Computer Vision I _2018

Homework assignment #6

R07522717 機械所製造組碩一 林温雅

```
#使用 python
#import 套件
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
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

# 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)
# 將 downsampling 後之圖檔進行 Yokoi 計算
yokoi = YokoiNum(downsampling_lena)
# 將 yokoi 計算後的圖檔輸出成 image，容易判讀
text_lena = show_text_image(yokoi, 100)

#輸出上述處理後之圖檔
cv2.imwrite('binarize_lena.bmp', binarize_lena)
cv2.imwrite('downsampling_lena.bmp', downsampling_lena)
cv2.imwrite('yokoi_num.bmp', text_lena)
```