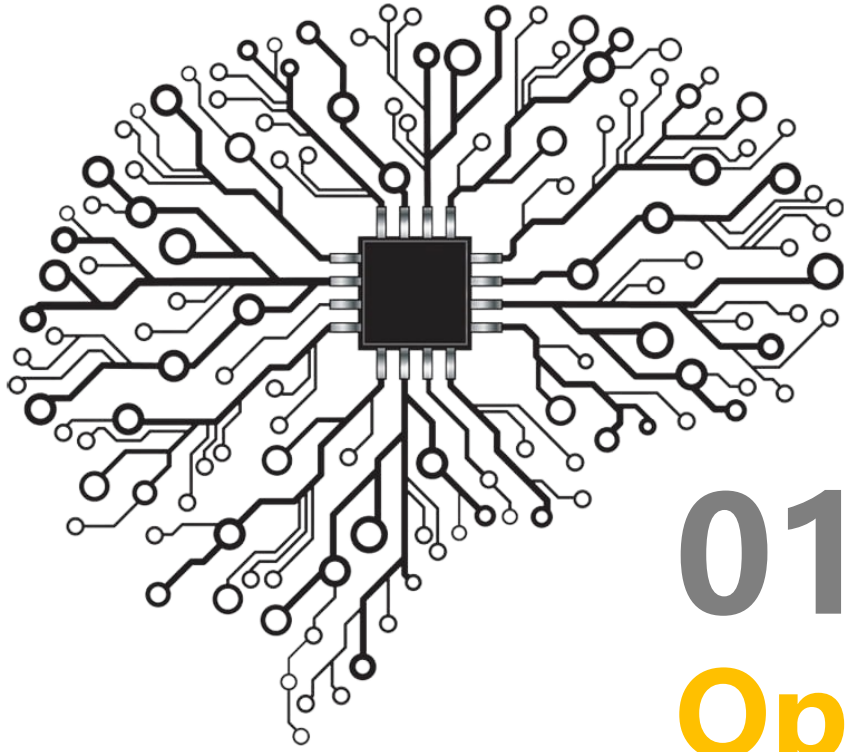
MIT Lab.
Multimedia
Intelligent Technical
Laboratory

去除雜訊干擾

目標任務

- **對此次課程提供的影像使用中值濾波、均值濾波、高斯濾波，以及形態學進行去除雜訊干擾**
- **使用函數:**
  - cv2.medianBlur
  - cv2.blur
  - cv2.GaussianBlur
  - cv2.morphologyEx
- **使用以下參數對影像做(中值、均值、高斯)濾波操作:**
  - Kernel_size：3
  - sigmaX：0 (only for gaussian)
  - sigmaY：0 (only for gaussian)
- **使用以下參數對影像做形態學操作:**
  - Kernel_size：3
  - Operation：cv2.MORPH_RECT



