影像處理

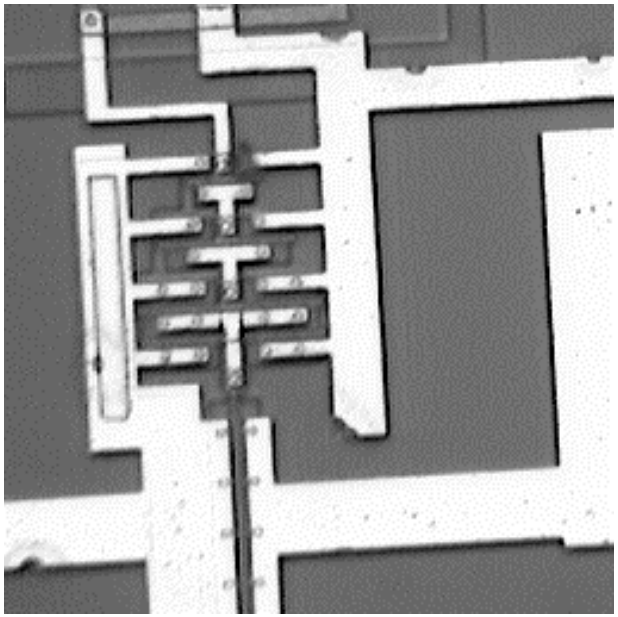
(Image Processing)

真理大學 資訊工程系 吳汶涓老師

Course 9

影像分割

Outline



9.1 前言

9.2 閥值運算

9.3 閥值的應用

9.4 設定合適的閥值

9.5 可適性閥值運算

9.6 邊緣偵測

9.7 導數與邊緣

9.8 二階導數

9.9 Canny邊緣偵測

9.10 Hough轉換

9.11 在Matlab中執行Hough轉換

9.1 前言

- **切割**(segmentation)指的是將影像按照組成部分分割，或將影像中各個物體分割開來。
- 本章將會探討兩個非常重要的主題：

  1. 閥值運算
  2. 邊緣偵測

9.2 閥值運算

- **單一閥值**：依照門檻值T將灰階影像轉換成二元黑白的數位影像。(基礎分割處理)

```
>> r=imread('rice.tif');
>> imshow(r),figure,imshow(r>110)
```

灰階值$>T$，則像素轉換為白色
灰階值$\leq T$，則像素轉換為黑色

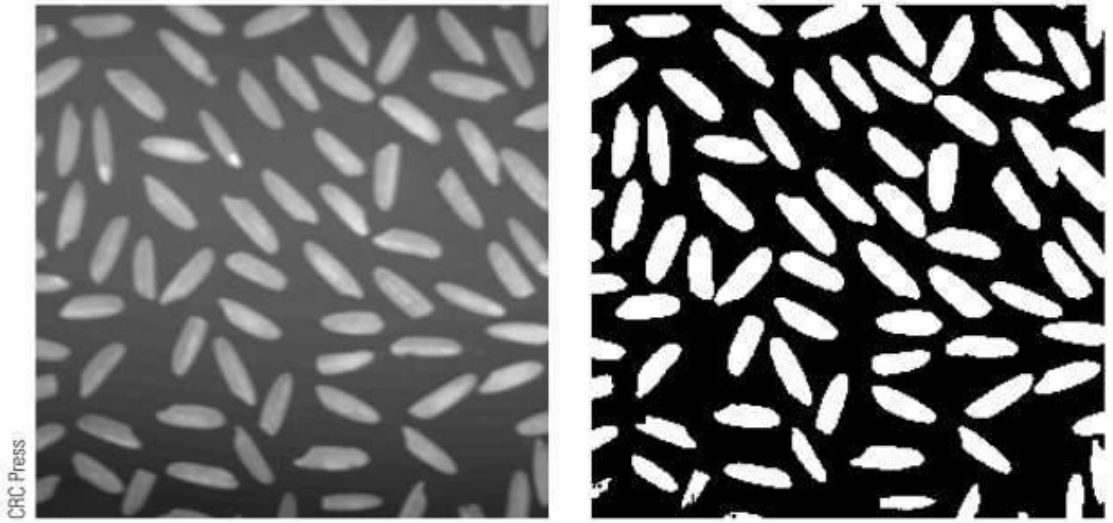


圖 9.1 稻穀的閥值影像

```
>> b=imread('bacteria.tif');
>> imshow(b),figure,imshow(b>100)
```

