# HW 4

> *b05902019 資工三 蔡青邑*

`Using "lena.bmp" as input image.`

## Python Packages I used

- `skimage.io`: for basic image i/o.
- `numpy`: for convience of array manipulation.
- `tqdm`: for showing the progress of the executing of the code.

## Some Other Functions I Build

- **blank_image(height, width)**: returning an all-black image of the given input height and width.
- **binarize(img, lower_expand, upper_expand, threshold)**: binarize the image(img) according to the threshold and return it.

# Dilation, Erosion, Hit-and-miss, Opening, and Closing

> *I wrote a function for each of them, below is how I implement and the image it create.*

## Dilation

- my function: `dilation(img, kernel)`
- $A \oplus B = \{c \in E^N | c = a + b \text{ for some} a \in A and b \in B\}$

It create a blank image first. After that, this function traverse every pixel of the image(`img`); if the pixel's color is white, it then change the color of that blank image into white according to the the shape of the kernel with that pixel's position being the origin. At the end, return the image we create.

## Erosion

- my function: `erosion(img, kernel)`
- $A \ominus B = \{x \in E^N | x + b \in A \text{ for every } b \in B\}$

It create a blank image first, too. Then it travers every pixel of the image(`img`), applying the kernel on each of them. If the kernel's position applied to a given pixel are all white, then we change the position of that pixel on the blank image into white, by declaring a flag in a loop. At the end, return the image we create.

### Hit_and_miss

- my function: `hit_and_miss(img, j, k)`
- $A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$

This part is easier. Simply run `erosion(imj, j)` and `erosion(255 - img, k)`, and then find their intercept white pixels.

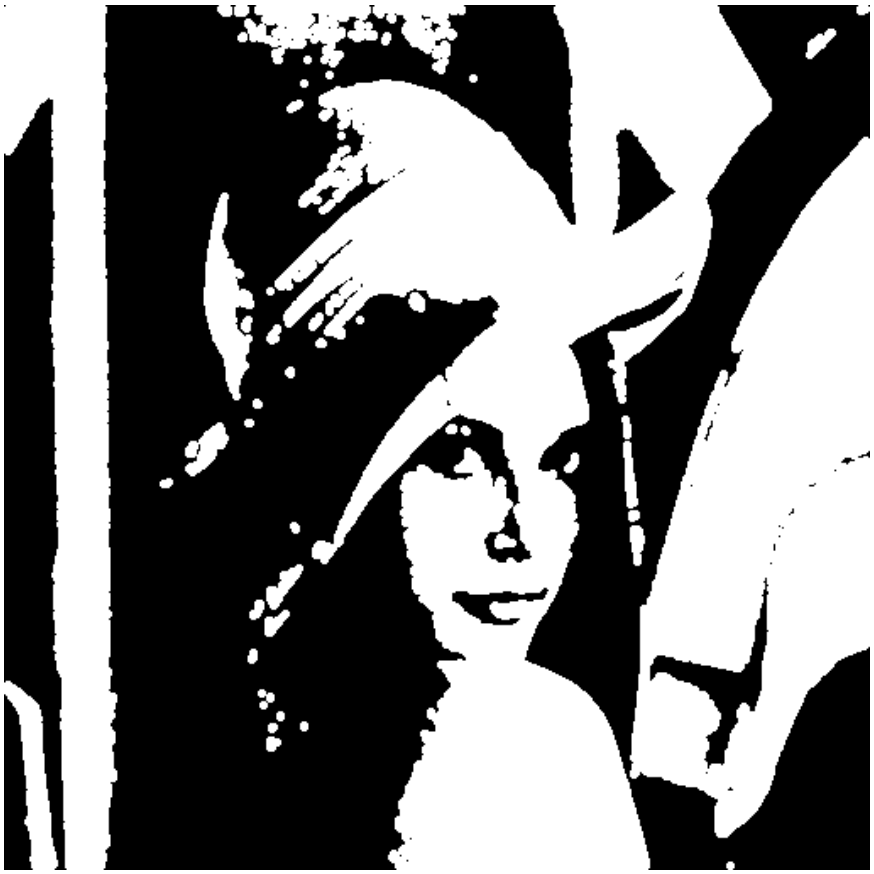
```
def hit_and_miss(img, j, k):
    img_erosed_by_j = erosion(img, j)
    img_dilated_by_k = erosion(255 - img, k)
    return_img = blank_image(height, width)
    print("---hit and miss intersecttion start---")
    for i in tqdm(range(height)):
        for j in range(width):
            if(img_erosed_by_j[i, j] == img_dilated_by_k[i, j] and
img_erosed_by_j[i, j] == 255):
                return_img[i, j] = 255
    print("---hit and miss intersecttion end---")
```

## Opening

- my function: `opening(img, kernel)`
- $B \circ K = (B \ominus K) \oplus K$

Simply apply the formula:

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



## Closing

- my function: `closing(img, kernel)`
- $B \bullet K = (B \oplus K) \ominus K$

Simply apply the formula:

```
def closing(img, kernel):
    temp = dilation(img, kernel)
    temp = erosion(temp, kernel)
    return temp
```