

## 電腦視覺 (一) Homework4

R11521701 程懷恩

### 1. Dilation



Code fragment:

```
def dilation(img,kernel):  
    arr = np.zeros(img.shape, np.uint8)  
    for i in range(0, img.shape[0]):  
        for j in range(0, img.shape[1]):  
            if img[i][j]==255:  
                for ite in kernel:  
                    x, y = ite  
                    if x+i>=0 and x+i<img.shape[0] and y+j>=0 and y+j<img.shape[1]:  
                        arr[x+i][y+j]=255  
    return arr
```

根據課本的 dilation 方法，在原陣列圖形中存在實物之點(RGB=255)疊合 structuring element 陣列後所產生之新點，加入擴張後的圖形中。(註：設定邊界範圍避免錯誤)

### 2. Erosion



Code fragment:

```
def erosion(img, kernel):
    arr = np.zeros(img.shape, np.uint8)
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            count = 0
            for ite in kernel:
                x, y = ite
                if x+i < 0 or x+i>=img.shape[0] or y+j<0 or y+j>=img.shape[1]:
                    break
                elif img[i+x][j+y]==255:
                    count+=1
            if count == len(kernel): arr[i][j]=255
    return arr
```

以 structing element 之中心為基準，尋找原始圖片陣列中與 structing element 疊合後完全相同之區域，並將其中心設為 255。程式的部分設置一個計數器 count 初始化為 0，以原陣列的點逐個為中心搜尋與 structing element 相同的元素數量，若是完全相符 (count = len(kernel)) 即給予 255，反之為 0。

### 3. Opening



Code fragment:

```
def opening(img, kernel):
    arr = np.zeros(img.shape, np.uint8)
    arr = erosion(img, kernel)
    return dilation(arr, kernel)
```

根據課本定義： $B \circ K = (B \ominus K) \oplus K$

先執行 erosion 後執行 dilation 即為 openings

#### 4. Closing



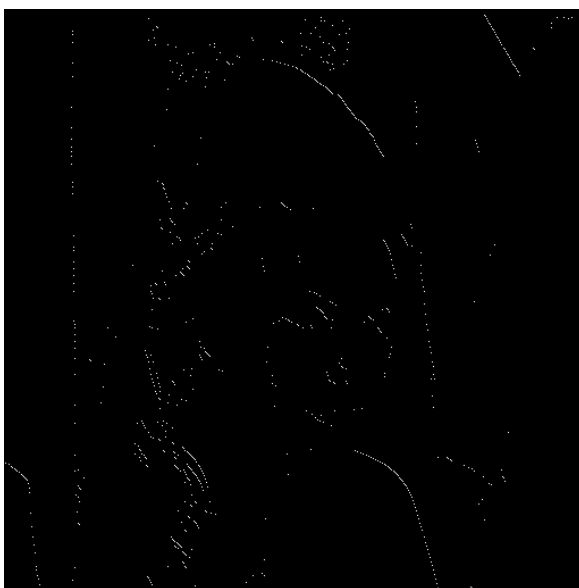
Code fragment:

```
def closing(img, kernel):  
    arr = np.zeros(img.shape, np.uint8)  
    arr = dilation(img, kernel)  
    return erosion(arr, kernel)
```

根據課本定義： $B \bullet K = (B \oplus K) \ominus K$

先執行 dilation 再執行 erosion 即為 closing

#### 5. Hit-and-Miss transform



Code fragment:

```
def contrary(img):
    arr = np.zeros(img.shape, np.uint8)
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            if img[i][j]==0:arr[i][j]=255
    return arr
```

首先，找出原陣列之補集。

```
def HNM(img,J,K):
    img_c = contrary(img)
    arr1 = erosion(img,J)
    arr2 = erosion(img_c,K)
    arr = np.zeros(img.shape, np.uint8)
    for i in range (0,img.shape[0]):
        for j in range(0, img.shape[1]):
            if arr1[i][j]==255 and arr2[i][j]==255:arr[i][j]=255
    return arr
```

將原陣列與 J 做 erosion；原陣列補集與 K 做 erosion 後，取兩個結果同時

為 255 之交集。