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。

1. DownSampling:

每8\*8個像素取左上角為代表來downsampling。

1. 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

1. 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

1. 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

1. 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

1. 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

1. Outside of function

original\_img #讀取原始圖檔

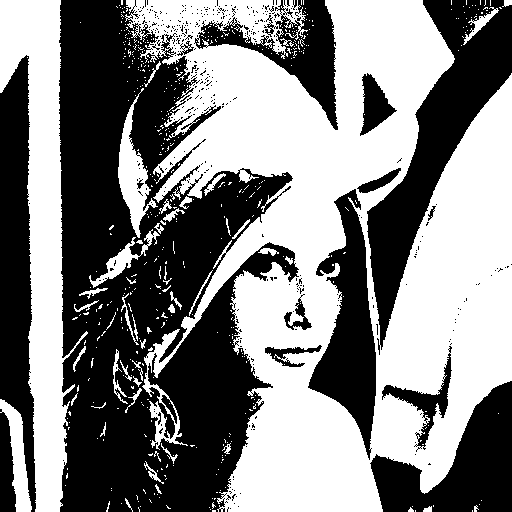
Principal code fragment:

# 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

