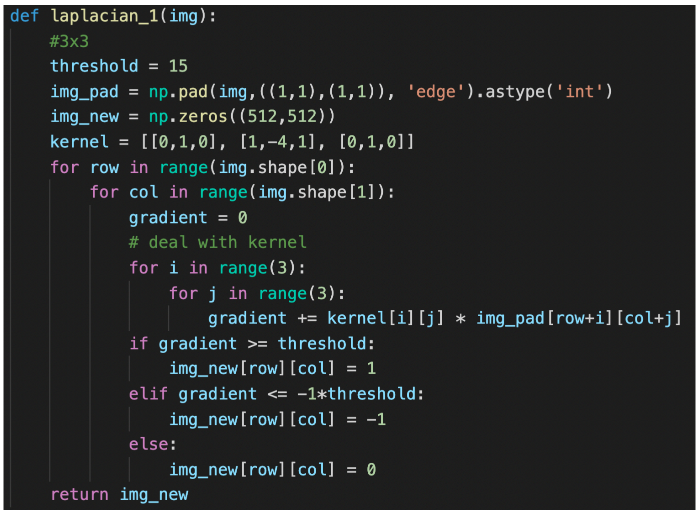
HW10
r10944013 網媒所碩一 馮啟倫

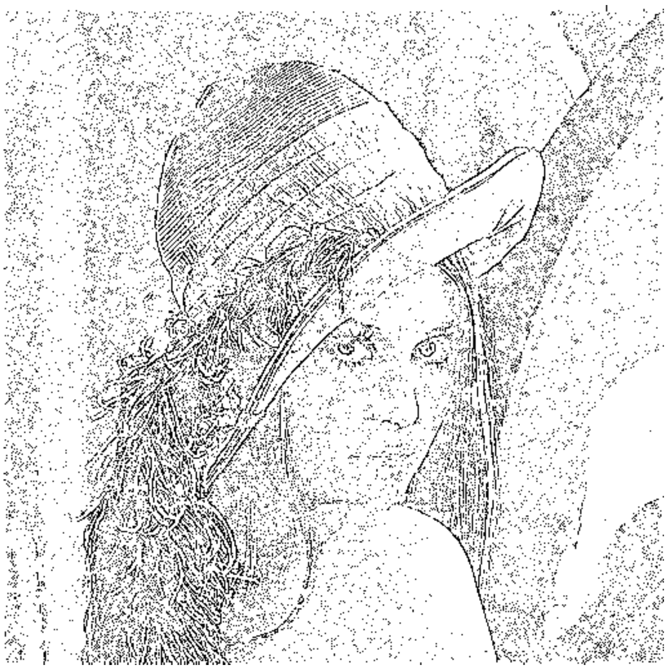
HW10-a:

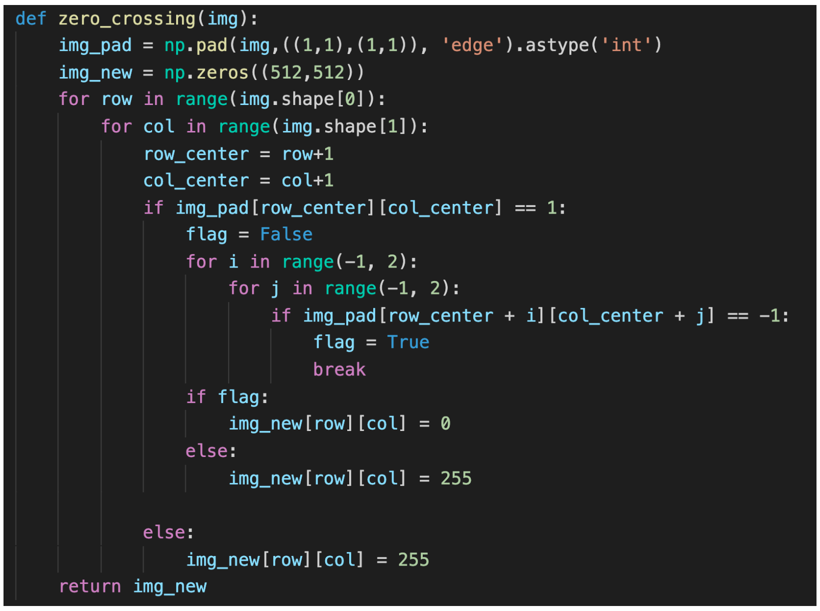
演算法：Laplace Mask1，使用PPT上的mask，並且**threshold=15**

