國立臺灣科技大學
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

實習三
去雜訊干擾

課程大綱

- 實習00: Colab 環境
- 實習03: 去雜訊干擾

實習 00
Colab 環境

Colab Env.

Colab Env.

Before we start...

```
#mount drive
from google.colab import drive
drive.mount('/content/drive')
```

```
# import libraries
import sys
import os
import cv2
import numpy as np
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow
```

實習 03
去雜訊干擾

實驗題目說明

對影像做去雜訊干擾(Denoise)，使用:

- **均值濾波(Average filter)**
- **中值濾波(Medium filter)**
- **高斯濾波(Gaussian filter)**
- **形態學(Morphology)**

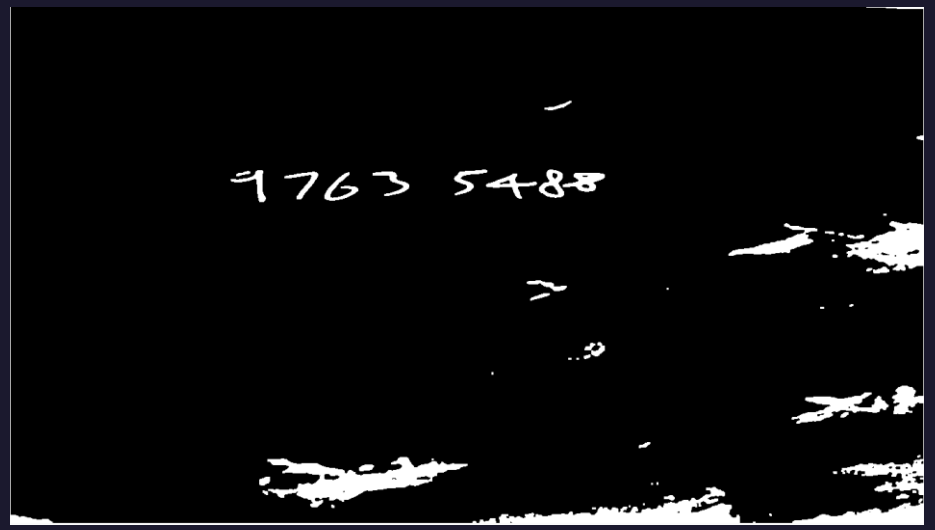
補充資料:

- **https://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm**
- **(developed by the Department of Artificial Intelligence in the University of Edinburgh)**

Lena_noise.bmp

https://drive.google.com/file/d/1a0VbzH-NcbfBnQat0wGflNO1vnJcEpuZ/view?usp=sharing

https://reurl.cc/m3e8zj



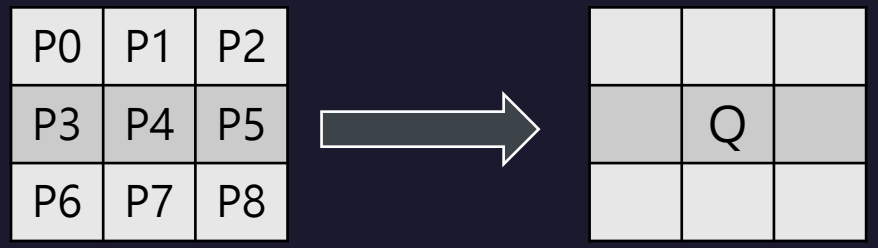
Number_noise.bmp

https://drive.google.com/file/d/1MYGgj5DBc7wOIM7aG0g_TapFNrbvB-YG/view?usp=sharing

https://reurl.cc/YXZgmL

TASK: Averaging Blurring (均值濾波/平滑)

- 包括移動平均法、算數平均、幾何平均、調和平均。
- 移動平均法是最簡單的雜訊去除法。
- 缺點: 無法分辨雜訊、邊緣，會使得影像模糊不清。



Q = (P0+P1+P2+P3+P4+P5+P6+P7+P8) / 9

移動平均法(算數平均)

Arithmetic mean filter 算數平均

$$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}} g(s,t)$$

Geometric mean filter 幾何平均

$$\hat{f}(x,y)=\left[\prod_{(s,t)\in S_{xy}} g(s,t)\right]^{1/mn}$$

Harmonic mean filter 調和平均

$$\hat{f}(x,y)=\frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

反向調和平均

Contraharmonic mean filter

$$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

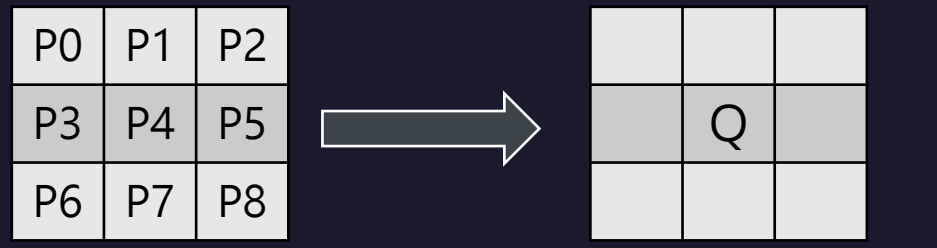
TASK: Averaging Blurring (均值濾波/平滑)

```
cv.blur( src, ksize[, dst[, anchor[, borderType]]] ) -> dst
```

- Parameters:
  - `src` input image; it can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
  - `dst` output image of the same size and type as src.
  - `ksize` blurring kernel size.
  - `anchor` anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.
  - `borderType` border mode used to extrapolate pixels outside of the image, see BorderTypes. BORDER_WRAP is not supported.

```python
# read an image
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'lena_saltpepper.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```python
def Average_Blurring(img):
  ksize=3
  img_blur = cv2.blur(img,(ksize,ksize))
  res = np.hstack([img,img_blur])
  cv2_imshow(res)
Average_Blurring(img_gray)
```



$$Q = (P0+P1+P2+P3+P4+P5+P6+P7+P8) / 9$$

移動平均法

TASK: Median Blurring (中值平滑/濾波)

- 由於移動平均法無法分辨雜訊及邊緣，結果會造成影像模糊不清。所以，盡可能地使目標影像的邊緣不被淡化，只除去雜訊。而此種最有名的解決方法即為中值濾波(Median Filter)。
- 中值濾波(Median Filter): 在某個像素周圍的區域內，尋求像素灰階濃度的中央值，並將其作為目標像素灰階濃度值的處理過程。

TASK: Median Blurring (中值平滑/濾波)

`cv.medianBlur( src, ksize[, dst] ) -> dst`

- Parameters:
  - `src` input 1-, 3-, or 4-channel image; when ksize is 3 or 5, the image depth should be CV_8U, CV_16U, or CV_32F, for larger aperture sizes, it can only be CV_8U.
  - `dst` destination array of the same size and type as src.
  - `ksize` aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ...

