R09922190 王祥任

CV Homework 10

Abstract

本次作業中，我們將實5種不同的edge detection 方式(zero-crossing 搭配不同種類的kernel): 2 Laplacian Mask, Minimum Variance Laplacian, Laplacian of Gaussian, and Difference of Gaussian。以上操作除輸入、輸出以外，不得直接套用現成套件。

Implementation

- Programming Language: Python3
- Python Package: opencv-python, matplotlib, numpy, copy
- Execution: python3 hw10-main.py

在執行時，請務必將 lena.bmp 放置在和 hw10-main.py 相同的檔案目錄下，並確認檔名相同。opencv-python用於讀取和寫出圖片，matplotlib僅用於runtime時即時顯示和儲存圖片，須配合jupyter使用，在此不贅述，而最後的copy和numpy則用於純計算上。

Zero-Crossing

```python
def zero_crossing(lena, mask, thres = 15):
    h, w = lena.shape
    m = mask.shape[0] // 2

    pad_lena = padding(lena, m)
    lap_tmp = np.zeros((h, w))

    for r in range(m, h + m):
        for c in range(m, w + m):
            gradient = np.sum( \
                pad_lena[(r-m):(r+m+1), c-m:(c+m+1)] * mask)
            if (gradient >= thres):
                lap_tmp[r - m][c - m] = 1
            elif (gradient <= -thres):
                lap_tmp[r - m][c - m] = -1
            else:
                lap_tmp[r - m][c - m] = 0

    pad_lap = padding(lap_tmp, 1)
    after = np.zeros((h, w))

    for r in range(1, h + 1):
        for c in range(1, w + 1):
            if (pad_lap[r][c] >= 1) and (pad_lap[r-1:r+2, c-1:c+2] == -1).any():
                after[r - 1][c - 1] = 0
            else:
                after[r - 1][c - 1] = 255

    return after
```

首先根據mask(也就是kernel)的大小，將原圖片進行對應大小的reflection padding。再針對所有原圖像素和其周邊的像素乘上Mask的數值並計算總和，若總和大於等於閾值則將該像素設為1，若總和小於等於負閾值則將該像素設為-1，若以上皆非則設為0。而遍歷完全部像素一遍後會得到原圖轉換為僅含1， 0， -1的圖片。而針對該張圖片，再做一次長度為1的reflection padding。然後再遍歷一次全部像素，若中心像素不大於等於1，則該點設為白色(255)，但若中心像素大於等於1且其鄰近像素中有人為-1，則代表zero crossing在此範圍發生，故將此點設為黑色(0)。而遍歷完全部像素後，我們便可以得到zero-crossing edge detection的結果。

```python
def Laplacian1(lena, thres=15):
    mask = np.array([   [0,  1, 0],
                        [1, -4, 1],
                        [0,  1, 0]])
    after = zero_crossing(lena, mask, thres)
    plt.imshow(after, cmap='gray', vmin=0, vmax=255)
    cv2.imwrite("lena-Laplacian-1-15.bmp", after)
    cv2.imwrite("lena-Laplacian-1-15.png", after)

Laplacian1(lena)
```

```python
def Laplacian2(lena, thres=15):
    mask = 1/3 * np.array([ [1,  1, 1],
                            [1, -8, 1],
                            [1,  1, 1]])
    after = zero_crossing(lena, mask, thres)
    plt.imshow(after, cmap='gray', vmin=0, vmax=255)
    cv2.imwrite("lena-Laplacian-2-15.bmp", after)
    cv2.imwrite("lena-Laplacian-2-15.png", after)

Laplacian2(lena)
```

```python
def MinVarLaplacian(lena, thres=20):
    mask = 1/3 * np.array([ [ 2, -1,  2],
                            [-1, -4, -1],
                            [ 2, -1,  2]])
    after = zero_crossing(lena, mask, thres)
    plt.imshow(after, cmap='gray', vmin=0, vmax=255)
    cv2.imwrite("lena-Min-Var-Laplacian-20.bmp", after)
    cv2.imwrite("lena-Min-Var-Laplacian-20.png", after)

MinVarLaplacian(lena)
```

```python
def LaplacianOfGaussian(lena, thres=3000):
    mask = 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]])
    after = zero_crossing(lena, mask, thres)
    plt.imshow(after, cmap='gray', vmin=0, vmax=255)
    cv2.imwrite("lena-Laplacian-Of-Gaussian-3000.bmp", after)
    cv2.imwrite("lena-Laplacian-Of-Gaussian-3000.png", after)

LaplacianOfGaussian(lena)
```

```python
def DifferenceOfGaussian(lena, thres=1):
    mask = 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]])
    after = zero_crossing(lena, mask, thres)
    plt.imshow(after, cmap='gray', vmin=0, vmax=255)
    cv2.imwrite("lena-Differnece-Of-Gaussian-1.bmp", after)
    cv2.imwrite("lena-Differnece-Of-Gaussian-1.png", after)

DifferenceOfGaussian(lena)
```

