Homework 10

Zero Crossing Edge Detection

R09921119 許凱荃

- 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.

- Threshold Values listed below are for reference:
  - Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15
  - Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1) : 15
  - Minimum variance Laplacian: 30
  - Laplace of Gaussian: 3000
  - Difference of Gaussian: 1

Outline :

- Used Kernel
- Method
- Image Result

Used Kernel

Laplacian1

(threshold = 15)

	1	
1	-4	1
	1	

Laplacian2

(threshold = 15)

$\frac{1}{3}$

1	1	1
1	-8	1
1	1	1

minimum-variance Laplacian

(threshold = 20)

$\frac{1}{3}$

2	-1	2
-1	-4	-1
2	-1	2

Laplacian of Gaussian (threshold = 3000)

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 (threshold = 1)

(inhibitory σ = 3, excitatory σ = 1, kernel size=11)

-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

Method

所有方法都是先套 Mask，再統一用 Zero-crossing 得到 Filter 後的結果。

- **Laplace Mask1** : Kernel 設定好，算加權總合。

```
def get_laplacian1_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image (0, 1)
    """

    w, h = img.shape
    new_img = img.copy().astype('int16')
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1              x1y1   x1y   x1y2
            #    x            x y1   x y   x y2
            #      x2         x2y1   x2y   x2y2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y]), int(img[x1][y2]), int(img[x][y1]), int(img[x][y]), int(
            magitude = np.dot(np.array( [0, 1, 0, 1, -4, 1, 0, 1, 0] ), coordinate)

            if magitude >= threshold :
                new_img[x-1][y-1] = 1
            elif magitude <= threshold*(-1):
                new_img[x-1][y-1] = -1
            else:
                new_img[x-1][y-1] = 0

    return new_img
```

- **Laplace Mask2** : 同上，換一個 Kernel 而已。

```
def get_laplacian2_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image (0, 1)
    """

    w, h = img.shape
    new_img = img.copy().astype('int16')
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1              x1y1   x1y   x1y2
            #    x            x y1   x y   x y2
            #      x2         x2y1   x2y   x2y2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y]), int(img[x1][y2]), int(img[x][y1]), int(img
            magitude = np.dot(np.array( [1, 1, 1, 1, -8, 1, 1, 1, 1] ), coordinate)*(1/3)

            if magitude >= threshold :
                new_img[x-1][y-1] = 1
            elif magitude <= threshold*(-1):
                new_img[x-1][y-1] = -1
            else:
                new_img[x-1][y-1] = 0

    return new_img
```

- **Minimum variance Laplacian**：同上，換一個 Kernel 而已。

```
def get_minvar_laplacian_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image (0, 1)
    """

    w, h = img.shape
    new_img = img.copy().astype('int16')
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1               x1y1   x1y   x1y2
            #    x             x y1   x y   x y2
            #       x2         x2y1   x2y   x2y2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y]), int(img[x1][y2]), int(img[x][y1
            magitude = np.dot(np.array( [2, -1, 2, -1, -4, -1, 2, -1, 2] ), coordinate)*(1/3)

            if magitude >= threshold :
                new_img[x-1][y-1] = 1
            elif magitude <= threshold*(-1):
                new_img[x-1][y-1] = -1
            else:
                new_img[x-1][y-1] = 0

    return new_img
```

- **Laplace of Gaussian**：按照 table 計算權重。

```
def get_gauss_laplacian_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image (0, 1)
    """

    w, h = img.shape
    new_img = img.copy().astype('int16')
    img = extend_padding(img, 5)

    for x in range( 5,w+5 ):
        for y in range( 5,h+5 ):

            # x1               x1y1   x1y   x1y2
            #    x             x y1   x y   x y2
            #       x2         x2y1   x2y   x2y2
            x1, y1, x2, y2, x3, y3, x4, y4, x5, y5 = x-5, y-5, x-4, y-4, x-3, y-3, x-2, y-2, x-1, y
            x6, y6, x7, y7, x8, y8, x9, y9, x10, y10 = x+1, y+1, x+2, y+2, x+3, y+3, x+4, y+4, x+5,

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y2]), int(img[x1][y3]), int(img[x
                                    int(img[x2][y1]), int(img[x2][y2]), int(img[x2][y3]), int(img[x
                                    int(img[x3][y1]), int(img[x3][y2]), int(img[x3][y3]), int(img[x
                                    int(img[x4][y1]), int(img[x4][y2]), int(img[x4][y3]), int(img[x
                                    int(img[x5][y1]), int(img[x5][y2]), int(img[x5][y3]), int(img[x
                                    int(img[x ][y1]), int(img[x ][y2]), int(img[x ][y3]), int(img[x
                                    int(img[x6][y1]), int(img[x6][y2]), int(img[x6][y3]), int(img[x
                                    int(img[x7][y1]), int(img[x7][y2]), int(img[x7][y3]), int(img[x
                                    int(img[x8][y1]), int(img[x8][y2]), int(img[x8][y3]), int(img[x
                                    int(img[x9][y1]), int(img[x9][y2]), int(img[x9][y3]), int(img[x
                                    int(img[x10][y1]), int(img[x10][y2]), int(img[x10][y3]), int(im
                                    ])
            magitude = np.dot(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 \
                                    ] ), coordinate)

            if magitude >= threshold :
                new_img[x-5][y-5] = 1
            elif magitude <= threshold*(-1):
                new_img[x-5][y-5] = -1
            else:
                new_img[x-5][y-5] = 0

    return new_img
```