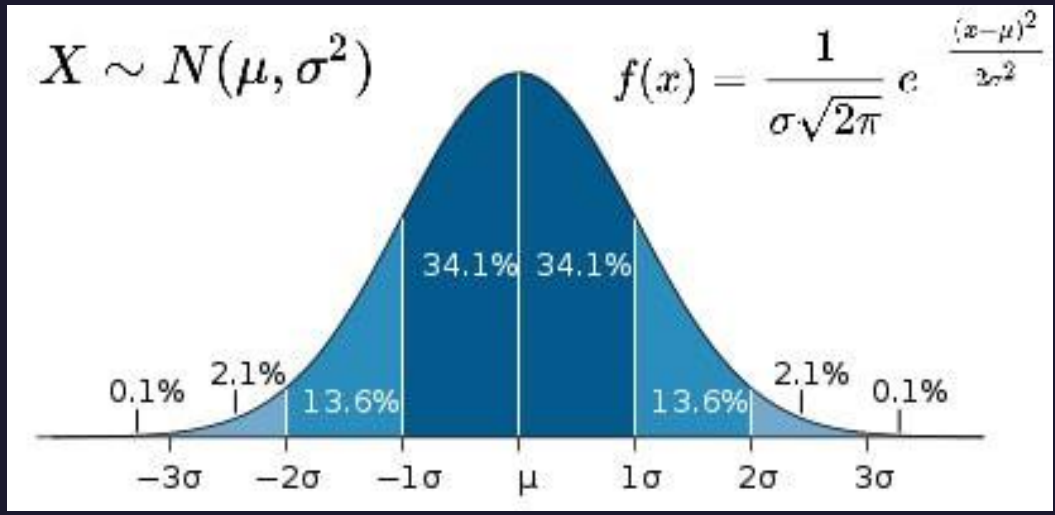

```python
# read an image
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'lena_saltpepper.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```python
def Medium_Blurring(img):
  ksize=3
  img_median = cv2.medianBlur(img,ksize)
  res = np.hstack([img,img_median])
  cv2_imshow(res)
Medium_Blurring(img_gray)
```

TASK: Gaussian Blurring (高斯平滑/濾波)

```
cv.GaussianBlur( src, ksize, sigmaX[, dst[, sigmaY[, borderType]]] ) -> dst
```

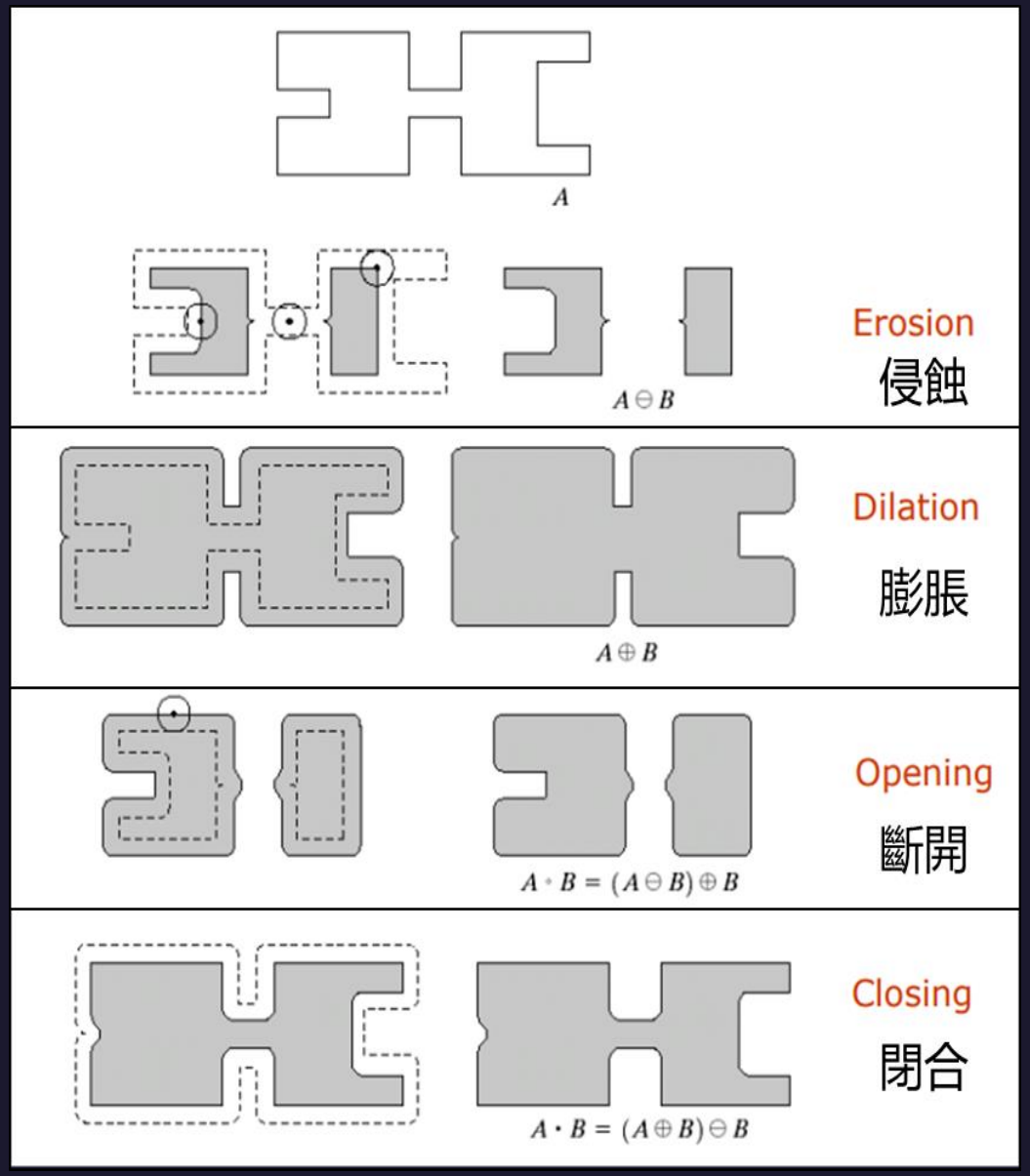
- Parameters:
  - `src` input image; the image can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
  - `dst` output image of the same size and type as src.
  - `ksize` Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma.
  - `sigmaX` Gaussian kernel standard deviation in X direction.
  - `sigmaY` Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively (see getGaussianKernel for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY.
  - `borderType` pixel extrapolation method, see BorderTypes. BORDER_WRAP is not supported.

