I. Problem Statement

Implement 2 Laplacian Mask, Minimum Variance Laplacian, Laplacian of Gaussian, and Difference of Gaussian(inhibitory sigma=3, excitatory sigma=1, kernel size 11x11). Please list the kernels and the thresholds(for zero crossing) you used.

II. Programming Tools

- Programming language: Python 3.8.5
- Library: Numpy 1.19.1, OpenCV 4.0.1

III. Problem-Solving Process

對 image 的由上到下、由左到右計算使用不同 kernel 去 pixel gradient，若 gradient 大於等於 threshold，則 mask 填 1，若 gradient 小於等於-threshold，則 mask 填-1，其餘 mask 填 0，之後再對 mask 做 zero crossing 檢測邊緣。

以下為計算 gradient 與 mask 的程式碼。

```
def gradient(mask, k, b, threshold, img_padding):
    for r in range(img.shape[0]):
        for c in range(img.shape[1]):
            gradient_magnitude = np.sum(img_padding[r:r+b,c:c+b] * k)
            if gradient_magnitude >= threshold:
                mask[r,c] = 1
            elif gradient_magnitude <= -threshold:
                mask[r,c] = -1
            else:
                mask[r,c] = 0
```

以下為 zero crossing 的程式碼，先判斷 mask 的 value 是否為 1，若等於 1 再判斷他的八個鄰居是否有值為-1 的，若有則為 edge。

```
def zero_crossing(img,mask):
    mask_padding = cv2.copyMakeBorder(mask,1,1,1,1,cv2.BORDER_REPLICATE)
    img_return = np.full_like(img,255,dtype=np.uint8)
    for r in range(img.shape[0]):
        for c in range(img.shape[1]):
            if mask_padding[r+1,c+1]==1 and (mask_padding[r:r+3,c:c+3]==-1).any():
                img_return[r,c] = 0
    return img_return
```

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

使用 k 作為 kernel，並設定 threshold 為 15。

```
def Laplacian_1(img, threshold=15):
    k = np.array([[0,1,0],[1,-4,1],[0,1,0]],dtype=np.int32)
    img_padding = cv2.copyMakeBorder(img,1,1,1,1,cv2.BORDER_REPLICATE).astype(np.int32)
    mask = np.zeros_like(img,dtype=np.int32)
    gradient(mask, k, k.shape[0], threshold, img_padding)
    return zero_crossing(img, mask)
```

(b) Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1): 15

使用 k 作為 kernel，並設定 threshold 為 15。

```python
def Laplacian_2(img, threshold=15):
    k = np.array([[1,1,1],[1,-8,1],[1,1,1]],dtype=np.float) / 3
    img_padding = cv2.copyMakeBorder(img,1,1,1,1,cv2.BORDER_REPLICATE).astype(np.float)
    mask = np.zeros_like(img,dtype=np.int32)
    gradient(mask, k, k.shape[0], threshold, img_padding)
    return zero_crossing(img, mask)
```

(c) Minimum variance Laplacian: 20

使用 k 作為 kernel，並設定 threshold 為 20。

```python
def Minimum_variance_Laplacian(img, threshold=20):
    k = np.array([[2,-1,2],[-1,-4,-1],[2,-1,2]],dtype=np.float) / 3
    img_padding = cv2.copyMakeBorder(img,1,1,1,1,cv2.BORDER_REPLICATE).astype(np.float)
    mask = np.zeros_like(img,dtype=np.int32)
    gradient(mask, k, k.shape[0], threshold, img_padding)
    return zero_crossing(img, mask)
```

(d) Laplace of Gaussian: 3000

使用 k 作為 kernel，並設定 threshold 為 3000。

```python
def Laplacian_of_Gaussian(img, threshold=3000):
    k = 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]
        ],dtype=np.float)
    img_padding = cv2.copyMakeBorder(img,5,5,5,5,cv2.BORDER_REPLICATE).astype(np.float)
    mask = np.zeros_like(img,dtype=np.int32)
    gradient(mask, k, k.shape[0], threshold, img_padding)
    return zero_crossing(img, mask)
```

(e) Difference of Gaussian: 1

使用 k 作為 kernel，並設定 threshold 為 1。

```python
def Difference_of_Gaussian(img, threshold=1):
    k = 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],
        ],dtype=np.float)
    img_padding = cv2.copyMakeBorder(img,5,5,5,5,cv2.BORDER_REPLICATE).astype(np.float)
    mask = np.zeros_like(img,dtype=np.int32)
    gradient(mask, k, k.shape[0], threshold, img_padding)
    return zero_crossing(img, mask)
```

IV. 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): 15

(c) Minimum variance Laplacian: 20

(d) Laplace of Gaussian: 3000

資工所 r08922123
Author: 王韻豪

(e) Difference of Gaussian: 1