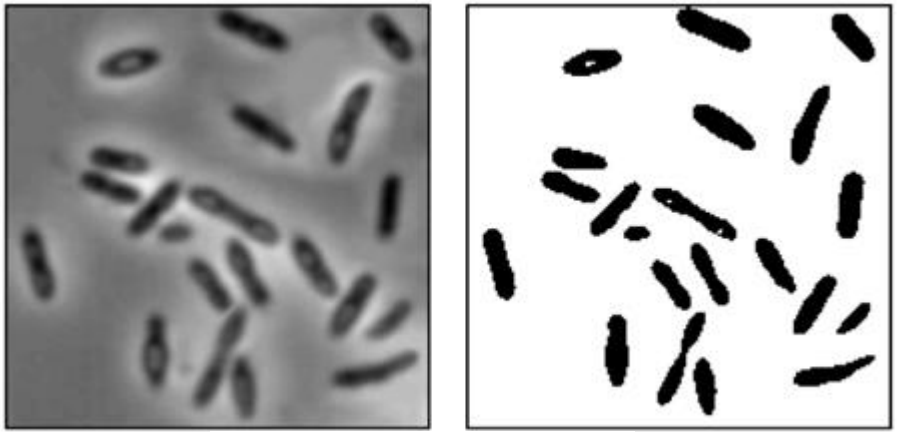


圖 9.2 細菌的閥值影像

- 也可以使用函數**im2bw**，對**任何資料形態**的影像進行閥值運算，語法為：im2bw(image,level)

```
>> im2bw(r,0.43);
>> im2bw(b,0.39);
```

level: 介於0~1的數值

- 除了擷取物體外，也可顯示影像中隱藏的特性

```
>> p=imread('paper1.tif');
>> imshow(p),figure,imshow(p>241)
```

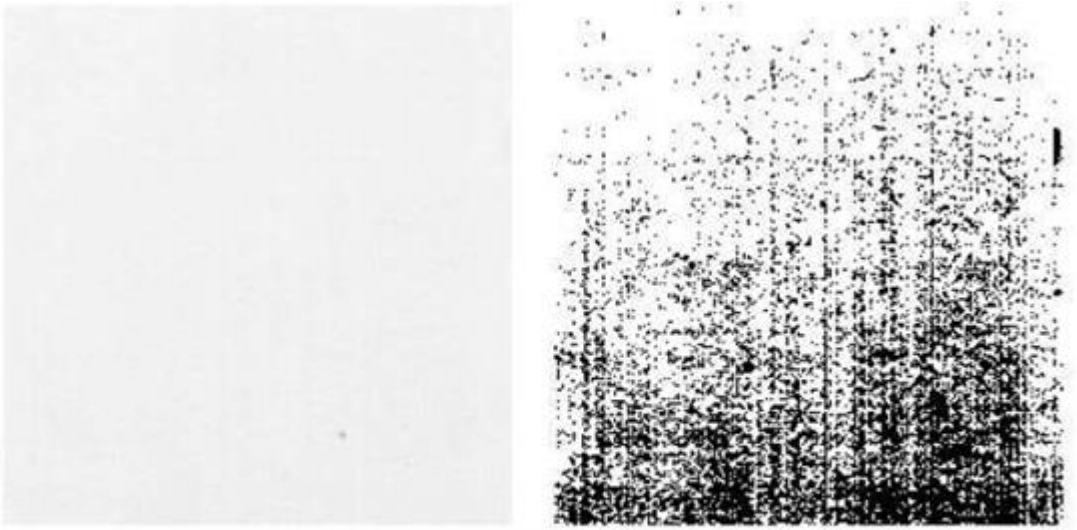
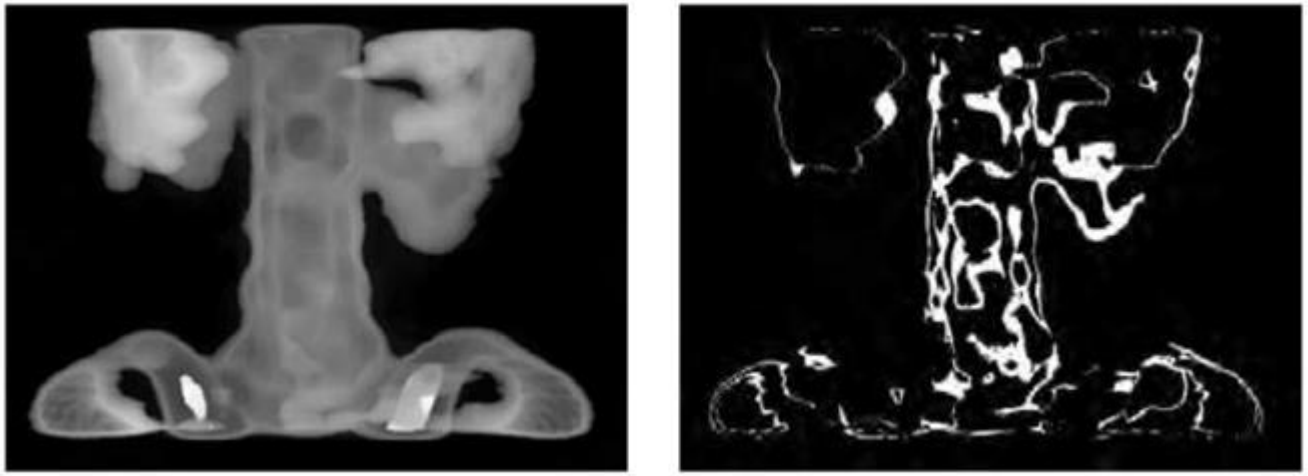