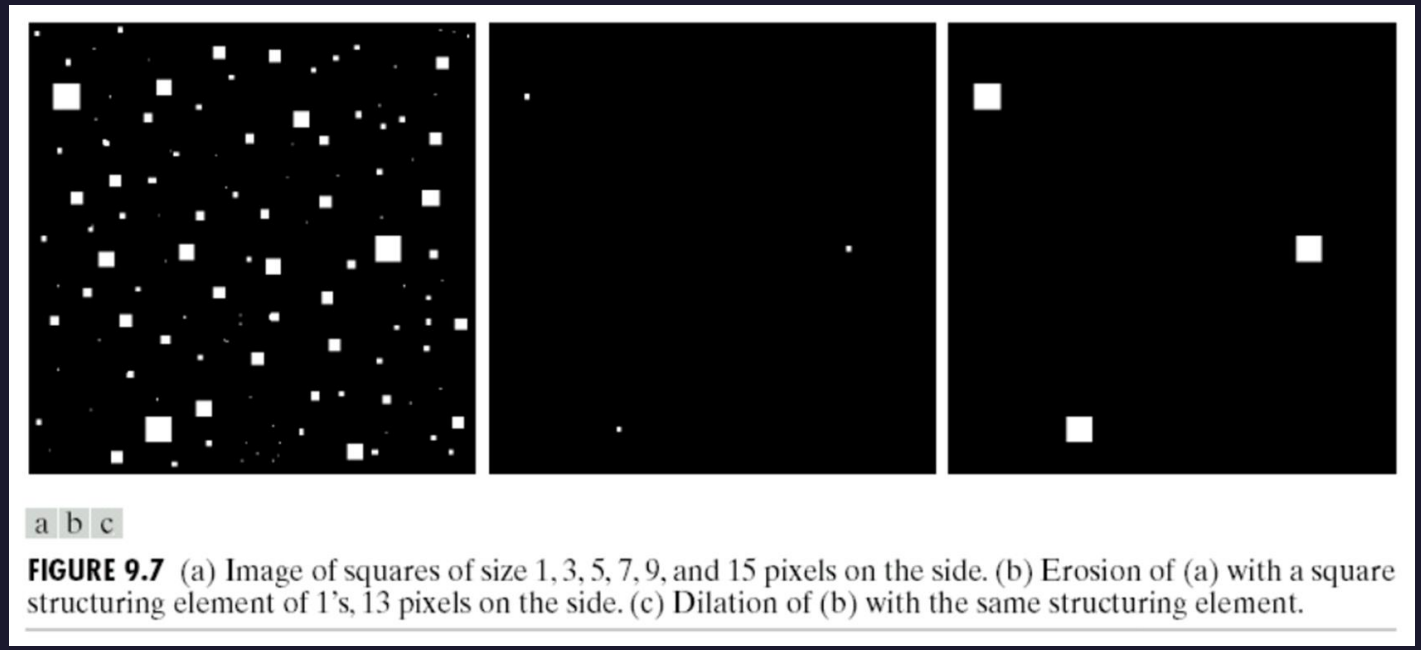



```python
# read an image
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'lena_saltpepper.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```python
def Gaussian_Blurring(img):
  ksize=3
  img_gaussianBlur = cv2.GaussianBlur(img,(ksize,ksize),0)
  res = np.hstack([img, img_gaussianBlur])
  cv2_imshow(res)
Gaussian_Blurring(img_gray)
```

TASK: Morphological Transformations (形態學變化)

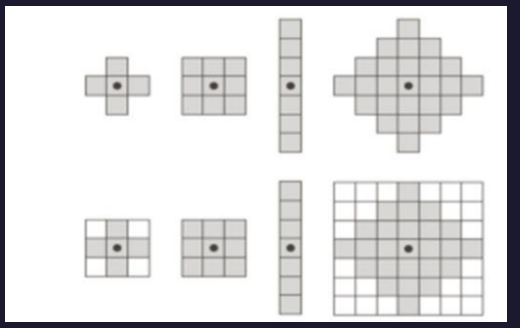




a b c

FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

Morphological Transformations: Structuring Element (結構元素)



- 取得結構元素，使用: cv2.getStructuringElement(shape, kernel_size)

```python
# Rectangular Kernel
kernel_rectangular = cv2.getStructuringElement (cv2.MORPH_RECT,(5,5))
print(kernel_rectangular)
print(type(kernel_rectangular))
```

```
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
<class 'numpy.ndarray'>
```

```python
# Elliptical Kernel
kernel_elliptical = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
print(kernel_elliptical)
print(type(kernel_elliptical))
```

```
[[0 0 1 0 0]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [0 0 1 0 0]]
<class 'numpy.ndarray'>
```

```python
# Cross-shaped Kernel
kernel_cross_shaped = cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))
print(kernel_cross_shaped)
print(type(kernel_cross_shaped))
```

```
[[0 0 1 0 0]
 [0 0 1 0 0]
 [1 1 1 1 1]
 [0 0 1 0 0]
 [0 0 1 0 0]]
<class 'numpy.ndarray'>
```

```python
# Rectangular Kernel, by numpy
kernel = np.ones((5,5),np.uint8)
print(kernel)
print(type(kernel))
```

```
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
<class 'numpy.ndarray'>
```

