Computer Vision I _2018

Homework assignment #4

R07522717 機械所製造組碩一 林温雅

```
#使用 python
#import 套件
import cv2
import numpy as np

#dilation function
def dilation(img, ker):
    #獲得輸入圖檔之行列數
    img_rows, img_columns = img.shape
    #獲得 kernel 之行列數
    ker_rows, ker_columns = ker.shape
    #計算 kernel 中心距離邊界有多遠，為的是擴大原始圖檔，方便後續迴圈處理
    row_dist, column_dist = int((ker_rows-1)/2), int((ker_columns-1)/2)
    #根據上述計算，製作一個比原始圖檔大的暫存圖檔，以 img 為 512*512, kernel 為 5*5 來說，暫存圖檔為 516*516，暫存圖檔為往上、往下、往左、往右分別外擴兩列/行，外擴新增的 pixel 值為 0，中間則就是原本輸入圖檔的值
    temp_img = np.zeros((img_rows+2*row_dist, img_columns+2*column_dist), np.int)
    temp_img[row_dist:img_rows+row_dist, column_dist:img_columns+column_dist] = img
    #製作一個新圖檔準備接受 dilation 後的圖
    new_img = np.zeros((img_rows, img_columns), np.int)
    #進行 dilation 邏輯計算
    for i in range(row_dist, img_rows+row_dist):
        for j in range(column_dist, img_columns+column_dist):
            if np.any(np.logical_and(ker, temp_img[i-row_dist: i+row_dist+1, j-column_dist: j+column_dist+1])):
                new_img[i-row_dist, j-column_dist] = 255
    return new_img
```

```
def erosion(img, ker):
    #獲得輸入圖檔之行列數
    img_rows, img_columns = img.shape
    #獲得 kernel 之行列數
    ker_rows, ker_columns = ker.shape
    #計算 kernel 中心距離邊界有多遠，為的是擴大原始圖檔，方便後續迴圈處理
    row_dist, column_dist = int((ker_rows-1)/2), int((ker_columns-1)/2)
    #根據上述計算，製作一個比原始圖檔大的暫存圖檔，以 img 為 512*512, kernel 為 5*5 來說，暫存圖檔為 516*516，暫存圖檔為往上、往下、往左、往右分別外擴兩列/行，外擴新增的 pixel 值為 1，中間則就是原本輸入圖檔的值
    ###特別注意這邊外擴 pixel 的值為 1，而在 dilation function 內是 0###
    temp_img = np.ones((img_rows+2*row_dist, img_columns+2*column_dist), np.int)
    temp_img[row_dist:img_rows+row_dist, column_dist:img_columns+column_dist] = img
    #製作一個新圖檔準備接受 dilation 後的圖
    new_img = np.zeros((img_rows, img_columns), np.int)
    #進行 erosion 邏輯計算
    for i in range(row_dist, img_rows+row_dist):
        for j in range(column_dist, img_columns+column_dist):
            if not np.any(ker - np.logical_and(ker, temp_img[i-row_dist: i+row_dist+1, j-column_dist: j+column_dist+1])):
                new_img[i-row_dist, j-column_dist] = 255
    return new_img

#Opening，對影像進行 Erosion 後再 Dilation
def opening(img, ker):
    return dilation(erosion(img, ker), ker)

#Closing，對影像進行 Dilation 後再 Erosion
def closing(img, ker):
    return erosion(dilation(img, ker), ker)
```

```
#對 binary image 製作黑白互補圖用的 function
def binary_image_complement(img):
    img_rows, img_columns = img.shape
    new_img = np.zeros((img_rows, img_columns), np.int)

    for i in range(img_rows):
        for j in range(img_columns):
            new_img[i,j] = 255-img[i,j]
    return new_img

#尋找 upper-right-corner 用的 hit_and_miss function
def hit_and_miss_ur_corner(img, ker_j, ker_k):
    img_rows, img_columns = img.shape
    new_img = np.zeros((img_rows, img_columns), np.int)
    temp_img1 = erosion(img, ker_j)
    temp_img2 = erosion(binary_image_complement(img), ker_k)
    for i in range(img_rows):
        for j in range(img_columns):
            if temp_img1[i,j]==255 and temp_img2[i,j]==255:
                new_img[i,j] = 255
    return new_img

###開始處理###

#讀取 hw2 實作之 binary 圖檔
original_img = cv2.imread('128binary_lena.bmp', 0)

###製作 kernel###
#dilation, erosion, opening, closing 要用的 kernel
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]])
#hit_and_miss 要用的 j 和 k kernel
kernel_j = np.array([[0,0,0], [1,1,0], [0,1,0]])
kernel_k = np.array([[0,1,1], [0,0,1], [0,0,0]])
```

```
###輸出圖片###

#輸出 dilation 圖片
cv2.imwrite('dilation_lena.bmp', dilation(original_img, kernel))

#輸出 erosion 圖片
cv2.imwrite('erosion_lena.bmp', erosion(original_img, kernel))

#輸出 opening 圖片
cv2.imwrite('opening_lena.bmp', opening(original_img, kernel))

#輸出 closing 圖片
cv2.imwrite('closing_lena.bmp', closing(original_img, kernel))

#輸出 hit_and_miss 右上 corner 圖片
cv2.imwrite('hit_and_miss_ur_corner.bmp',
hit_and_miss_ur_corner(original_img, kernel_j, kernel_k))
```