

影像處理

(Image Processing)

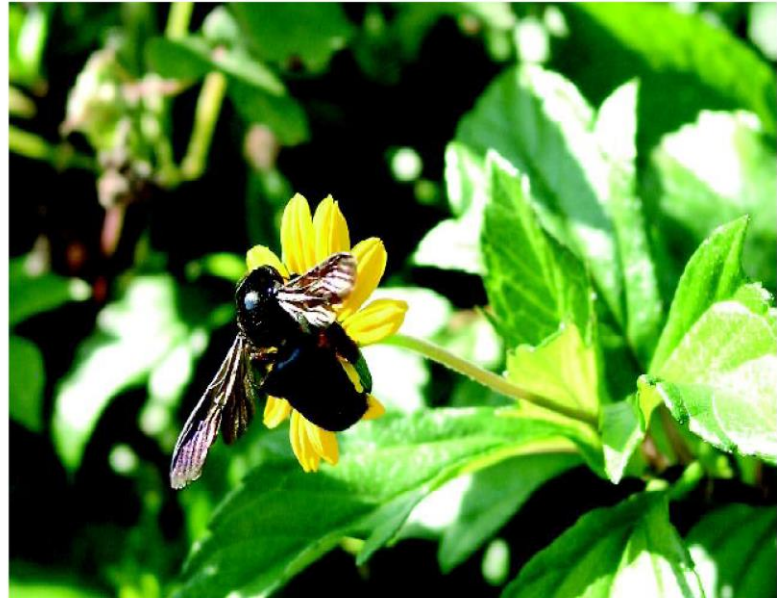
真理大學 資訊工程系 吳汶涓老師

Course 3
影像顯示



Outline

- 3.1 前言
- 3.2 影像顯示的基礎
- 3.3 imshow函數
- 3.4 位元平面
- 3.5 空間解析度
- 3.6 量化與混色



3.2 影像顯示的基礎

- 使用 `imshow` 指令視窗開啟灰階影像

```
>> w=imread('wombats.tif');
```

```
>> figure, imshow(w), pixval on
```



圖 2.1 執行 `pixval on` 指令的袋熊影像

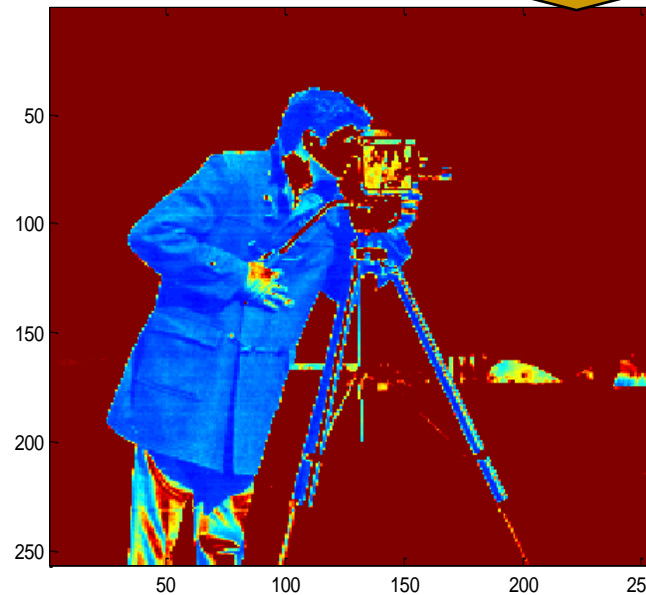
■ 另一個顯示影像的基本函數為image

```
>> c=imread('cameraman.tif');  
>> image(c)
```

因為image指令使用內定的色譜(64色)




原圖




有座標軸
axis

```
>> c=imread('cameraman.tif');
```

```
>> image(c)
```



```
>> image(c), truesize, axis off, colormap(gray(247))
```



```
>> size(unique(c))
```

```
ans =
```

```
247    1
```

```
>> image(c),true size,axis off, colormap(gray(247))
```

```
colormap(gray(247))
```



```
colormap(gray(512))
```



太暗

```
colormap(gray(128))
```



偏亮

- `image` 函數可顯示索引色影像，但需讀入色譜

```
>> [x, map] = imread('cat.tif');  
>> image(x), truesize, axis off, colormap(map)
```