Morphological Transformations: Erosion(侵蝕)

```python
kernel = np.ones((5,5),np.uint8)
erosion = cv2.erode(img_gray,kernel,iterations = 1)

res = np.hstack((img_gray,erosion))
cv2_imshow(res)
```

$$A \ominus B = \{z|(B)_z \subseteq A\}$$
$$= \{z|(B)_z \cap A^c = \emptyset\}$$

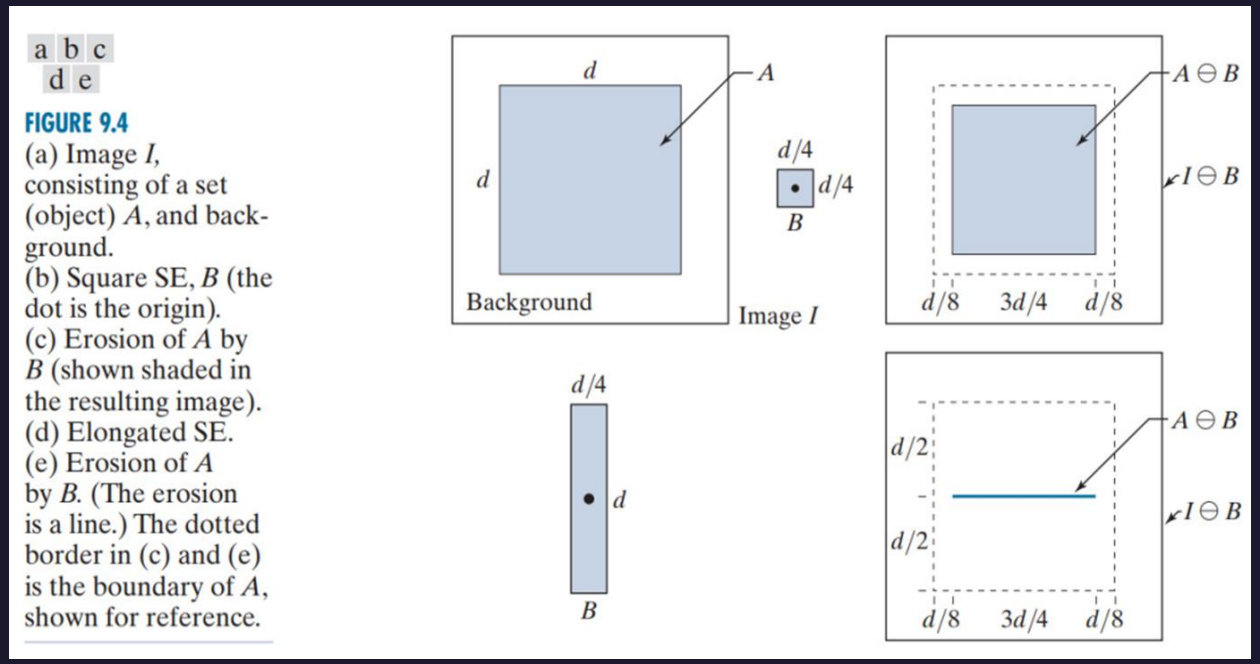
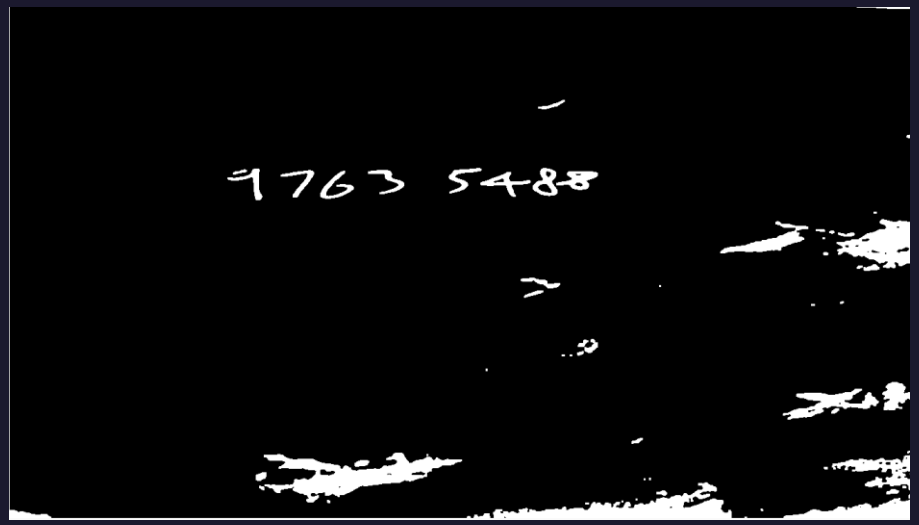
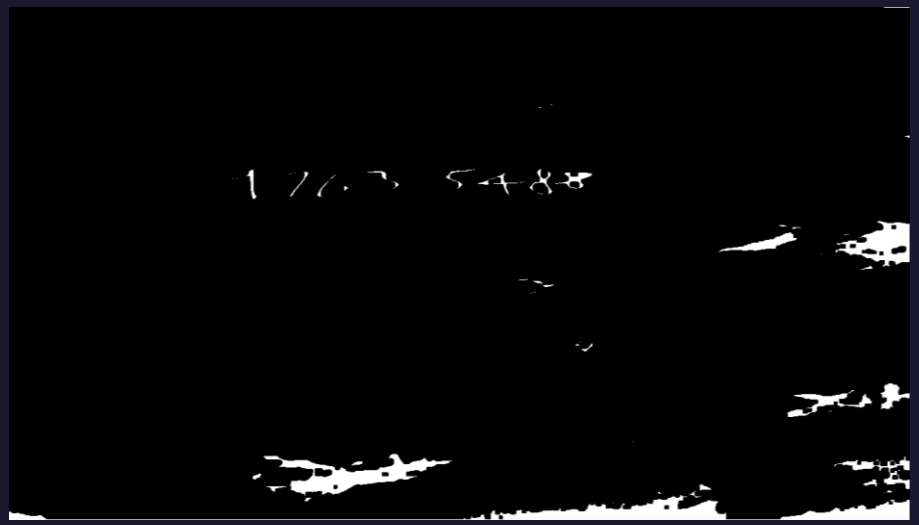


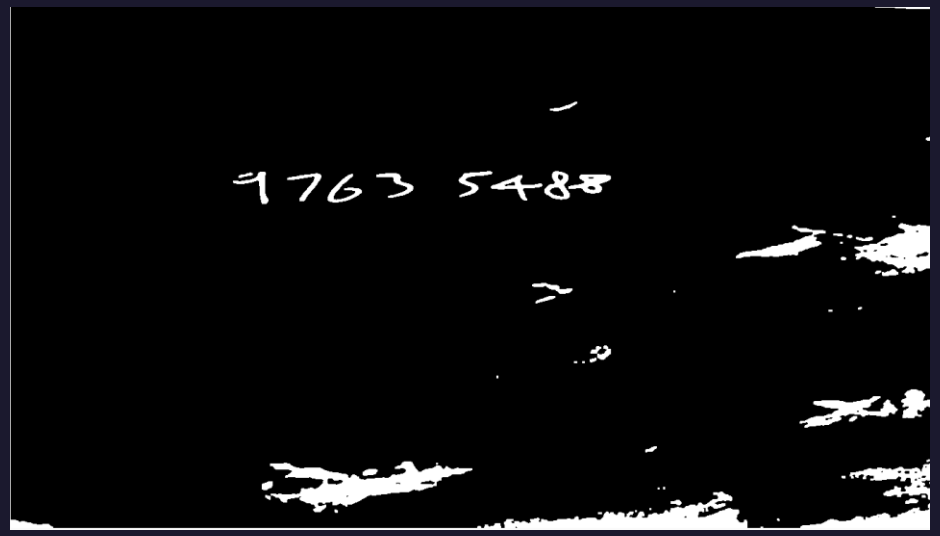
FIGURE 9.4
(a) Image I, consisting of a set (object) A, and background.
(b) Square SE, B (the dot is the origin).
(c) Erosion of A by B (shown shaded in the resulting image).
(d) Elongated SE.
(e) Erosion of A by B. (The erosion is a line.) The dotted border in (c) and (e) is the boundary of A, shown for reference.

