

Answer

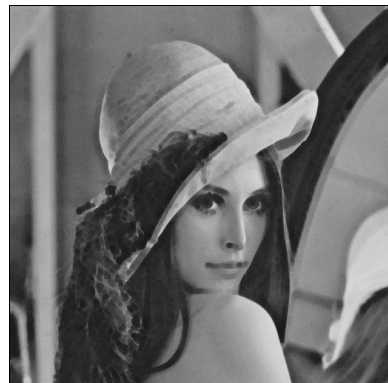
lena.bmp



(a) dilation_lena.bmp



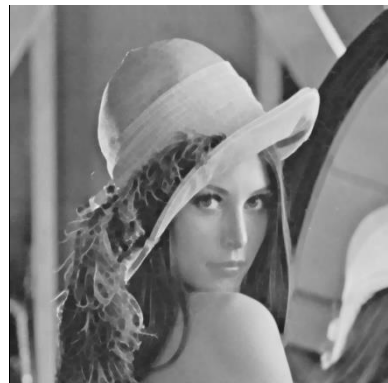
(b) erosion_lena.bmp



(c) opening_lena.bmp



(d) closing_.bmp



Description

1. Problem Formation

I. Write programs which do gray-scale morphology on a gray-scale image(lena.bmp):

(a) Dilation (b) Erosion (c) Opening (d) Closing

II. Please use the octogonal 3-5-5-5-3 kernel.

2. Method of Algorithms

(a) Dilation:

$$(f \oplus k)(x) = \max\{f(x-z) + k(z) \mid z \in K\}$$

(b) Erosion:

$$(f \ominus k)(x) = \min\{f(x+z) - k(z) \mid z \in K\}$$

(c) Opening:

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

(d) Closing:

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

3. 本次作業使用 python, IDE 為 Spyder

4. Source Code [HW5.py]說明如下

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Sun Oct 20 10:32:58 2019
4
5  @author: vincent黃國郡
6  """
7  from PIL import Image
8  import numpy as np
9  import myMorphology
10 kernel = np.array([
11     [0, 1, 1, 1, 0],
12     [1, 1, 1, 1, 1],
13     [1, 1, 1, 1, 1],
14     [1, 1, 1, 1, 1],
15     [0, 1, 1, 1, 0]])
16 centerKernel = tuple([x // 2 for x in kernel.shape])
17
18 if __name__ == '__main__':
19     lena = Image.open("lena.bmp")
20     # HW5a)
21     dilation_lena = myMorphology.dilation(lena, kernel, centerKernel)
22     dilation_lena.save('dilation_lena.bmp')
23
24     # HW5(b)
25     erosion_lena = myMorphology.erosion(lena, kernel, centerKernel)
26     erosion_lena.save('erosion_lena.bmp')
27
28     # HW5(c)
29     opening_lena = myMorphology.opening(lena, kernel, centerKernel)
30     opening_lena.save('opening_lena.bmp')
31
32     # HW5(d)
33     closing_lena = myMorphology.closing(lena, kernel, centerKernel)
34     closing_lena.save('closing_lena.bmp')
35
```

Import 需要用到的 library

宣告 kernel[35553]

定義 kernel 中心點

呼叫我的 myMorphology.py 副函式進行處理作業

5. Source Code (副程式) [[myMorphology.py](#)]說明如下

```

13 ▼ def dilation(originalImage, kernel, centerKernel):
14     dilationImage = Image.new('L', originalImage.size)
15     for r in range(originalImage.size[0]):
16         for c in range(originalImage.size[1]):
17             localMaxPixel = 0
18             for ii in range(kernel.shape[0]):
19                 for jj in range(kernel.shape[1]):
20                     if (kernel[ii, jj]):
21                         targetX = r + (ii - centerKernel[0])
22                         targetY = c + (jj - centerKernel[1])
23                         if ((0 <= targetX < originalImage.size[0])
24                             and (0 <= targetY < originalImage.size[1])):
25                             originalPixel = originalImage.getpixel(
26                                 (targetX, targetY))
27                             localMaxPixel = max(originalPixel, localMaxPixel)
28             dilationImage.putpixel((r, c), localMaxPixel)
29     return dilationImage
30
31
32 ▼ def erosion(originalImage, kernel, centerKernel):
33     erosionImage = Image.new('L', originalImage.size)
34     for r in range(originalImage.size[0]):
35         for c in range(originalImage.size[1]):
36             localMinPixel = 255
37             for ii in range(kernel.shape[0]):
38                 for jj in range(kernel.shape[1]):
39                     if (kernel[ii, jj]):
40                         targetX = r + (ii - centerKernel[0])
41                         targetY = c + (jj - centerKernel[1])
42                         if ((0 <= targetX < originalImage.size[0])
43                             and (0 <= targetY < originalImage.size[1])):
44                             originalPixel = originalImage.getpixel(
45                                 (targetX, targetY))
46                             localMinPixel = min(originalPixel, localMinPixel)
47             erosionImage.putpixel((r, c), localMinPixel)
48     return erosionImage
49
50
51 def opening(originalImage, kernel, centerKernel):
52     return dilation(erosion(originalImage, kernel, centerKernel), kernel, centerKernel)
53
54
55 def closing(originalImage, kernel, centerKernel):
56     return erosion(dilation(originalImage, kernel, centerKernel), kernel, centerKernel)

```

先產生一張跟 input 一樣大的 8bit original Image

初始一個 localMaxPixel 來抓 kernel 對應到的方框最小值

掃過每一個 pixel kernel 對應上的 each pixel 來抓最大值

最後把 localMaxPixel 值填回去 kernel 中心點作用的 Pixel

如同 dilation 程序，只是換抓取 localmin，最小值填回該中心 pixel 值

先 erosion 再 dilation

先 dilation 再 erosion