電腦視覺作業報告

**Homework 5**

學號：D03922030

姓名：林君憲

中華民國103年11月18日

程式語言：Python

程式名稱：grayErosion.py

函式名稱：grayErosion()

函式引數：imageBuffer、kernel

說明：

1. 本函式模組係供main.py呼叫的函式，需要傳入兩個引數：一個是圖檔資料imageBuffer、一個是kernel串列。在主程式中開啟lena.bmp後傳入grayErosion()函式中做mathematical morphology erosion，最後把erosion的圖片資料傳回。
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. 依據灰階erosion的定義，取原圖被kernel涵蓋範圍內的像素與kernel的值相減後之最小值，以該值作為新的灰階值，即完成灰階erosion運算。據此，將上開步驟寫成之程式碼片段如下：

for i in range(width):

for j in range(height):

minValue = 255

for k in range(len(kernel)):

p = kernel[k][0]

q = kernel[k][1]

value = kernel[k][2]

temp = inputIm.getpixel((i,j))

x = i + p

y = j + q

if (x >= 0) and (x < width) and (y >= 0) and (y < height):

temp = inputIm.getpixel((x,y)) - value

if temp < minValue:

minValue = temp

eroIm.putpixel((i,j), minValue)

1. 本習題使用的kernel為3-5-5-5-3，定義在main()函式中的octangon變數中，以供傳入其他mathematical morphology之用。

經過灰階erosion後的lena.bmp如下：



程式語言：Python

程式名稱：grayDilation.py

函式名稱：grayDilation()

函式引數：imageBuffer、kernel

說明：

1. 本函式模組係供main.py呼叫的函式，需要傳入兩個引數：一個是圖檔資料imageBuffer、一個是kernel串列。在主程式中開啟lena.bmp後傳入grayDilation()函式中做mathematical morphology dilation，最後把dilation的圖片資料傳回。
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. 依據灰階Dilation的定義，以kernel在原圖上掃描，取原圖上的有像素點強度值與kernel的強度值相加後取最大值，並將此最大值作為新圖的像素強度值，即為灰階dilation運算。進行dilation運算的程式碼片段如下：

for i in range(width):

for j in range(height):

maxValue = 0

for k in range(len(kernel)):

p = kernel[k][0]

q = kernel[k][1]

value = kernel[k][2]

temp = inputIm.getpixel((i,j))

x = i - p

y = j - q

if (x >= 0) and (x < width) and (y >= 0) and (y < height):

temp = inputIm.getpixel((x,y)) + value

if temp > maxValue:

maxValue = temp

dilIm.putpixel((i,j), maxValue)

經過灰階dilation後的lena.bmp如下：



程式名稱：main.py

函式名稱：main()

函式引數：無

說明：

1. 本函式模組會將lena.bmp檔讀入之後呼叫grayErosion()和grayDilation()函式，接著做灰階opening及closing。
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. 由於closing是先做dilation後再把所得結果做erosion，所以，在呼叫了grayDilation()之後把結果存放到dilImage中，再把dilImage及kernel一併傳入grayErosion()後，就可以得到closing的結果了。所使用的kernel是3-5-5-5-3，kernel中各點的值為0，每個點以tuple的型態存放在octangon中。
4. 由於opening程序則是和closing相反，是先做erosion後再把所得結果做dilation，所以，在呼叫了grayErosion()之後把結果存放到eroImage中，再把eroImage及kernel一併傳入grayDilation()後，就可以得到opening的結果了。所使用的kernel與上述的 octangon相同。

主程式中做呼叫灰階ersion、dilation、opening、closing運算的程式碼片段如下：

octangon = [(-1,2,0),(0,2,0),(1,2,0),(-2,1,0),(-1,1,0),(0,1,0),(1,1,0),(2,1,0),(-2,1,0),(-1,1,0),(0,1,0),(1,1,0),(2,1,0),(-2,0,0),(-1,0,0),(0,0,0),(1,0,0),(2,0,0),(-2,-1,0),(-1,-1,0),(0,-1,0),(1,-1,0),(2,-1,0),(-1,-2,0),(0,-2,0),(1,-2,0)]

im = Image.open(*"lena.bmp"*, *"r"*)

# Erosion

eroImage = grayErosion.grayErosion(im, octangon)

eroImage.show()

eroImage.save(*"Erosion\_lena.bmp"*)

# Dilation

dilImage = grayDilation.grayDilation(im, octangon)

dilImage.show()

dilImage.save(*"Dilation\_lena.bmp"*)

# Opening

openingImage = grayDilation.grayDilation(eroImage, octangon)

openingImage.show()

openingImage.save(*"Opening\_lena.bmp"*)

# Closing

closingImage = grayErosion.grayErosion(dilImage, octangon)

closingImage.show()

closingImage.save(*"Closing\_lena.bmp"*)

灰階Opening運算後的lena.bmp



灰階Closing運算後的lena.bmp