- 對全彩影像，`image` 函數則會忽略設定的色譜

```
>> t = imread('twins.tif');  
>> image(t), truesize, axis off
```



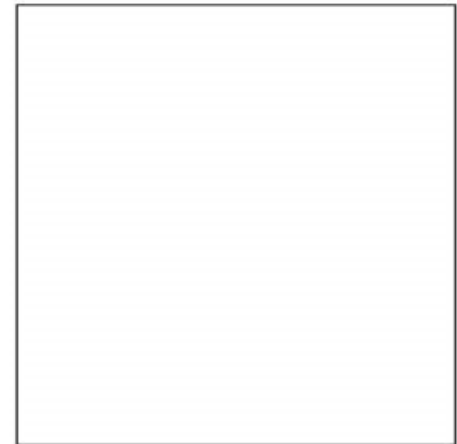
3.3 imshow 函數

- Imshow 函數可正確顯示 uint8 型態的影像，但對於 double 型態則會出錯

```
>> c=imread('caribou.tif');  
>> whos c  


| Name | Size    | Bytes | Class       |
|------|---------|-------|-------------|
| c    | 256x256 | 65536 | uint8 array |

  
Grand total is 65536 elements using 65536 bytes  
  
>> cd=double(c);  
>> imshow(c),figure, imshow(cd)
```



➔ 解決方式: `imshow(cd/255)`
或 `cd=im2double(c); imshow(cd)`


```
>> imshow(cd/512)
>> imshow(cd/128)
```

太暗



(a)



偏亮

(b)

圖 3.2 將影像矩陣除以一個常數 (a) 用 512 除 cd 矩陣
(b) 用 128 除 cd 矩陣

■ 產生二元黑白影像

```
>> c=imread('caribou.tif');  
>> c1=c>120;  
>> whos c1;
```

Name	Size	Bytes	Class
c1	256x256	65536	logical array

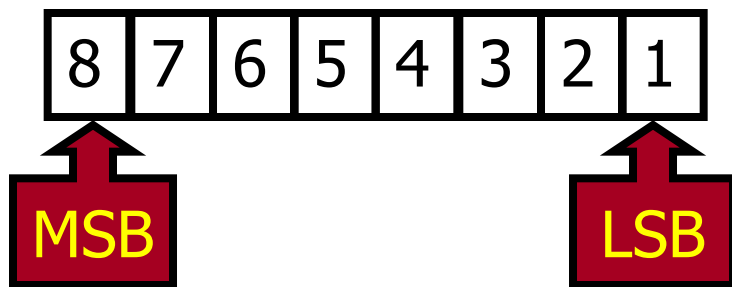
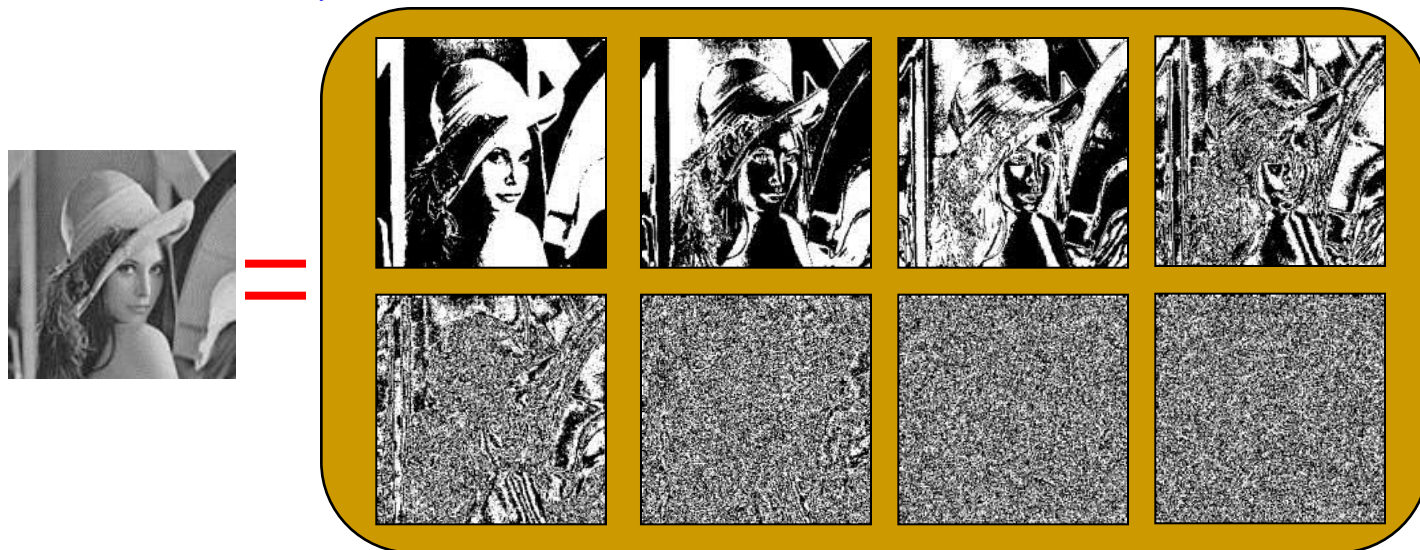
```
Grand total is 65536 elements using 65536 bytes
```

```
>> imshow(c1)
```



