

# Digital Image Processing

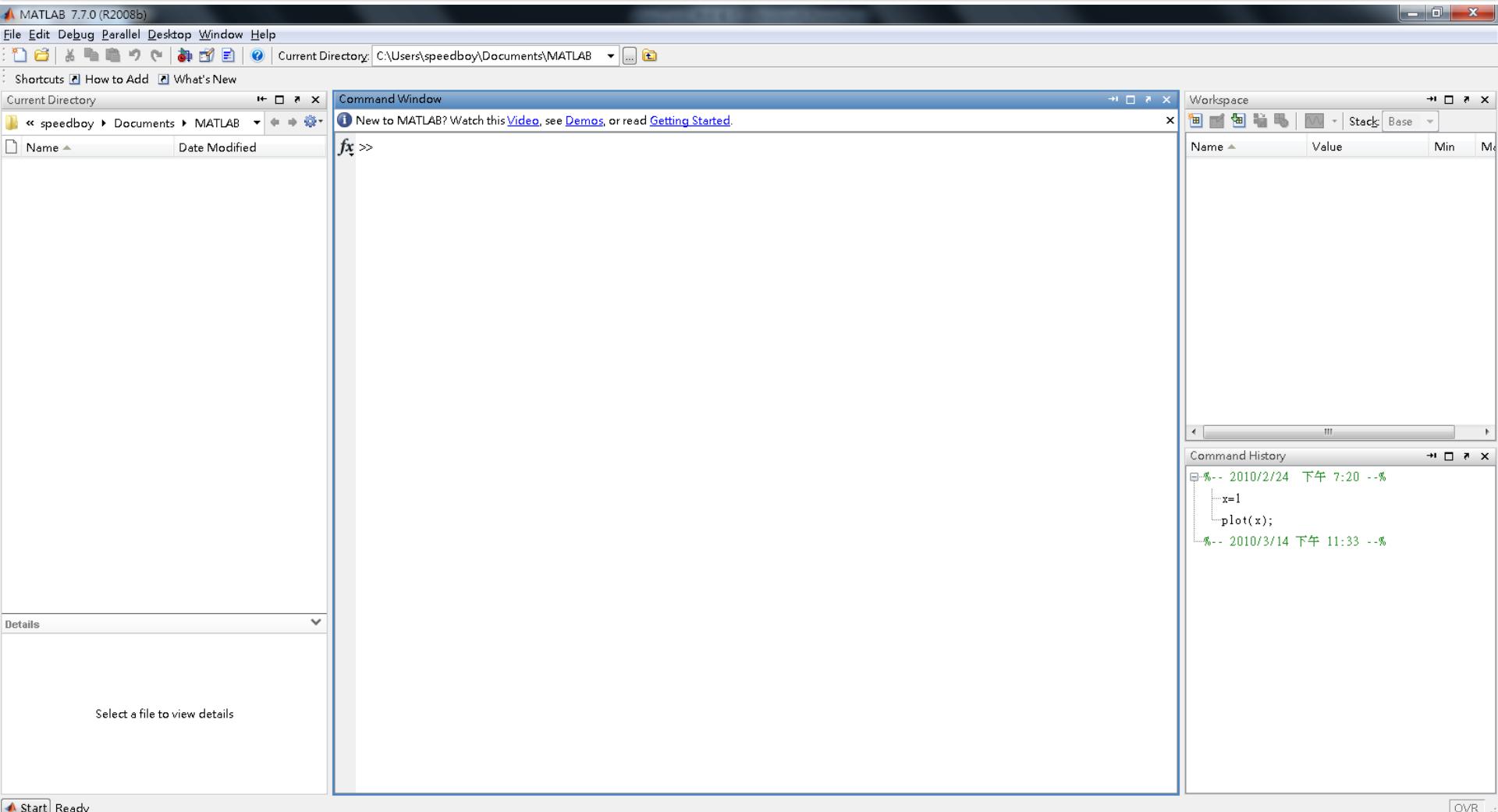
## Matlab

Ming-Chin Chuang

# Outline

- Matlab之基本介紹
- Matlab之繪圖
- Matlab之影像處理
  - 影像讀/存檔
  - 影像濾波器
  - 影像加強
  - 影像移動模糊及還原
  - 影像的擴張與侵蝕

# Matlab畫面



# Matlab畫面

The screenshot shows the MATLAB Editor window with the file 'image\_process.m' open. The code is a script for creating a GUI. It defines a state structure, handles command-line arguments, and sets up an opening function for the GUI.

```
Editor - C:\Users\speedboy\Desktop\文件\美圖\數位影像處理(莊明睿)\03-15\0chp9\image_process.m
File Edit Text Go Cell Tools Debug Desktop Window Help
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0 + ÷ 11 × ✎ fx
Stack Base
1
28 - gui_Singleton = 1;
29 - gui_State = struct('gui_Name',         filename, ...
30 -                      'gui_Singleton',   gui_Singleton, ...
31 -                      'gui_OpeningFcn', @image_process_OpeningFcn, ...
32 -                      'gui_OutputFcn',  @image_process_OutputFcn, ...
33 -                      'gui_LayoutFcn', [], ...
34 -                      'gui_Callback', []);
35 - if nargin && ischar(varargin{1})
36 -     gui_State.gui_Callback = str2func(varargin{1});
37 - end
38
39 - if nargout
40 -     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 - else
42 -     gui_mainfcn(gui_State, varargin{:});
43 - end
44 % End initialization code - DO NOT EDIT
45
46
47 % --- Executes just before image_process is made visible.
48 function image_process_OpeningFcn(hObject, eventdata, handles, varargin)
49 % This function has no output args, see OutputFcn.
50 % hObject    handle to figure
51 % eventdata   reserved - to be defined in a future version of MATLAB
52 % handles    structure with handles and user data (see GUIDATA)
53 % varargin   command line arguments to image_process (see VARARGIN)
54
55 % Choose default command line output for image_process
56 handles.output = hObject;
57
58 % Update handles structure
59 guidata(hObject, handles);
60
61 % UIWAIT makes image_process wait for user response (see UIRESUME)
62 % uiwait(handles.figure1);
63
```

# MATLAB簡介

- 由MathWorks公司於1984年推出的數學軟體。
- 名稱是由「矩陣實驗室」（MATrix LABoratory）所合成。
- MATLAB為各種動態系統模擬、數位訊號處理、科學計算、科學目視等領域的標準程式語言。
- MATLAB 的許多的核心計算技術是源自於 LINPACK 及 EISPACK 。

# MATLAB簡介

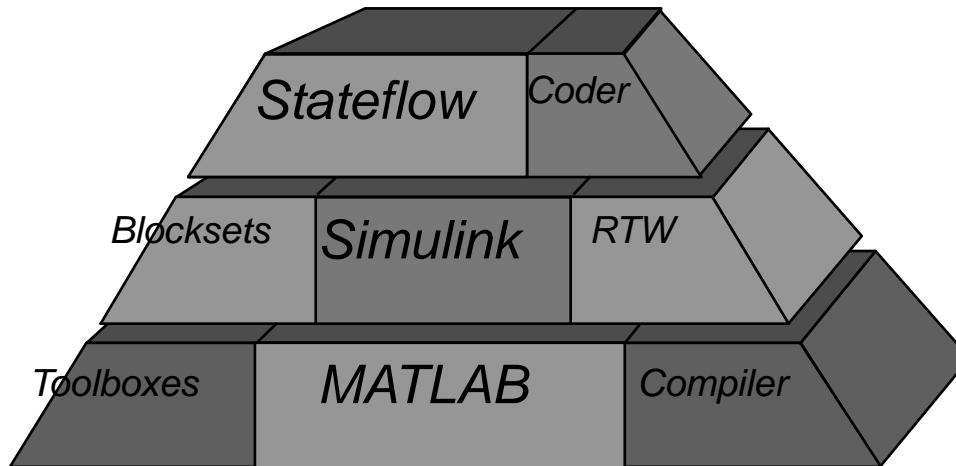
- MATLAB早在 1978 年即已現身，是用 **Fortran** 撰寫的免費軟體，其作者是當時任教於新墨西哥大學的 Cleve Moler 教授。
- Jack Little（又稱為 John Little）將 MATLAB 以 **C** 語言重寫，並於 1984 年成立 MathWorks 公司，首次推出 MATLAB 商用版。
- MathWorks 在 Newsgroup 上進行對使用者的技術指導，在 WWW 興起之後，就提供各項技術支援與搜尋功能，並在內聯網（Intranet）方面，以 Web 與資料庫的整合來進行軟體 bug 的追蹤、修復與管理。

# Simulink 及 Stateflow

- Simulink 專用於連續或離散時間的動態系統模擬。Simulink 是一個模擬核心，圍繞著這個核心所開發的應用程式稱為方塊集（Blocksets）。
- Stateflow 則用於模擬有限狀態機（Finite State Machines）或事件驅動系統（Event-driven Systems）。

# MATLAB、Simulink 及 Stateflow

- MATLAB、Simulink 及 Stateflow三者的關係：



- 由現有 Simulink 與 Stateflow 的 C 程式碼自動產生功能，以  
及定點運算方塊集（Fixed-point Blockset）與 C 程式碼至  
VHDL 的自動轉換功能，可看出「高階的系統模擬」或「低  
階的晶片演算法設計」，都可用  
MATLAB/Simulink/Stateflow 及相關的工具箱來達成。

# 使用變數與基本運算

## ■ 一般數學符號運算

- 在 MATLAB 命令視窗（Command Window）內的提示符號（ $>>$ ）之後輸入運算式，並按入 Enter 鍵即可。例如：

```
>> (5*2+3.5)/5
```

```
ans =
```

```
2.7000
```

- 若不想讓 MATLAB 每次都顯示運算結果，只需在運算式最後加上分號（;）即可，例如：

```
>> (5*2+3.5)/5;
```

# 變數命名規則與使用

- 第一個字母必需是英文字母。
- 字母間不可留空格。
- 最多只能有 31 個字母，MATLAB 會忽略多餘字母（在 MATLAB 第 4 版，則是 19 個字母）。
- MATLAB 在使用變數時，不需預先經過變數宣告（Variable Declaration）的程序，而且所有數值變數均以預設的 double 資料型式儲存。

# 加入註解

- 若要加入註解（Comments），可以使用百分比符號（%）例如：

```
>> y = (5*2+3.5)/5;      % 將運算結果儲存在變數 y，但不用顯示於螢幕  
>> z = y^2                % 將運算結果儲存在變數 z，並顯示於螢幕  
z =  
    7.2900
```

# Variables, Vectors, Matrices, and Arrays

1. Variable: 一個單獨的元素(整數、實數、複數)。
2. 向量(Vector): 一維的矩陣，可分為 Column Vectors 及 Row Vectors。
3. 矩陣(Matrices): 二維的陣列，一維向量為其特例
4. 陣列(Arrays): Using array as an inclusive term to designate a vector or a matrix.

# 向量與矩陣的處理

- MATLAB 中的變數還可用來儲存向量（Vectors）及矩陣（Matrix），以進行各種運算，例如：

```
>> s = [1 3 5 2];% 注意[]的使用，及各數字間的空白間隔
```

```
>> t = 2*s+1
```

```
t =
```

```
3 7 11 5
```

# 矩陣的各種處理

- MATLAB 亦可取出向量中的一個元素或一部份來做運算，例如：

```
>> t(3) = 2 % 將向量 t 的第三個元素更改為 2
```

```
t =
```

```
3 7 2 5
```

```
>> t(6) = 10 % 在向量 t 加入第六個元素，其值為 10
```

```
t =
```

```
3 7 2 5 0 10
```

```
>> t(4) = [] % 將向量 t 的第四個元素刪除，[] 代表空集合
```

```
t =
```

```
3 7 2 0 10
```

# 建立大小為 $m \times n$ 的矩陣

- 在每一橫列結尾加上分號 (;) ，例如：

```
>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12]; % 建立 3x4 的矩陣 A
```

```
>> A      % 顯示矩陣 A 的內容
```

```
A =
```

```
1  2  3  4  
5  6  7  8  
9  10 11 12
```

