Computer Vision I _2018

Homework assignment #10

R07522717 機械所製造組碩一 林温雅

Part1（此次作業僅 one part）

Description:

implement 2 Laplacian Mask, Minimum Variance Laplacian, Laplacian of Gaussian, and Difference of Gaussian(inhibitory sigma=1, excitatory sigma=3, kernel size 11x11 [1][1])

Algorithm:

根據不同的 edge detector 所使用的 kernel，對影像做 convolution。最後自行挑選合適的 threshold，對影像做 reverse thresholding(黑白轉換一下方便看)。

Parameters:

i,j	#迴圈計數用參數
original_img	#原始圖檔
ker	#各式不同 kernel
rows, cols	#圖檔的長與寬
temp_img	#擴大後的圖檔，為了 convolution 邊界所製作
temp	#在迴圈中擷取影像中和 kernel 一樣大的矩陣，以方便計算
new_img	#用來接收新 data 的輸出圖檔

Principal code fragment:

```
    def Laplacian(img, mode=None):
    # mode1 是第一種 kernel、mode2 是第二種、 mode3 是
minimum-variance
    #ker = None
    if mode==1:
        ker = np.array([[0,1,0],[1,-4,1],[0,1,0]])
    elif mode==2:
        ker = np.array([[1,1,1],[1,-8,1],[1,1,1]]) / 3
    elif mode==3:
        ker = np.array([[2,-1,2],[-1,-4,-1],[2,-1,2]]) / 3

    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=1, bottom=1, left=1,
right=1, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i + 3, j:j + 3]
            new_img[i, j] = np.sum(ker * temp)
    return new_img

def Laplacian_Gaussian(img):
    ker = 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]])
```

```
    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=5, bottom=5, left=5,
right=5, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+11, j:j+11]
            new_img[i, j] = np.sum(ker * temp)
    return new_img


def Difference_Gaussian(img):
    ker = 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]])

    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=5, bottom=5, left=5,
right=5, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+11, j:j+11]
            new_img[i, j] = np.sum(ker * temp)
    return new_img
```

Kernels & Thresholds:

Laplacian1:

Kernel: ([[0,1,0],[1,-4,1],[0,1,0]])

Threshold:30

Laplacian2:

Kernel: ([[1,1,1],[1,-8,1],[1,1,1]]) / 3

Threshold:25

Minimum-Variance Laplacian:

Kernel: ([[2,-1,2],[-1,-4,-1],[2,-1,2]]) / 3

Threshold:25

Laplacian of Gaussian:

Kernel: ([[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]])

Threshold:7000

Difference of Gaussian:

Kernel: ([[-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]])

Threshold:11000

Resulting Image:

Laplacian1_30

Laplacian2_25

minimum_variance_Laplacian_20

Laplacian_of_Gaussian_7000

Difference_of_Gaussian_11000