3.4 位元平面

- 灰階影像可看成8張位元平面的組合



$$255 = (1111\ 1111)_2$$
$$100 = (0110\ 0100)_2$$

■ 範例:

```
>> c=imread('cameraman.tif');  
>> cd=double(c);
```

```
>> c0=mod(cd,2);  
>> c1=mod(floor(cd/2),2);  
>> c2=mod(floor(cd/4),2);  
>> c3=mod(floor(cd/8),2);  
>> c4=mod(floor(cd/16),2);  
>> c5=mod(floor(cd/32),2);  
>> c6=mod(floor(cd/64),2);  
>> c7=mod(floor(cd/128),2);
```

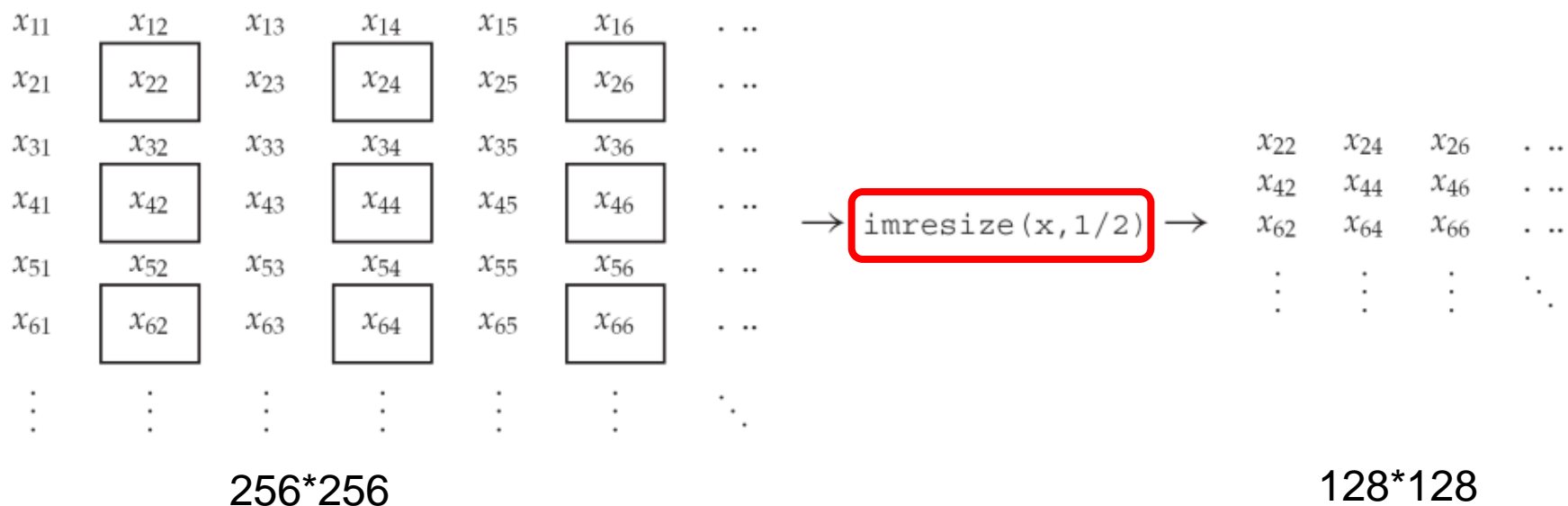


C7等於影像經過閾值127
的運算結果:

```
>> ct = c>127;  
>> imshow(ct)
```

3.5 空間解析度

- 空間解析度是指影像像素的密度；解析度越高，用來顯示影像的像素越多。
- 這裡將利用 `imresize` 函數來實驗解析度的影響

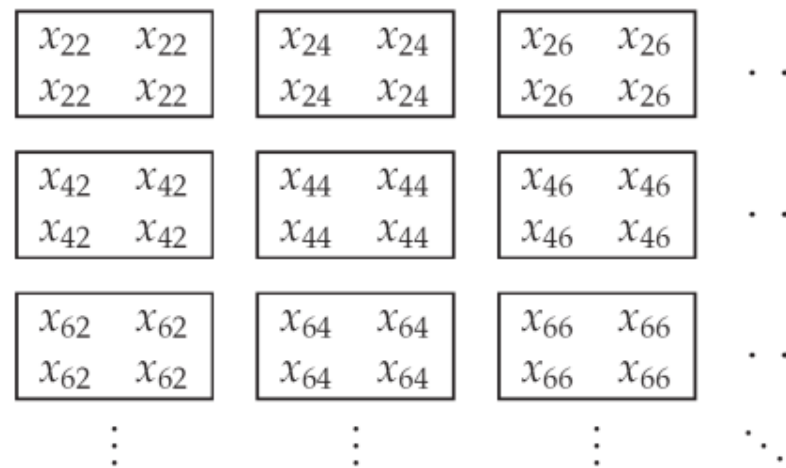


■ 範例:

- 先將圖縮小再放大，讓新影像解低度只有 128×128 。

```
>> x=imread('newborn.tif');
```

```
>> x2=imresize(imresize(x,1/2),2);
```



256*256

指令	有效解析度
<code>imresize(imresize(x,1/4),4);</code>	64×64
<code>imresize(imresize(x,1/8),8);</code>	32×32
