Computer Vision I _2018

Homework assignment #9

R07522717 機械所製造組碩一 林温雅

Part1（此次作業僅 one part）

Description:

Write programs to generate the following gradient magnitude images and choose proper thresholds to get the binary edge images:

- Roberts operator
- Prewitt edge detector
- Sobel edge detector
- Frei and Chen gradient operator
- Kirsch compass operator
- Robinson compass operator
- Nevatia-Babu 5X5 operator

Algorithm:

根據不同的 edge detector 所使用的 kernel，對影像做 convolution。在 roberts、perwitt、sobel、frei & chen 中，將不同 kernel 的 convolution 值進行平方後相加開根號處理，設為新影像的值。而在 kirsch、robinson、nevatia-babu 中，將不同 kernel 的 convolution 值做比較，挑選 max 值設為新影像的值。

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_XX	#各式不同 kernel

list_XX #儲存 kernel 用的 list
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
max #用來計算最大值的參數

Principal code fragment:

```
def roberts(img):
    ker_r1 = np.array([[-1, 0],[0, 1]])
    ker_r2 = np.array([[0, -1],[1, 0]])
    rows, cols = img.shape
    # for center 在左上角
    temp_img = cv2.copyMakeBorder(src=img, top=0, bottom=1, left=0, 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+2, j:j+2]

            new_img[i,j] = np.sqrt(np.sum(np.multiply(ker_r1, temp))**2 + np.sum(np.multiply(ker_r2, temp))**2)
            #new_img[i,j] = np.abs(np.sum(np.multiply(ker_r1, temp))) + np.abs(np.sum(np.multiply(ker_r2, temp)))

    return new_img


def krisch(img):
    ker_k0 = np.array([[-3,-3,5], [-3,0,5], [-3,-3,5]])
    ker_k1 = np.array([[-3,5,5], [-3,0,5], [-3,-3,-3]])
```

```
    ker_k2 = np.array([[5,5,5], [-3,0,-3], [-3,-3,-3]])
    ker_k3 = np.array([[5,5,-3], [5,0,-3], [-3,-3,-3]])
    ker_k4 = np.array([[5,-3,-3], [5,0,-3], [5,-3,-3]])
    ker_k5 = np.array([[-3,-3,-3], [5,0,-3], [5,5,-3]])
    ker_k6 = np.array([[-3,-3,-3], [-3,0,-3], [5,5,5]])
    ker_k7 = np.array([[-3,-3,-3], [-3,0,5], [-3,5,5]])
    list_kn = [ker_k0, ker_k1, ker_k2, ker_k3, ker_k4, ker_k5, ker_k6, ker_k7]
    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]
            max=0 #  初始化 max 值
            for ker in list_kn:
                temp_sum = np.sum(ker * temp)
                if temp_sum > max:
                    max = temp_sum
            new_img[i, j] = max

    return new_img

def reverse_thresholding(img, threshold=128):
    new_img = np.empty(img.shape)
    new_img.fill(255)
    mask = img >= threshold
    new_img[mask] = 0
    return   new_img
```

Resulting Image:

roberts_30

perwitt_90

sobel_130

frei_chen_110

krisch_230

robinson_120

nevatia_babu_30000