圖 9.3 紙張影像及閥值結果

- **雙重閥值**：設定兩個門檻值 *T1* 與 *T2*

$$\begin{cases} \text{灰階值介於 } T_1 \text{ 與 } T_2\text{，則像素轉換為白色} \\ \text{灰階值為其他數值，則像素轉換為黑色} \end{cases}$$

```
>> [x,map]=imread('spine.tif');
>> s=ind2gray(x,map);
>> imshow(s),figure,imshow(s>115 & s<125)
```



(可顯示出脊椎一些細緻的地方)

圖 9.4 影像 spine.tif 及雙重閥值之結果

9.3 閥值的應用

- 去除影像中不需要的細節
- 顯示隱藏的細節
- 去除文字或圖畫中雜亂的背景
- …

```
>> r=rand(256)*128+127;
>> t=imread('text.tif');
>> tr=uint8(r.*double(not(t));
>> imshow(tr)
```

```
>> imshow(tr>100)
```

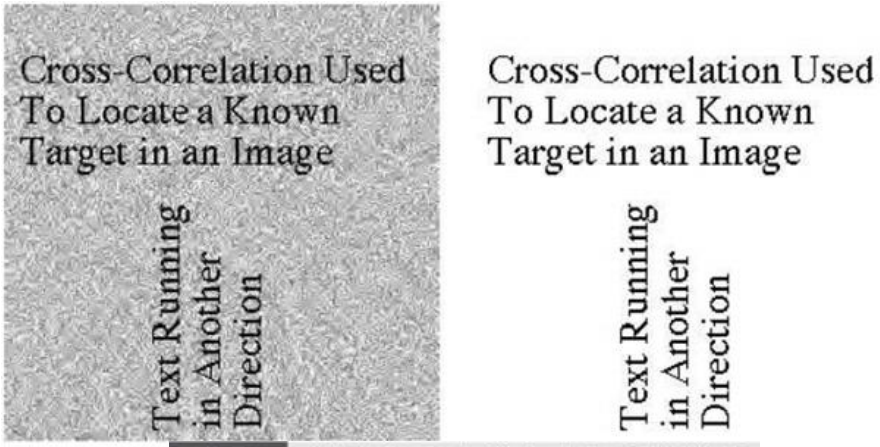


圖 9.5 雜亂背景中的文字及閥值結果

9.4 設定合適的閥值

```
>> n=imread('nodules1.tif');
>> imshow(n);
>> n1=im2bw(n,0.35);
>> n2=im2bw(n,0.75);
>> figure,imshow(n1),figure,imshow(n2)
```

圖 9.6 閥值設定

Q: 如何才能找到合適的閥值呢?