# **mxn矩陣的各種處理之一**

- $\text{>> } A(2,3) = 5$  % 將矩陣 A 第二列、第三行的元素值，改變為 5

A =

```
1  2  3  4
5  6  5  8
9  10 11 12
```

- $\text{>> } B = A(2,1:3)$  % 取出矩陣 A 的第二橫列、第一至第三直行，並儲存成矩陣 B

B =

```
5  6  5
```

# **mxn矩陣的各種處理之二**

- `>> A = [A B'] % 將矩陣 B 轉置後、再以行向量併入矩陣 A`

A =

```
1 2 3 4 5  
5 6 5 8 6  
9 10 11 12 5
```

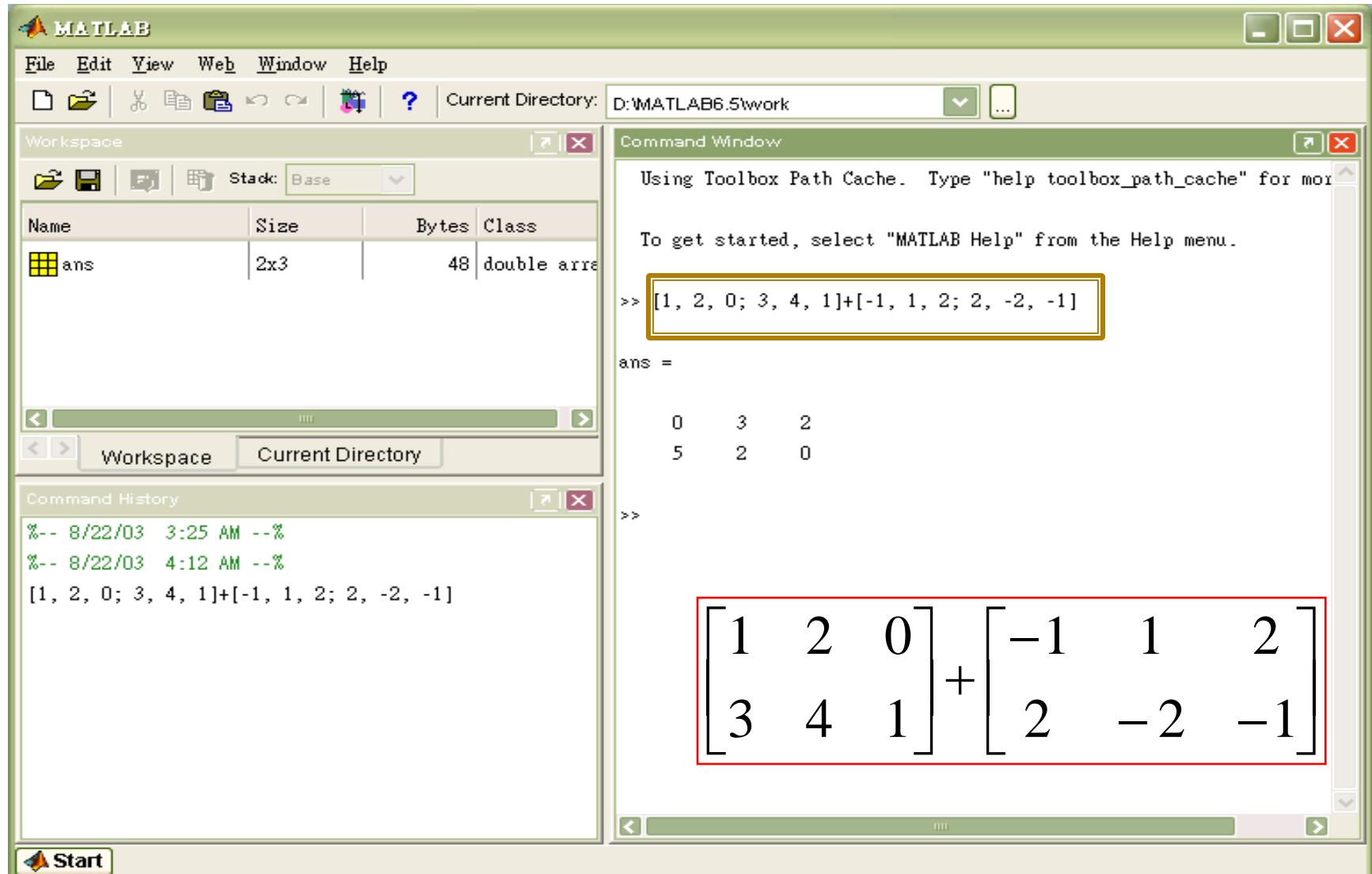
- `>> A(:, 2) = [] % 刪除矩陣 A 第二行 ( : 代表所有橫列 , [] 代表空矩陣 )`

A =

```
1 3 4 5  
5 5 8 6  
9 11 12 5
```

# Array Operation

## - Sum(+) and Difference(-)



# **mxn矩陣的各種處理之三**

- `>> A = [A; 4 3 2 1] % 在原矩陣 A 中，加入第四列`

`A =`

```
1 3 4 5
5 5 8 6
9 11 12 5
4 3 2 1
```

- `>> A([1 4], :) = [] % 刪除第一、四列（：代表所有直行，[]是空矩陣）`

`A =`

```
5 5 8 6
```

# 常用數學函數

- MATLAB 是一個科學計算軟體，因此可以支援很多常用到的數學函數
  - `>> y = abs(x) % 取 x 的絕對值`
  - `>> y = sin(x) % 取 x 的正弦值`
  - `>> y = exp(x) % 自然指數 exp(x)`
  - `>> y = log(x) % 自然對數 ln(x)`
- MATLAB 也支援複數運算，通常以 i 或 j 代表單位虛數

# 向量矩陣的運算

- 有一些函數是特別針對向量而設計
  - `>> y = min(x)` % 向量 x 的極小值
  - `>> y = max(x)` % 向量 x 的極大值
  - `>> y = mean(x)` % 向量 x 的平均值
  - `>> y = sum(x)` % 向量 x 的總和
  - `>> y = sort(x)` % 向量 x 的排序

# 線上支援

- `help`：用來查詢已知指令的用法。
- `lookfor`：用來尋找未知的指令。找到所需的指令後，即可用 `help` 進一步找出其用法。
- `helpwin` 或 `helpdesk`：產生線上支援視窗，其效果和直接點選 MATLAB 命令視窗工作列的圖示是一樣的。
- `doc`：產生特定函數的線上支援。

# 程式流程控制

- MATLAB 提供重複迴圈（Loops）及條件判斷（Conditions）等程式流程控制（Flow Control）的指令
  - `for` 迴圈

```
For 變數 = 向量  
    運算式;  
end
```

# 流程控制

- while 迴圈 (While-loop)

while 條件式

運算式;

end

- if – else – end

if 條件式

運算式;

else

運算式;

end

# M 檔案

- 若要一次執行大量的 MATLAB 指令，可將這些指令存放於一個副檔名為 m 的檔案，並在 MATLAB 指令提示號下鍵入此檔案的主檔名即可。

# 搜尋路徑

- 若要檢視 MATLAB 已設定的搜尋路徑，鍵入 path 指令即可：  
`>> path`
- 若要查詢某一特定指令所在的搜尋路徑，可用 which 指令
- 要將目錄加入 MATLAB 的搜尋路徑，可使用 addpath 指令

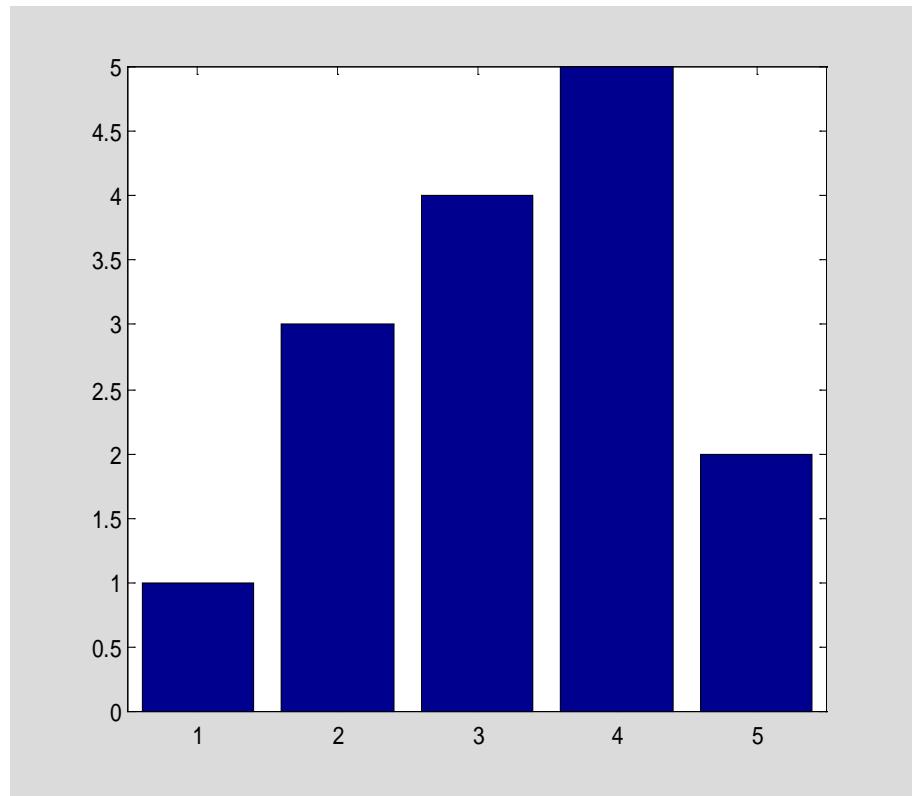
# 離開 MATLAB

- 在命令視窗內，鍵入 `exit` 指令。
- 在命令視窗內，鍵入 `quit` 指令。
- 直接關閉 MATLAB 的命令視窗。

# 長條圖之繪製

- 長條圖 (Bar Graphs)  
特別適用於少量且離散的資料。欲畫出垂直長條圖，可用 bar 指令。

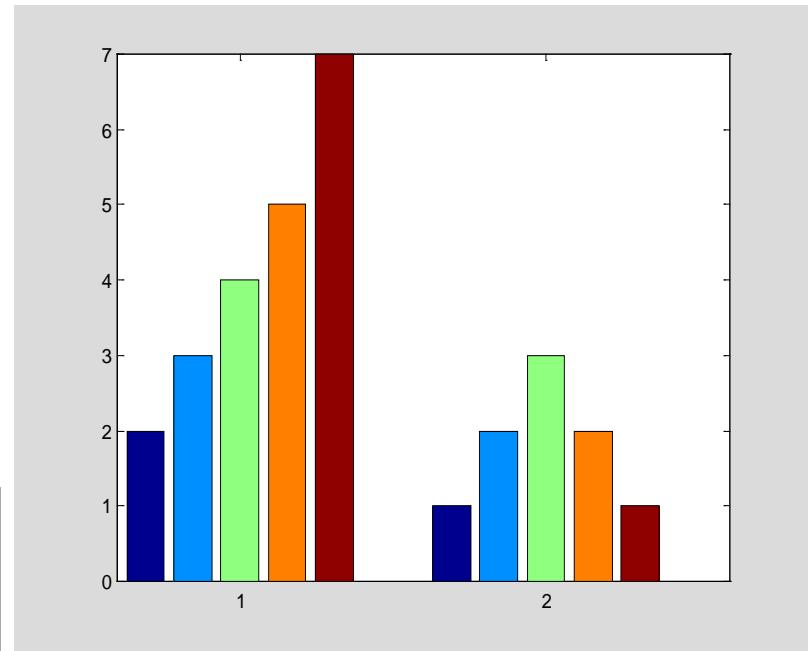
```
x = [1 3 4 5 2];  
bar(x);
```



# 長條圖之繪製(cont.)

- bar 指令也可接受矩陣輸入  
它會將同一橫列的資料聚集在一起。

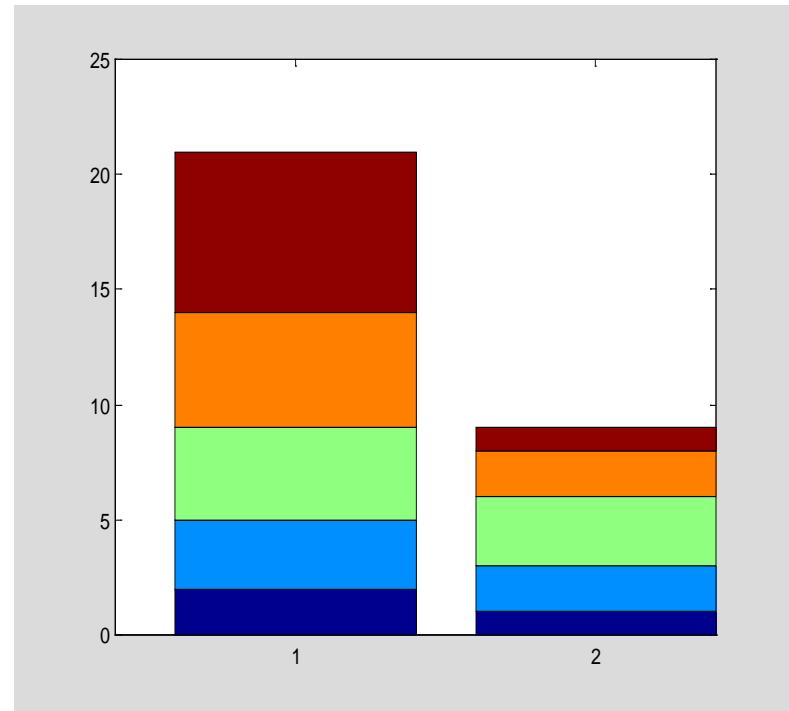
```
x = [2 3 4 5 7; 1 2 3 2 1];  
bar(x);
```



# 長條圖之繪製(cont.)

- bar 及 barh 指令還有一項特異功能，就是可以將同一橫列的資料以堆疊（Stack）方式來顯示。

```
x = [2 3 4 5 7; 1 2 3 2 1];  
bar(x,'stack')
```



# 長條圖之繪製(cont.)

- 除了平面長條圖之外，MATLAB 亦可使用 `bar3` 指令來畫出立體長條圖。

```
x = [2 3 4 5 7; 1 2 3 2 1];  
bar3(x)
```

