Computer Vision Homework #10

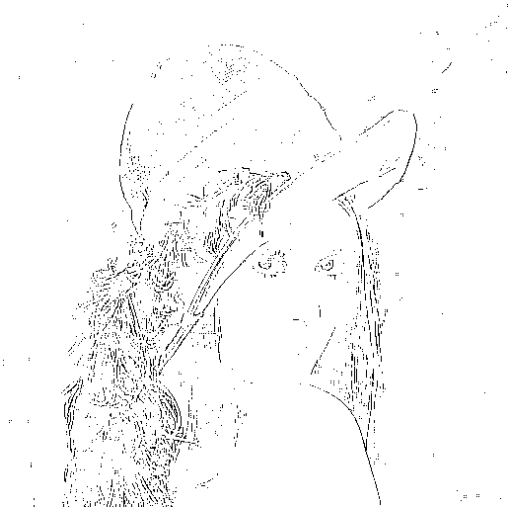
Student-ID: r10944020; Name: 林顯倫; Department: GINM

[Results]

(a) Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15

(b) Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1)

(c) Minimum variance Laplacian: 20



(d) Laplace of Gaussian: 3000

(e) Difference of Gaussian: 1

[Code Fragment & Explanation]

Part-1 Function for edge detection via 3*3 kernel

針對(a)(b)(c)這三個小題，會把不同的3 * 3 kernel傳入這個function進行運算。

```
30 def laplace(img, kernel, threshold):
31     imageW, imageH = img.shape
32     res = np.zeros((imageW-2, imageH-2), dtype='int32')
33     for x in range(imageW-2):
34         for y in range(imageH-2):
35             val = convolution(img[x:x+3, y:y+3], kernel)
36             if val >= threshold:
37                 res[x][y] = 1
38             elif val <= -threshold:
39                 res[x][y] = -1
40             else:
41                 res[x][y] = 0
42     return zero_crossing_detector(res, 3)
```

Part-2 Function for edge detection via 10*10 kernel

針對(d)(e)這三個小題，會把不同的11 * 11 kernel傳入這個function進行運算。

```
44 def edge_Gaussian(img, kernel, threshold):
45     imageW, imageH = img.shape
46     res = np.zeros((imageW-10, imageH-10), dtype='int32')
47     for x in range(imageW-10):
48         for y in range(imageH-10):
49             val = convolution(img[x:x+11, y:y+11], kernel)
50             if val >= threshold:
51                 res[x][y] = 1
52             elif val <= -threshold:
53                 res[x][y] = -1
54             else:
55                 res[x][y] = 0
56     return zero_crossing_detector(res, 3)
```

Part-3 Zero-Crossing Edge Detection 此部分依照講義公式與助教講解實作

```
13 def validPixel(x, bound):
14     return (x >= 0 and x <= bound)
15
16 def zero_crossing_detector(img, kernel_size):
17     imageW, imageH = img.shape
18     res = np.full(img.shape, 255, dtype='int32')
19     for x in range(imageW):
20         for y in range(imageH):
21             if img[x][y] == 1:
22                 for ex in range(-kernel_size//2+1, kernel_size//2+1):
23                     for ey in range(-kernel_size//2+1, kernel_size//2+1):
24                         dest_x, dest_y = x + ex, y + ey
25                         if validPixel(dest_x, imageW-1) and validPixel(dest_y, imageH-1) \
26                             and img[dest_x][dest_y] == -1:
27                             res[x][y] = 0
28     return res
```

Part-4 Padding 圖片傳入之前，會先做不同的Padding (for 3*3 and 11*11)

```
58 def padding_img(img, size):
59     return np.pad(img, (size, size), 'edge')
```

Part-5 主函式呼叫（見下頁）

```python
if __name__ == '__main__':
    img = cv2.imread('lena.bmp', cv2.IMREAD_GRAYSCALE)
    img_pad1 = padding_img(img, 1)
    img_pad2 = padding_img(img, 5)

    # (a) Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15
    kernel_laplace_1 = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
    res_a = laplace(img_pad1, kernel_laplace_1, 15)
    cv2.imwrite('res_a.png', res_a)
    # (b) Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1)
    kernel_laplace_2 = np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]]) / 3
    res_b = laplace(img_pad1, kernel_laplace_2, 15)
    cv2.imwrite('res_b.png', res_b)
    # (c) Minimum variance Laplacian: 20
    kernel_laplace_minvar = np.array([[2, -1, 2], [-1, -4, -1], [2, -1, 2]]) / 3
    res_c = laplace(img_pad1, kernel_laplace_minvar, 20)
    cv2.imwrite('res_c.png', res_c)
    # (d) Laplace of Gaussian: 3000
    kernel_LOG = np.array([
        [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]
    ])
    res_d = edge_Gaussian(img_pad2, kernel_LOG, 3000)
    cv2.imwrite('res_d.png', res_d)
    # (e) Difference of Gaussian: 1
    kernel_DOG = np.array([
        [-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]
    ])
    res_e = edge_Gaussian(img_pad2, kernel_DOG, 1)
    cv2.imwrite('res_e.png', res_e)
```