01

OpenCV：Denoise

中值濾波

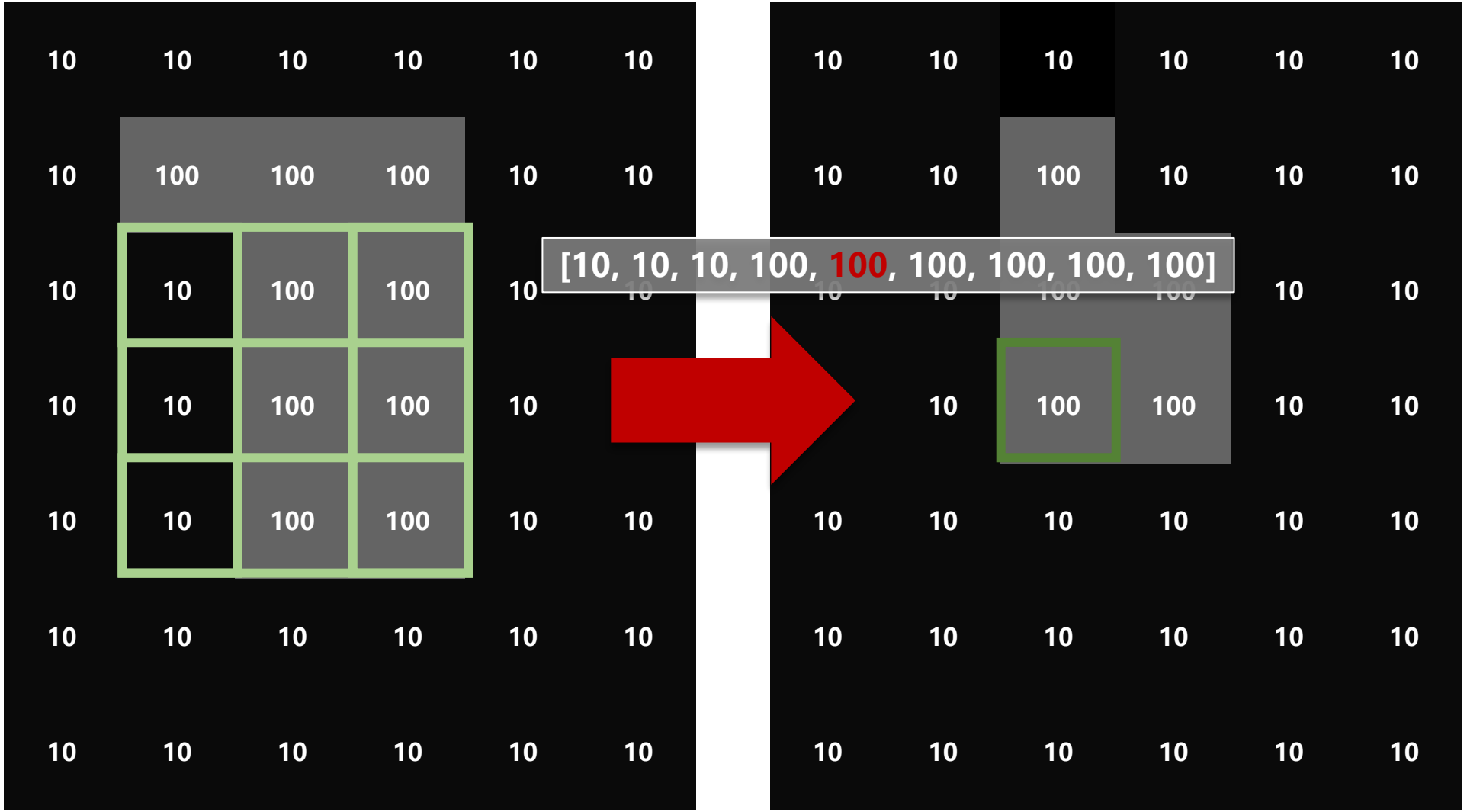
MIT Lab.
Multimedia
Intelligent Technical
Laboratory

```
cv2.medianBlur(src, ksize[, dst]) -> dst
```

Parameters	Description
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 ...

中值濾波

MIT Lab.
Multimedia
Intelligent Technical
Laboratory



均值濾波

MIT Lab.
Multimedia
Intelligent Technical
Laboratory

```
cv2.blur(src, ksize[, dst[, anchor[, borderType]]]) -> dst
```

Parameters	Description
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	blurring kernel size.
anchor	anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.
borderType	pixel extrapolation method. BORDER_WRAP is not supported.

均值濾波

MIT Lab.
Multimedia
Intelligent Technical
Laboratory

(100+100+100+10+100+100+10+100+100) / 9

Kernel

1	1	1
1	1	1
1	1	1

$$* \frac{1}{9}$$

10	10	10	10	10	10
10	100	100	100	10	10
10	10	100	100	10	10
10	10	100	100	10	10
10	10	100	100	10	10
10	10	10	10	10	10
10	10	10	10	10	10

50	50	70	50	30	10
30	40	60	50	30	10
30	50	80	70	40	10
40	70	70	40	10	
10	30	50	50	30	10
10	20	30	30	20	10
10	10	10	10	10	10

高斯濾波

MIT Lab.
Multimedia
Intelligent Technical
Laboratory

```
cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]]) -> dst
```

Parameters	Description
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; 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. BORDER_WRAP is not supported.

