Computer Vision I _2018

Homework assignment #5

R07522717 機械所製造組碩一 林温雅

Part1（此次作業僅 one part）

Description:

Write programs which do binary morphological dilation, erosion, opening, closing, and hit-and-miss transform on a binary image.

Algorithm:

1. Dilation:

獲得 kernel 與輸入 image 的行列數以後，製作一個『外擴』圖檔，為的是處理 kernel 在尋訪時超出邊界的情況（以 image 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，才不會影響後續取 max 的計算，中間則就是原本輸入圖檔的值）。

接著再進行計算，kernel 設計為[[np.nan,0,0,0,np.nan], [0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0], [np.nan,0,0,0,np.nan]]。在計算時會忽略遇到 nan 的運算，因此計算結果即為 3553 形狀 kernel 中的最大值，將最大值填入 kernel 中心覆蓋的某 p 點。

特別的是，在 dilation 進行 convolution，因此 kernel 須先 flip 180 度，但作業指定的是點對稱 kernel，有沒有 flip 不對結果造成影響。

2. Erosion:

獲得 kernel 與輸入 image 的行列數以後，製作一個『外擴』圖檔，為的是處理 kernel 在尋訪時超出邊界的情況（以 image 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 255，才不會影響後續取 min 的計算，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 255，而在 dilation function 內是 0）。

接著再進行計算，kernel 設計為[[np.nan,0,0,0,np.nan], [0,0,0,0,0],

[0,0,0,0,0], [0,0,0,0,0], [np.nan,0,0,0,np.nan]]。在計算時會忽略遇到 nan 的運算，因此計算結果即為 3553 形狀 kernel 中的最小值，將最小值填入 kernel 中心覆蓋的某 p 點。

3. Opening:
   對影像進行 GrayScale_Erosion 後再進行 GrayScale_Dilation

4. Closing:
   對影像進行 GrayScale_Dilation 後再 GrayScale_Erosion

Parameters:

1. In function “dilation”:

img_rows, img_columns	#輸入圖檔的行列數
ker_rows, ker_columns	#kernel 的行列數
row_dist, column_dist	#計算 kernel 中心距離邊界有多遠，主要目的是看原始圖檔要擴大多少
temp_img	#原始圖檔擴大後的暫存圖檔
kernel_flip	#flip 180 度後的 kernel
new_img	#新圖檔準備接受 dilation 後的圖
i,j	#迴圈計數用參數

2. In function “erosion”:

img_rows, img_columns	#輸入圖檔的行列數
ker_rows, ker_columns	#kernel 的行列數
row_dist, column_dist	#計算 kernel 中心距離邊界有多遠，主要目的是看原始圖檔要擴大多少
temp_img	#原始圖檔擴大後的暫存圖檔
new_img	#新圖檔準備接受 Erosion 後的圖
i,j	#迴圈計數用參數

3. Outside of function

original_img　　#讀取原始圖檔
kernel　　#作業指定會用到的 kernel 矩陣

Principal code fragment:

```
def GrayScale_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 值另為 0，中間則就是原本輸入圖檔的值

    #dilation 要找最大的，所以外擴的填 0
    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
    #為了 for 迴圈裡面 index 好寫 ,這邊一樣把 new_img 改成擴大後的 ,之後再來裁 ,和 hw4 做法有一點點不一樣
    new_img = np.zeros((img_rows+2*row_dist, img_columns+2*column_dist), np.int)

    #為了矩陣相乘，先 flip kernel，erosion 不用這樣
    kernel_flip = np.flip(ker)

    #進行 dilation 計算
    for i in range(row_dist, img_rows+row_dist):
        for j in range(column_dist, img_columns+column_dist):
            new_img[i, j] = np.nanmax(temp_img[i-row_dist: i+row_dist+1, j-column_dist: j+column_dist+1]+kernel_flip)
```

```
    new_img = new_img[row_dist:img_rows+row_dist,
column_dist:img_columns+column_dist]
    return new_img

def GrayScale_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 值另為 0，中間則就是原本輸入圖檔的值

    #erosion 要找最小的，所以外擴的填 255
    temp_img = 255 * 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
    #為了 for 迴圈裡面 index 好寫，這邊一樣把 new_img 改成擴大後的，之後再來裁，和 hw4 做法有一點點不一樣
    new_img = 255*np.ones((img_rows+2*row_dist, img_columns+2*column_dist),
np.int)

    #進行 erosion 計算
    for i in range(row_dist, img_rows+row_dist):
        for j in range(column_dist, img_columns+column_dist):
            new_img[i, j] = np.nanmin(temp_img[i-row_dist: i+row_dist+1,
j-column_dist: j+column_dist+1]-ker)
    new_img = new_img[row_dist:img_rows+row_dist,
column_dist:img_columns+column_dist]
```

Resulting images

gray_scale_dilation

gray_scale_erosion

gray_scale_opening

gray_scale_closing