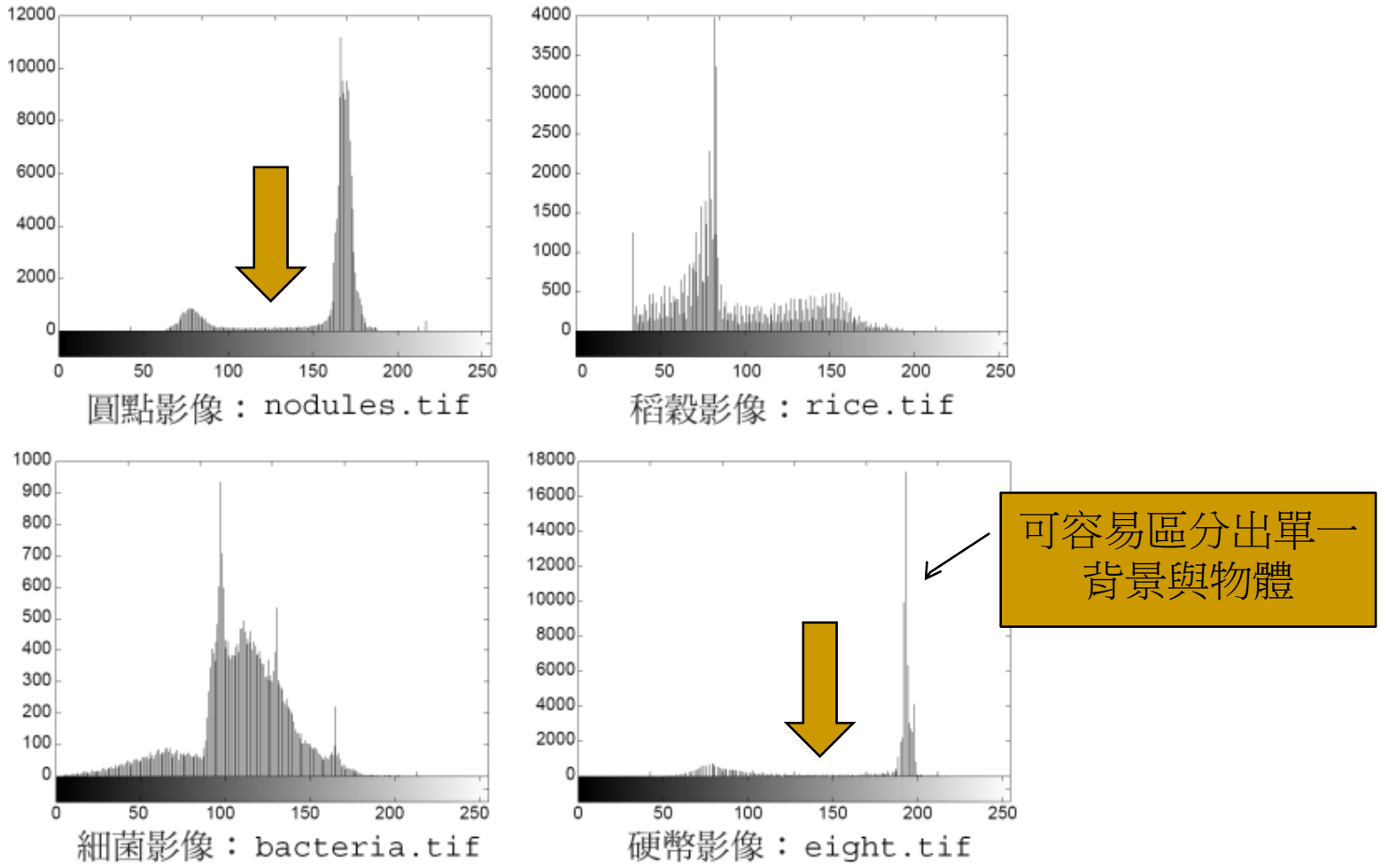

圓點影像：nodules.tif
稻穀影像：rice.tif
細菌影像：bacteria.tif
硬幣影像：eight.tif
可容易區分出單一背景與物體

圖 9.7 直方圖

- 將直方圖視為一種機率分布： $p_i = n_i / N$

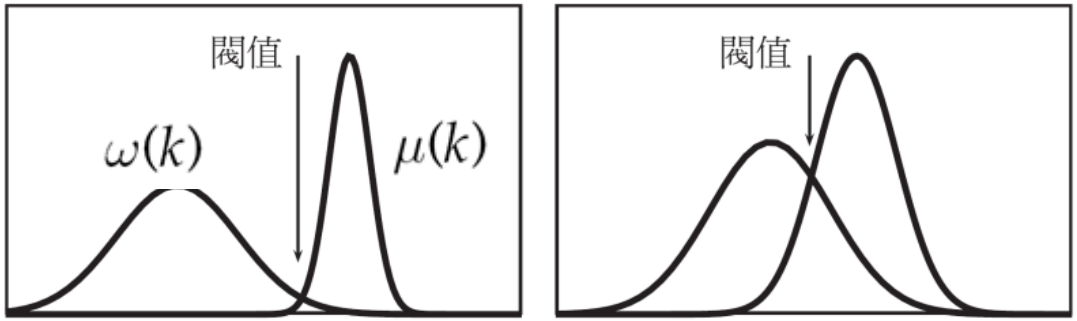


圖 9.8 分割直方圖以便設定閥值

$$\omega(k) = \sum_{i=0}^{k} p_i \qquad \mu(k) = \sum_{i=k+1}^{L-1} p_i \quad \Rightarrow \quad \omega(k) + \mu(k) = \sum_{i=0}^{L-1} p_i = 1$$

➔ 希望計算出$\omega(k)$ 與$\mu(k)$ 差異最大時的k 值 **(Otsu's method)**

$$\frac{\left(\mu_T \omega(k) - \mu(k)\right)^2}{\omega(k)\mu(k)}$$

```
>> tn=graythresh(n)
tn = 0.5804

>> r=imread('rice.tif');
>> tr=graythresh(r)
tr = 0.4902

>> b=imread('bacteria.tif');
>> tb=graythresh(b)
tb = 0.3765

>> e=imread('eight.tif');
>> te=graythresh(e)
te = 0.6490
```

```
>> imshow(im2bw(n,tn))
>> figure,imshow(im2bw(r,tr))
>> figure,imshow(im2bw(b,tb))
>> figure,imshow(im2bw(e,te))
```