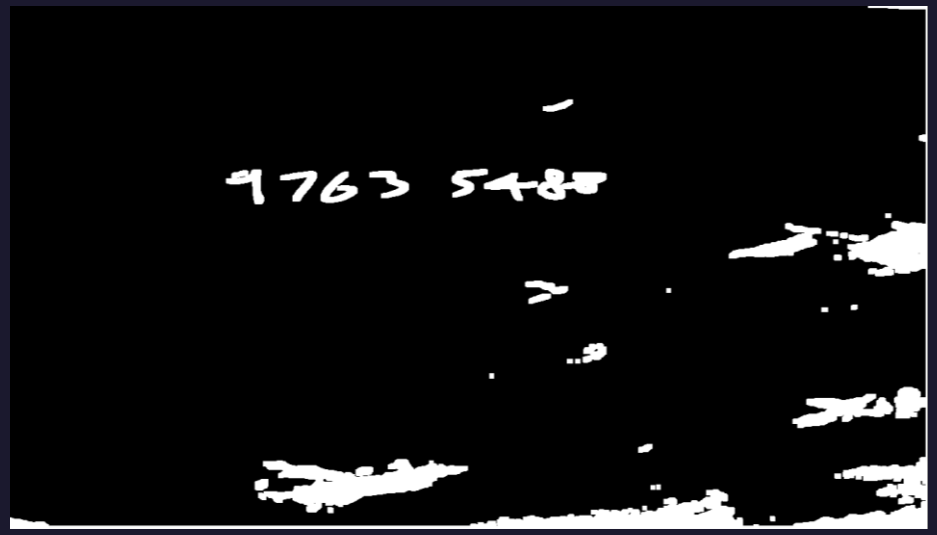




Morphological Transformations: Dilation (膨脹)

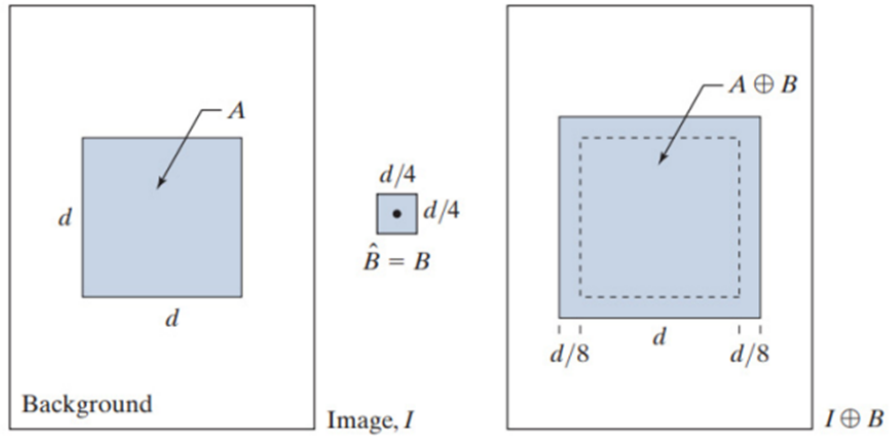
```python
kernel = np.ones((5,5),np.uint8)
dilation = cv2.dilate(img_gray,kernel,iterations = 1)

res = np.hstack((img_gray,dilation))
cv2_imshow(res)
```



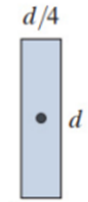


a b c
d e

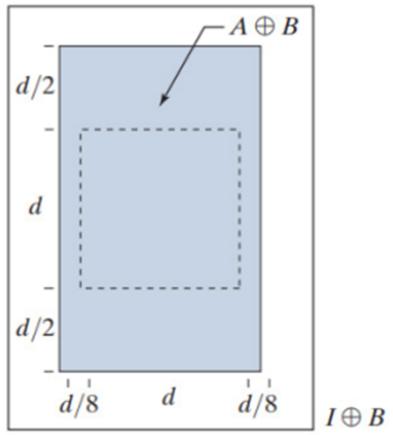
FIGURE 9.6
(a) Image I, composed of set (object) A and background.
(b) Square SE (the dot is the origin).
(c) Dilation of A by B (shown shaded).
(d) Elongated SE.
(e) Dilation of A by this element. The dotted line in (c) and (e) is the boundary of A, shown for reference.



