R10922123 周昱豪

Code:

基本上就是套 kernel 下去，然後跟原圖做 convolution，而 zero crossing 就是如果現在的值 >= t，就去檢查鄰居是否有 <= -t，然後執行順序都是先做 Laplace 再做 zero crossing，出來的結果存成圖片

Laplacian

```
def Laplacian(img, kernel, threshold):
    row, col = img.shape
    res_img = np.zeros((row - 2, col - 2))

    for i in range(1, row-1):
        for j in range(1, col-1):
            rk, ck = kernel.shape
            magnitude_gradient = 0
            for ki in range(-rk // 2 + 1, rk // 2 + 1):
                for kj in range(-ck // 2 + 1, ck // 2 + 1):
                    magnitude_gradient += img[i+ki][j+kj] * kernel[ki+(rk//2)][kj+(ck//2)]

            if magnitude_gradient >= threshold:
                res_img[i-1][j-1] = 1
            elif magnitude_gradient <= -threshold:
                res_img[i-1][j-1] = -1
            else:
                res_img[i-1][j-1] = 0
    return res_img
```

Zero crossing

```
26  def zero_crossing(pad_img, t):
27      row, col = pad_img.shape
28      res_img = np.zeros((row-2, col-2))
29
30      for i in range(1, row-1):
31          for j in range(1, col-1):
32              res_img[i-1][j-1] = 255
33              if pad_img[i][j] >= t:
34                  for ki in range(-1, 2):
35                      for kj in range(-1, 2):
36                          if pad_img[i+ki][j+kj] <= -t:
37                              res_img[i-1][j-1] = 0
38                              break
39      return res_img
```

Laplacian of Gaussian

```python
def Laplacian_of_Gaussian(pad_img, threshold):
    kernel = np.array([ ···
    row, col = pad_img.shape
    res_img = np.zeros((row - 10, col - 10))
    rk, ck = kernel.shape

    for i in range(5, row-5):
        for j in range(5, col-5):
            magnitude_gradient = 0
            for ki in range(-rk // 2 + 1, rk // 2 + 1):
                for kj in range(-ck // 2 + 1, ck // 2 + 1):
                    magnitude_gradient += pad_img[i+ki][j+kj] * kernel[ki+(rk//2)][kj+(ck//2)]

            if magnitude_gradient >= threshold:
                res_img[i-5][j-5] = 1
            elif magnitude_gradient <= -threshold:
                res_img[i-5][j-5] = -1
            else:
                res_img[i-5][j-5] = 0
        print(i)
    return res_img
```

Difference of Gaussian

```python
def Difference_of_Gaussian(pad_img, threshold):
    kernel = np.array([ ···
    row, col = pad_img.shape
    res_img = np.zeros((row - 10, col - 10))
    rk, ck = kernel.shape

    for i in range(5, row-5):
        for j in range(5, col-5):
            magnitude_gradient = 0
            for ki in range(-rk // 2 + 1, rk // 2 + 1):
                for kj in range(-ck // 2 + 1, ck // 2 + 1):
                    magnitude_gradient += pad_img[i+ki][j+kj] * kernel[ki+(rk//2)][kj+(ck//2)]

            if magnitude_gradient >= threshold:
                res_img[i-5][j-5] = 1
            elif magnitude_gradient <= -threshold:
                res_img[i-5][j-5] = -1
            else:
                res_img[i-5][j-5] = 0
        print(i)
    return res_img
```

Result:

Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15	Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1)
	

Minimum variance Laplacian: 20

Laplace of Gaussian: 3000

Difference of Gaussian: 1