高斯濾波

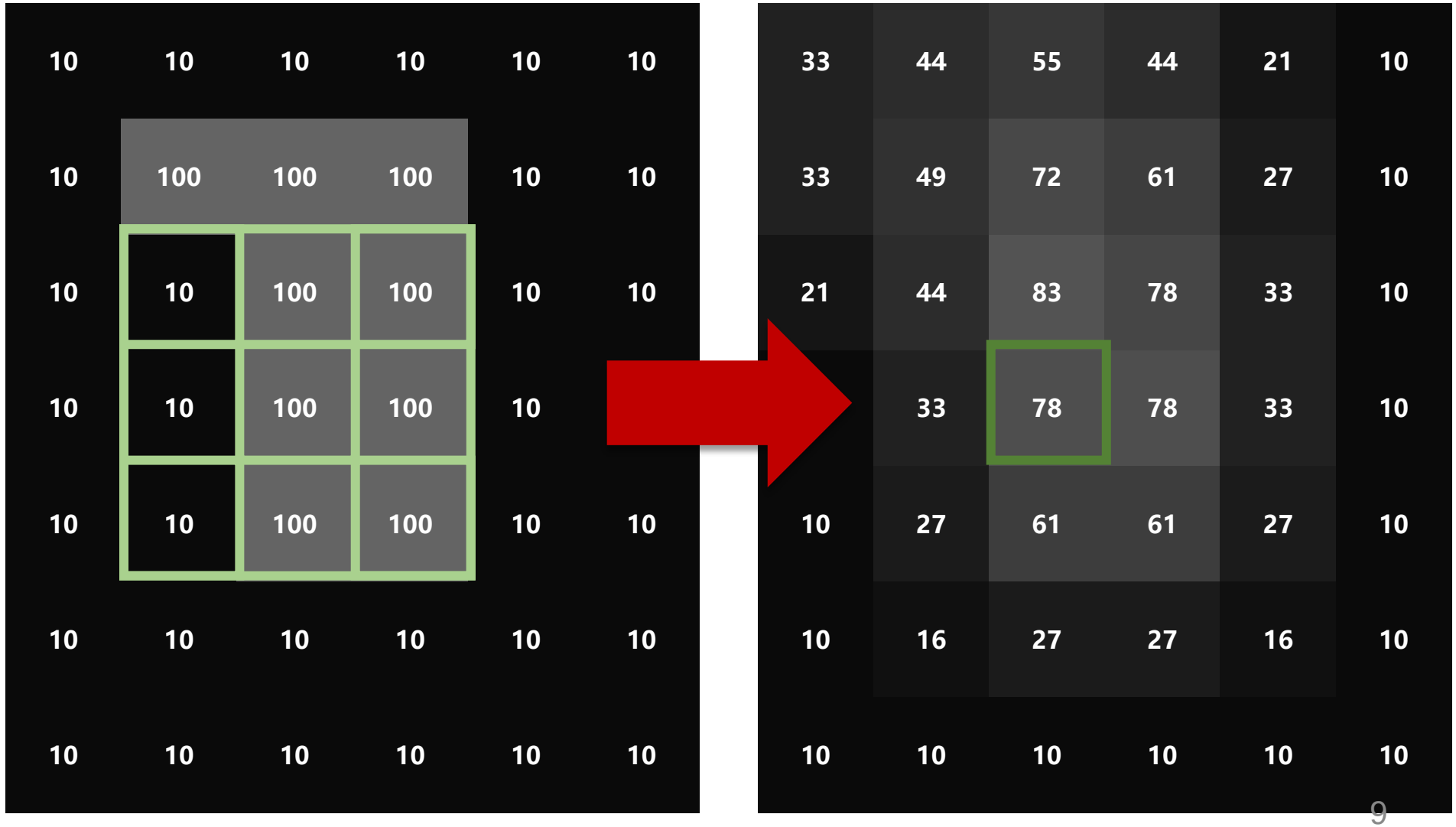
MIT Lab.
Multimedia
Intelligent Technical
Laboratory

(100*1+100*2+100*1+10*2+100*4+100*2+10*1+100*2+100*1) / 16

Kernel

1	2	1
2	4	2
1	2	1

$$* \frac{1}{16}$$



濾波操作

MIT Lab.
Multimedia
Intelligent Technical
Laboratory

```
def average_filter(grayimage, kernel_size=3):
    return cv2.blur

def median_filter(grayimage, kernel_size=3):
    return cv2.medianBlur

def gaussian_filter(grayimage, kernel_size=3):
    return cv2.GaussianBlur
```

