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)