Howework 10 Report

Student:鞠之浩 ID:P10942A08

Output:

Part(a)

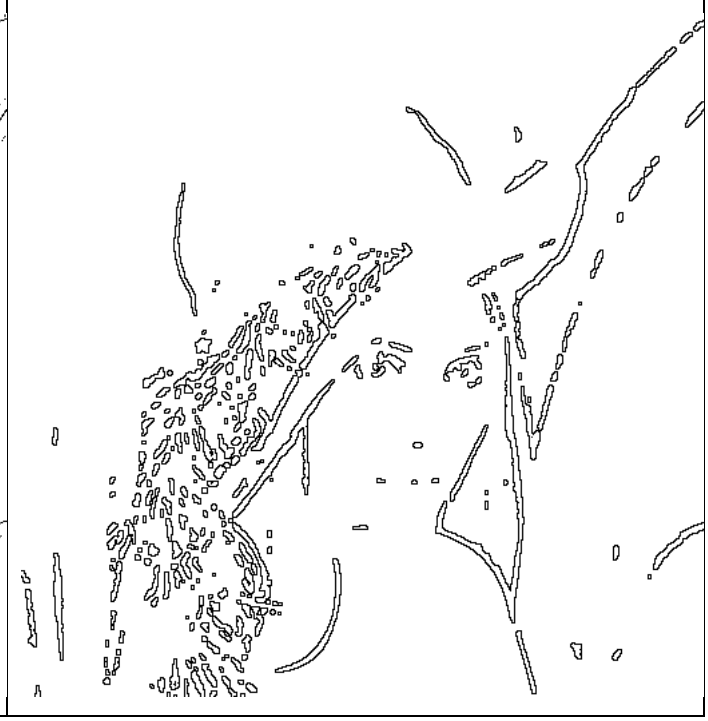
Laplacian Mask1, threshold: 15

Part(b)

Laplacian Mask2, threshold: 15

Part(c)

Minimum-variance Laplacian, threshold: 20

Part(d)	Part(e)
Laplace of Gaussian, threshold: 3000	Difference of Gaussian, threshold: 1
	

Code describe:

Part(a):

```
def Laplacian_Mask_1(img, threshold):
    laplacian_img = np.zeros(img.shape, np.int)
    img = padding(img)
    row, col = img.shape
    for i in range(1, row-1):
        for j in range(1, col-1):
            gradient = int(img[i-1, j]) + int(img[i, j-1]) + int(img[i, j+1]) + int(img[i+1, j]) - 4*int(img[i, j])
            if gradient >= threshold:
                laplacian_img[i-1, j-1] = 1
            elif gradient <= -threshold:
                laplacian_img[i-1, j-1] = -1
            else:
                laplacian_img[i-1, j-1] = 0
    zero_crossing_img = np.zeros(laplacian_img.shape, np.int)
    laplacian_img = padding(laplacian_img)
    for i in range(1, row-1):
            for j in range(1, col-1):
                zero_crossing_img[i-1, j-1] = 255
                if laplacian_img[i, j] == 1:
                    if laplacian_img[i-1, j-1] == -1 or laplacian_img[i-1, j] == -1 or laplacian_img[i-1, j+1] == -1 or
                        zero_crossing_img[i-1, j-1] = 0
                else:
                    zero_crossing_img[i-1, j-1] = 255
    return zero_crossing_img
```

do Laplacian Mask1 first then do zero-crossing for every pixel.

Part(b):

```
def Laplacian_Mask_2(img, threshold):
    laplacian_img = np.zeros(img.shape, np.int)
    img = padding(img)
    row, col = img.shape
    for i in range(1, row-1):
        for j in range(1, col-1):
            gradient = (int(img[i-1, j-1]) + int(img[i-1, j]) + int(img[i-1, j+1]) + int(img[i, j-1]) + int(img[i, j+1]
            if gradient >= threshold:
                laplacian_img[i-1, j-1] = 1
            elif gradient <= -threshold:
                laplacian_img[i-1, j-1] = -1
            else:
                laplacian_img[i-1, j-1] = 0
    zero_crossing_img = np.zeros(laplacian_img.shape, np.int)
    laplacian_img = padding(laplacian_img)
    for i in range(1, row-1):
            for j in range(1,col-1):
                zero_crossing_img[i-1, j-1] = 255
                if laplacian_img[i, j] == 1:
                    if laplacian_img[i-1, j-1] == -1 or laplacian_img[i-1, j] == -1 or laplacian_img[i-1, j+1] == -1 or
                        zero_crossing_img[i-1, j-1] = 0
                else:
                    zero_crossing_img[i-1, j-1] = 255
    return zero_crossing_img
```

do Laplacian Mask2 first then do zero-crossing for every pixel.