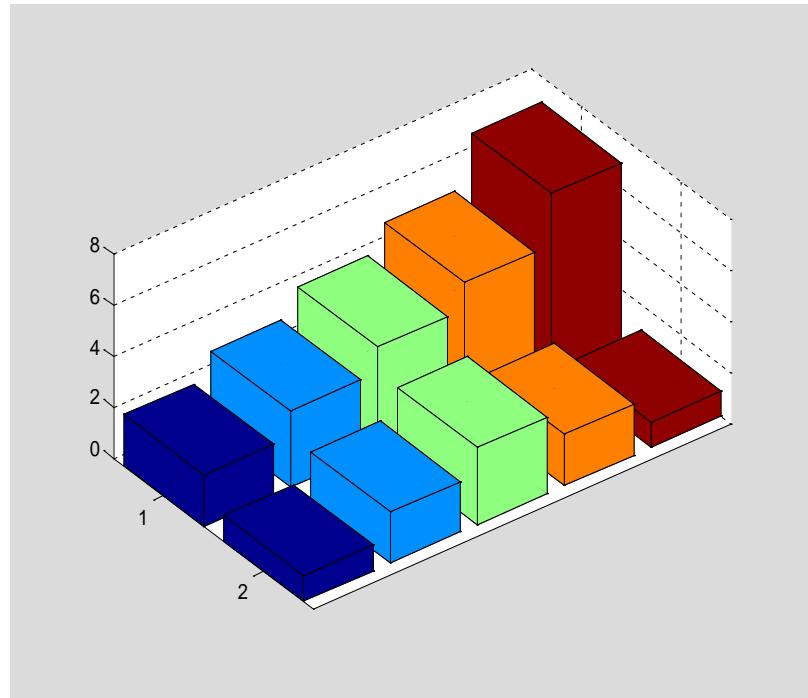


Fig. 5-4

# 長條圖之繪製(cont.)

- `bar3` 指令還可以使用群組 (Group) 方式來呈現長條圖

```
x = [2 3 4 5 7; 1 2 3 2 1];  
bar3(x, 'group')
```

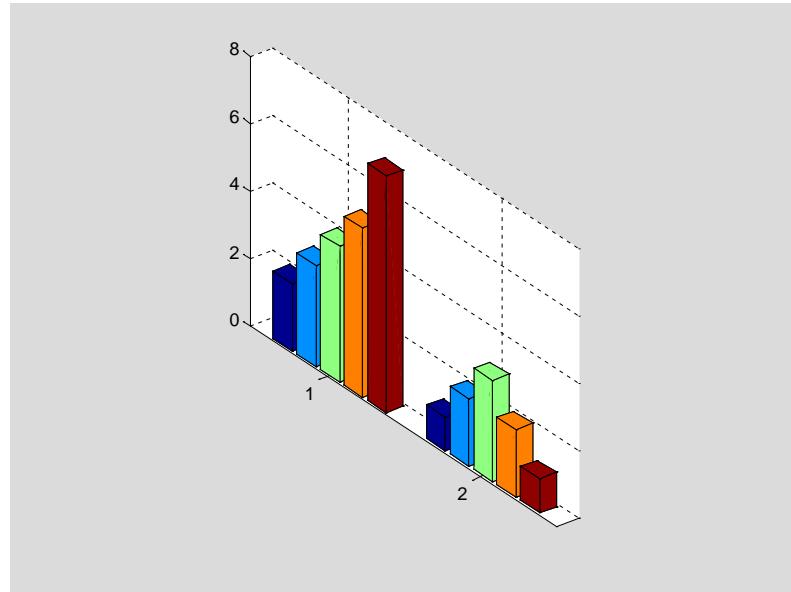


Fig. 5-5

# 長條圖之繪製(cont.)

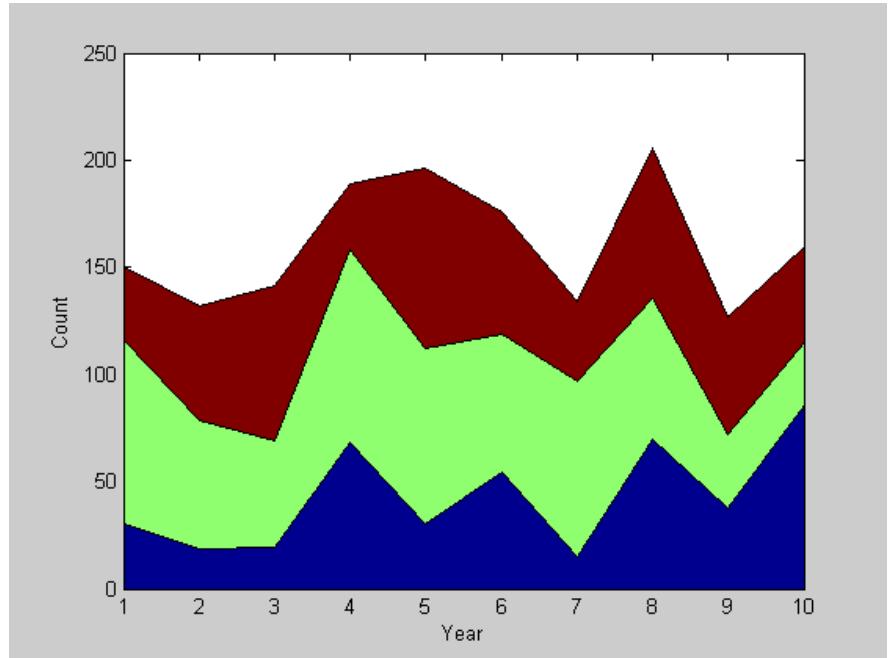
- 長條圖的指令和類別：

	垂直長條圖	水平長條圖
平面	bar	barh
立體	bar3	bar3h

# 面積圖之繪製

- 面積圖（Area Graphs）和以堆疊方式呈現的長條圖很類似，特別適用於具有疊加關係的資料。舉例來說，若要顯示清華大學在過去 10 年來的人數（含大學部，研究生，及教職員）變化情況，可用面積圖顯示。

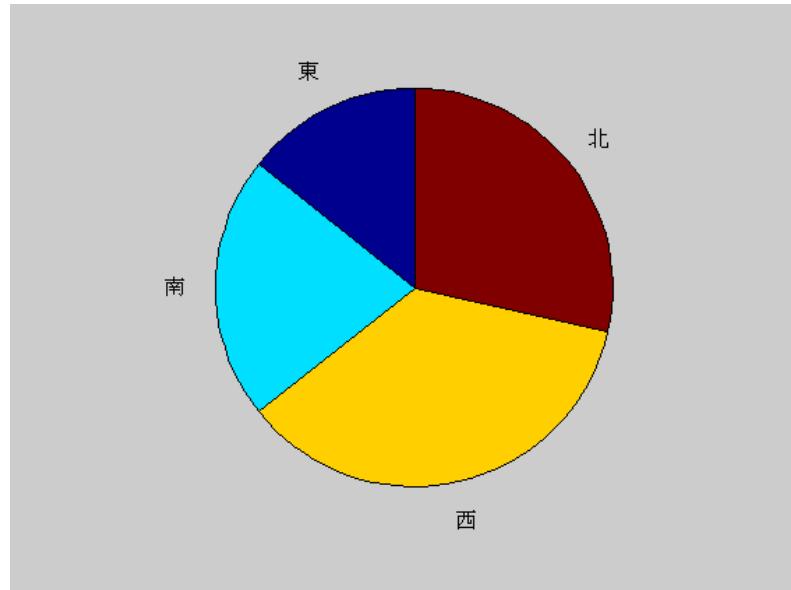
```
y = rand(10,3)*100;  
x = 1:10;  
area(x, y);  
xlabel('Year');  
ylabel('Count')
```



# 扇形圖之繪製

- 使用 pie 指令，可畫出平面扇形圖（Pie Charts），並可加上說明。

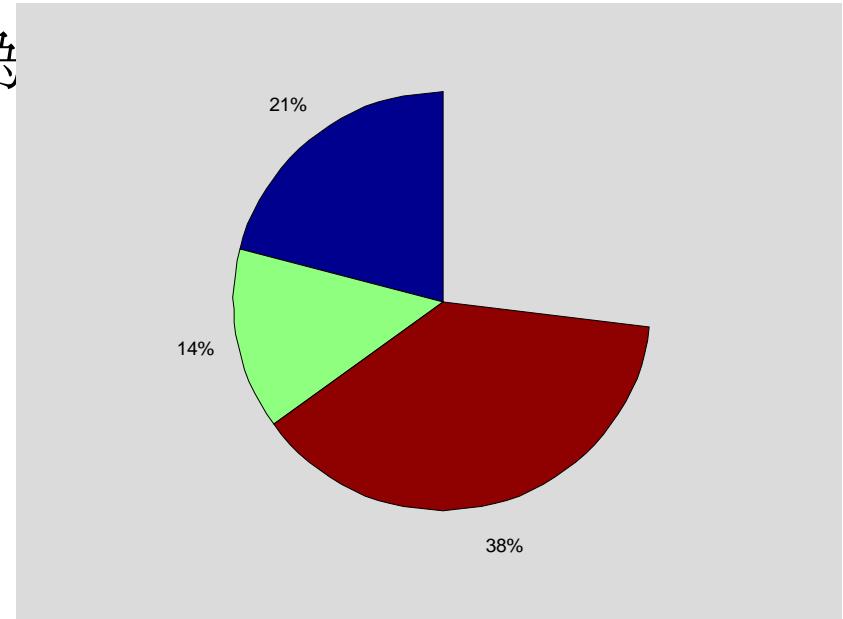
```
x = [2 3 5 4];  
label={'東','南','西','北'};  
pie(x, label);
```



# 扇形圖之繪製(cont.)

- `pie` 指令直接將 `x` 元素視為面積百分比，因此可畫出不完全的扇形圖。

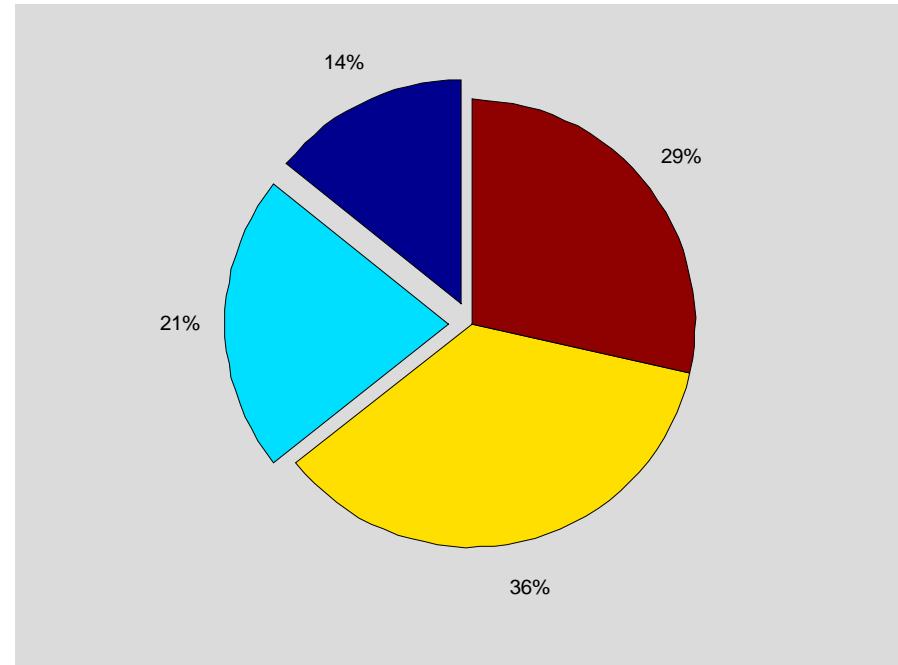
```
x = [0.21, 0.14, 0.38];  
pie(x);
```



# 扇形圖之繪製(cont.)

- pie 指令還有一特異功能，可將某個或數個扇形圖向外拖出，以強調部份資料。

```
x = [2 3 5 4];  
explode = [1 1 0 0];  
pie(x, explode);
```

