Computer Vision I _2018

Homework assignment #6

R07522717 機械所製造組碩一 林温雅

Part1（此次作業僅 one part）

Description:

Downsampling Lena from 512x512 to 64x64: Binarize the benchmark image lena as in HW2, then using 8x8 blocks as a unit, take the topmost-left pixel as the downsampled data.

Algorithm:

1. Binarize:
   先製作一個全部像素亮度值皆為 0 的圖檔，再將原始圖檔相對應位置像素亮度值大於等於 128 者之亮度設為 255。
2. DownSampling:
   每 8*8 個像素取左上角為代表來 downsampling。
3. Yokoi:
   做一個 dictionary 存 q, s, r 被數的次數，根據 dictionary 存的資料來建立 Yokoi 矩陣。

Parameters:

1. In function “Binarize”:

rows, columns	#輸入圖檔的行列數
new_img	#新圖檔準備接受 Binarize 後的圖
i,j	#迴圈計數用參數

2. In function “DownSampling”:

rows, columns	#輸入圖檔的行列數
new_img	#新圖檔準備接受 DOwnSampling 後的圖
i,j	#迴圈計數用參數
scale	#看要用多少倍率來 scaling

3. In function “YokoiCalc”:

b, c, d, e	#用來判斷這組 2*2 像素組是 q 還是 r 還是 s

4. In function “YokoiNum”:

rows, columns	#輸入圖檔的行列數
temp_img	#外擴圖檔，為了邊界數值而設計的
new_img	#新圖檔準備接受 Yokoi 計算後的圖
i,j	#迴圈計數用參數
dict_f	#用來儲存 q, r, s 的判斷結果

5. In function “show_text_image”:

rows, columns	#輸入圖檔的行列數
text_img	#新圖檔準備接受 Yokoi 的矩陣轉成影像的圖
i,j	#迴圈計數用參數

6. Outside of function

original_img	#讀取原始圖檔

Principal code fragment:

```
# 進行 Yokoi 計算的輔助 function，供 YokoiNum function 使用
def YokoiCalc(b, c, d, e):
    if b == c:
        if (d != b) or (e != b):
            return 'q'
        elif (d == b) and (e == b):
            return 'r'
    elif b != c:
        return 's'

# 對整張圖檔進行 Yokoi 計算的 function，使用 4-connectivity
def YokoiNum(img):
    #獲得輸入圖檔之行列數
    rows, columns = img.shape
    #擴大圖檔每邊各一條
    temp_img = np.zeros((rows+2, columns+2), np.int)
    temp_img[1:rows+1, 1:columns+1] = img
    #製作一個新圖檔準備接受 dilation 後的圖
    new_img = np.zeros((rows, columns), np.int)
    #f = np.chararray(4, itemsize=1)
    dict_f = dict({'q': 0, 's':0, 'r':0})
    for i in range(1, 1+rows):
        for j in range(1, 1+columns):
            if temp_img[i,j] != 255:
                continue
            dict_f['q'], dict_f['s'], dict_f['r']= 0, 0, 0
            dict_f[YokoiCalc(temp_img[i,j], temp_img[i,j+1],
temp_img[i-1,j+1], temp_img[i-1,j])] +=1
            dict_f[YokoiCalc(temp_img[i,j], temp_img[i-1,j],
temp_img[i-1,j-1], temp_img[i,j-1])] +=1
            dict_f[YokoiCalc(temp_img[i,j], temp_img[i,j-1],
temp_img[i+1,j-1], temp_img[i+1,j])] +=1
            dict_f[YokoiCalc(temp_img[i,j], temp_img[i+1,j],
```

```
temp_img[i+1,j+1], temp_img[i,j+1])] +=1

                if dict_f['r'] is 4:
                    new_img[i-1, j-1] = 5
                else:
                    new_img[i-1, j-1] = dict_f['q']
    return new_img

#將一個矩陣輸出成一個 image 檔用的 function，能將 yokoi 結果輸出為清晰易讀之圖檔
def show_text_image(img, scale):
    text_img = np.empty(tuple(scale*i for i in img.shape))
    rows, columns = img.shape
    text_img.fill(255)
    for i in range(0,scale*rows, scale):
        for j in range(0,scale*columns, scale):
            if img[int(i/scale),int(j/scale)] ==0:
                continue

cv2.putText(text_img,str(img[int(i/scale),int(j/scale)]),(int(j+scale/2.2),int(i+scale/1.8)),cv2.FONT_HERSHEY_COMPLEX,2,(100,10,80),5)
    return text_img
```

Resulting images

binarize_lena

downsampling_lena

yokoi_number