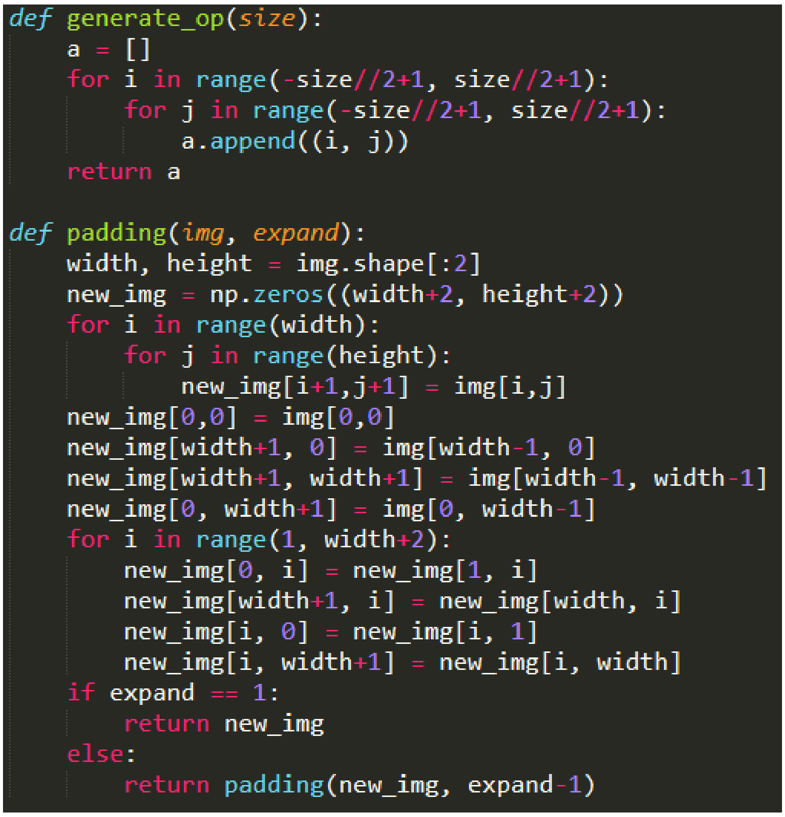
CV HW10

資工四 陳光裕 B07902072

Usage:

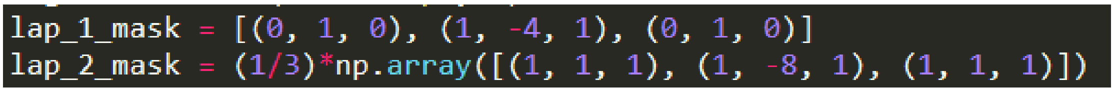
python3 main.py

產生 filter 跟 padding 這兩個 function 跟上次作業一樣：



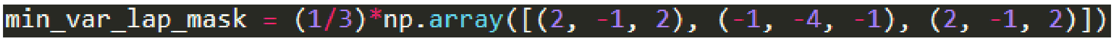
```python
def generate_op(size):
    a = []
    for i in range(-size//2+1, size//2+1):
        for j in range(-size//2+1, size//2+1):
            a.append((i, j))
    return a

def padding(img, expand):
    width, height = img.shape[:2]
    new_img = np.zeros((width+2, height+2))
    for i in range(width):
        for j in range(height):
            new_img[i+1,j+1] = img[i,j]
    new_img[0,0] = img[0,0]
    new_img[width+1, 0] = img[width-1, 0]
    new_img[width+1, width+1] = img[width-1, width-1]
    new_img[0, width+1] = img[0, width-1]
    for i in range(1, width+2):
        new_img[0, i] = new_img[1, i]
        new_img[width+1, i] = new_img[width, i]
        new_img[i, 0] = new_img[i, 1]
        new_img[i, width+1] = new_img[i, width]
    if expand == 1:
        return new_img
    else:
        return padding(new_img, expand-1)
```

Laplace Mask:



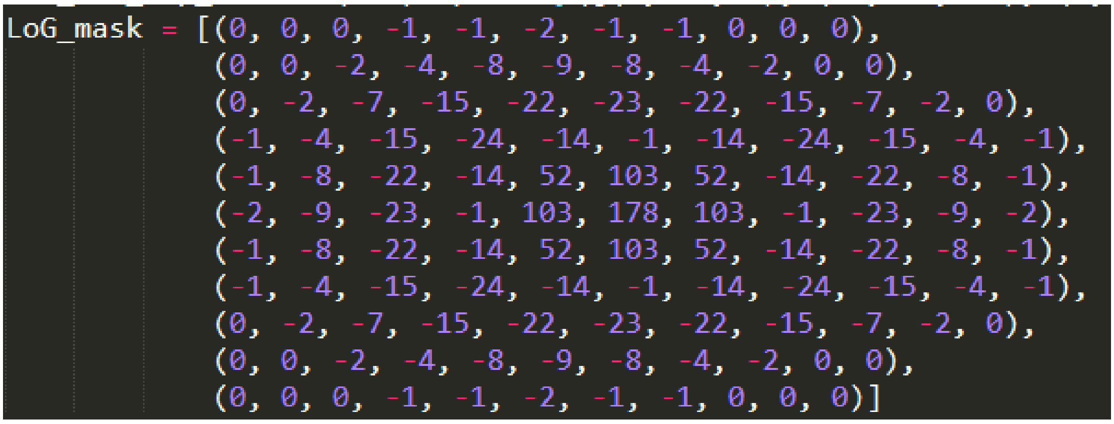
```python
lap_1_mask = [(0, 1, 0), (1, -4, 1), (0, 1, 0)]
lap_2_mask = (1/3)*np.array([(1, 1, 1), (1, -8, 1), (1, 1, 1)])
```

Minimum variance Laplacian:



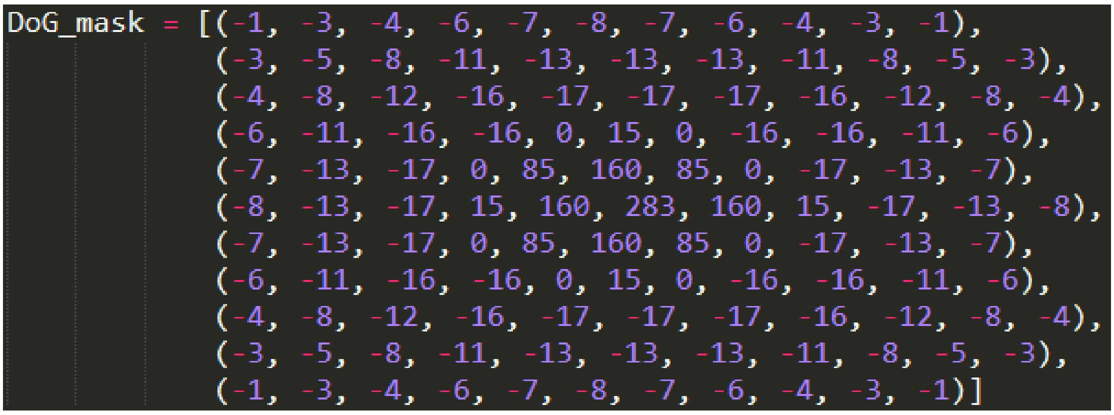
```python
min_var_lap_mask = (1/3)*np.array([(2, -1, 2), (-1, -4, -1), (2, -1, 2)])
```

Laplace of Gaussian:



```python
LoG_mask = [(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)]
```