<code>imresize(imresize(x,1/16),16);</code>	16×16
<code>imresize(imresize(x,1/32),32);</code>	8×8



原始影像



128×128 解析度影像



64×64 解析度影像



32×32 解析度影像



16×16 解析度影像



8×8 解析度影像

3.6 量化與混色

- **量化** (Quantization): 用來顯示影像的灰階數目
 - 均勻量化 (uniform) 將灰階範圍分割成 n 個平均範圍

原始數值	輸出數值
0 ~ 63	0
64 ~ 127	1
128 ~ 191	2
192 ~ 255	3

```
x=imread('newborn.tif');  
f=floor(double(x)/64);  
q=uint8(f*64);
```

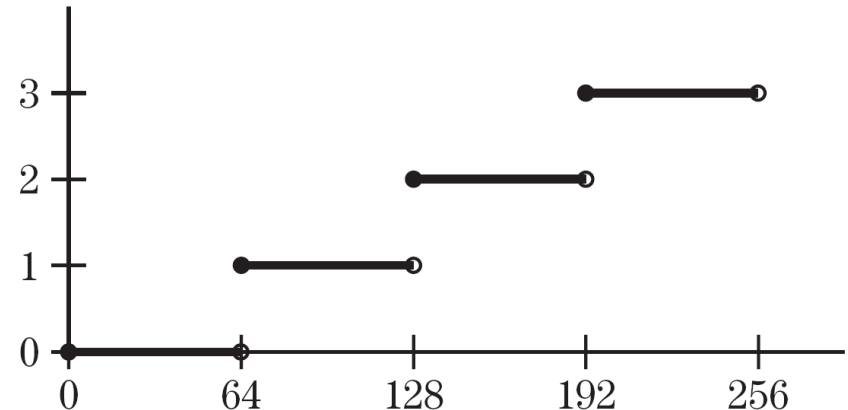


圖 3.8 均勻量化的映射圖

- 或，使用 **grayslice** 函數

```
>> q=grayslice(x,4);  
>>imshow(q, gray(4))
```

指令	灰階數
<code>imshow(grayslice(x,128),gray(128))</code>	128
<code>imshow(grayslice(x,64),gray(64))</code>	64
<code>imshow(grayslice(x,32),gray(32))</code>	32
<code>imshow(grayslice(x,16),gray(16))</code>	16
<code>imshow(grayslice(x,8),gray(8))</code>	8
<code>imshow(grayslice(x,4),gray(4))</code>	4
<code>imshow(grayslice(x,2),gray(2))</code>	2



