Computer Vision HW4　　B07902127 資工三 羅義鈞

(a) Dilation

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

A: threshold 過後的圖, B: 3-5-5-5-3 八邊形

遇到 A 中 255 的點，就把 return 圖片的對應座標加上 kernel 中的對應座標後，如果在邊界內該座標設為 255。

```
def dilation(img, ker):
    ret = np.zeros((img.shape))
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i, j] == 255:
                for idx in ker:
                    if 0 <= i + idx[0] < img.shape[0] and 0 <= j + idx[1] < img.shape[1]:
                        ret[i + idx[0], j + idx[1]] = 255
    return ret
```

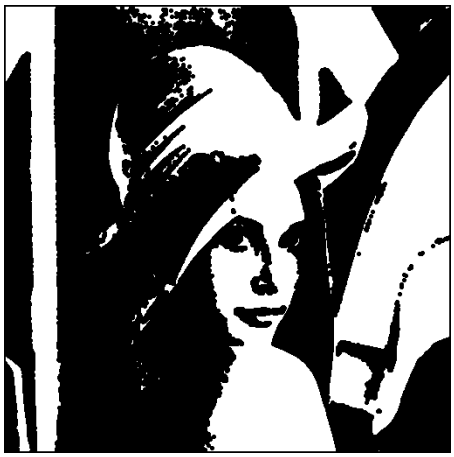
dilation.bmp

(b) Erosion

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

如果加上了 kernel 的座標會超出邊界或是加上之後的該點座標灰階值不為 255，則將灰階值設為 0。

```
def erosion(img, ker):
    ret = np.zeros((img.shape))
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            flag = 1
            for idx in ker:
                if not (0 <= i + idx[0] < img.shape[0] and 0 <= j + idx[1] < img.shape[1]) \
                    or img[i + idx[0], j + idx[1]] != 255:
                    flag = 0
                    break
            if flag:
                ret[i, j] = 255
    return ret
```


erosion.bmp

(c) Opening

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

先 erosion 再 dilation

```
def opening(img, ker):
    ret = erosion(img, ker)
    ret = dilation(img, ker)
    return ret
```


opening.bmp

(d) Closing

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

先 dilation 再 erosion

```
def closing(img, ker):
    ret = dilation(img, ker)
    ret = erosion(img, ker)
    return ret
```

closing.bmp

(e) Hit-and-miss transform

$$A \otimes (L_1, L_2) = (A \ominus L_1) \cap (A^c \ominus L_2)$$

A^c 表示將 A 翻轉，L_1 和 L_2 為 L 型的 kernel 座標。首先先 reverse，將 0 改成 255 以及 255 改成 0。再來用 intersec 找交集部分，分別做完 erosion 後進行 intersection 即為 hit-and-miss 的 image。

```python
def reverse(img):
    ret = np.zeros((img.shape), np.int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i, j] == 0:
                ret[i, j] = 255
            else:
                ret[i, j] = 0
    return ret

def intersec(img1, img2):
    ret = np.zeros((img1.shape), np.int)
    for i in range(img1.shape[0]):
        for j in range(img1.shape[1]):
            if img1[i, j] == 255 and img2[i, j] == 255:
                ret[i, j] = 255
    return ret

def hit_and_miss(img):
    # L shaped kernel
    L1 = [[0, 0], [0, -1], [1, 0]]
    L2 = [[0, 1], [-1, 0], [-1, 1]]

    ret = np.zeros((img.shape), np.int)
    i1 = reverse(img)
    i2 = erosion(img, L1)
    i3 = erosion(i1, L2)
    ret = intersec(i2, i3)
    return ret
```

hit_and_miss.bmp