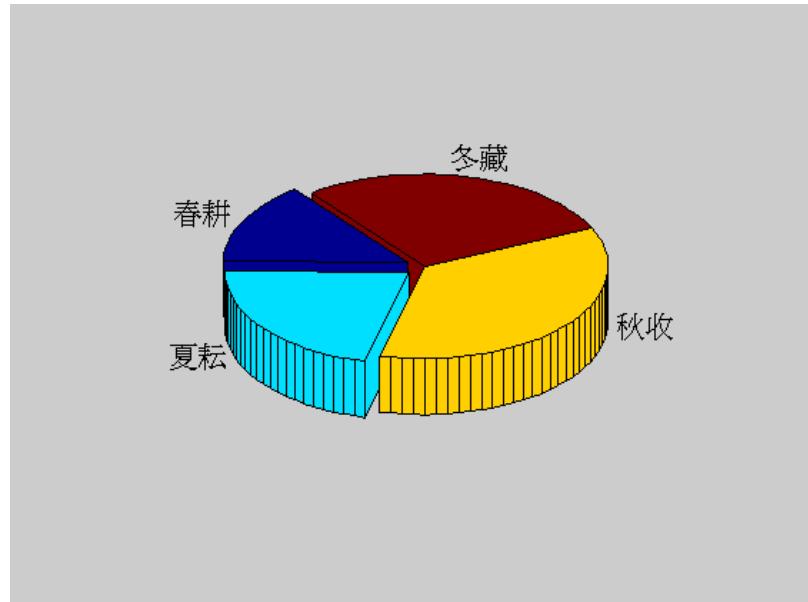


其中指令 explode 中非零的元素即代表要向外拖出的扇形。

# 扇形圖之繪製(cont.)

- 欲畫出立體扇形圖，可用 pie3 指令。

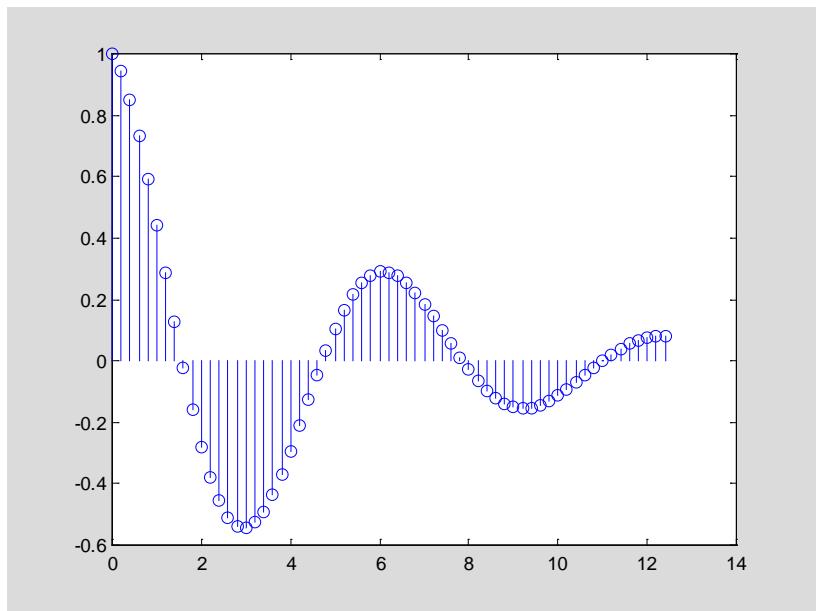
```
x = [2 3 5 4];  
explode = [1 1 0 0];  
label = {'春','夏','秋','冬'};  
pie3(x, explode, label);
```



# 針頭圖之繪製

- 顧名思義，針頭圖（*Stem Plots*）就是以一個大頭針來表示某一點資料，其指令為 `stem`。

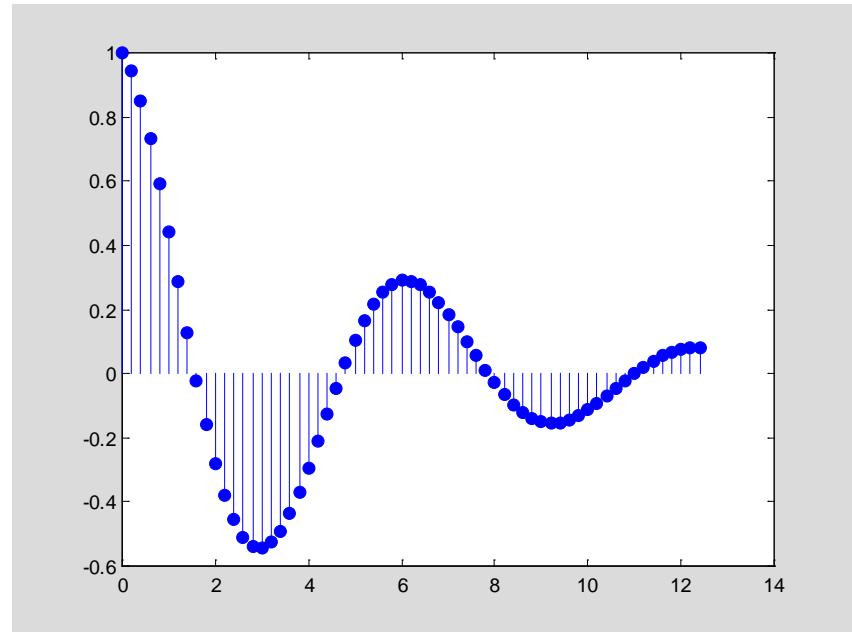
```
t = 0:0.2:4*pi;  
y = cos(t).*exp(-t/5);  
stem(t, y)
```



# 針頭圖之繪製(cont.)

- 針頭圖特別適用於表示「數位訊號處理」（DSP，Digital Signal Processing）中的數位訊號。若要畫出實心的針頭圖，可加 “fill”選項。

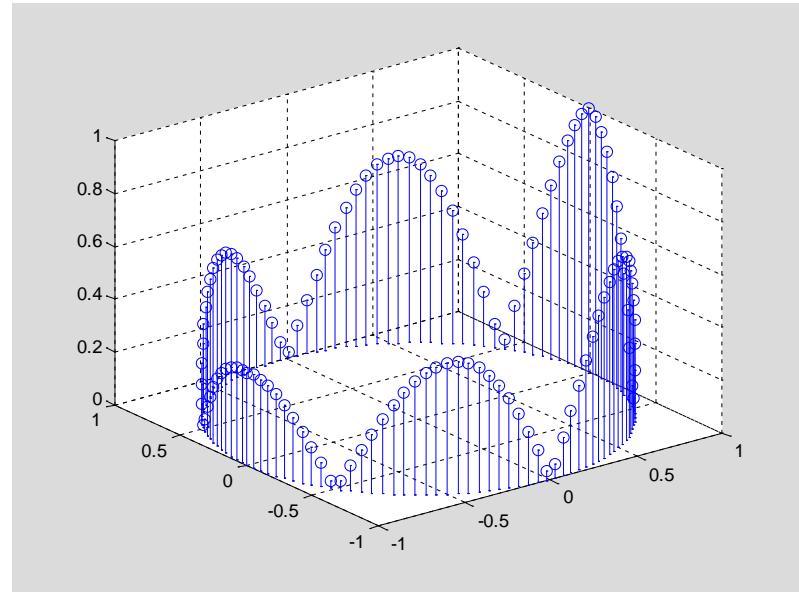
```
t = 0:0.2:4*pi;  
y = cos(t).*exp(-t/5);  
stem(t, y, 'fill');
```



# 針頭圖之繪製(cont.)

- 欲畫出立體的針頭圖，可用 `stem3` 指令。

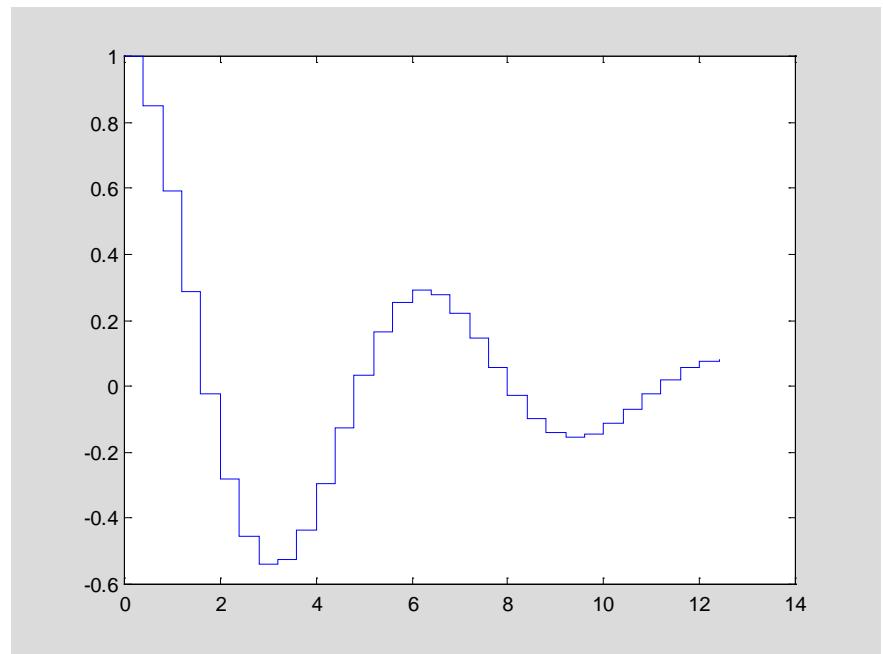
```
theta = -pi:0.05:pi;  
x = cos(theta);  
y = sin(theta);  
z = abs(cos(3*theta)).*exp(-abs(theta/3));  
stem3(x, y, z);
```



# 階梯圖之繪製

- 使用 `stairs` 指令，可畫出階梯圖（Stairstep Plots），其精神和針頭圖很相近，只是將目前資料點的高度向右水平畫至下一點為止。（在數位訊號處理，此種作法稱為 Zero-order Hold。）

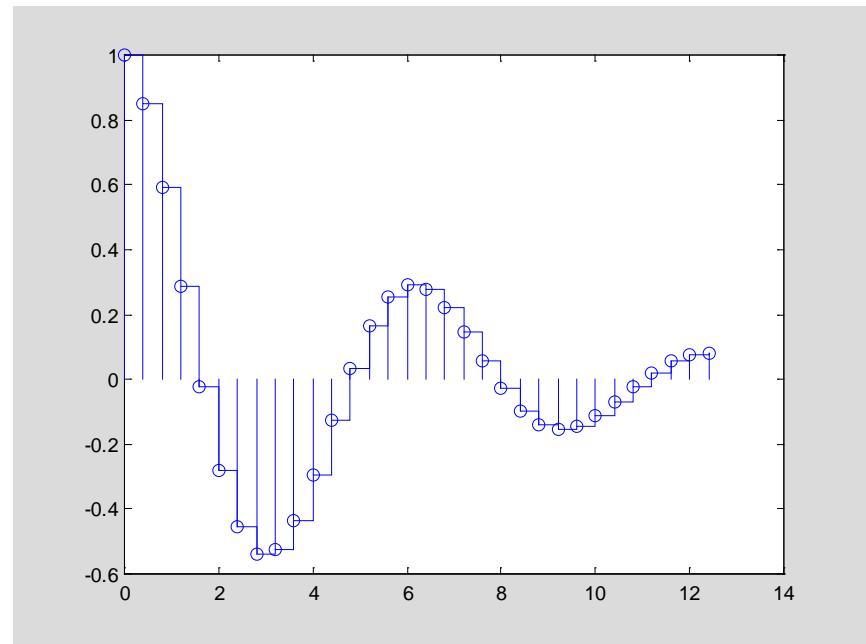
```
t = 0:0.4:4*pi;  
y = cos(t).*exp(-t/5);  
stairs(t, y);
```



# 階梯圖之繪製(cont.)

- 若再加上針頭圖，則可見兩者相似之處。

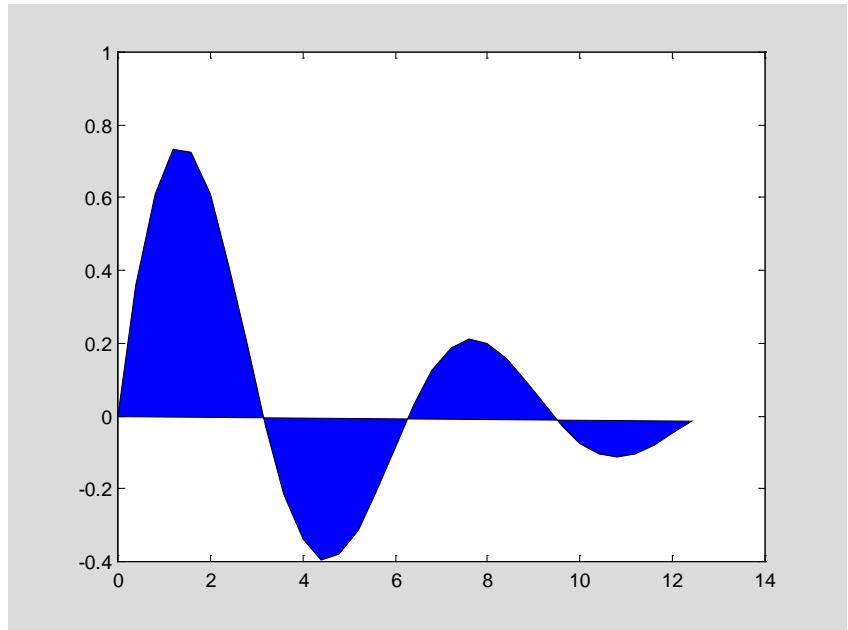
```
t = 0:0.4:4*pi;  
y = cos(t).*exp(-t/5);  
stairs(t, y);  
hold on % 保留舊圖形  
stem(t, y); % 覆上針頭圖  
hold off
```



