

# Computer Vision\_HW4

---

R09922093 楊子萱

In this homework, I use python3 and import cv2 to read and write images and use cv2 function to translate image into the binary image. To run my code, use command line and enter python3 hw4.py\_(<http://hw4.py>). .

( a ) Dilation

Search for the whole image, if pixel value == 255, then do the "dilation\_check" for doing the dilation process.

In the dilation\_check function, I use the algorithm mentioned in the lecture pdf.

- dilation of  $A$  by  $B$ :  $A \oplus B$

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

where  $A$  is image,  $B$  is kernel.

```
def dilation_check(image, kernel, x, y):
    p_x = x - (kernel.shape[0]//2)
    p_y = y - (kernel.shape[1]//2)
    for i in range(kernel.shape[0]):
        for j in range(kernel.shape[1]):
            if kernel[i][j] == 1:
                if p_x + i >= 0 and p_x + i < image.shape[0]
                   and p_y + j >= 0 and p_y + j < image.shape[1]:
                    image[p_x+i][p_y+j] = 255

def dilation(image, kernel):
    result = np.full((image.shape[0], image.shape[1]), 0)
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            if image[i][j] == 255:
                dilation_check(result, kernel, i, j)
    return result
```



( b ) Erosion

Search for the whole image, then do the "erosion\_check" for doing the erosion process. In the erosion\_check function, I use the algorithm mentioned in the lecture pdf.



erosion of  $A$  by  $B$ : set of all  $x$  s.t.  $x + b \in A$  for every  $b \in B$

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

where  $A$  is image and  $B$  is kernel.

```
def erosion_check(image, kernel, x, y):
    p_x = x - (kernel.shape[0]//2)
    p_y = y - (kernel.shape[1]//2)
    for i in range(kernel.shape[0]):
        for j in range(kernel.shape[1]):
            if kernel[i][j] == 1:
                if p_x + i < 0 or p_x + i >= image.shape[0]
                   or p_y + j < 0 or p_y + j >= image.shape[1] :
                    return 0
                if image[p_x+i][p_y+j] != 255:
                    return 0
    return 255
def erosion(image, kernel):
    result = np.full((image.shape[0],image.shape[1]),0)
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            result[i][j] = erosion_check(image, kernel, i, j)
    return result
```



( c ) Opening	( d ) Closing
erosion then dilation.	dilation then erosion.
	

( c ) Opening  
erosion then dilation.

( e ) Hit-and-miss transform  
Use the algorithm mentioned in the lecture pdf.

hit-and-miss of set  $A$  by  $(J,K)$

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

$A$  is image, and  $J$  and  $K$  are kernels.

so, do erosion twice and find the intersection of two images after erosion.

```

def hit_and_miss(image, image1, J, K):
    result = np.full((image.shape[0],image.shape[1]),0)
    erosion_1 = erosion(image,J)
    erosion_2 = erosion(image1,K)
    cv2.imwrite("erosion_1.bmp",erosion_1)
    cv2.imwrite("erosion_2.bmp",erosion_2)
    for i in range(erosion_1.shape[0]):
        for j in range(erosion_1.shape[1]):
            if(erosion_1[i][j] == 255 and erosion_2[i][j] == 255):
                result[i][j] = 255
    return result

```