主程式

MIT Lab.
Multimedia
Intelligent Technical
Laboratory

```
image = cv2.imread('./Lena_noise.bmp')
grayimage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

average = average_filter(grayimage)
median = median_filter(grayimage)
gaussian = gaussian_filter(grayimage)

titles = ['GRAY', 'AVERAGE', 'MEDIAN', 'GAUSSIAN']
images = [grayimage, average, median, gaussian]
for i in range(4):
    plt.subplot(2, 2, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

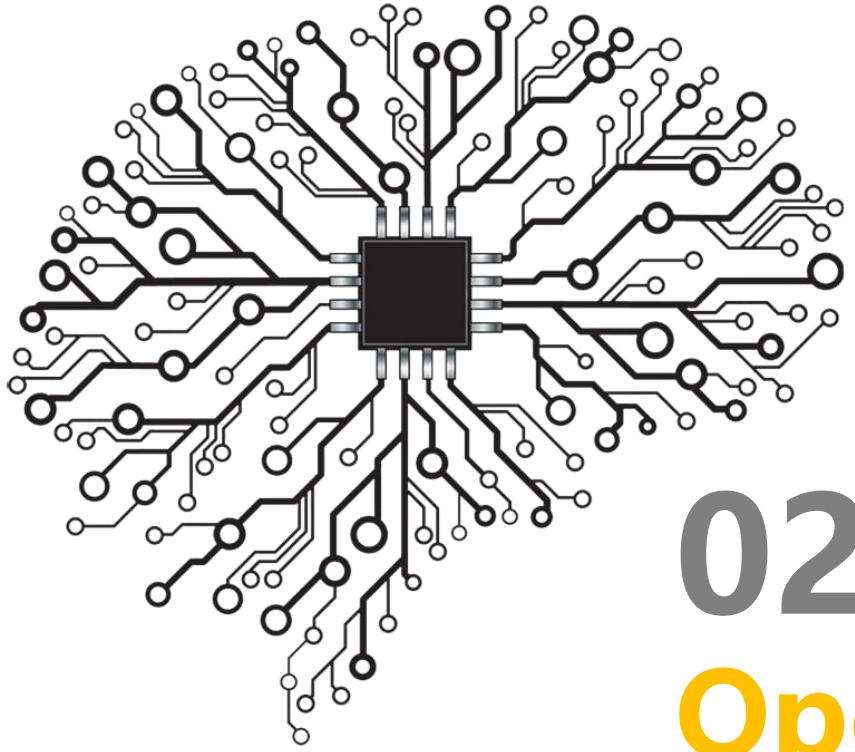
實作結果

GRAY

AVERAGE

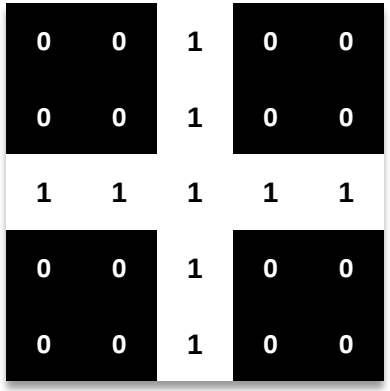
MEDIAN

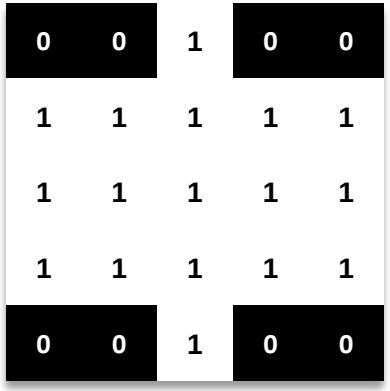
GAUSSIAN

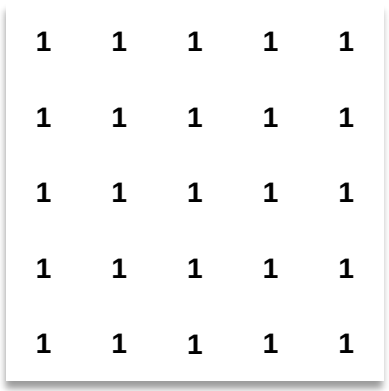


02
OpenCV：Morphology

卷積核結構







cv2.MORPH_CROSS **cv2.MORPH_ELLIPSE** **cv2.MORPH_RECT**

Shape	Description
cv2.MORPH_CROSS	生成由0、1構成的十字架卷積核
cv2.MORPH_ELLIPSE	生成由0、1構成的橢圓形卷積核
cv2.MORPH_RECT	生成全為1的方形卷積核

膨脹 & 侵蝕

形態學


```
cv2.morphologyEx(src, op, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) -> dst
```

Parameters	Description
src	Source image. The number of channels can be arbitrary. The depth should be one of CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
dst	Destination image of the same size and type as source image.
op	Type of a morphological operation.
kernel	Structuring element. It can be created using getStructuringElement.
anchor	Anchor position with the kernel. Negative values mean that the anchor is at the kernel center.
iterations	Number of times erosion and dilation are applied.
borderType	pixel extrapolation method. BORDER_WRAP is not supported.
borderValue	Border value in case of a constant border. The default value has a special meaning.

取得卷積核

MIT Lab.
Multimedia
Intelligent Technical
Laboratory