圓點

稻穀

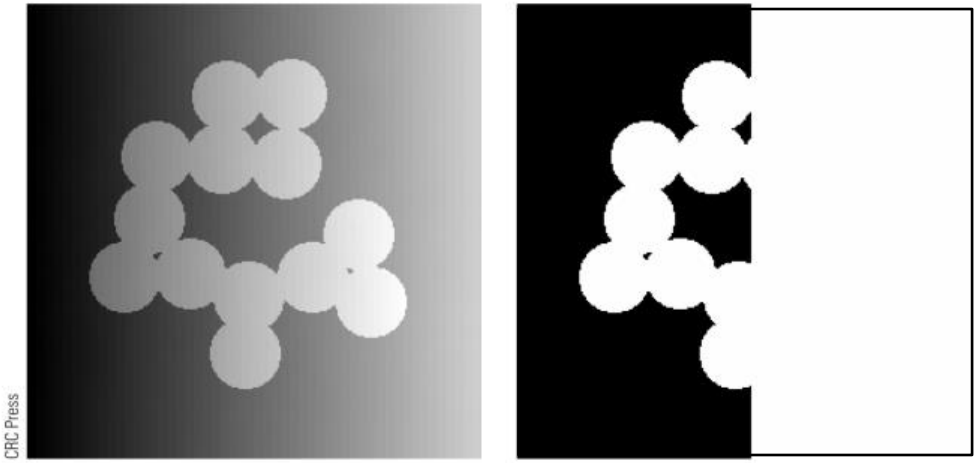
細菌

硬幣

9.5 可適性閥值運算

- 當物體和背景的**亮度都不均**時：

```
>> c=imread('circles.tif');
>> x=ones(256,1)*[1:256];
>> c2=double(c).*(x/2+50)+(1-double(c)).*x/2;
>> c3=uint8(255*mat2gray(c2));
```




```
>> t=graythresh(c3)

t =

    0.4196

>> ct=im2bw(c3,t);
```

圖 9.10 閥值運算 (a) 圓形影像：c3 (b) 閥值運算：ct

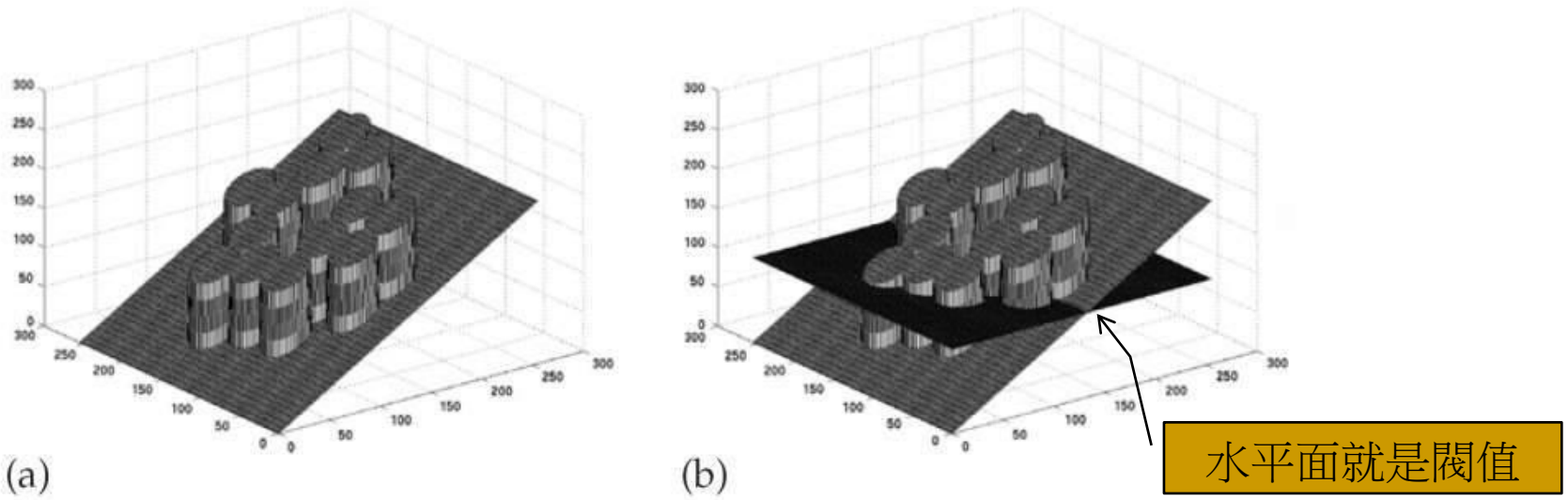


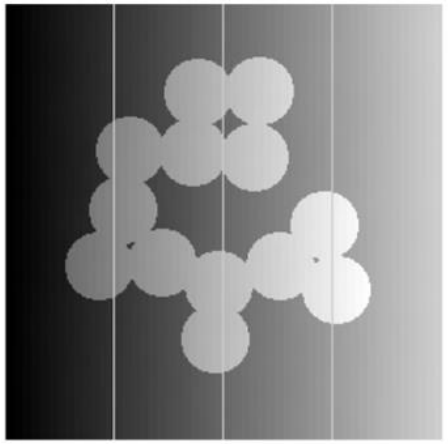
圖 9.11 閥值運算：函數顯示 (a) 以函數顯示影像 (b) 閥值運算

➔解決方式：將影像切成小塊

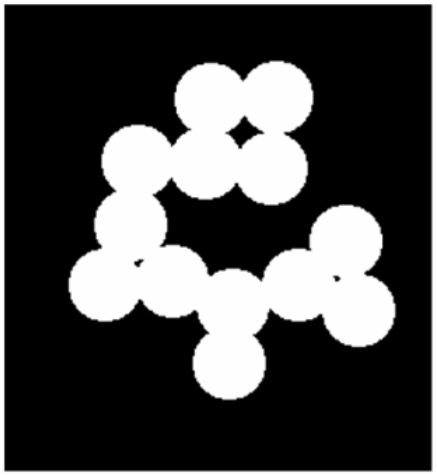
```
>> p1=c3(:,1:64);
>> p2=c3(:,65:128);
>> p3=c3(:,129:192);
>> p4=c3(:,193:256);
```

```
>> g1=im2bw(p1,graythresh(p1));
>> g2=im2bw(p2,graythresh(p2));
>> g3=im2bw(p3,graythresh(p3));
>> g4=im2bw(p4,graythresh(p4));
```

```
>> imshow([g1 g2 g3 g4])
```



