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。

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。

1. Marked:

new一個boolean array，要mark的就是True，其他是False

1. 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

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 “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

1. 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:

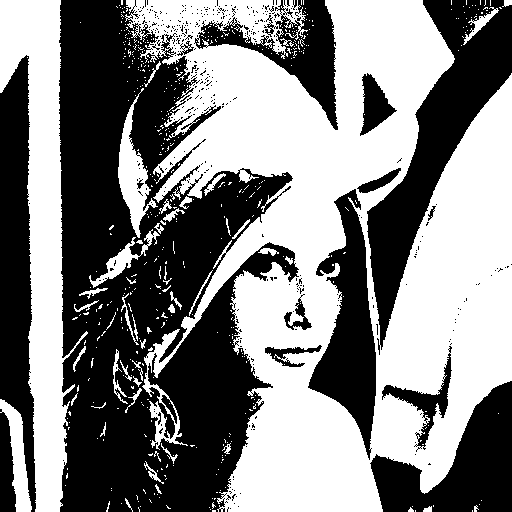
break

else:

processed\_original\_img = final\_img.copy()

Resulting images

binarize\_lena



downsampling\_lena

image after thinning