- **Difference of Gaussian**：按照 table 計算權重。

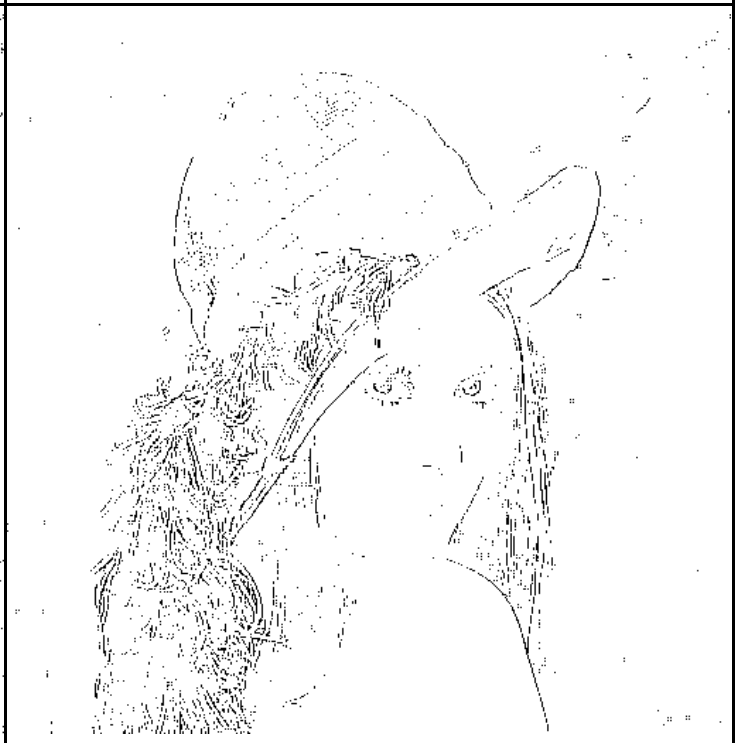
```
def get_diff_gauss_laplacian_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image (0, 1)
    """

    w, h = img.shape
    new_img = img.copy().astype('int16')
    img = extend_padding(img, 5)

    for x in range( 5,w+5 ):
        for y in range( 5,h+5 ):

            # x1               x1y1   x1y   x1y2
            #    x             x y1   x y   x y2
            #      x2          x2y1   x2y   x2y2
            x1, y1, x2, y2, x3, y3, x4, y4, x5, y5 = x-5, y-5, x-4, y-4, x-3, y-3, x-2, y-2, x-1, y-1
            x6, y6, x7, y7, x8, y8, x9, y9, x10, y10 = x+1, y+1, x+2, y+2, x+3, y+3, x+4, y+4, x+5, y+5

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y2]), int(img[x1][y3]), int(img[x1][y4]), int(img[x1][y5]), int(img[x1][y]), i
                                    int(img[x2][y1]), int(img[x2][y2]), int(img[x2][y3]), int(img[x2][y4]), int(img[x2][y5]), int(img[x2][y]), i
                                    int(img[x3][y1]), int(img[x3][y2]), int(img[x3][y3]), int(img[x3][y4]), int(img[x3][y5]), int(img[x3][y]), i
                                    int(img[x4][y1]), int(img[x4][y2]), int(img[x4][y3]), int(img[x4][y4]), int(img[x4][y5]), int(img[x4][y]), i
                                    int(img[x5][y1]), int(img[x5][y2]), int(img[x5][y3]), int(img[x5][y4]), int(img[x5][y5]), int(img[x5][y]), i
                                    int(img[x ][y1]), int(img[x ][y2]), int(img[x ][y3]), int(img[x ][y4]), int(img[x ][y5]), int(img[x ][y]), i
                                    int(img[x6][y1]), int(img[x6][y2]), int(img[x6][y3]), int(img[x6][y4]), int(img[x6][y5]), int(img[x6][y]), i
                                    int(img[x7][y1]), int(img[x7][y2]), int(img[x7][y3]), int(img[x7][y4]), int(img[x7][y5]), int(img[x7][y]), i
                                    int(img[x8][y1]), int(img[x8][y2]), int(img[x8][y3]), int(img[x8][y4]), int(img[x8][y5]), int(img[x8][y]), i
                                    int(img[x9][y1]), int(img[x9][y2]), int(img[x9][y3]), int(img[x9][y4]), int(img[x9][y5]), int(img[x9][y]), i
                                    int(img[x10][y1]), int(img[x10][y2]), int(img[x10][y3]), int(img[x10][y4]), int(img[x10][y5]), int(img[x10][
                                    ])
            magitude = np.dot(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, -1, \
                                         -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, -1, \
                                         -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 \
                                    ] ), coordinate)

            if magitude >= threshold :
                new_img[x-5][y-5] = 1
            elif magitude <= threshold*(-1):
                new_img[x-5][y-5] = -1
            else:
                new_img[x-5][y-5] = 0

    return new_img
```

Image Result

Original Lena

Laplace Mask1 : 15

Laplace Mask1 : 15



Minimum variance Laplacian: 30

Laplace of Gaussian: 3000

Difference of Gaussian: 1

Tips : Zero-crossing 只要判斷原值為 1 時周圍是否有 -1 即可

助教辛苦了 :)

(· ∀ ·)ﾉｼ