Computer Vision I _2018

Homework assignment #7

R07522717 機械所製造組碩一 林温雅

```
#使用 python
#import 套件
# -*- coding: utf-8 -*-

import cv2
import numpy as np

#  讀取原始影像
original_img = cv2.imread('lena.bmp', 0)

#  進行二值化用的 function
def Binarize(img):
    rows, columns = img.shape
    new_img = np.zeros((rows, columns), np.int)
    for i in range(rows):
        for j in range(columns):
            if img[i, j] >= 128:
                new_img[i, j] = 255
    return new_img



#  進行 DownSampling 用的 function
def DownSampling(img, scale):
    rows, columns = img.shape
    new_img = np.zeros((int(rows / scale), int(columns / scale)), np.int)
    for i in range(0, rows, scale):
        for j in range(0, columns, scale):
            new_img[int(i / scale), int(j / scale)] = img[i, j]

    return new_img
```

```
# 進行 Yokoi 計算的輔助 function，供 YokoiNum function 使用，使用 4-connectivity
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'

def Yokoi_Single_Point(img, i, j):
    # 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處理後的圖
    new_img = np.zeros((rows, columns), np.int)
    dict_f = dict({'q': 0, 's': 0, 'r': 0})
    # dict_f['q'], dict_f['s'], dict_f['r'] = 0, 0, 0
    i += 1
    j += 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
    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:
        return 5
    else:
```

```
        return dict_f['q']


# 對整張圖檔進行 Yokoi 計算的 function(4,8 通用)
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.copy()
    # 製作一個新圖檔準備接受處理後的圖
    new_img = np.zeros((rows, columns), np.int)
    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


# 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()


cv2.imwrite('thin_lena.bmp', final_img)
```