電腦視覺作業報告

**Homework 7**

學號：D03922030

姓名：林君憲

中華民國103年12月1日

程式語言：Python

程式名稱：Thinning.py

說明：

1. 本程式會讀入lena.bmp影像檔，將其二元化後重新取樣為64 x 64大小的縮圖，接著依序掃描re-sample後的每個像素，做thinning operator part A(移除左上方向的點)及thinning operator part B(移除右下方向的點)。
2. 引用Python之Pillow影像程式庫（http://pillow.readthedocs.org/en/latest/index.html）來處理圖檔讀寫的工作。
3. 依據Thinning operator的演算法定義，thinning的過程可以分為兩段，一個是移除圖形右上方向的像素點，接著是移除圖形左下方向的像素點，重複上述步驟至圖形內容不再改變時，即完成thinning operation。
4. 移除圖形右上方像素點的程式片段如下：

for i in range(1, downWidth-1):

for j in range(1, downHeight-1):

con1 = False

con2 = False

con3 = False

con4 = False

# 0 (255 is white background), 1 (zero is region). Inverse each neighborhood pixels.

P1 = (255 - thinIm.getpixel((i+1, j))) / 255

P2 = (255 - thinIm.getpixel((i, j-1))) / 255

P3 = (255 - thinIm.getpixel((i-1, j))) / 255

P4 = (255 - thinIm.getpixel((i, j+1))) / 255

P5 = (255 - thinIm.getpixel((i+1, j+1))) / 255

P6 = (255 - thinIm.getpixel((i+1, j-1))) / 255

P7 = (255 - thinIm.getpixel((i-1, j-1))) / 255

P8 = (255 - thinIm.getpixel((i-1, j+1))) / 255

if thinIm.getpixel((i, j)) == 0:

neighborValue = P1 + P2 + P3 + P4 + P5 + P6 + P7 + P8

if neighborValue >= 2 and neighborValue <= 6:

con1 = True

counter = 0

if P7 < P2:

counter = counter + 1

if P2 < P6:

counter = counter + 1

if P6 < P1:

counter = counter + 1

if P1 < P5:

counter = counter + 1

if P5 < P4:

counter = counter + 1

if P4 < P8:

counter = counter + 1

if P8 < P3:

counter = counter + 1

if P3 < P7:

counter = counter + 1

if counter == 1:

con2 = True

# If on of P2, P1, P4 is background (0), then set condition 3 to TRUE.

if P2 \* P1 \* P4 == 0:

con3 = True

# If on of P1, P4, P3 is background (0), then set condition 4 to TRUE.

if P1 \* P4 \* P3 == 0:

con4 = True

# If the 4 conditions above are all True, remove the pixel that scan.

if con1 == con2 == con3 == con4 == True:

thinIm.putpixel((i,j), 255) # 255 is white, removed.

1. 移除圖形左下方像素點的程式片段如下：

for i in range(1, downWidth-1):

for j in range(1, downHeight-1):

con1 = False

con2 = False

con3 = False

con4 = False

# 0 (255 is white background), 1 (zero is region). Inverse each neighborhood pixels.

P1 = (255 - thinIm.getpixel((i+1, j))) / 255

P2 = (255 - thinIm.getpixel((i, j-1))) / 255

P3 = (255 - thinIm.getpixel((i-1, j))) / 255

P4 = (255 - thinIm.getpixel((i, j+1))) / 255

P5 = (255 - thinIm.getpixel((i+1, j+1))) / 255

P6 = (255 - thinIm.getpixel((i+1, j-1))) / 255

P7 = (255 - thinIm.getpixel((i-1, j-1))) / 255

P8 = (255 - thinIm.getpixel((i-1, j+1))) / 255

if thinIm.getpixel((i, j)) == 0: # zero is region

# If region pixels between 2 and 6, condition one is true.

neighborValue = P1 + P2 + P3 + P4 + P5 + P6 + P7 + P8

if neighborValue >= 2 and neighborValue <= 6:

con1 = True

# Go through clockwise from top, once pixel from 0 (gray 255 is background) turn to 1 (gray zero is region), counter++. Counter = 1, TRUE.

counter = 0

if P7 < P2:

counter = counter + 1

if P2 < P6:

counter = counter + 1

if P6 < P1:

counter = counter + 1

if P1 < P5:

counter = counter + 1

if P5 < P4:

counter = counter + 1

if P4 < P8:

counter = counter + 1

if P8 < P3:

counter = counter + 1

if P3 < P7:

counter = counter + 1

if counter == 1:

con2 = True

# If one of P3, P2, P1 is background, set condition 3 to TRUE

if P3 \* P2 \* P1 == 0:

con3 = True

# If one of P4, P3, P2 is background, set condition 4 to TRUE

if P4 \* P3 \* P2 == 0:

con4 = True

# If the 4 conditions above are all True, remove the pixel that scan.

if con1 == con2 == con3 == con4 == True:

thinIm.putpixel((i,j), 255) # 255 is white background, pixel removed.