```
def get_kernel(kernel_size, kernel_shape):
    assert kernel_shape in ['rect', 'cross', 'ellipse'], 'The kernel_size must be either rect, cross, or ellipse'
    if kernel_shape == 'rect':
        shape = cv2.MORPH_RECT
    elif kernel_shape == 'cross':
        shape = cv2.MORPH_CROSS
    elif kernel_shape == 'ellipse':
        shape = cv2.MORPH_ELLIPSE
    return cv2.getStructuringElement
```

形態學操作

MIT Lab.
Multimedia
Intelligent Technical
Laboratory

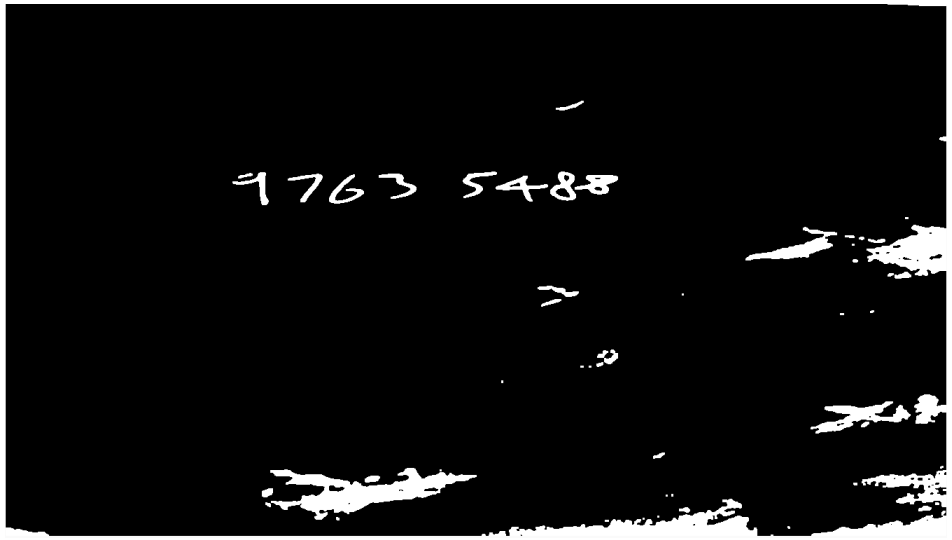
```
def morphology(grayimage, kernel_size=  kernel_shape=  operation=  :
    assert operation in ['open', 'close'], 'The operation must be either open or close.'
    if operation == 'open':
        op = cv2.MORPH_OPEN
    elif operation == 'close':
        op = cv2.MORPH_CLOSE
    kernel = get_kernel(kernel_size, kernel_shape)
    return cv2.morphologyEx
```

主程式

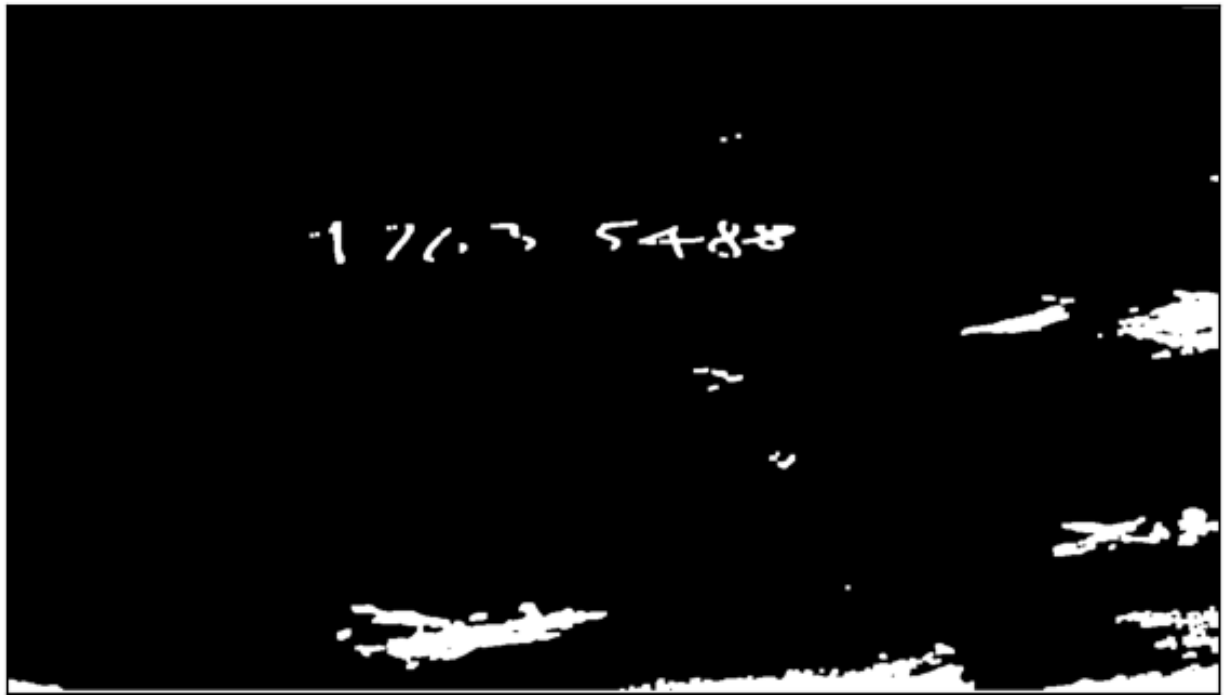