量化為 128 灰階



量化為 64 灰階



量化為 32 灰階



量化為 16 灰階



量化為 8 灰階



量化為 4 灰階



量化為 2 灰階

- **混色** (dithering)：也就是**減少影像色彩**的數目
 - 其中，將影像用兩種色調表示的方法稱為**半色調** (halftoning)



灰階



二元



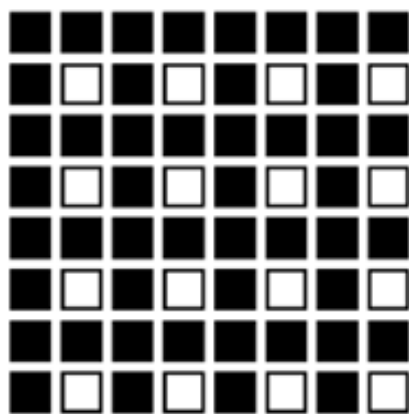
□ 利用人眼對高頻的不敏銳性

- 網點越密，灰度值越低
- 網點越疏，灰度值越高

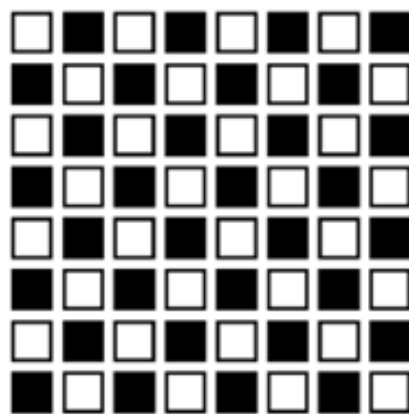
可解決量化後的輪廓
失真現象



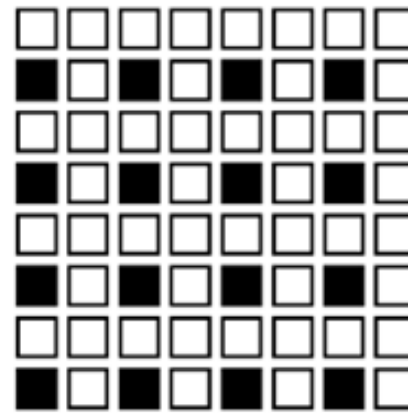
圖 3.13 混色的圖樣



深灰



中灰



淺灰

- 作法:使用 混色矩陣，加入亂數

$$D = \begin{bmatrix} 0 & 128 \\ 192 & 64 \end{bmatrix} \quad D_2 = \begin{bmatrix} 0 & 128 & 32 & 160 \\ 192 & 64 & 224 & 96 \\ 48 & 176 & 16 & 144 \\ 240 & 112 & 208 & 80 \end{bmatrix}$$

$d(i, j)$ 是重複 D 後的矩陣，則輸出像素 $p(i, j)$ 為：

$$p(i, j) = \begin{cases} 1 & \text{若 } x(i, j) > d(i, j) \\ 0 & \text{若 } x(i, j) \leq d(i, j) \end{cases}$$

```
>> x=imread('newborn.tif');  
>> D=[0 128;192 64]  
>> r= repmat(D,128,128);  
>> x2=x>r;imshow(x2)  
>> D2=[0 128 32 160;192 64 224 96;48 176 16 144;240 112 208 80];  
>> r2=repmat(D2,64,64);  
>> x4=x>r2;imshow(x4)
```




(a)



(b)

圖 3.14 混色法範例 (a) 使用 D 進行混色法的新生兒影像
(b) 使用 $D2$ 進行混色法的新生兒影像

```
>> D = [0 56; 84 28];
>> r = repmat(D, 128, 128);
>> x = double(x);
>> q = floor(x/85);
>> x4 = q+(x-85*q>r);
>> imshow(uint8(85*x4))
```

```
>> D = [0 24; 36 12];
>> r = repmat(D, 128, 128);
>> x = double(x);
>> q = floor(x/37);
>> x8 = q+(x-37*q>r);
>> imshow(uint8(37*x8))
```

混色成
四色調



(a)

混色成
八色調



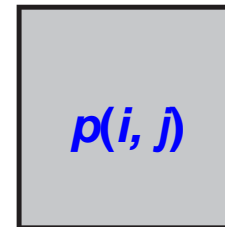
(b)

