Computer Vision I _2018

Homework assignment #4

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，中間則就是原本輸入圖檔的值）。

接著再進行邏輯運算，若 kernel 中心覆蓋某 p 點時，kernel 與 image 有 and 計算為 true 的情況，則將 p 點設為白色。

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 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。ppt 內之 erosion 照片應是忽略此點，才造成邊界都是一圈黑色）。

接著再進行邏輯運算，若 kernel 中心覆蓋某 p 點時，kernel 與 image 之 and 計算結果和 kernel 本身相等（及整個 kernel 的 1 值都在 image 為白色處），則將 p 點設為白色。

3. Opening:

對影像進行 Erosion 後再 Dilation

4. Closing:
   對影像進行 Dilation 後再 Erosion

5. Hit-and-Miss:
   先對 binary image 製作黑白互補圖的互補圖，接著製作此原 image 與 kernel_j 的 erosion 圖&互補圖與 kernel_k 的 erosion 圖。最後輸出此兩張 erosion 結果圖重疊的部分。

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
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. In function “binary_image_complement”:

img_rows, img_columns	#輸入圖檔的行列數
new_img	#新圖檔準備接受 Erosion 後的圖
i,j	#迴圈計數用參數

4. In function “hit_and_miss_ur_corner”:

img_rows, img_columns	#輸入圖檔的行列數
new_img	#新圖檔準備接受 Erosion 後的圖
temp_img1	#原圖檔與 kernel_j 進行 erosion 後的圖
temp_img2	#原圖的互補圖與kernel_k進行erosion後的圖
i,j	#迴圈計數用參數

5. Outside of function

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

Principal code fragment:

```
#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)
    ###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

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
```

Resulting images

Dilation

Erosion

Opening

Closing

Hit-and-Miss