1. 以上兩段程式碼使用一個while loop包圍起來，並且比較每次做完thinning operation後的圖形內容（亦即比較preThinIm及 thinIm兩圖形變數內容），直到前後兩次的圖形內容相同時，代表thinning operation 已經完成，就跳出while loop。
2. 以下為完成thinning operation的圖形，以\*代表有像素的點。

\*\*\*\* \*\*

\*\* \* \* \*\* \* \*\*\* \*

\* \* \* \* \* \*\*

\* \* \* \* \*\*\*\*\* \*\* \*

\*\* \* \* \*\*\*\* \* \* \* \*\* \* \*

\* \*\* \*\* \* \*\* \*\* \*\*\* \* \*\*\*\*

\*\*\*\* \*\* \*\*\*\* \* \*\*\* \*

\* \* \*\*\*\*\*\*\* \*

\* \*\* \* \* \*

\* \*\*\*\*\* \* \*

\* \*\*\* \* \* \*\*

\* \*\* \* \* \*\*

\* \* \*\* \* \*\*

\* \*\*\*\*\*\*\* \* \* \*\*

\* \* \* \* \* \*

\* \* \* \* \* \*\*

\* \* \* \*\* \* \*

\* \*\* \* \*\*\* \*

\* \* \*\*\* \*\*

\* \* \* \* \*

\* \* \*\* \* \*\*

\* \* \* \* \*\* \*\*

\* \* \*\* \* \*\* \*\*\*\*

\* \* \*\* \* \* \*\*\* \*\*\*

\* \* \* \* \* \*\* \*\* \*

\* \* \*\* \*\* \* \*

\* \*\* \* \*\* \*\*\* \*\*\* \*\* \*\*\*

\* \* \*\* \* \* \*\*\* \* \* \* \*

\* \* \*\*\*\* \*\*\* \*\*\*\*\*\*\*\*

\* \* \*\* \* \* \* \*

\* \*\*\*\* \*\*\* \* \* \*

\* \* \* \*\*\*\* \*\*\*\* \* \*

\* \*\* \*\*\* \*\*\* \*\*\*\* \*\*\* \*\*

\* \* \* \* \* \*\*\* \* \* \* \*

\* \*\*\* \*\*\*\* \* \* \* \* \* \* \*

\* \* \*\* \* \*\* \* \* \* \*\*\* \*\*

\* \*\*\* \*\*\* \* \* \* \*

\* \* \*\*\*\*\*\* \*\*\*\* \*\* \* \*\*

\* \* \*\* \* \* \* \* \* \*\*\* \*

\* \* \*\*\*\*\*\*\*\*\*\*\*\* \* \* \* \*

\* \*\*\*\* \* \* \* \*\*\*\* \* \* \* \*

\* \*\*\*\*\*\* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \*\*\* \* \* \* \* \*

\* \* \* \*\*\* \* \*\* \*\*\* \*\* \* \*

\* \*\*\*\*\*\* \*\* \*\* \* \* \*\*\*\*\*\* \*

\* \*\*\*\*\*\* \*\*\*\*\* \* \* \*

\* \* \*\*\*\*\* \*\*\*\*\*\*\*\* \* \*

\* \*\*\*\*\*\* \* \* \* \*

\* \*\* \* \* \* \* \*\*\*\*\* \* \*

\* \* \* \* \*\*\* \* \* \*\*\*\* \*

\* \*\*\*\*\* \* \* \*\*\* \* \* \*

\* \* \* \*\*\*\*\*\*\*\*\*\*\* \*\* \* \*

\* \* \*\*\* \* \* \* \* \* \* \*

\* \* \* \*\*\*\*\*\*\*\* \* \* \* \*

\* \*\* \*\*\* \* \* \*\*\* \* \* \*

\* \* \* \*\*\*\*\*\*\*\* \* \* \* \*

\* \* \*\*\*\* \*\*\* \*\* \*

\* \* \* \* \* \*\*\* \*

\* \* \*\*\*\*\* \*\* \*\*\* \*

\* \* \*\* \* \* \* \*

\* \* \*\*\*\*\*\*\*\*\*\*\*\*\*\* \* \*

\* \* \* \* \*\*\*\*\*\*\*\*\*\*\*\*\*

\* \* \*\*