Part(c):

```
def minimum_variance_Laplacian(img, threshold):
    laplacian_img = np.zeros(img.shape, np.int)
    img = padding(img)
    row, col = img.shape
    for i in range(1, row-1):
        for j in range(1, col-1):
            gradient = (2*int(img[i-1, j-1]) - int(img[i-1, j]) + 2*int(img[i-1, j+1]) - int(img[i, j-1]) - int(img[i,
            if gradient >= threshold:
                laplacian_img[i-1, j-1] = 1
            elif gradient <= -threshold:
                laplacian_img[i-1, j-1] = -1
            else:
                laplacian_img[i-1, j-1] = 0
    zero_crossing_img = np.zeros(laplacian_img.shape, np.int)
    laplacian_img = padding(laplacian_img)
    for i in range(1, row-1):
            for j in range(1, col-1):
                zero_crossing_img[i-1, j-1] = 255
                if laplacian_img[i, j] == 1:
                    if laplacian_img[i-1, j-1] == -1 or laplacian_img[i-1, j] == -1 or laplacian_img[i-1, j+1] == -1 or
                        zero_crossing_img[i-1, j-1] = 0
                else:
                    zero_crossing_img[i-1, j-1] = 255
    return zero_crossing_img
```

do Minimum-variance Laplacian first then do zero-crossing for every pixel.

Part(d):

```
def Laplacian_of_Gaussian(img, threshold):
    mask = [[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]
           ]
    laplacian_img = np.zeros(img.shape, np.int)
    img = padding(img)
    img = padding(img)
    img = padding(img)
    img = padding(img)
    img = padding(img)
    row, col = img.shape
    for i in range(5, row-5):
        for j in range(5, col-5):
            gradient = 0
            for m in range(-5, 6):
                for n in range(-5, 6):
                    gradient += int(img[i+m, j+n]) * mask[m+5][n+5]
            if gradient >= threshold:
                laplacian_img[i-5, j-5] = 1
            elif gradient <= -threshold:
                laplacian_img[i-5, j-5] = -1
            else:
                laplacian_img[i-5, j-5] = 0
    zero_crossing_img = np.zeros(laplacian_img.shape, np.int)
    laplacian_img = padding(laplacian_img)
    row, col = laplacian_img.shape
    for i in range(1, row-1):
            for j in range(1, col-1):
                zero_crossing_img[i-1, j-1] = 255
                if laplacian_img[i, j] == 1:
                    if laplacian_img[i-1, j-1] == -1 or laplacian_img[i-1, j] == -1 or laplacian_img[i-1, j+1] == -1 or
                        zero_crossing_img[i-1, j-1] = 0
                else:
                    zero_crossing_img[i-1, j-1] = 255
    return zero_crossing_img
```

do Laplace of Gaussian first then do zero-crossing for every pixel.

Part(e):

```
def Difference_of_Gaussian(img, threshold):
    mask = [[-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]
           ]
    laplacian_img = np.zeros(img.shape, np.int)
    img = padding(img)
    img = padding(img)
    img = padding(img)
    img = padding(img)
    img = padding(img)
    row, col = img.shape
    for i in range(5, row-5):
        for j in range(5, col-5):
            gradient = 0
            for m in range(-5, 6):
                for n in range(-5, 6):
                    gradient += int(img[i+m, j+n]) * mask[m+5][n+5]
            if gradient >= threshold:
                laplacian_img[i-5, j-5] = 1
            elif gradient <= -threshold:
                laplacian_img[i-5, j-5] = -1
            else:
                laplacian_img[i-5, j-5] = 0
    zero_crossing_img = np.zeros(laplacian_img.shape, np.int)
    laplacian_img = padding(laplacian_img)
    row, col = laplacian_img.shape
    for i in range(1, row-1):
            for j in range(1, col-1):
                zero_crossing_img[i-1, j-1] = 255
                if laplacian_img[i, j] == 1:
                    if laplacian_img[i-1, j-1] == -1 or laplacian_img[i-1, j] == -1 or laplacian_img[i-1, j+1] == -1 or
                        zero_crossing_img[i-1, j-1] = 0
                else:
                    zero_crossing_img[i-1, j-1] = 255
    return zero_crossing_img
```

do Difference of Gaussian first then do zero-crossing for every pixel.