圖 3.15 2 灰階以上的混色法 (a) 輸出 4 個灰階影像的混色法
(b) 輸出 8 個灰階影像的混色法

■ 誤差擴散法 (error diffusion)

- 由Floyd 和Steinberg 發展，將影像**量化**成兩個層次
- 計算每個像素 $p(i, j)$ 的灰階值及其量化值的**誤差**，並將誤差分散到鄰近的像素

$$E = \begin{cases} p(i, j) & \text{若 } p(i, j) < 128 \\ p(i, j) - 255 & \text{若 } p(i, j) \geq 128 \end{cases}$$



$$+ \frac{7}{16}E$$

$$+ \frac{3}{16}E$$

$$+ \frac{5}{16}E$$

$$+ \frac{1}{16}E$$


```
function y = fl_stein(x)
```

```
%
% FL_STEIN applies Floyd-Steinberg error diffusion to an image x, which is
% assumed to be of type uint8.
%
height=size(x,1);
width=size(x,2);
y=uint8(zeros(height,width));
z=zeros(height+2,width+2);
z(2:height+1,2:width+1)=x;
for i=2:height+1,
    for j=2:width+1,
        if z(i,j) < 128
            y(i-1,j-1) = 0;
            e = z(i,j);
        else
            y(i-1,j-1) = 255;
            e = z(i,j)-255;
        end
        z(i,j+1)=z(i,j+1)+7*e/16;
        z(i+1,j-1)=z(i+1,j-1)+3*e/16;
        z(i+1,j)=z(i+1,j)+5*e/16;
        z(i+1,j+1)=z(i+1,j+1)+e/16;
    end
end
```

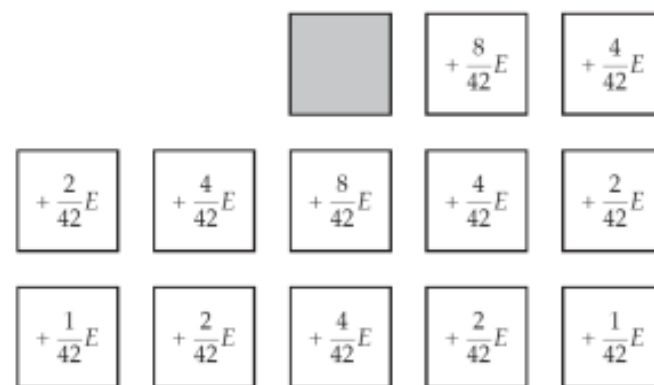
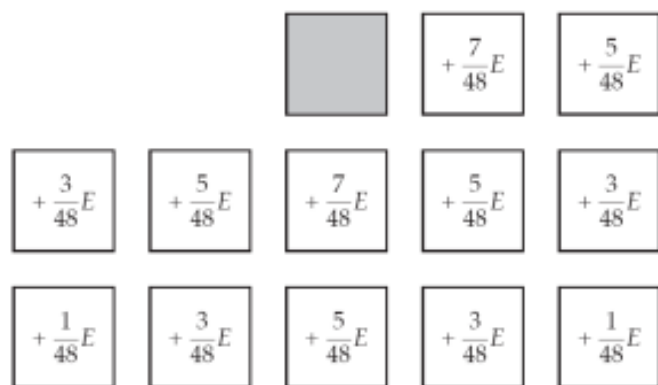
Z

(1,1)	(1,2)	(1,3)	(1,4)	(1,w+1)	(1,w+2)
(2,1)	(1,1)	(1,2)	(1,3)	(1,w)	
(3,1)	(2,1)				
(4,1)	(3,1)				
(h+1,1)	(h,1)			X	
(h+2,1)					

圖 3.16 MATLAB 函數將 Floyd-Steinberg 誤差擴散法應用到灰階影像



圖 3.17 進行 Floyd-Steinberg 誤差擴散法之後的新生兒影像



(a)



(b)

圖 3.18 使用其他誤差擴散結構 (a) Jarvis-Judice-Ninke 誤差擴散法結果
(b) Stucki 誤差擴散法結果

練習

- 將彩色影像(cat.tiff)轉變成灰階、二元影像，使用均勻量化技術量化成2灰階影像、使用混色技術(D矩陣)產生2灰階的影像、使用誤差擴散法產生2灰階的影像，比較以上4張二元影像，請問哪個視覺效果較佳呢？