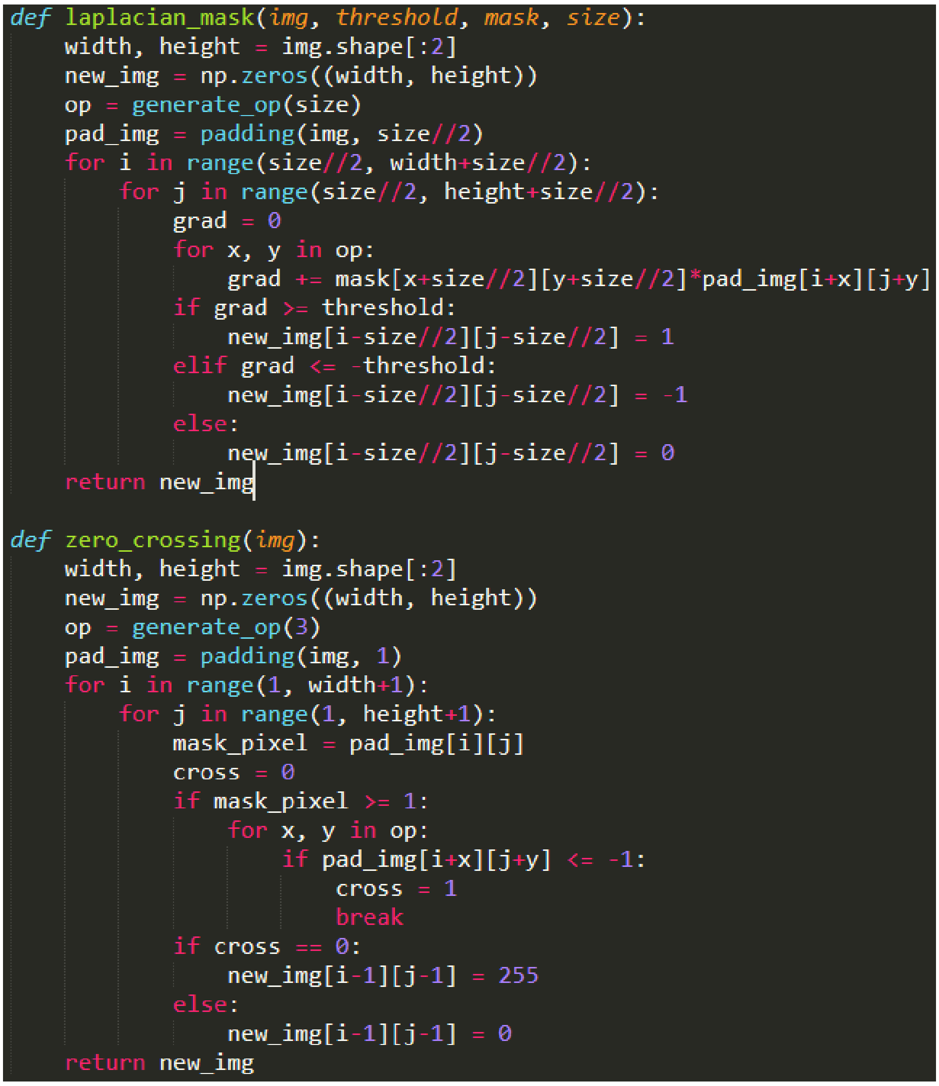
Difference of Gaussian:



```
DoG_mask = [(-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1),
            (-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3),
            (-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4),
            (-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6),
            (-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7),
            (-8, -13, -17, 15, 160, 283, 160, 15, -17, -13, -8),
            (-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7),
            (-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6),
            (-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4),
            (-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3),
            (-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1)]
```

核心演算法: Zero crossing

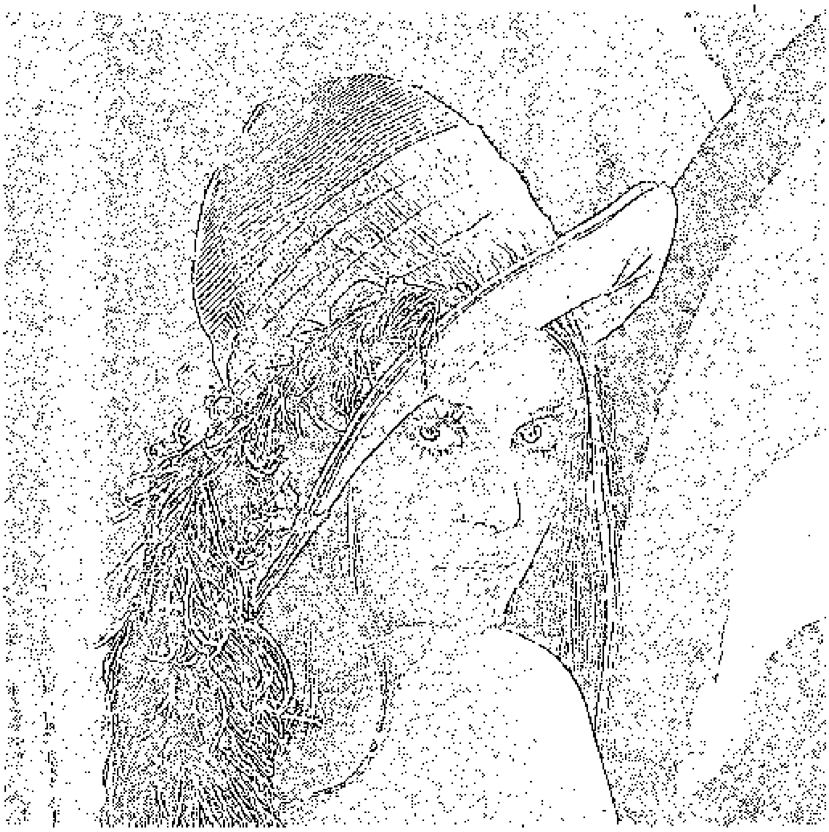
先將原圖依照五個不同的 mask size 去做 padding，然後每種作法用相對應的 mask 來算 gradient magnitude，然後跟 threshold 比較，大於 threshold 的為 1，小於-threshold 的為-1，其餘為 0，然後用產生的陣列去找 zero crossing，先 padding 一次，然後開始尋找：如果自己是 1，並且周圍 8 個點有任一個點是 -1，則這裡發生 zero crossing，這點設為 255，其餘都設成 0。



