

CV Homework 4

Abstract

本次作業中，我們要使用 octagonal 3-5-5-5-3 kernel 來對原圖實作 (1) Dilation (2) Erosion (3) Opening (4) Closing 並使用兩個 "L" shaped kernel 來實作 (5) Hit-and-Miss Transform. 以上都需針對白色點進行操作，另外以上操作除輸入、輸出以外，不得直接套用現成套件。

Implementation

- Programming Language: Python3
- Python Package: opencv-python, matplotlib, numpy, copy
- Execution: python3 hw4-main.py

在執行時，請務必將 lena.bmp 放置在和 hw4-main.py 相同的檔案目錄下，並確認檔名相同。opencv-python用於讀取和寫出圖片，matplotlib僅用於runtime時即時顯示和儲存圖片，須配合jupyter使用，在此不贅述，而最後的copy和numpy則用於純計算上。

Dilation and Erosion

```

25  def dilation(bina, kernel):
26      after = copy.deepcopy(bina)
27      for r in range(bina.shape[0]):
28          for c in range(bina.shape[1]):
29              # For White Only
30              if (bina[r][c] == 255):
31                  for k in kernel:
32                      # Check Bound
33                      if (r + k[0] >= 0 and r + k[0] < bina.shape[0] and
34                          c + k[1] >= 0 and c + k[1] < bina.shape[1]):
35                          after[r + k[0]][c + k[1]] = 255
36      return after

42  def erosion(bina, kernel):
43      after = copy.deepcopy(bina)
44      for r in range(bina.shape[0]):
45          for c in range(bina.shape[1]):
46              # Assume it will be white
47              after[r][c] = 255
48              for k in kernel:
49                  # Check Bound
50                  if (r + k[0] >= 0 and r + k[0] < bina.shape[0] and
51                      c + k[1] >= 0 and c + k[1] < bina.shape[1]):
52                      if (bina[r + k[0]][c + k[1]] != 255):
53                          after[r][c] = 0
54      return after

```

Dilation之輸入的圖片為已經以128為界進行二值化的圖片，詳細在之前的作業中有提到，在此不細講。搜尋原圖中為白色的點，並對套用Kernel後範圍內的像素，在變形後的圖中對應的像素設定為白色。成果貼於報告最後的圖組中。

Erosion輸入的原圖同Dilation。對所有的點先預設其為白色，假設原圖中其套用Kernel後範圍內的像素皆為白色，則變形後的圖的該像素維持白色，但若範圍內有人為黑色，則將變形後的圖的該像素設定為黑色。成果貼於報告最後的圖組中。

Opening and Closing

```
61  def opening(bina, kernel):
62      return dilation(erosion(bina, kernel), kernel)
```

$$B \circ K = (B \ominus K) \oplus K$$

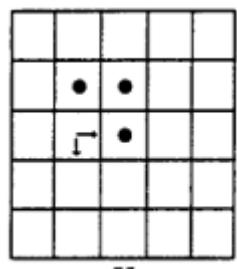
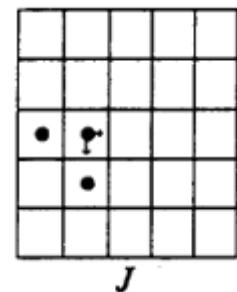
```
68  def closing(bina, kernel):
69      return erosion(dilation(bina, kernel), kernel)
```

$$B \bullet K = (B \oplus K) \ominus K$$

有了dilation和erosion，我們便可套用公式來實作opening和closing。opening便是先作erosion後再使用同樣的kernel對erosion後的結果作dilation，而closing則是順序相反過來。成果同樣貼於報告最後的圖組中。

Hit-and-Miss Transform

```
78  def hit_and_miss(bina, kernel_1, kernel_2):
79      # Get bina's complement
80      c_bina = 255 - bina
81      # Do erosion
82      AJ = erosion(bina, kernel_1)
83      AcK = erosion(c_bina, kernel_2)
84      # Get Intersection
85      after = copy.deepcopy(bina)
86      for r in range(bina.shape[1]):
87          for c in range(bina.shape[1]):
88              if AJ[r][c] == AcK[r][c]:
89                  after[r][c] = AJ[r][c]
90              else:
91                  after[r][c] = 0
92      return after
```



$$A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$$

hit and miss 根據公式，首先對二值化原圖跟L形Kernel, J 作erosion，然後用原圖的互補(即255-原圖數值)和L形Kernel, K 作erosion，最後取兩者的交集(即同一位置上有相同顏色者，則設定該像素為白色，否則為黑色)即可。成果同樣貼於報告最後的圖組中

Results

下一頁的成果圖組因為排版的因素，故圖片的尺寸都略有縮小。