$\hat{B} = B$



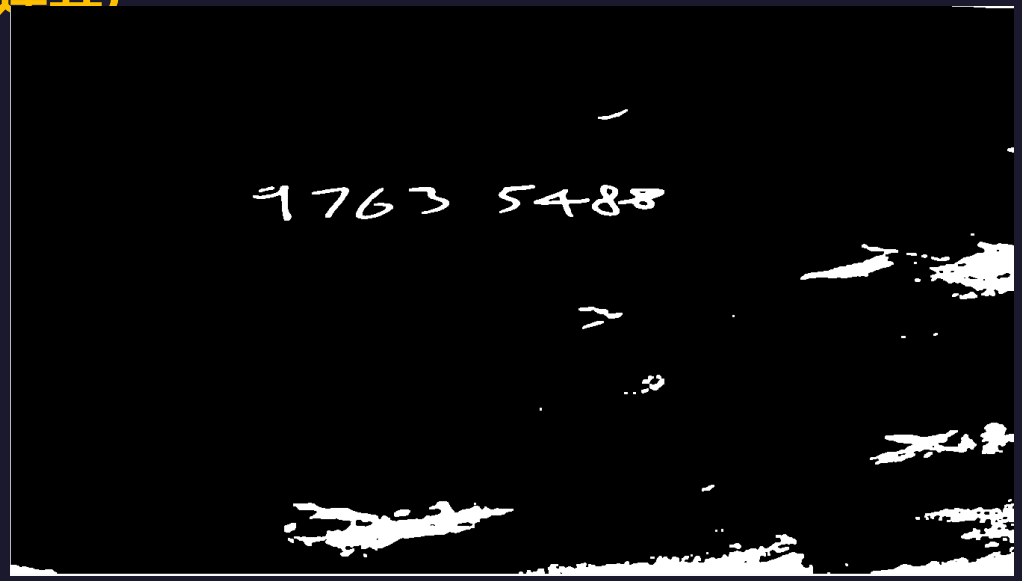
$\hat{B} = B$



$$A \oplus B = \left\{ z \mid \left(\hat{B}\right)_z \cap A \neq \phi \right\}$$
$$= \left\{ z \mid \left[\left(\hat{B}\right)_z \cap A\right] \subseteq A \right\}$$

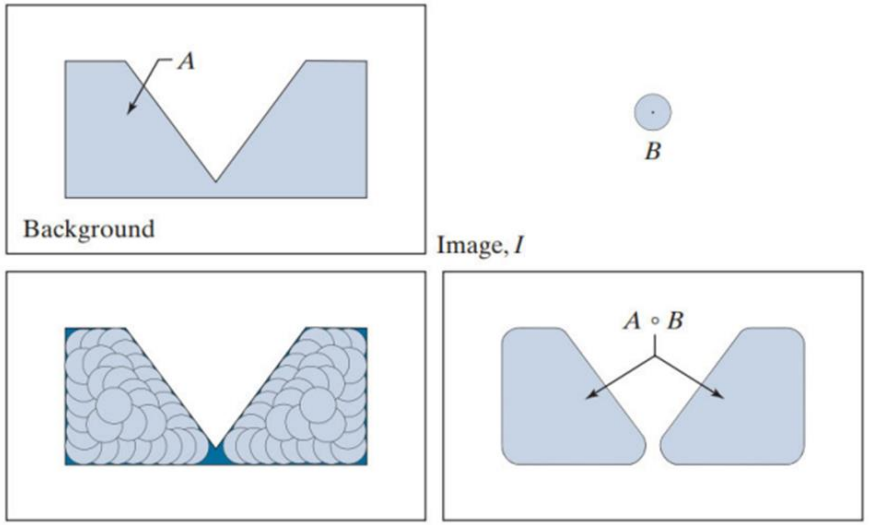
Morphological Transformations: Opening (斷開/開運算)

```
kernel = np.ones((5,5),np.uint8)
opening = cv2.morphologyEx(img_gray, cv2.MORPH_OPEN, kernel)
res = np.hstack((img_gray,opening))
cv2_imshow(res)
```



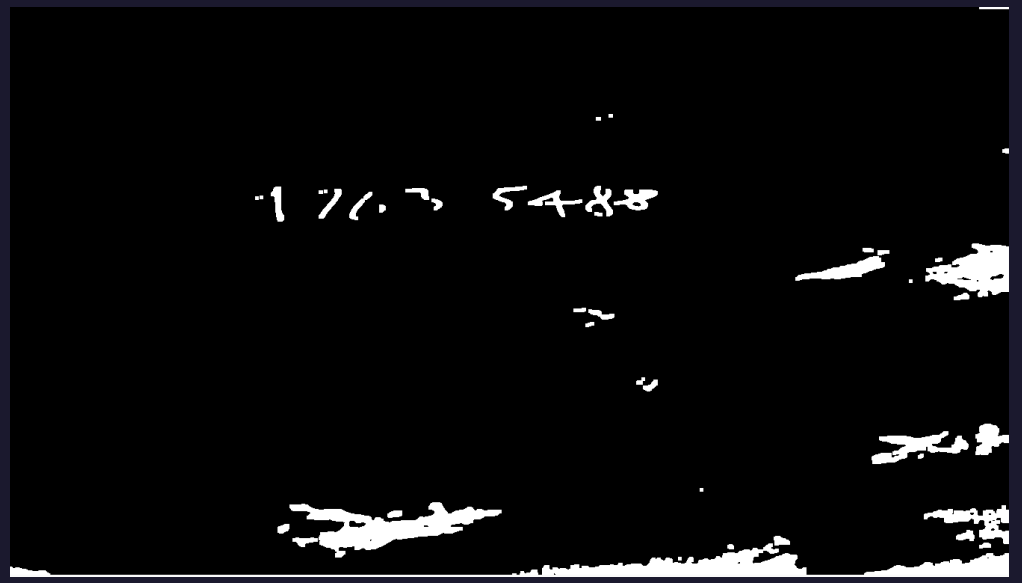
a b
c d

FIGURE 9.8
(a) Image I, composed of set (object) A and background.
(b) Structuring element, B.
(c) Translations of B while being contained in A. (A is shown dark for clarity.)
(d) Opening of A by B.



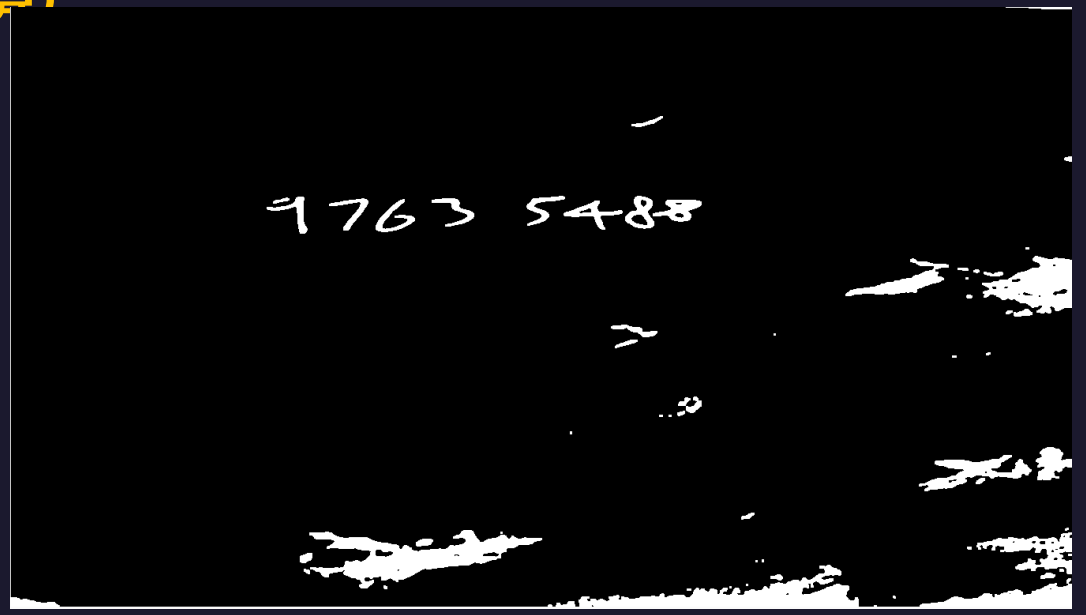
$$A \circ B = \bigcup \{(B)_z \mid (B)_z \subseteq A\}$$