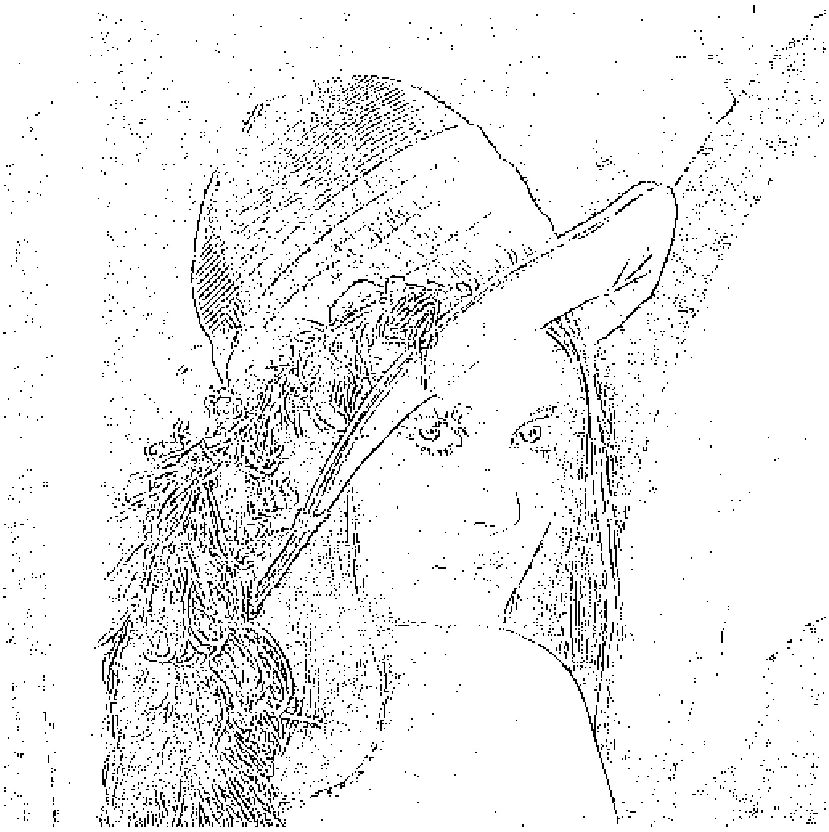
```
def laplacian_mask(img, threshold, mask, size):
    width, height = img.shape[:2]
    new_img = np.zeros((width, height))
    op = generate_op(size)
    pad_img = padding(img, size//2)
    for i in range(size//2, width+size//2):
        for j in range(size//2, height+size//2):
            grad = 0
            for x, y in op:
                grad += mask[x+size//2][y+size//2]*pad_img[i+x][j+y]
            if grad >= threshold:
                new_img[i-size//2][j-size//2] = 1
            elif grad <= -threshold:
                new_img[i-size//2][j-size//2] = -1
            else:
                new_img[i-size//2][j-size//2] = 0
    return new_img

def zero_crossing(img):
    width, height = img.shape[:2]
    new_img = np.zeros((width, height))
    op = generate_op(3)
    pad_img = padding(img, 1)
    for i in range(1, width+1):
        for j in range(1, height+1):
            mask_pixel = pad_img[i][j]
            cross = 0
            if mask_pixel >= 1:
                for x, y in op:
                    if pad_img[i+x][j+y] <= -1:
                        cross = 1
                        break
            if cross == 0:
                new_img[i-1][j-1] = 255
            else:
                new_img[i-1][j-1] = 0
    return new_img
```

Result:

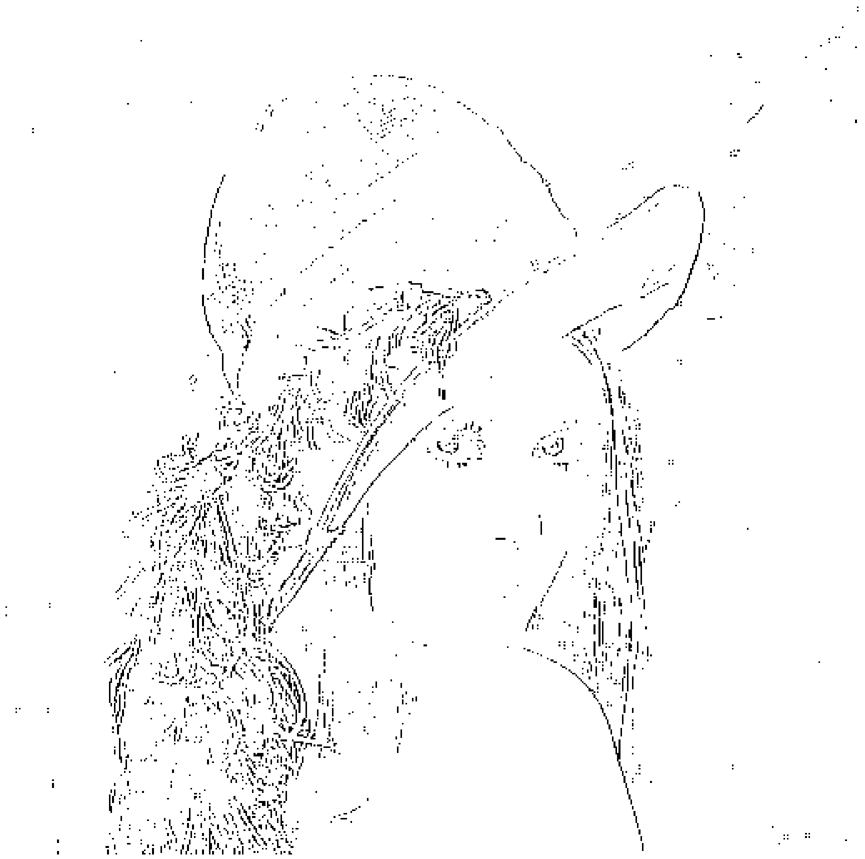
(A) Laplacian mask 1 with threshold 15



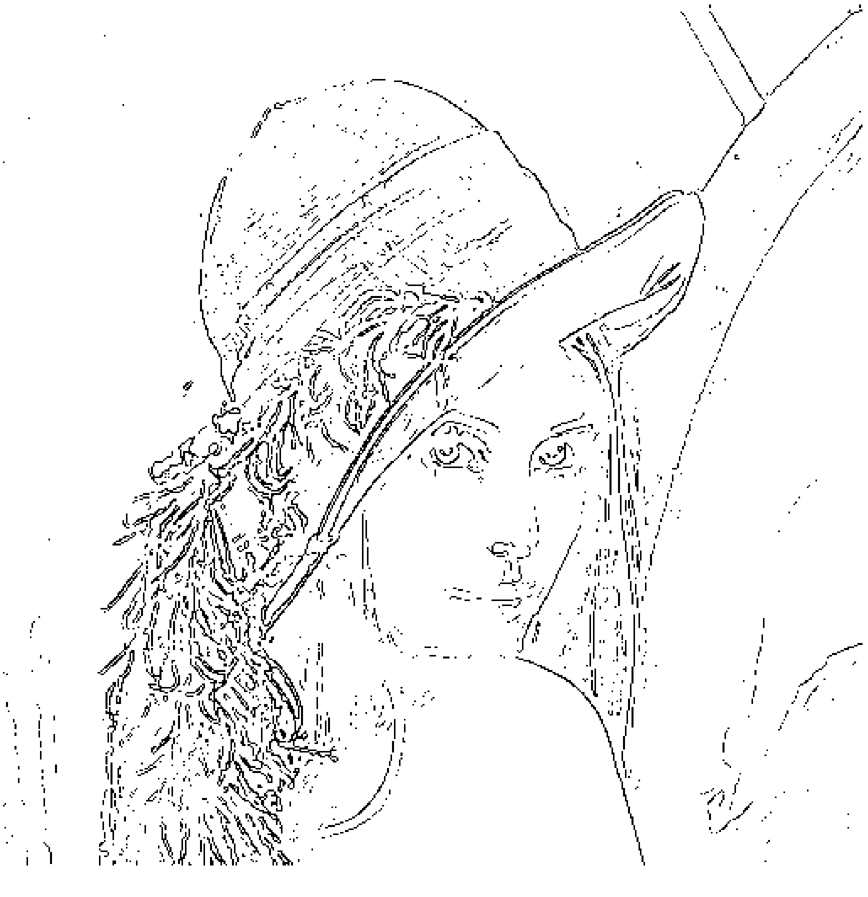
(B) Laplacian mask 2 with threshold 15



(C) Minimum variance Laplacian with threshold 20



(D) Laplace of Gaussian with threshold 3000



(E) Difference of Gaussian with threshold 1