Code segment與result picture（包含laplacian以及zero-crossing，後面的mask就不再放上zero_crossing的code segment）：



```python
def laplacian_1(img):
    #3x3
    threshold = 15
    img_pad = np.pad(img,((1,1),(1,1)), 'edge').astype('int')
    img_new = np.zeros((512,512))
    kernel = [[0,1,0], [1,-4,1], [0,1,0]]
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            gradient = 0
            # deal with kernel
            for i in range(3):
                for j in range(3):
                    gradient += kernel[i][j] * img_pad[row+i][col+j]
            if gradient >= threshold:
                img_new[row][col] = 1
            elif gradient <= -1*threshold:
                img_new[row][col] = -1
            else:
                img_new[row][col] = 0
    return img_new
```



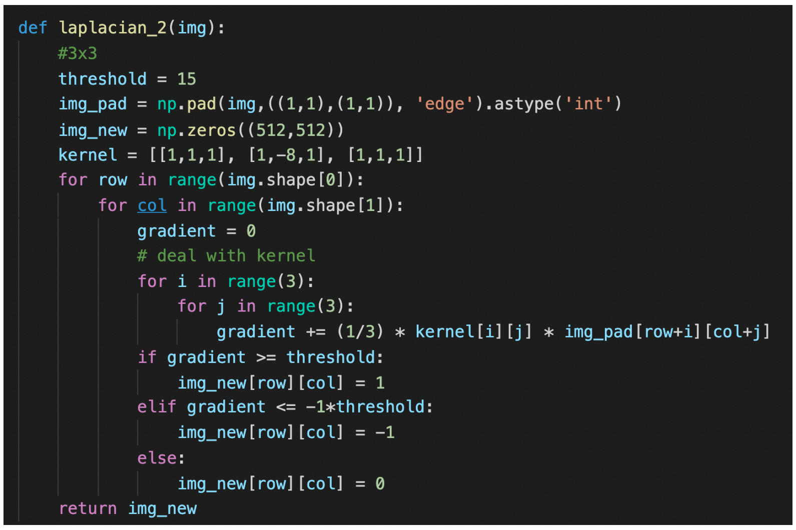


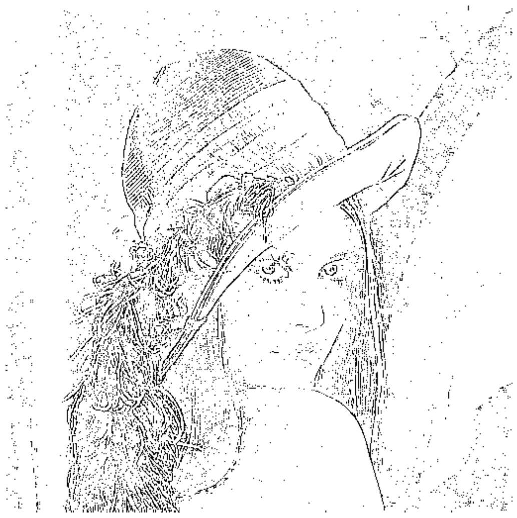
```python
def zero_crossing(img):
    img_pad = np.pad(img,((1,1),(1,1)), 'edge').astype('int')
    img_new = np.zeros((512,512))
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            row_center = row+1
            col_center = col+1
            if img_pad[row_center][col_center] == 1:
                flag = False
                for i in range(-1, 2):
                    for j in range(-1, 2):
                        if img_pad[row_center + i][col_center + j] == -1:
                            flag = True
                            break
                if flag:
                    img_new[row][col] = 0
                else:
                    img_new[row][col] = 255

            else:
                img_new[row][col] = 255
    return img_new
```

