B06902042 劉愷爲

Computer Vision HW4

Part1 Dilation

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

基本上是利用上面的公式實作，A即為經過threshold後的圖，B則為kernel也就是35553的八邊形，利用迴圈實作，每遇到灰階值為255的點就加上kernel中的座標，並改變灰階值為255。ind: kernel中的座標。kernel : 35553的八邊形

```
def dilation(img):
    ans = np.zeros((img.shape), np.int)
    for x in range(img.shape[0]):
        for y in range(img.shape[1]):
            if img[x, y] == 255:
                for idx in kernel:
                    if img.shape[0] > x + idx[0] >= 0 and img.shape[1] > y + idx[1] >= 0:
                        ans[x + idx[0], y + idx[1]] = 255
    return ans
```

Part2 Erosion

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

參考上面的式子，利用迴圈和假設實作，掃描所有的點。如果加上kernel的座標之後會超出邊界，則點的灰階值改為0，或是加上座標之後有任一點的灰階值不為255也將點原本的點的灰階值改為0。 flag: 用來記錄是否有點加上kernel座標後，任一點的灰階值不為255，如果沒有flag = 1，反之 flag = 0。

```
def erosion(img, k):
    ans = np.zeros((img.shape), np.int)
    for x in range(img.shape[0]):
        for y in range(img.shape[1]):
            flag = 1
            for idx in k:
                if not (img.shape[0] > x + idx[0] >= 0 and img.shape[1] > y + idx[1] >= 0):
                    flag = 0
                    break
                elif img[x + idx[0], y + idx[1]] != 255 :
                    flag = 0
                    break
            if flag:
                ans[x, y] = 255
    return ans
```

Part3 Opening

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

先做erosion, 再做dilation

```
def opening(img):
    ans = erosion(img, kernel)
    ans = dilation(ans)
    return ans
```

Part4 Closing

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

先做dilation, 再做erosion

Part5 Hit and Miss

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

Ac的圖代表將A翻轉，j和k則分別是L型的座標，如下圖j為1的部分，k為0的部分。

	0	0
1	1	0
	1	

實作上，reverse就是利用迴圈將0改為255，255改為0。
Intersection也是利用迴圈看兩張圖交 集的部分。A和j，Ac和k，分別erosion，再intersection，即完成hit and mess的圖像。

```
def reverse(img):
    ans = np.zeros((img.shape), np.int)
    for x in range(img.shape[0]):
        for y in range(img.shape[1]):
            if img[x, y] == 255:
                ans[x, y] = 0
            else:
                ans[x, y] = 255
    return ans

def inter(i1, i2):
    ans = np.zeros((i1.shape), np.int)
    for x in range(i1.shape[0]):
        for y in range(i1.shape[1]):
            if i1[x, y] == 255 and i2[x, y] == 255:
                ans[x, y] = 255
    return ans

def hit_and_miss(img):
    j = [ [0, 0], [0, -1], [1, 0] ]
    k = [ [0, 1], [-1, 0], [-1, 1] ]
    ans = np.zeros((img.shape), np.int)
    img_c = reverse(img)
    img1 = erosion(img, j)
    img2 = erosion(img_c, k)
    ans = inter(img1, img2)
    return ans
```

Threshold

Dilation

Erosion

Opening

Closing

Hit and Miss