# 實心圖之繪製

- MATLAB 指令 `fill` 將資料點視為多邊形頂點，並將此多邊形塗上顏色，呈現出實心圖（Filled Plots）的結果。

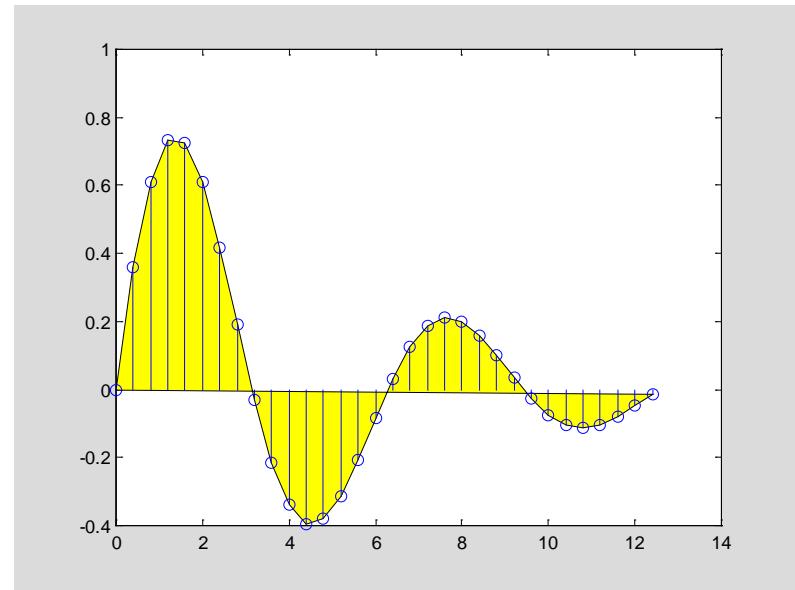
```
t = 0:0.4:4*pi;  
y = sin(t).*exp(-t/5);  
fill(t, y, 'b'); % 'b'為藍色
```



# 實心圖之繪製(cont.)

- 若與 stem 合用，則可創造出一些不同的視覺效果。

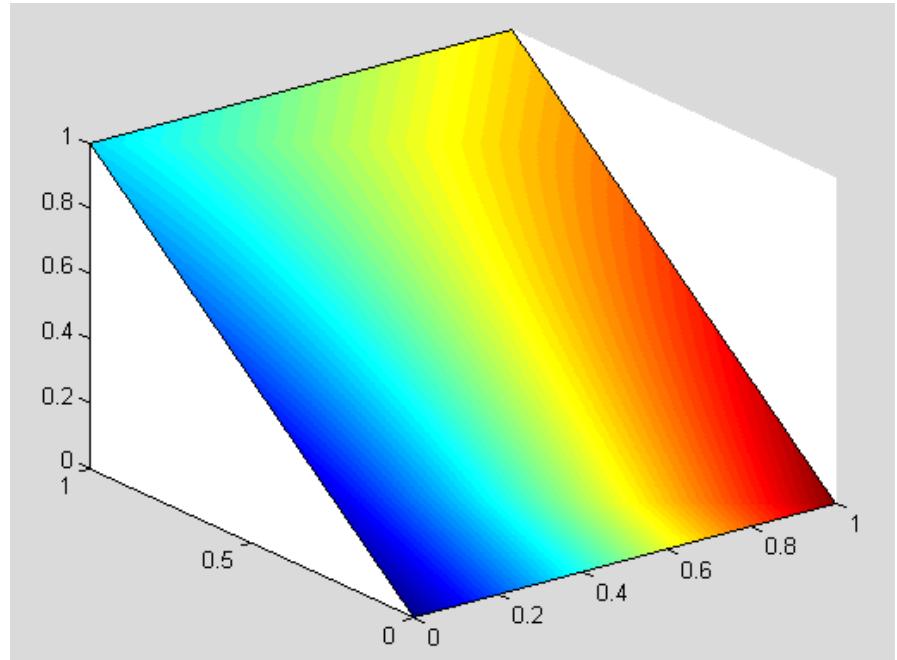
```
t = 0:0.4:4*pi;  
y = sin(t).*exp(-t/5);  
fill(t, y, 'y');          % 'y' 為黃色  
hold on                  % 保留舊圖形  
stem(t, y, 'b');         % 疊上藍色針頭圖  
hold off
```



# 實心圖之繪製(cont.)

- `fill3` 可用於三維的實心圖。

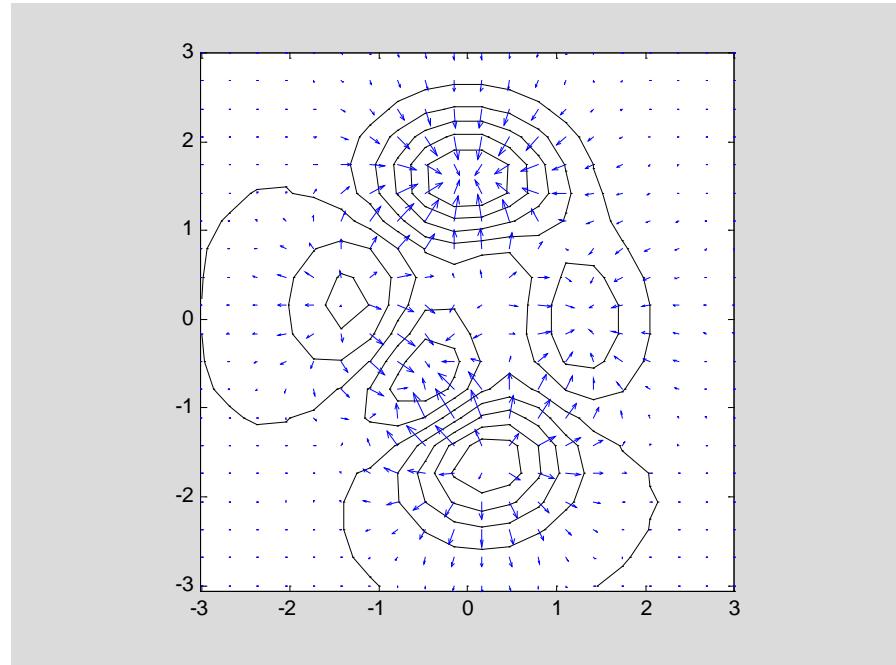
```
X = [0 0 1 1];  
Y = [0 1 1 0];  
Z = [0 1 1 0];  
C = [0 0.3 0.6 0.9]';  
fill3(X, Y, Z, C);
```



# 向量場圖之繪製

- 使用 quiver 指令可畫出平面上的向量場圖（Quiver Plots），特別適用於表示分布於平面的向量場（Vector Fields），例如平面上的電場分布，或是流速分布。

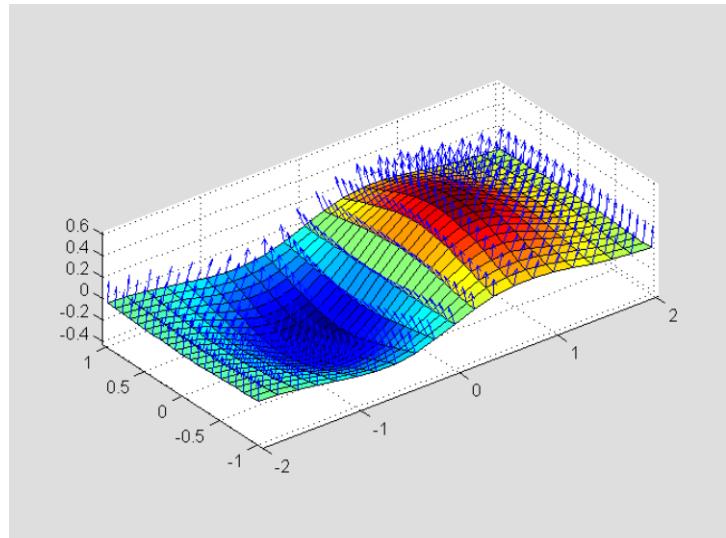
```
[x, y, z] = peaks(20);
[u, v] = gradient(z);
contour(x, y, z, 10);
hold on, quiver(x,y,u,v); hold off
axis image
```



# 向量場圖之繪製(cont.)

- 欲畫出空間中的向量場圖，可用 quiver3 指令。

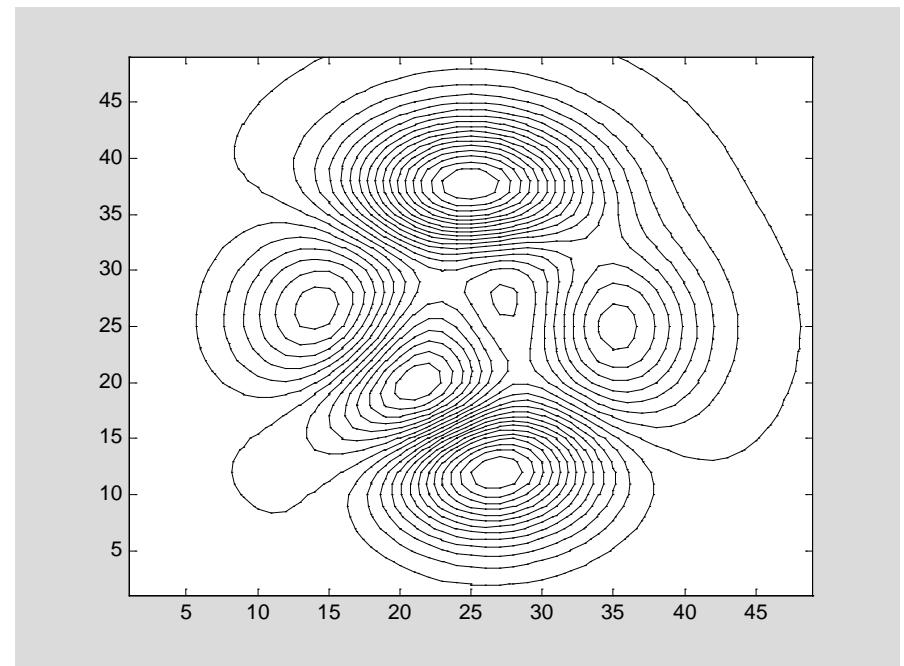
```
[x, y] = meshgrid(-2:0.2:2, -1:0.1:1);
z = x.*exp(-x.^2-y.^2);
[u, v, w] = surfnorm(x, y, z);
quiver3(x, y, z, u, v, w);
hold on, surf(x, y, z); hold off
axis equal
```



# 等高線圖之繪製

- 我們可用 `contour` 指令來畫出「等高線圖」(Contour Plots)。

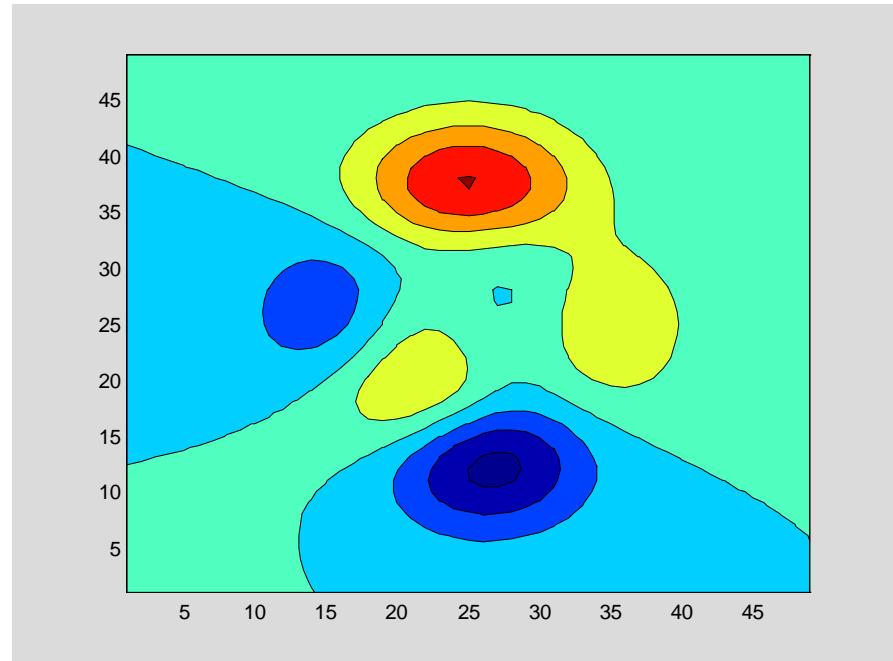
```
z = peaks;  
contour(z, 30);  
% 畫出 30 條等高線
```