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



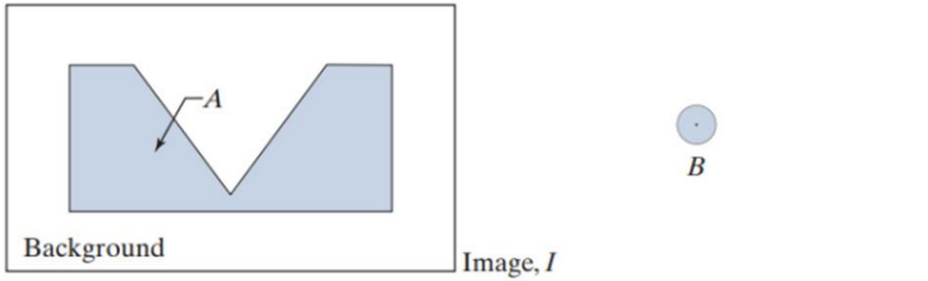
Morphological Transformations: Closing (閉合/閉運算)

```
1  kernel = np.ones((5,5),np.uint8)
2  closing = cv2.morphologyEx(img_gray, cv2.MORPH_CLOSE, kernel)
3  res = np.hstack((img_gray,closing))
4  cv2_imshow(res)
```

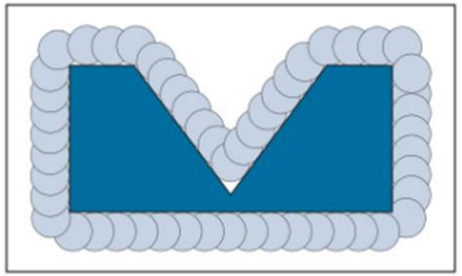


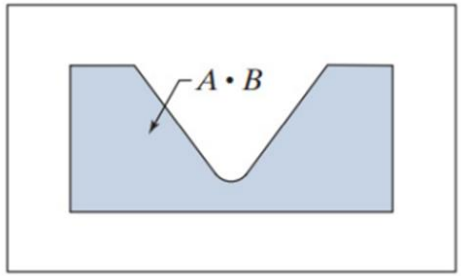
a b
c d

FIGURE 9.9
(a) Image I, composed of set (object) A, and background.
(b) Structuring element B.
(c) Translations of B such that B does not overlap any part of A. (A is shown dark for clarity.)
(d) Closing of A by B.



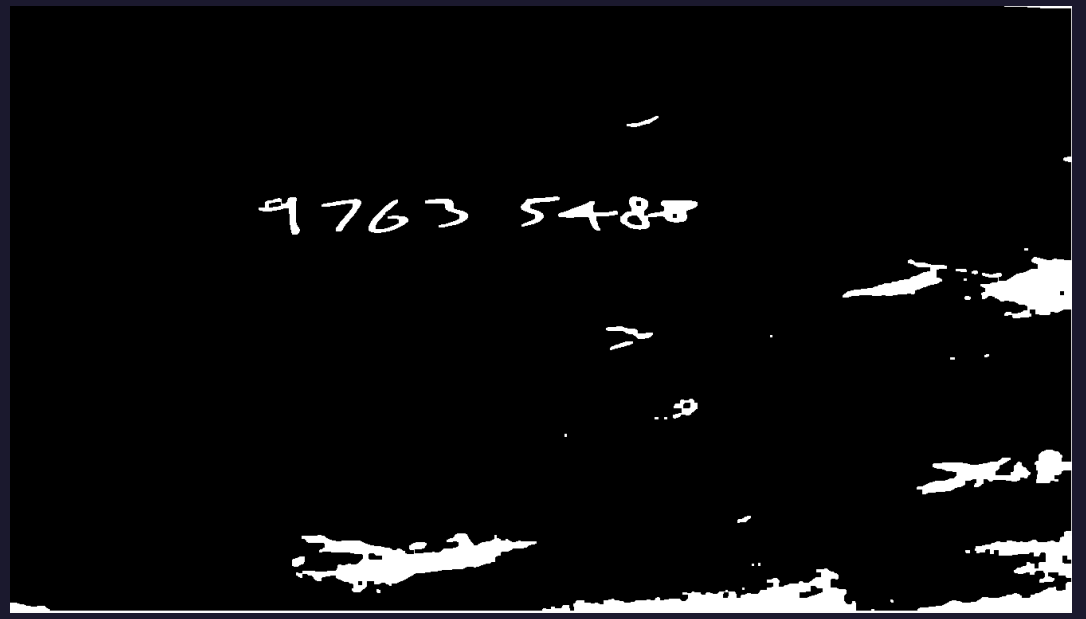
Image, I



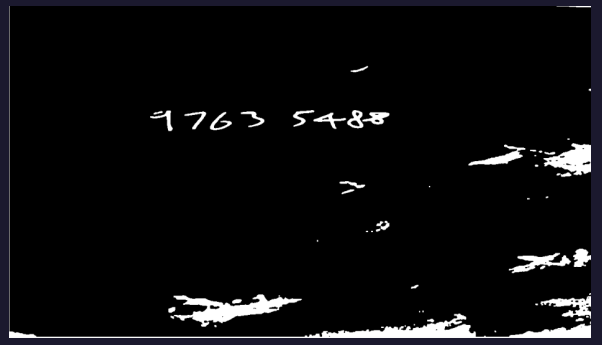


$$A \bullet B = \left[\bigcup\{(B)_z \mid (B)_z \cap A = \varnothing\}\right]^c$$