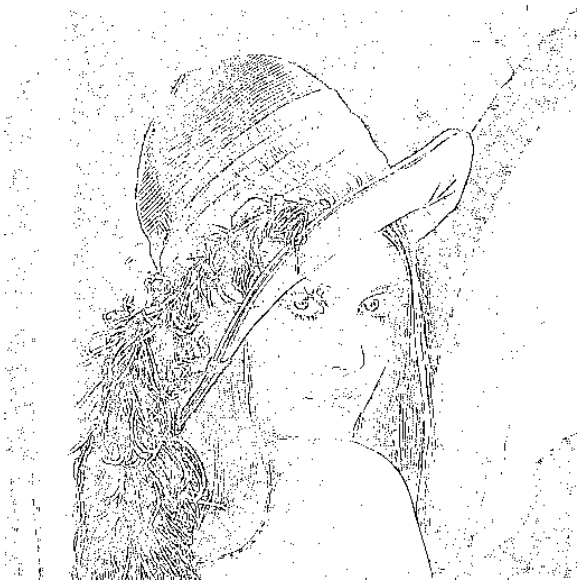
而有了zero-crossing的函數後，針對不同種類的kernel，我們只需要丟入特定的mask(kernel)和對應的threshold即可。而我採用的kernel和其threshold如下：

(1) Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15
(2) Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1): 15
(3) Minimum variance Laplacian: 20
(4) Laplace of Gaussian: 3000
(5) Difference of Gaussian: 1

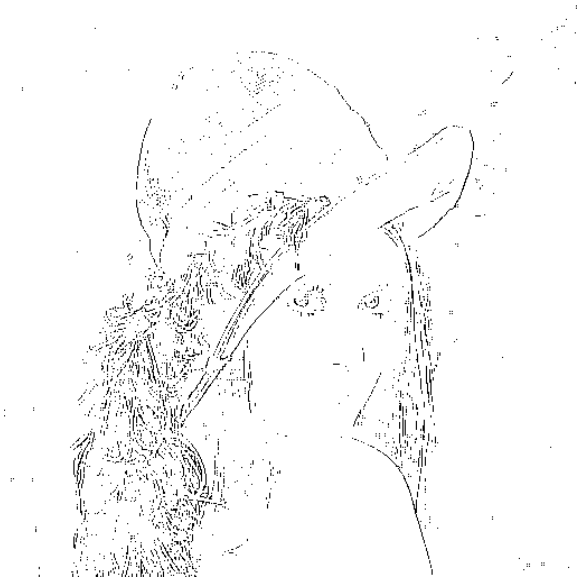
而以上全部的成果貼於最後的圖組中。

Results

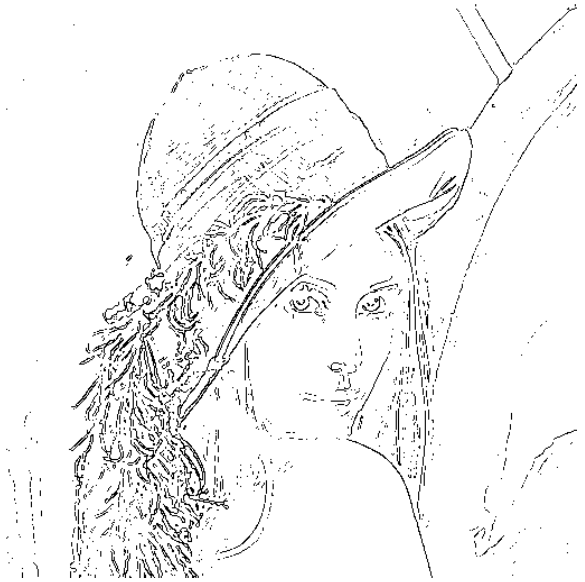
Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15



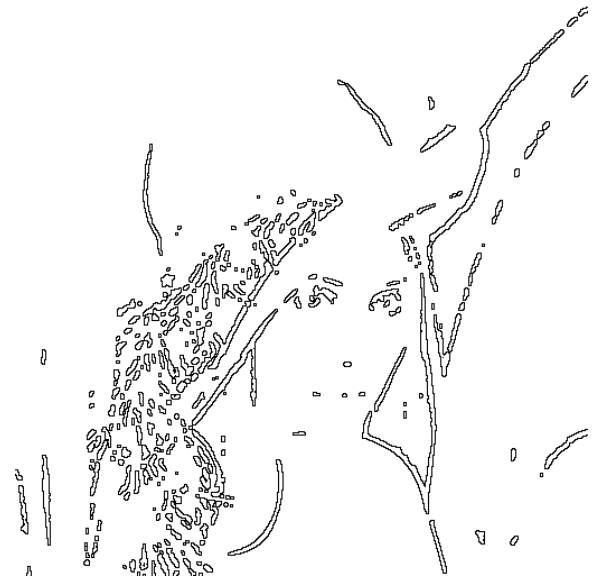
Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1): 15



Minimum variance Laplacian: 20



Laplace of Gaussian: 3000



Difference of Gaussian: 1