# 等高線圖之繪製(cont.)

- 若欲在等高線之間填入顏色，可用 `contourf` 指令。

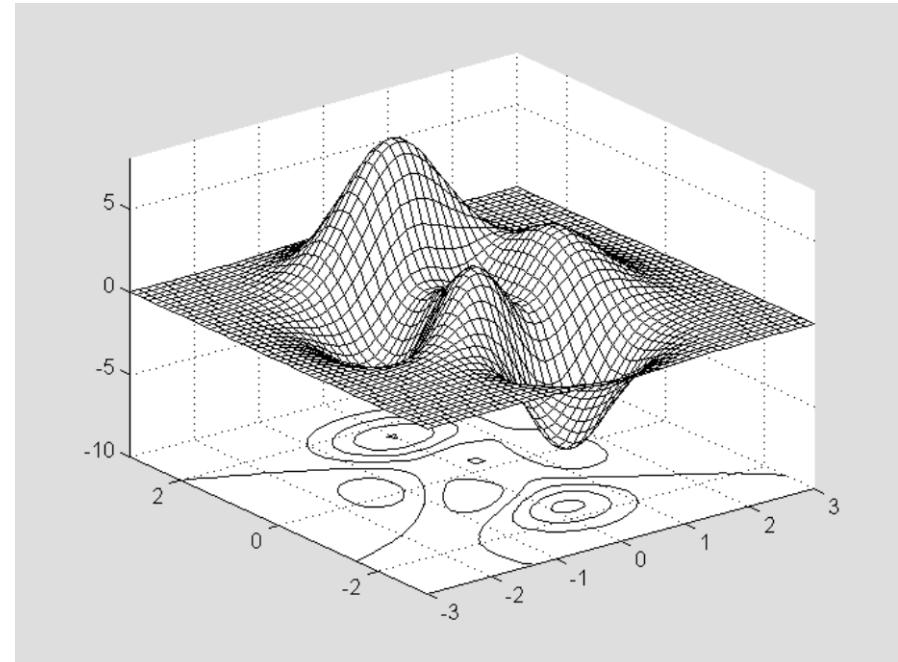
```
z = peaks;  
contourf(z);
```



# 等高線圖之繪製(cont.)

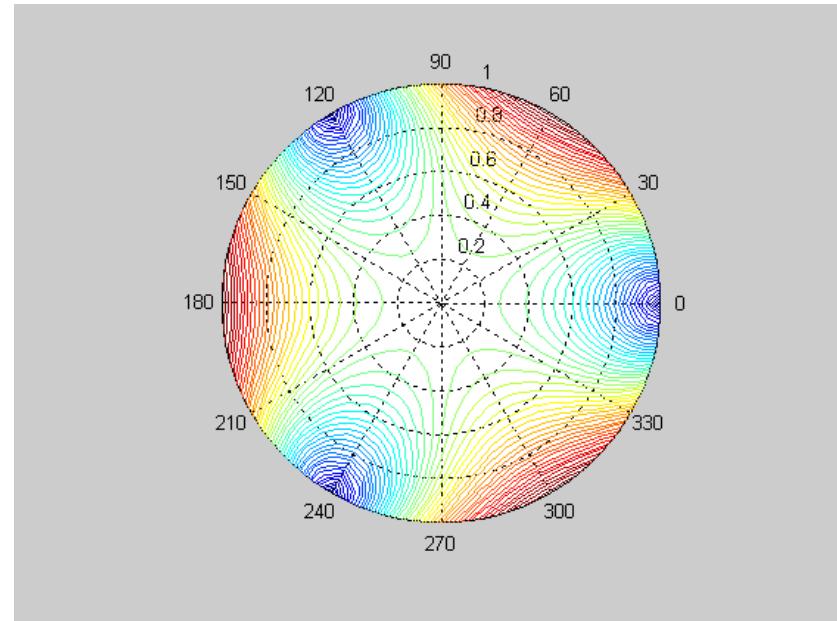
- contourf 亦可接受  $x$ 、 $y$ 、 $z$  輸入引數。若要將等高線畫在曲面的正下方，可用 surf 或 meshc 指令。

```
[x, y, z] = peaks;  
meshc(x, y, z);  
axis tight
```



# 等高線圖之繪製(cont.)

- 在上例中，座標的標示仍為直角座標。欲將等高線顯示於極座標上，需先用 `polar` 指令產生一個極座標圖，再移除圖形，留下圖軸，然後再進行作圖。



```
h = polar([0 2*pi], [0 1]);  
delete(h);  
hold on  
contour(x, y, abs(f), 30);  
hold off
```

% 產生在極座標上的一條直線  
% 移除上述圖形，但留下極座標圖軸

# 等高線圖之繪製(cont.)

特殊繪圖函數：

指令	說明
bar, barh, bar3, bar3h	長條圖
Area	面積圖
pie, pie3	扇形圖
stem, stem3	針頭圖
stairs	階梯圖
fill, fill3	實心圖
quiver, quiver3	向量場圖
contour, contourf, contour3	等高線圖

# 其他進階繪圖功能

- MATLAB 在 5.3 版後，開始支援「容積目視法」（Volume Visualization）、因此能夠畫出在三度空間中的流線圖、向量場圖、等高面圖（Isosurfaces）、切面圖（Slices）等，相關指令可列表如下頁：

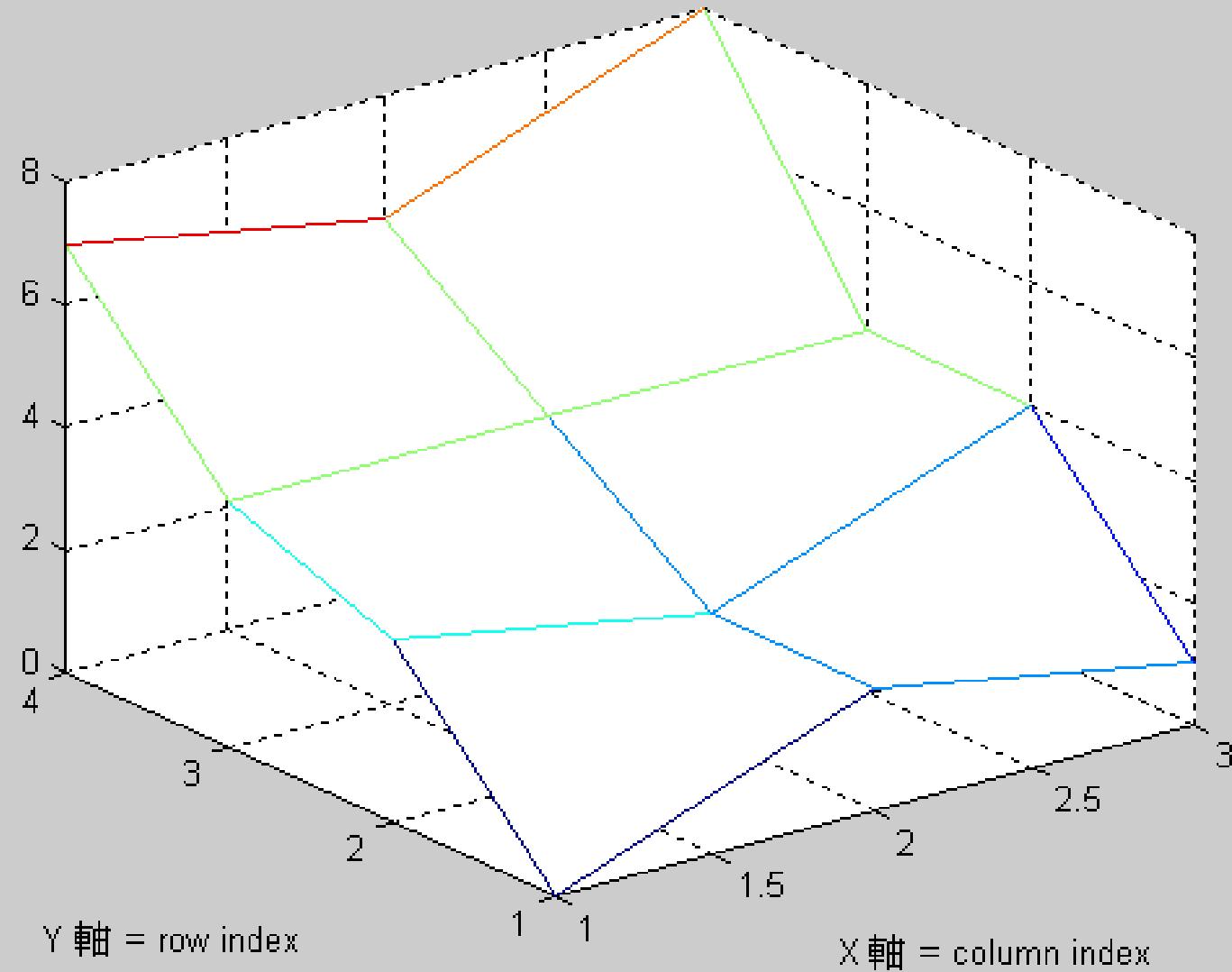
# 基本立體繪圖指令

## ■ mesh 和 surf :

- mesh : 可畫出立體的「網狀圖」 ( Mesh Plots )
- surf : 可畫出立體的「曲面圖」 ( Surface Plots )

```
z = [0 2 1; 3 2 4; 4 4 4; 7 6 8];
mesh(z);
xlabel('X 軸 = column index'); % X 軸的說明文字
ylabel('Y 軸 = row index'); % Y 軸的說明文字
```

# 基本立體繪圖指令



# 其他進階繪圖功能(cont.)

※ MATLAB 有關於「容積目視法」的指令：

指令	說明
coneplot	以圓錐瓶畫出三度空間的向量場圖
contourslice	在三度空間的切面上畫出等高線
isosurface	從容積資料中算出等高面資料
isocaps	計算等高面在端點切片的等高資訊
isonormals	計算等高面的法向量
slice	在三度空間的切片
streamline	從 2-D 或 3-D 的流線資料來畫出流線圖
isocolors	計算等高區面頂點的顏色（第六版才支援）
divergence	計算3-D向量場的亂度（Divergence）（第六版才支援）
curl	計算3-D向量場的curl 及垂直方向的角速度（第六版才支援）

# 影像讀/存檔

- 讀檔
  - `Imread('檔名')`
- 另存新檔
  - `Imwrite('source', 'destination', '影像格式')`
- 顯示
  - `Imshow('影像矩陣')`
- 顯示大小
  - `Truesize([寬 高])`

# MATLAB的影像格式

- MATLAB 最常處理的影像格式為索引影像（Indexed Images）
- 顯示此類型影像的語法如下：

image(X)

colormap(map)

其中X為影像的資料矩陣，map為色盤矩陣。

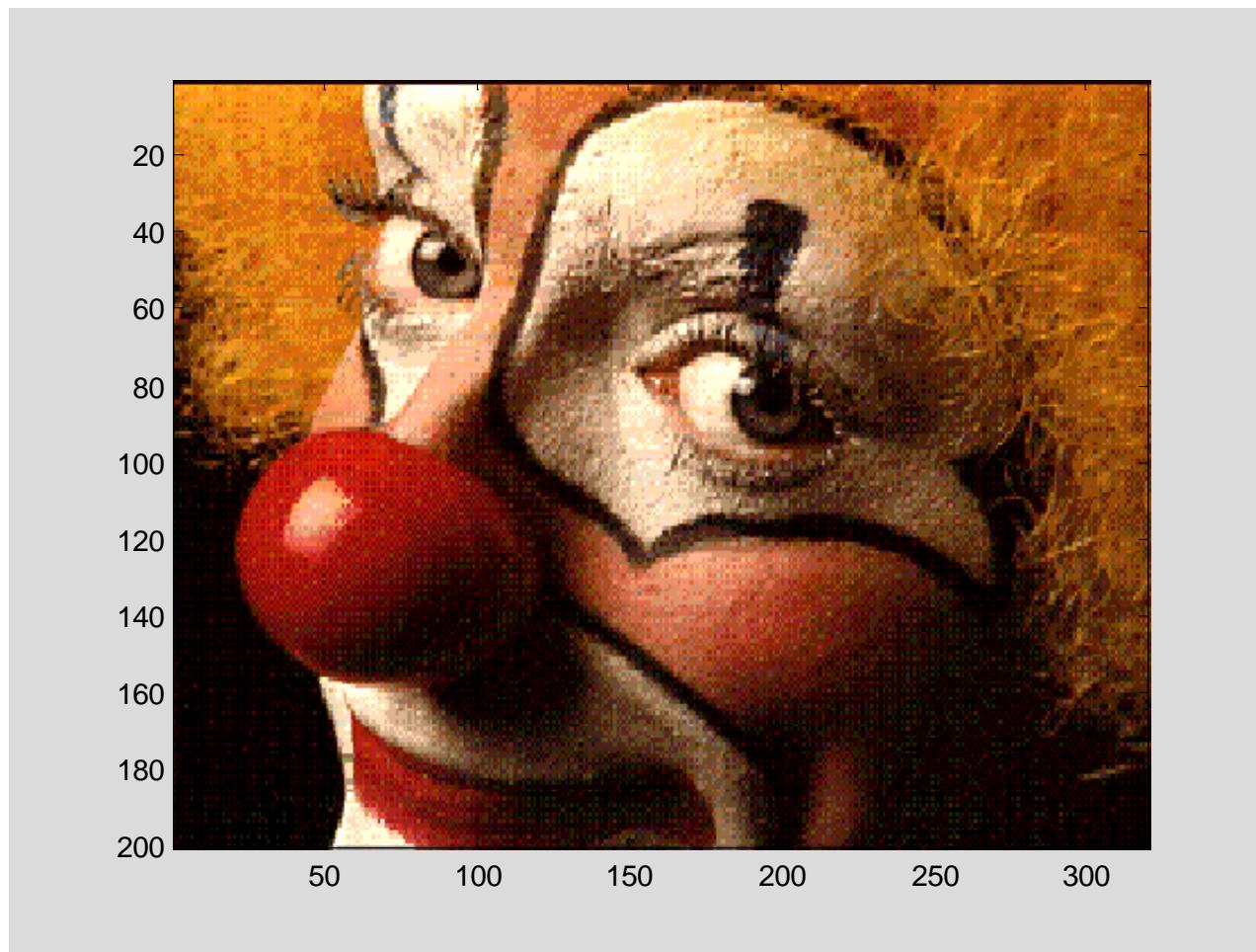
- 色盤矩陣的大小為Kx3，每個橫列由三個元素所組成，分別是R(紅)、G(綠)、B(藍)，每個元素的範圍為0~1
- X的值為1~K，也就是當X(i, j)的值為p，則像素點(i, j)的顏色為map(p, :)這一列的值所決定。