(a) 切割影像



(b) 分別進行閥值運算

9.6 邊緣偵測

- 最常用於偵測邊緣的指令：

```
edge(image,'method',parameters...)
```

- 所謂邊緣：為像素值局部的不連續狀況

51	52	53	59
54	52	53	62
50	52	53	68
55	52	53	55

(a)

50	53	155	160
51	53	160	170
52	53	167	190
51	53	162	155

(b)

圖 9.13 像素區塊

9.7 導數與邊緣

- 基本定義：

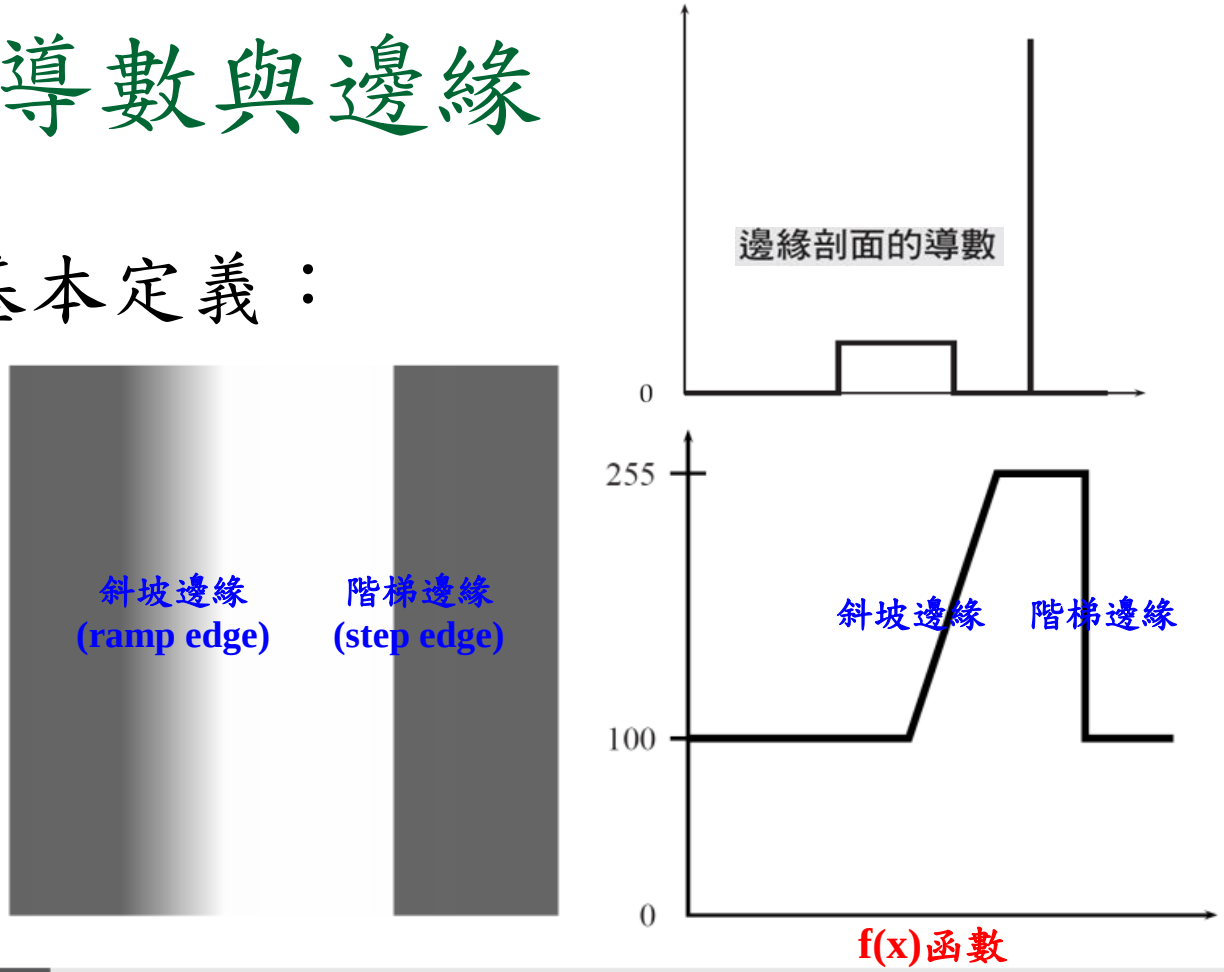


圖 9.14 邊緣及其剖面

微分(differentiation)→處理影像不連續之處

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

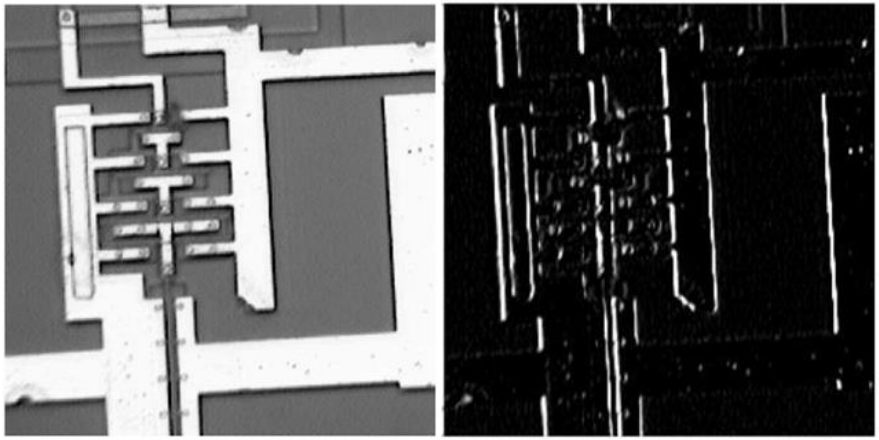
$$\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y)$$

- 邊緣偵測濾波器 -- **Prewitt** filter

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

```
>> ic=imread('ic.tif');
```

```
>> px=[-1 0 1;-1 0 1;-1 0 1];
>> icx=filter2(px,ic);
>> figure,imshow(icx/255)
```



垂直方向

```
>> py=px';
>> icy=filter2(py,ic);
>> figure,imshow(icy/255)
```

➔ icx矩陣是垂直方向的梯度值(gradient)，但每個像素點的梯度大小則為 $\sqrt{(\frac{\partial f}{\partial x})^2+(\frac{\partial f}{\partial y})^2}$ 或 $\max\{|\frac{\partial f}{\partial x}|,|\frac{\partial f}{\partial y}|\}$ 或 $|\frac{\partial f}{\partial x}|+|\frac{\partial f}{\partial y}|$

```
>> edge_p=sqrt(icx.^2+icy.^2);
>> figure,imshow(edge_p/255)
```

