊需要的kernel可能比較多，所以這邊選擇輸入kernel list，而計算gradient magnitude 的方法也不太一樣，這邊會將所有傳入的kernel都計算出一個值，比較取得最大的值設為該點的 gradient magnitude，如果此gradient magnitude超過threshold，則此點設為0 (edge)，反之，設為255，另外值得一提的是，這兩個function為了避免邊界都被設為edge，這邊會利用鏡像的方式，這樣超出邊緣的部分也可以取值，避免邊界都被判斷成edge

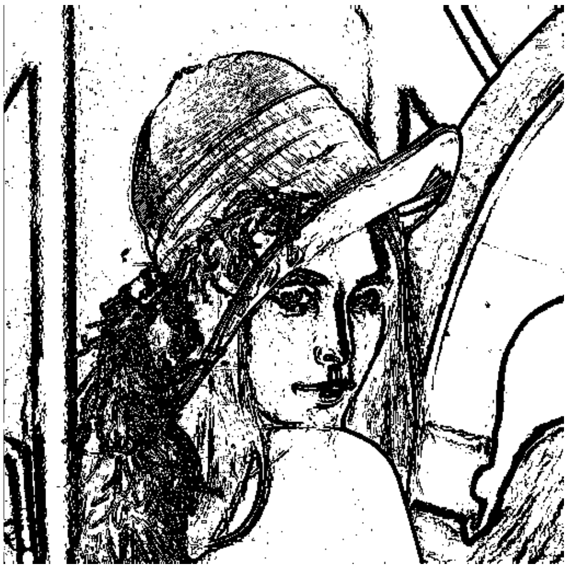


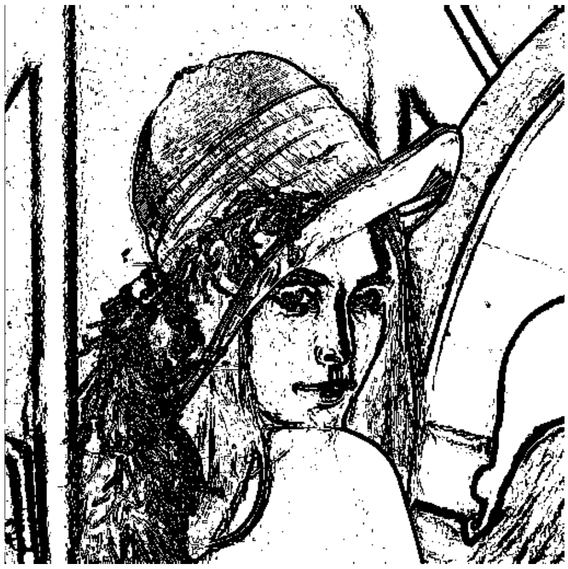
```
def max_edge_detector(img, kernel_set, threshold):
    img_edge = np.zeros((img.shape[0], img.shape[1]), dtype=int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            k_list = []
            for kernel in kernel_set:
                tmp = 0
                for [x1, 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 += img[a1][a2]*w
                k_list.append(tmp)
            img_edge[i][j] = 0 if max(k_list)>=threshold else 255
    return img_edge
```

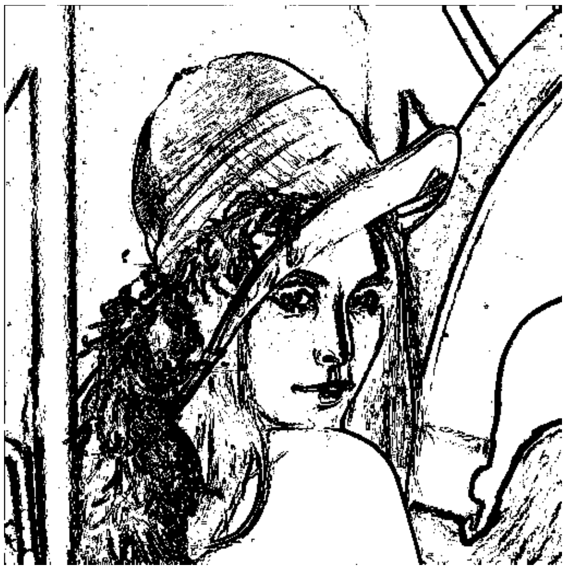
結果：

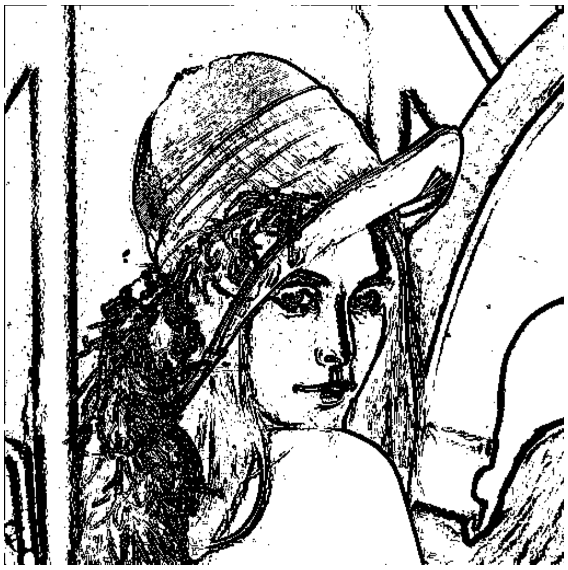

Robert's Operator: 12

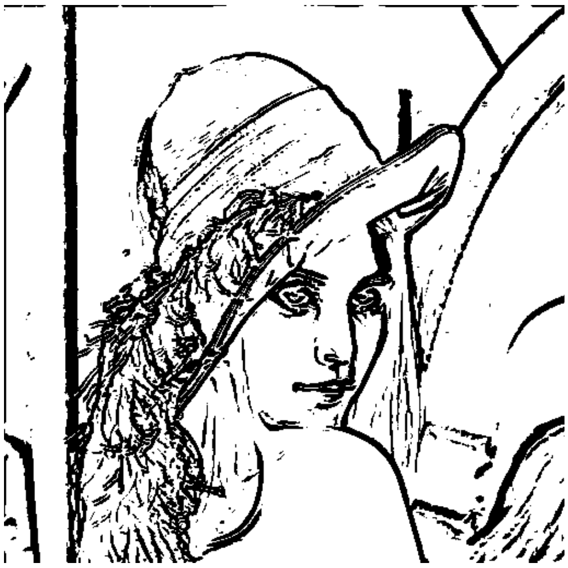

Prewitt's Edge Detector: 24


Sobel's Edge Detector: 38


Frei and Chen's Gradient Operator: 30


Kirsch's Compass Operator: 135


Robinson's Compass Operator: 43


Nevatia-Babu 5x5 Operator: 12500