# 顯示索引影像範例一

- 在下例中，我們使用MATLAB顯示內建的小丑圖。

```
load clown.mat          % 載入小丑影像資料，含變數 X 和 map  
image(X);              % 顯示影像  
colormap(map)          % 取用色盤矩陣
```

# 顯示索引影像範例一



# 顯示索引影像範例二

- 由於由  $X$  是索引影像，因此其最小值是 1, 最大值會等於  $map$  的列數（即「可顯示之顏色數目」），可驗証如下：

```
load clown.mat          % 載入小丑影像資料，含變數 X 和 map  
fprintf('min(min(X)) = %d\n', min(min(X)));  
fprintf('max(max(X)) = %d\n', max(max(X)));  
fprintf('size(map, 1) = %d\n', size(map, 1));
```

## 顯示索引影像範例二

$$\min(\min(X)) = 1$$

$$\max(\max(X)) = 81$$

$$\text{size}(map, 1) = 81$$

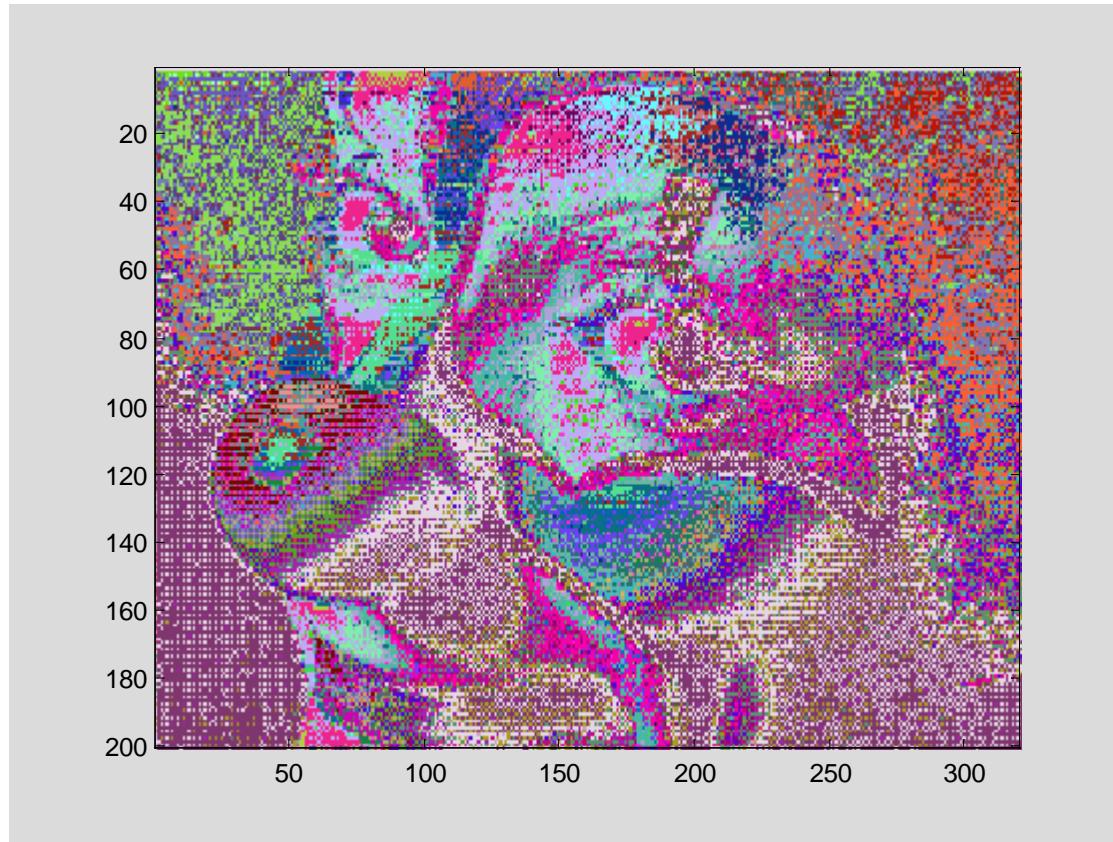
由範例可知，此小丑影像共含有 81 種不同的顏色。

# 顯示索引影像範例三

- 要正確地顯示索引影像則需要正確的色盤，以上面的小丑影像為例，如果使用亂數產生的色盤則會產生下面的結果：

```
load clown.mat          % 載入小丑影像資料，含變數 X 和 map  
newmap = rand(size(map));  
image(X);  
colormap(newmap)
```

# 顯示索引影像範例三



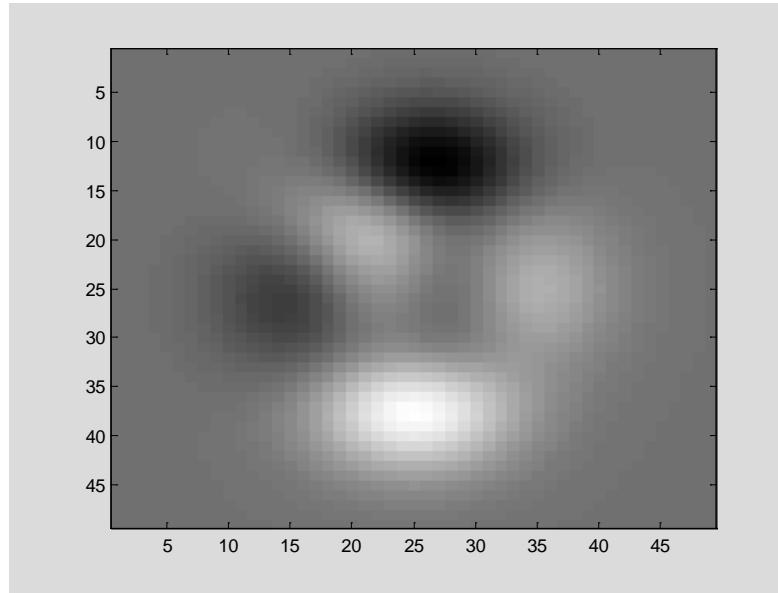
# 顯示索引影像範例四

- 如果我們的色盤矩陣只有  $K$  個橫列，但是  $X$  的某些元素值小於 1 或大於  $K$ ，則我們可以使用 `imagesc` 指令將  $X$  的最小值轉換成 1，最大值轉成  $K$ ，其他中間值則依線性關係轉換成介於 1 與  $K$  的值，舉例如下：

```
X = peaks;  
imagesc(X);  
colormap(gray);  
min(min(X)) % 顯示 X 的最小值  
max(max(X)) % 顯示 X 的最大值
```

# 顯示索引影像範例四

```
ans =  
-6.5466  
ans =  
8.0752
```



- 具有上述特性的影像資料稱為強度影像（Intensity Images），一般經由數值運算產生的矩陣均屬此類，因此均可由 `imagesc` 來顯示。

# 顯示索引影像範例五

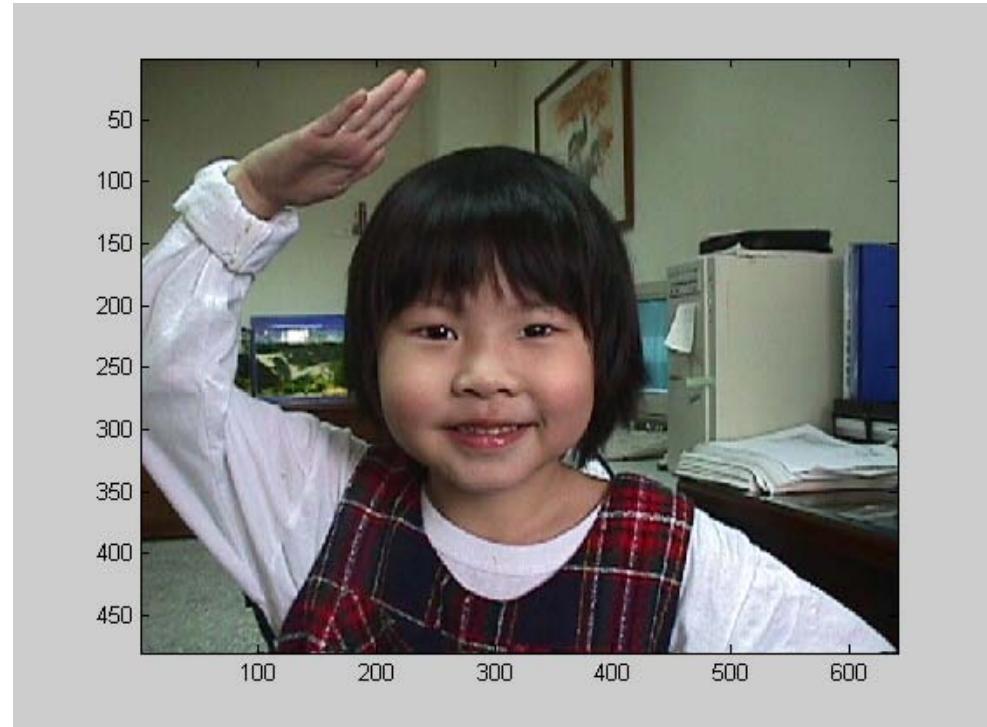
- `image` 指令亦接受全彩影像（Truecolor Images）。全彩影像可以表示成一個  $m \times n \times 3$  的矩陣  $X$ ，其中  $X(:, :, 1)$  代表紅色的強度。 $X(:, :, 2)$  代表綠色的強度， $X(:, :, 3)$  則代表藍色的強度。
- $X$ 的值的範圍可以是下列兩種：介於0~1的浮點數或是0~255的uint8，舉例來說：

```
X = imread('annie19980405.jpg');
image(X)
size(X)
```

# 顯示索引影像範例五

ans =

480 640 3

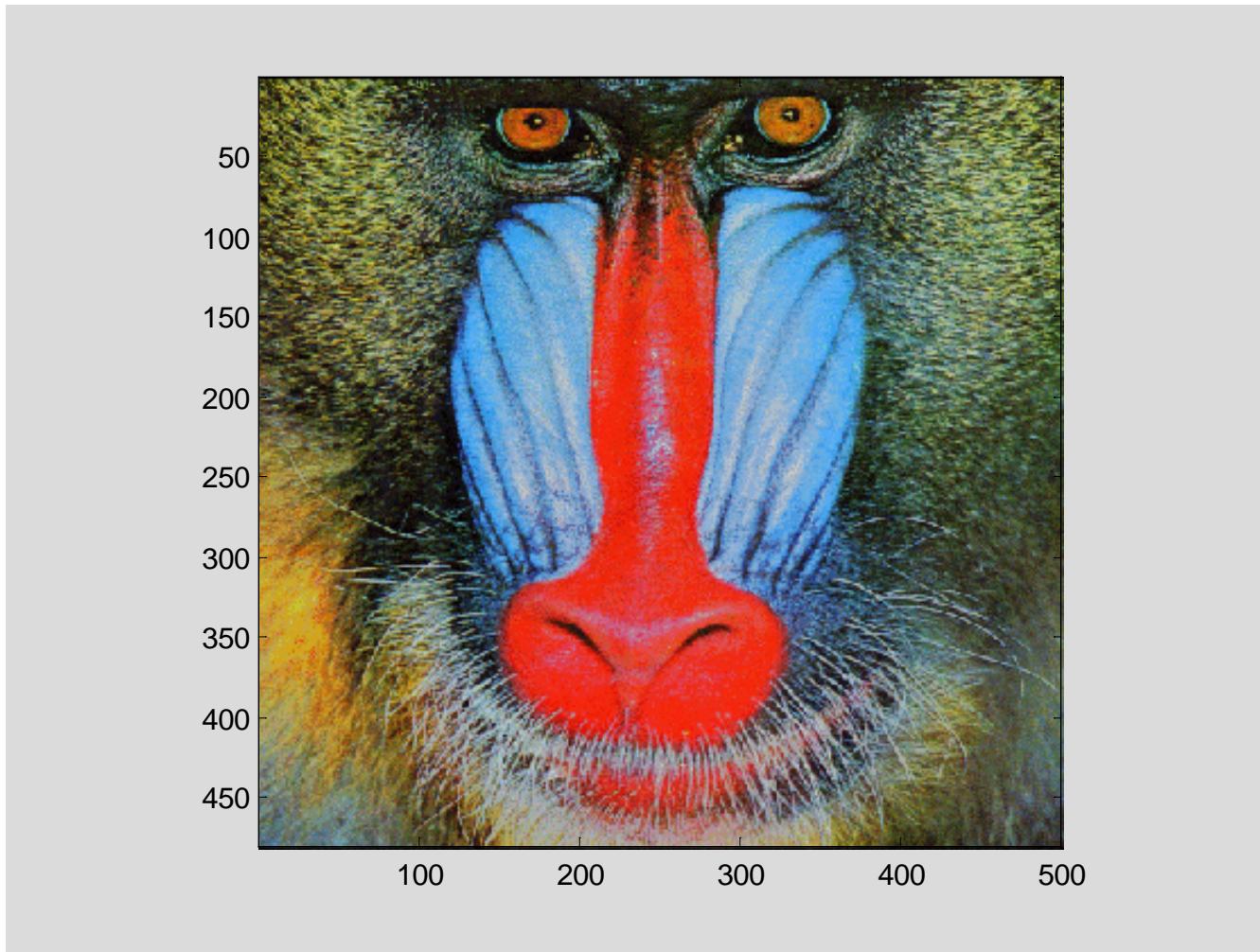


# 影像的顯示與列印

- MATLAB 在顯示影像時，會將之置於預設的圖軸之中，並以此圖軸的長寬比來成像，因而造成影像的失真。若要以影像本身的長寬比來成像，可加入 `axis image`，如下：

```
load mandrill.mat  
image(X);  
colormap(map);  
axis image
```

