Computer Vision HW5　　B07902127 資工三 羅義鈞

(a) Dilation

$$f : F \to E \text{ and } k : K \to E, \text{ then } f \oplus k : F \oplus K \to E$$

$$(f \oplus k)(x) = \max\{f(x - z) + k(z) | z \in K, x - z \in F\}$$

f: 灰階圖，k: kernel (35553 的八邊形，value = 0)

每個點以 kernel 為範圍找出最大值後修改為最大值並且存為新的圖片。

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

dilation.bmp

(b) Erosion

$$f : F \to E \text{ and } k : K \to E, \text{ then } f \ominus k : F \ominus K \to E$$

$$(f \ominus k)(x) = \min_{z \in K}\{f(x + z) - k(z)\}$$

跟 dilation 很類似，只是換成最小值，其餘一樣。

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

erosion.bmp

(c) Opening

$$f \circ k = (f \ominus k) \oplus k$$

先 erosion 再 dilation

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

opening.bmp

(d) Closing

$$f \cdot k = (f \oplus k) \ominus k$$

先 dilation 再 erosion

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

closing.bmp