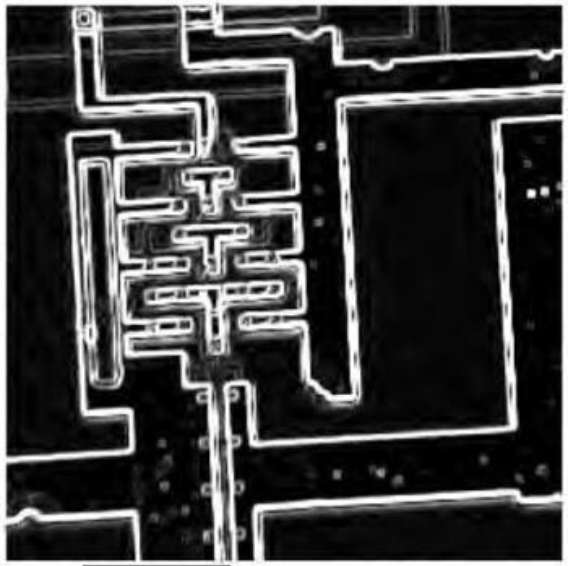


圖 9.18 (a)

```
>> edge_t=im2bw(edge_p/255,0.3);
```

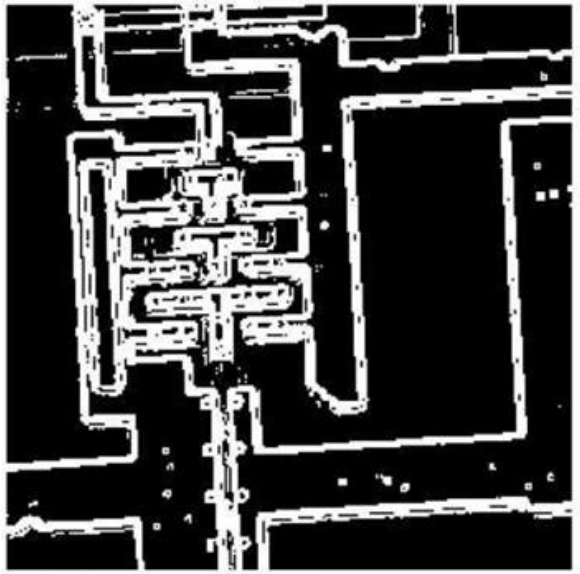


圖 9.18 (b)

```
>> edge_p=edge(ic,'prewitt');
```

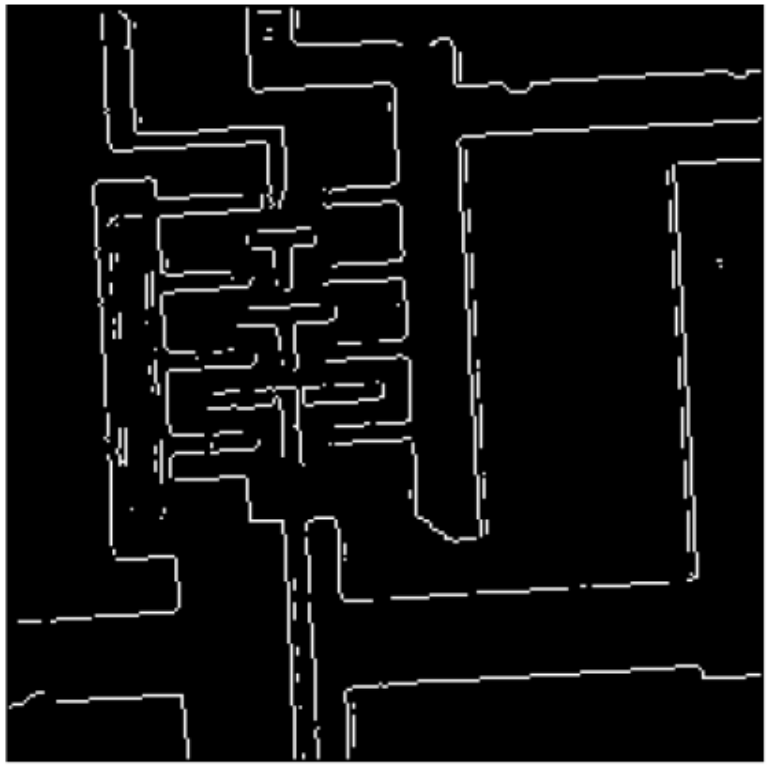


圖 9.19 edge 函數使用 Prewitt 濾波器

- 邊緣偵測濾波器 -- **Roberts** filter (交叉梯度)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad 與 \quad \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- 邊緣偵測濾波器 -- **Sobel** filter (凸顯中間像素)