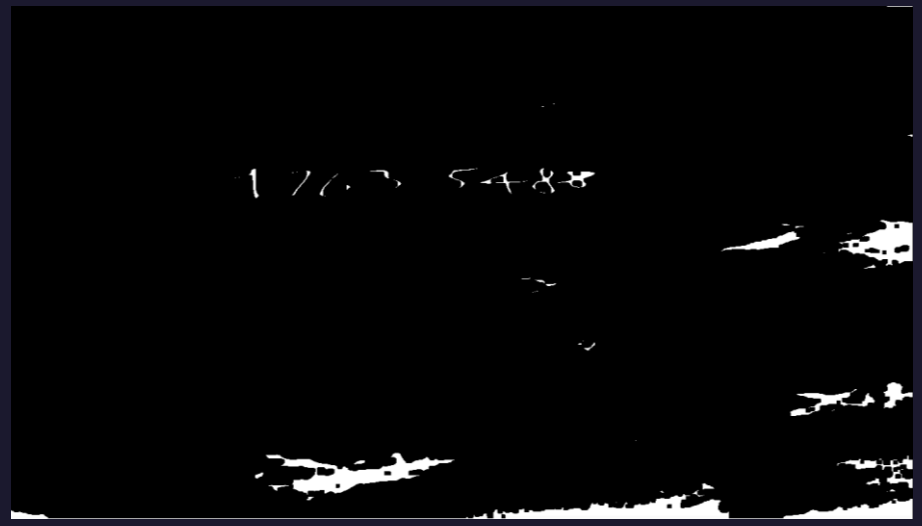
$$A \cdot B = (A \oplus B) \ominus B$$



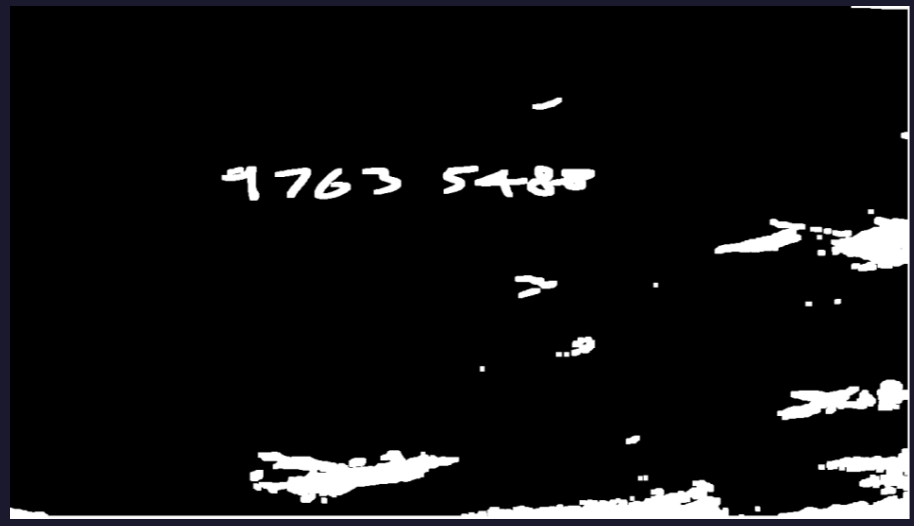
Morphological Transformations: result



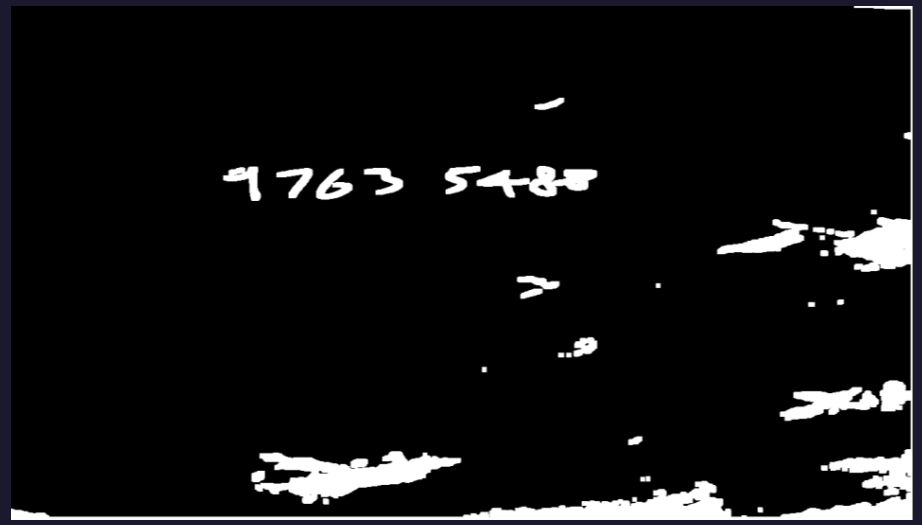
原圖



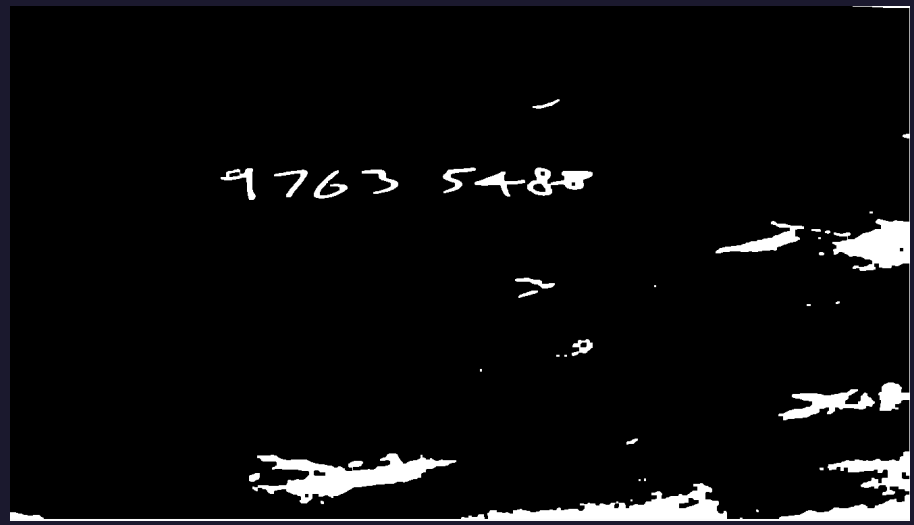
Erosion(侵蝕)



Dilation (膨脹)



Opening (斷開)



Closing (閉合)

國立臺灣科技大學
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Thanks for listening

Thank You

詹宏澤

Email: chanhts323@gmail.com

https://sites.google.com/view/peter-chan