Homework 4 R08921005 黃國郡

Answer

lena.bmp	(a) dilation_lena.bmp
	
(b) erosion_lena.bmp	(c) opening_lena.bmp
	
(d) closing_.bmp	(e) hitmiss_lena.bmp
	

Homework 4 R08921005 黃國郡

Description

1. Problem Formation

   I. Write programs which do binary morphology on a binary image:

   (a) Dilation (b) Erosion (c) Opening (d) Closing (e) Hit-and-miss transform

   II. Binarize Lena with the threshold 128 (0-127,128-255).

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

   IV. Please use the "L" shaped kernel (same as the text book) to detect the upper-right corner for hit-and-miss transform.

   V. Please process the white pixels (operating on white pixels).

2. Method of Algorithms

   (a) Dilation:

$$A \oplus B = \{c \in E^N \mid c = a + b \text{ for some } a \in A \text{ and } b \in B\}$$

   (b) Erosion:

$$A \ominus B = \{x \in E^N \mid x + b \in A \text{ for every } b \in B\}$$

   (c) Opening:

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

   (d) Closing:

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

   (e) Hit-and-Miss:

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

3. 本次作業使用 python, IDE 使用 Spyder

4. Source Code [HW4.py]說明如下

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 17 20:56:19 2019

@author: vincent黃國郡
"""
from PIL import Image
import numpy as np
import myMorphology
kernel = np.array([
    [0, 1, 1, 1, 0],
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1],
    [0, 1, 1, 1, 0]])
centerKernel = tuple([x // 2 for x in kernel.shape])

if __name__ == '__main__':
    lena = Image.open("lena.bmp")
    # binarization
    binary_lena = lena.point(lambda x: 0 if x < 128 else 255, '1')

    # HW4(a)
    dilation_lena = myMorphology.dilation(binary_lena, kernel, centerKernel)
    dilation_lena.save('dilation_lena.bmp')

    # HW4(b)
    erosion_lena = myMorphology.erosion(binary_lena, kernel, centerKernel)
    erosion_lena.save('erosion_lena.bmp')

    # HW4(c)
    opening_lena = myMorphology.opening(binary_lena, kernel, centerKernel)
    opening_lena.save('opening_lena.bmp')

    # HW4(d)
    closing_lena = myMorphology.closing(binary_lena, kernel, centerKernel)
    closing_lena.save('closing_lena.bmp')

    # HW4(e)
    kernel_J = np.array([
        [1, 1],
        [0, 1]])
    centerKernel_J = (1, 0)
    kernel_K = np.array([
        [1, 1],
        [0, 1]])
    centerKernel_K = (0, 1)
    hitmiss_lena = myMorphology.hitmiss(binary_lena,
                                        kernel_J, centerKernel_J,
                                        kernel_K, centerKernel_K)
    # Save image fo file.
    hitmiss_lena.save('hitmiss_lena.bmp')
```

Import 需要用到的 library

宣告 kernel[35553]

定義 kernel 中心點

因為要 program on a binary image，故先將 lena 二值化

呼叫我的 myMorphology.py 副函式進行處理作業

宣告 hitmiss 要用到的 Array

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

```python
def dilation(originalImage, kernel, centerKernel):
    dilationImage = Image.new('1', originalImage.size)
    for r in range(originalImage.size[0]):
        for c in range(originalImage.size[1]):
            originalPixel = originalImage.getpixel((r, c))
            # If this pixel is 1 which is white
            if (originalPixel != 0):
                for ii in range(kernel.shape[0]):
                    for jj in range(kernel.shape[1]):
                        if (kernel[ii, jj]):
                            targetX = r + (ii - centerKernel[0])
                            targetY = c + (jj - centerKernel[1])
                            if ((0 <= targetX < originalImage.size[0])
                                    and (0 <= targetY < originalImage.size[1])):
                                dilationImage.putpixel((targetX, targetY), 1)
    return dilationImage
```

先產生一張跟 input 一樣大的 binary original Image

掃過每一個 pixel 並判斷是否為 1(白色格子)

如果為 1(白色格子)，並將 kernel 貼上去那個 pixel，並將 kernel 中黑點的地方都填上 1(條件為在 dilation Image)大小內

將處理好的影像回傳

```python
def erosion(originalImage, kernel, centerKernel):
    erosionImage = Image.new('1', originalImage.size)
    for r in range(originalImage.size[0]):
        for c in range(originalImage.size[1]):
            matchFlag = 1
            for ii in range(kernel.shape[0]):
                for jj in range(kernel.shape[1]):
                    if (kernel[ii, jj] and matchFlag):
                        targetX = r + (ii - centerKernel[0])
                        targetY = c + (jj - centerKernel[1])
                        if ((0 <= targetX < originalImage.size[0])
                                and (0 <= targetY < originalImage.size[1])):
                            if (originalImage.getpixel(
                                    (targetX, targetY)) == 0):
                                matchFlag = False
                                break
                        else:
                            matchFlag = False
                            break
            if (matchFlag):
                erosionImage.putpixel((r, c), 1)
    return erosionImage
```

先產生一張跟 input 一樣大的 binary original Image

matchFlag 來判斷 kenel 是否和該 pixel 的鄰近 pixel 都符合

只要掃到和 kernel 不一樣的內容就 break 掉，以節省運算時間

只要通過檢核的 pixel，一律填值 1 給他(白色)，最後就可以得到我的 erosion Image

```python
def opening(originalImage, kernel, centerKernel):
    return dilation(erosion(originalImage, kernel, centerKernel), kernel, centerKernel)


def closing(originalImage, kernel, centerKernel):
    return erosion(dilation(originalImage, kernel, centerKernel), kernel, centerKernel)
```

先 erosion 再 dilation

先 dilation 再 erosion

Homework 4　　R08921005 黃國郡

```python
def complement(originalImage):
    complementImage = Image.new('1', originalImage.size)
    for r in range(originalImage.size[0]):
        for c in range(originalImage.size[1]):
            if (originalImage.getpixel((r, c)) == 0):
                complementImage.putpixel((r, c), 1)
            else:
                complementImage.putpixel((r, c), 0)
    return complementImage


def intersection(image1, image2):
    intersectionImage = Image.new('1', image1.size)
    for r in range(image1.size[0]):
        for c in range(image1.size[1]):
            image1Pixel = image1.getpixel((r, c))
            image2Pixel = image2.getpixel((r, c))
            if (image1Pixel and image2Pixel):
                intersectionImage.putpixel((r, c), 1)
            else:
                intersectionImage.putpixel((r, c), 0)
    return intersectionImage

def hitmiss(originalImage, kernel_J, centerKernel_J, kernel_K, centerKernel_K):
    return intersection(erosion(originalImage, kernel_J, centerKernel_J),
                        erosion(complement(originalImage), kernel_K, centerKernel_K))
```

先產生一張跟 input 一樣大的 binary original Image

一個一個 pixel 往下掃往右掃，如果是 1 就填 0，如果是 0 就填 1，做 complement

先產生一張跟 input 一樣大的 binary original Image

一個一個 pixel 往下掃往右掃，並同時比較兩張童位置的 pixel 做交集，將結果存入 intersectionImage

最後進行 hit and miss，交集(被 erosion 過的 oirginal Image 和被 erosion 過的 oirginal complement Image)