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad 與 \quad \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

```
>> edge_r=edge(ic,'roberts');
>> figure,imshow(edge_r)
```

```
>> edge_s=edge(ic,'sobel');
>> figure,imshow(edge_s)
```

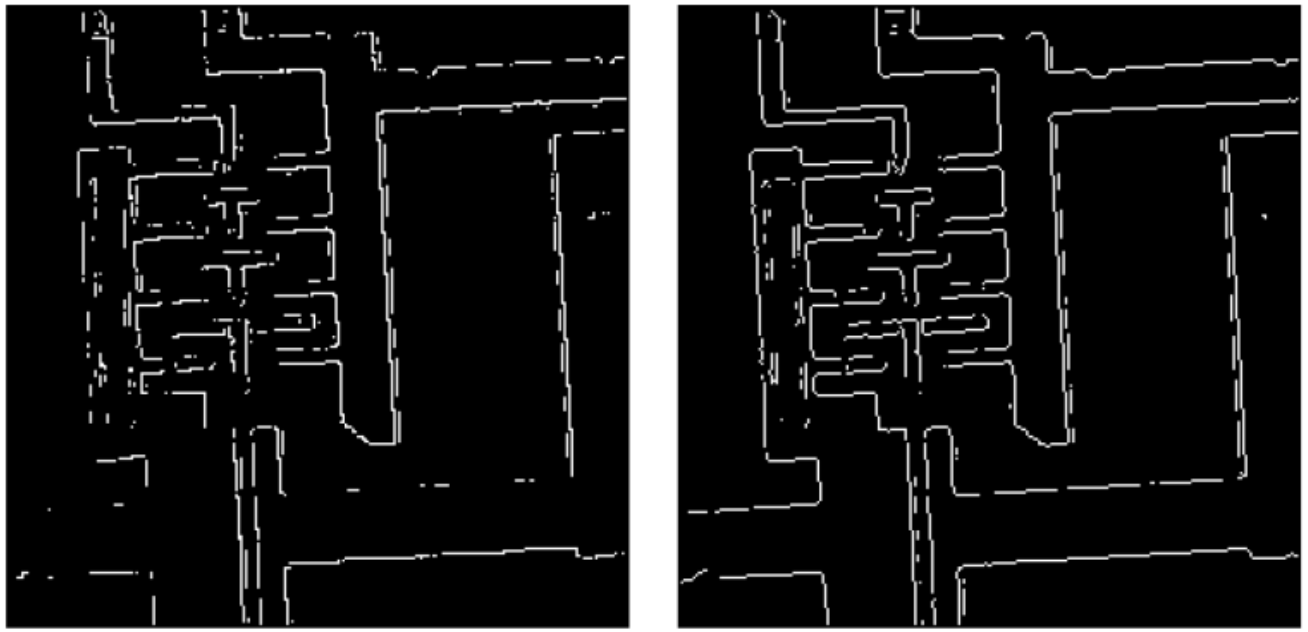


圖 9.20 Roberts 和 Sobel 濾波器的結果 (a) Roberts 邊緣偵測 (b) Sobel 邊緣偵測

練習

- 將自拍的彩色影像讀入並產生其灰階子影像：

```
>> f=imread('XXX.tif');
>> fg=rgb2gray(f);
```

➔ 找出合適的閥值，將影像變成二元黑白圖

➔ 帶入Roberts, Prewitt, Sobel的濾波器，利用filter2函數找出偵測的邊緣圖，並分析結果哪種效果較好，為什麼？

PS: 上傳時，除了繳交.m程式與word報告外，還需繳交自拍彩色影像。