HW10-b:

演算法：Laplace Mask2，使用PPT上的mask，並且**threshold=15**

Code segment與result picture：

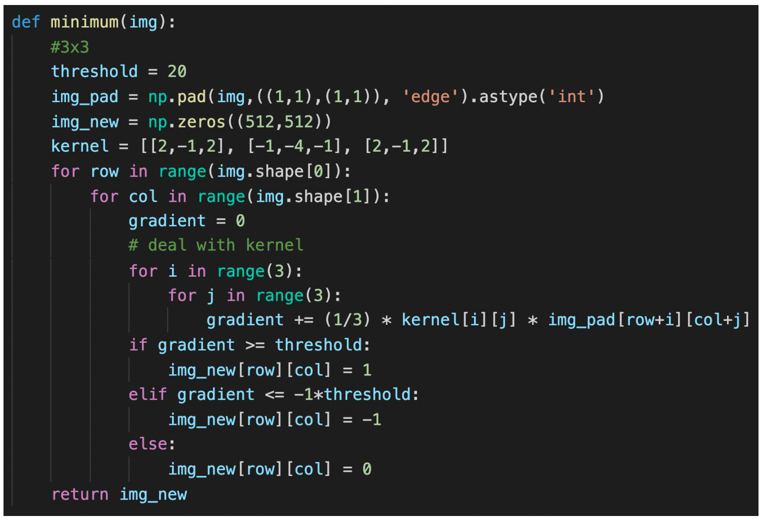

```python
def laplacian_2(img):
    #3x3
    threshold = 15
    img_pad = np.pad(img,((1,1),(1,1)), 'edge').astype('int')
    img_new = np.zeros((512,512))
    kernel = [[1,1,1], [1,-8,1], [1,1,1]]
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            gradient = 0
            # deal with kernel
            for i in range(3):
                for j in range(3):
                    gradient += (1/3) * kernel[i][j] * img_pad[row+i][col+j]
            if gradient >= threshold:
                img_new[row][col] = 1
            elif gradient <= -1*threshold:
                img_new[row][col] = -1
            else:
                img_new[row][col] = 0
    return img_new
```

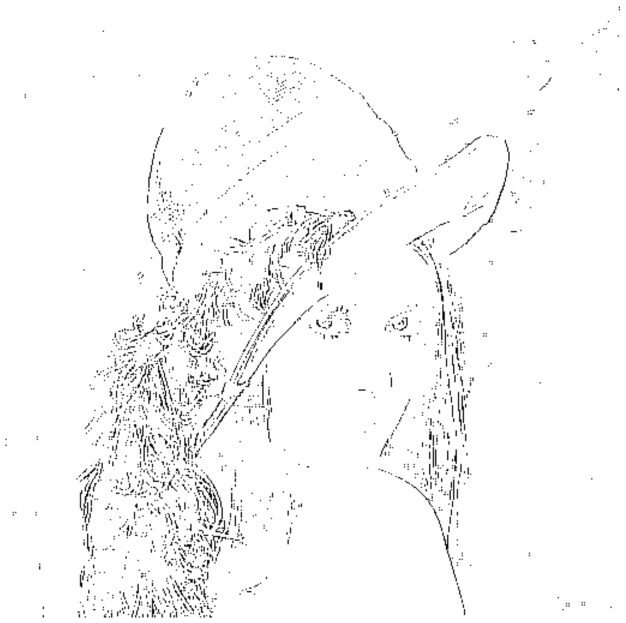


HW10-c:

演算法：Minimum variance Laplacian，使用PPT上的mask，並且**threshold=20**

Code segment與result picture：

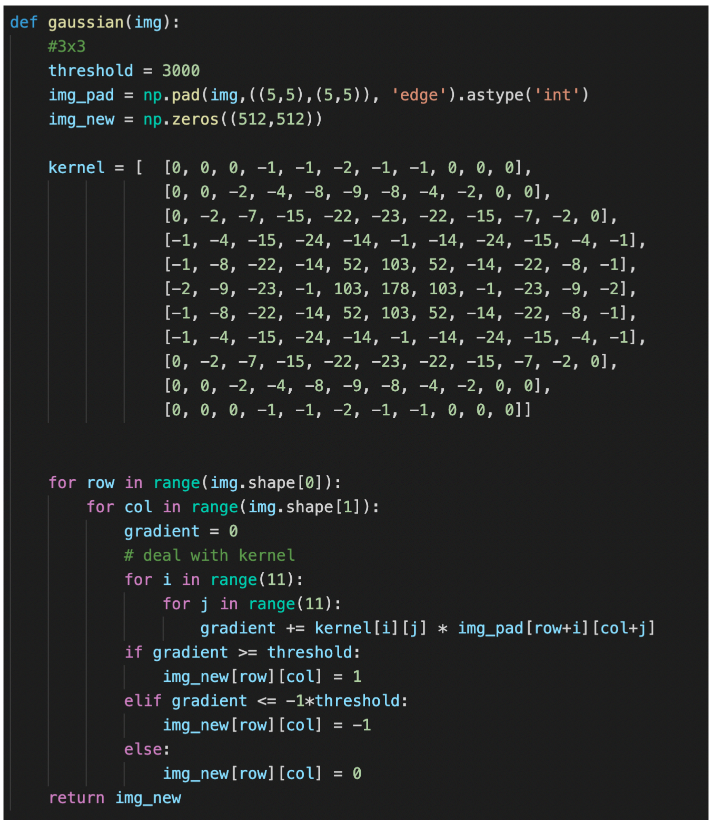

```python
def minimum(img):
    #3x3
    threshold = 20
    img_pad = np.pad(img,((1,1),(1,1)), 'edge').astype('int')
    img_new = np.zeros((512,512))
    kernel = [[2,-1,2], [-1,-4,-1], [2,-1,2]]
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            gradient = 0
            # deal with kernel
            for i in range(3):
                for j in range(3):
                    gradient += (1/3) * kernel[i][j] * img_pad[row+i][col+j]
            if gradient >= threshold:
                img_new[row][col] = 1
            elif gradient <= -1*threshold:
                img_new[row][col] = -1
            else:
                img_new[row][col] = 0
    return img_new
```



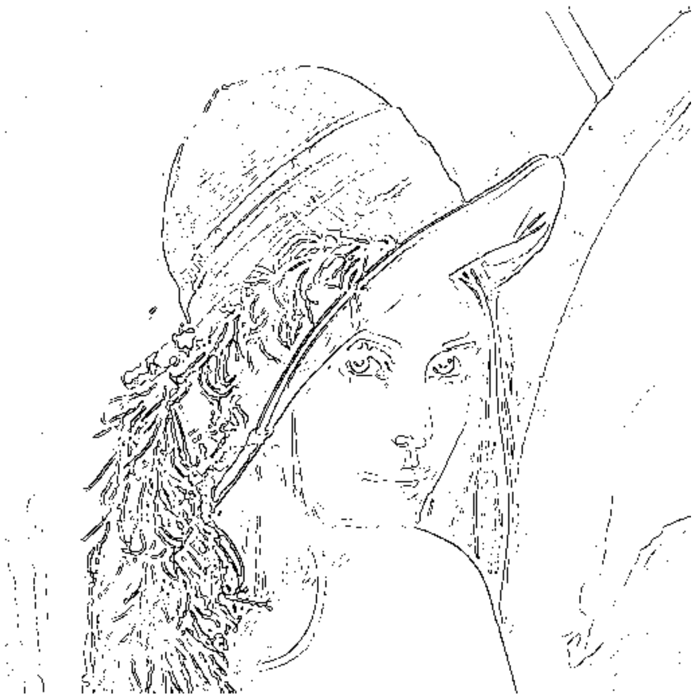
HW10-d:

演算法：Laplace of Gaussian，使用PPT上的mask，並且**threshold=3000**

Code segment與result picture：



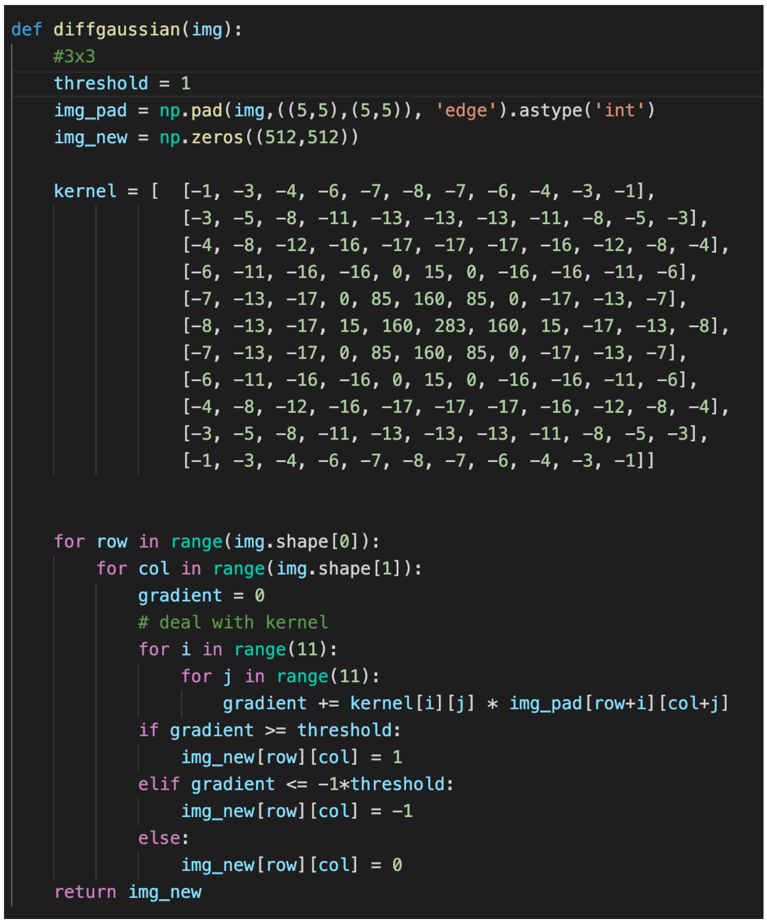
```python
def gaussian(img):
    #3x3
    threshold = 3000
    img_pad = np.pad(img,((5,5),(5,5)), 'edge').astype('int')
    img_new = np.zeros((512,512))

    kernel = [  [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]]


    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            gradient = 0
            # deal with kernel
            for i in range(11):
                for j in range(11):
                    gradient += kernel[i][j] * img_pad[row+i][col+j]
            if gradient >= threshold:
                img_new[row][col] = 1
            elif gradient <= -1*threshold:
                img_new[row][col] = -1
            else:
                img_new[row][col] = 0
    return img_new
```



HW10-e:

演算法：Difference of Gaussian，使用PPT上的mask，並且**threshold=1**

Code segment與result picture：



```python
def diffgaussian(img):
    #3x3
    threshold = 1
    img_pad = np.pad(img,((5,5),(5,5)), 'edge').astype('int')
    img_new = np.zeros((512,512))

    kernel = [  [-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]]


    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            gradient = 0
            # deal with kernel
            for i in range(11):
                for j in range(11):
                    gradient += kernel[i][j] * img_pad[row+i][col+j]
            if gradient >= threshold:
                img_new[row][col] = 1
            elif gradient <= -1*threshold:
                img_new[row][col] = -1
            else:
                img_new[row][col] = 0
    return img_new
```