Computer Vision I _2018

Homework assignment #7

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.

Then do the thinning operation.

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。
4. Marked:
   new 一個 boolean array，要 mark 的就是 True，其他是 False
5. 最後處理：
   用一個 while loop，不斷迭代直到結果不在變化為止

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 “Marked”:
   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
6. Outside of function
   original_img　　#讀取原始圖檔
   anythingchanged　　#判斷要不要結束迭代

Principal code fragment:

```
# 如果 yokoi 是 1，就可以砍(removable)，要 input 的是原始 binary image
def Connected_Shrink(img):
    # 用一個 boolean array，是 removable 的就是 True，其他是 False
    new_img = np.full(img.shape, False, dtype=bool)
    temp_img = YokoiNum(img)
    # 獲得輸入圖檔之行列數
    rows, columns = img.shape
    for i in range(rows):
        for j in range(columns):
            if temp_img[i, j] == 1:  # or temp_img[i, j] == 0:
                new_img[i, j] = True

    return new_img

#把 p 設為 true, q 設為 false
def Marked(img):
    rows, columns = img.shape
    # new_img = np.chararray(img.shape, unicode=True)
    #temp_img = np.chararray((rows + 2, columns + 2), unicode=True)
    temp_img = np.zeros((rows + 2, columns + 2), np.int)
    temp_img[1:rows + 1, 1:columns + 1] = img.copy()

    # new 一個 boolean array，要 mark 的就是 True，其他是 False
    new_img = np.full(img.shape, False, dtype=bool)
    for i in range(1, rows + 1):
        for j in range(1, columns + 1):
            if temp_img[i, j] == 1:
                templist = [temp_img[i][j+1], temp_img[i-1][j],
temp_img[i][j-1], temp_img[i+1][j]]
                if 1 in templist:
                    new_img[i - 1, j - 1] = True
    return new_img
```

```
# 將圖檔二值化
binarize_lena = Binarize(original_img)
# 將二值化之圖檔進行邊長 8 倍的 downscaling
downsampling_lena = DownSampling(binarize_lena, 8)
processed_original_img = downsampling_lena.copy()
final_img = processed_original_img.copy()

while True:
    anythingchanged = False
    # yokoi 數字圖
    # 本身是 1，而且 4-connected 周邊至少也有一個 1
    yokoi = YokoiNum(processed_original_img)
    marked_img = Marked(yokoi)
    for i in range(64):
        for j in range(64):
            if Yokoi_Single_Point(processed_original_img, i, j) == 1 and
marked_img[i, j]:
                final_img[i, j] = 0
                processed_original_img = final_img.copy()
                anythingchanged = True
    if not anythingchanged:
        break
    else:
        processed_original_img = final_img.copy()
```

Resulting images

binarize_lena

downsampling_lena

image after thinning