```
image = cv2.imread('./Number_noise.bmp')
grayimage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
morph_open = morphology(grayimage, operation='open')
morph_close = morphology(grayimage, operation='close')

titles = ['OPEN', 'CLOSE']
images = [morph_open, morph_close]

plt.figure(figsize=(10, 10))
for i in range(2):
    plt.subplot(1, 2, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

實作結果

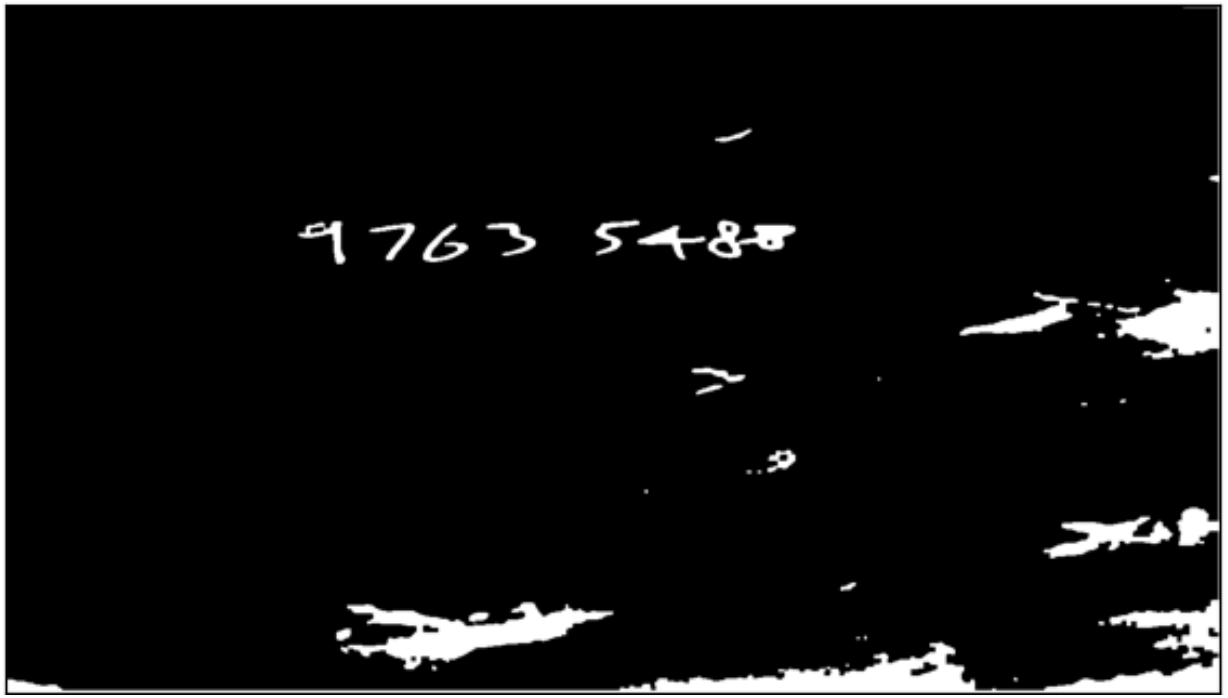
MIT Lab.
Multimedia
Intelligent Technical
Laboratory



OPEN



CLOSE



MIT Lab.
Multimedia
Intelligent Technical
Laboratory

Thanks for listening