# 以原影像長寬比例顯示範例



# 將影像對應到螢幕上的點的範例

- 若要使影像資料的每一點對應至螢幕上的一個像素（Pixel），可輸入如下：

```
load mandrill.mat  
[m, n] = size(X);  
figure ('unit', 'pixel', 'position', [200, 200, n, m]);  
image(X);  
colormap(map);  
set(gca, 'position', [0, 0, 1, 1]);
```

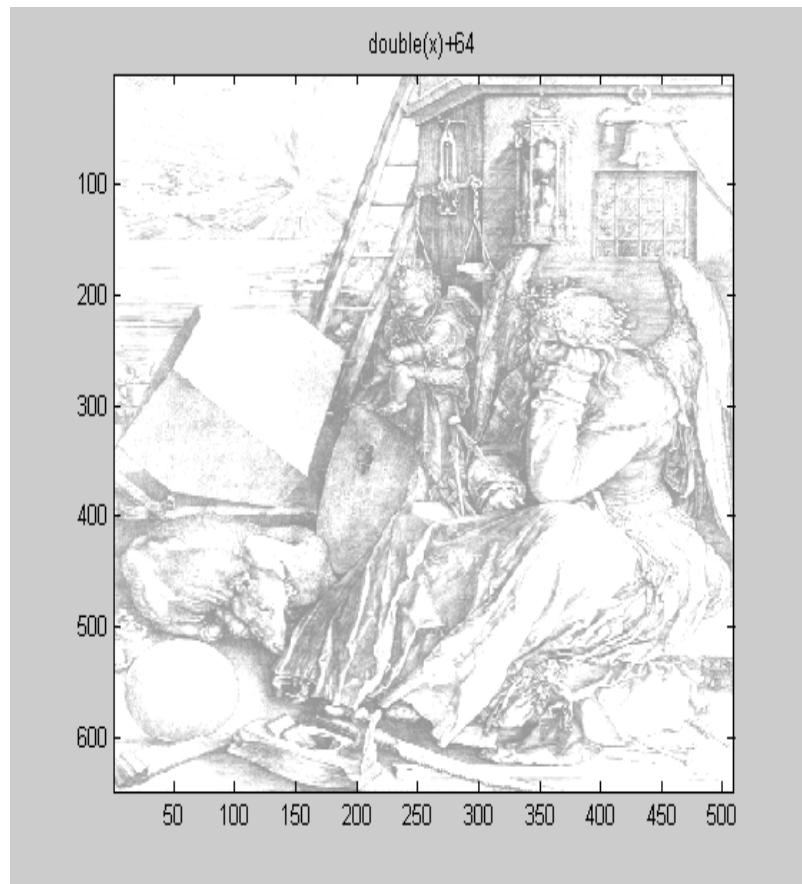
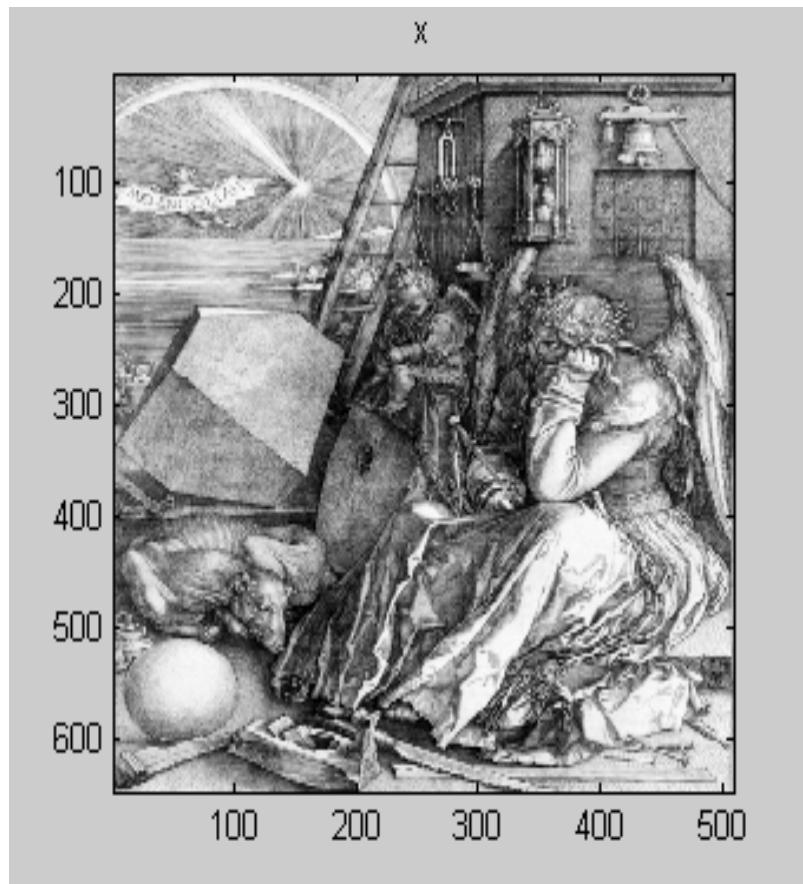
# 將影像對應到螢幕上的點

- 此範例產生圖形如同前一個範例，如果你的螢幕解析度較低，圖形會變大。
- 上述範例程式碼中，`figure` 的 ‘position’ 性質為 `[200, 200, n, m]`，代表視窗的左下角位置是 `[200, 200]`（以 pixel 為單位），而視窗的寬度為 `n`，高度為 `m`，正好可以符合影像的大小。
- `gca` 傳回使用中的圖軸，最後一個敘述將圖軸的位置設為整個視窗的大小，使用了正規化的單位。

# 影像的顯示與列印

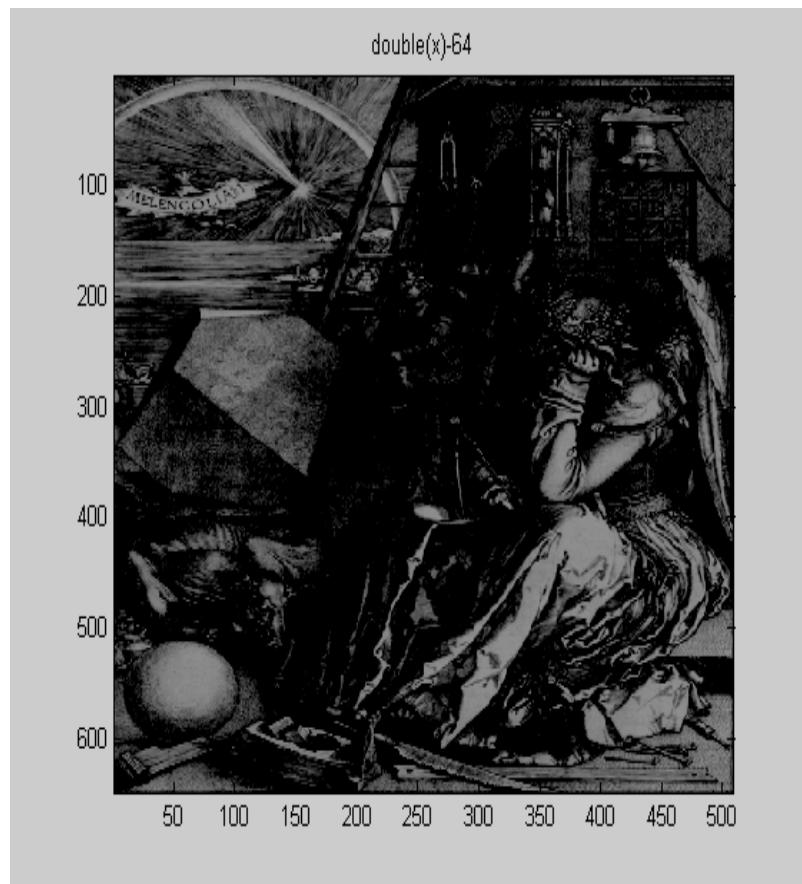
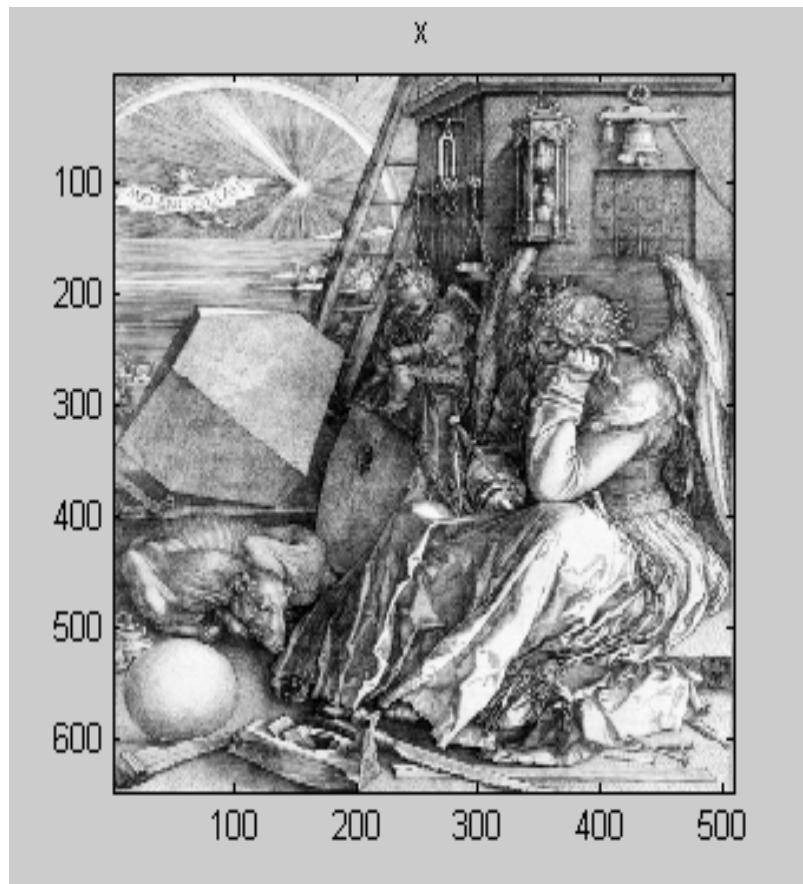
- 在列印影像時，MATLAB 會根據視窗的 **Paper position** 性質來調整圖形的長寬比，使得印出的影像再度變形。欲防止情況，可用下列指令：  
`>>set(gcf, 'PaperPositionMode', 'auto')`
- 若要使 **Paper Position Mode** 的預設值就是“**auto**”，可在 **startup.m** 檔案中加入下一行：  
`set(o, 'DefaultFigurePaperPositionMode', 'auto')`

# 點運算影像顯示



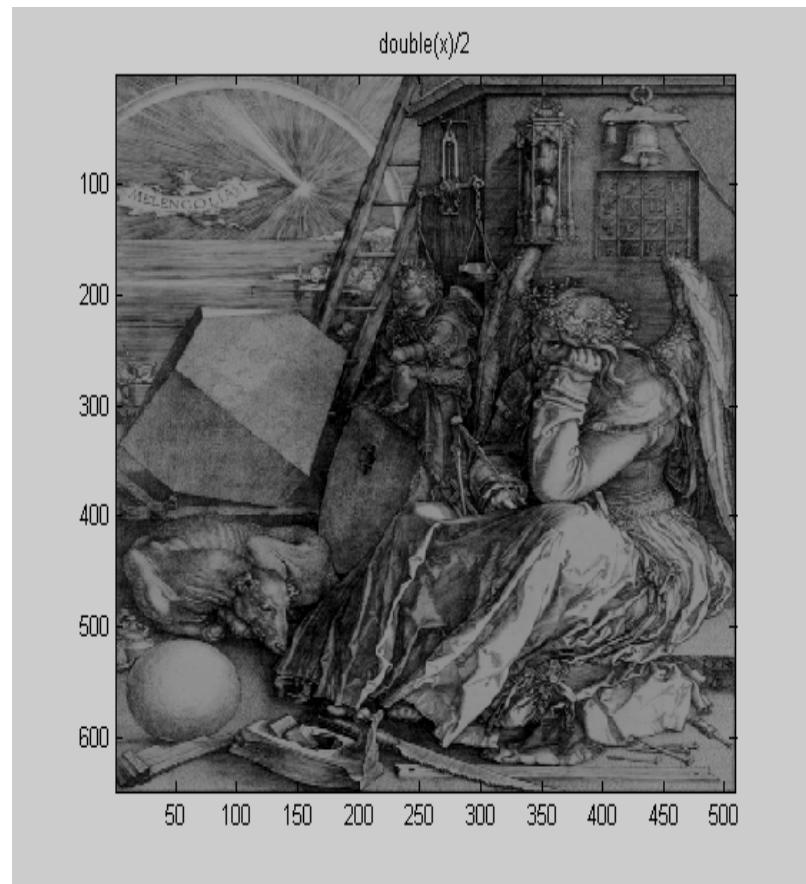
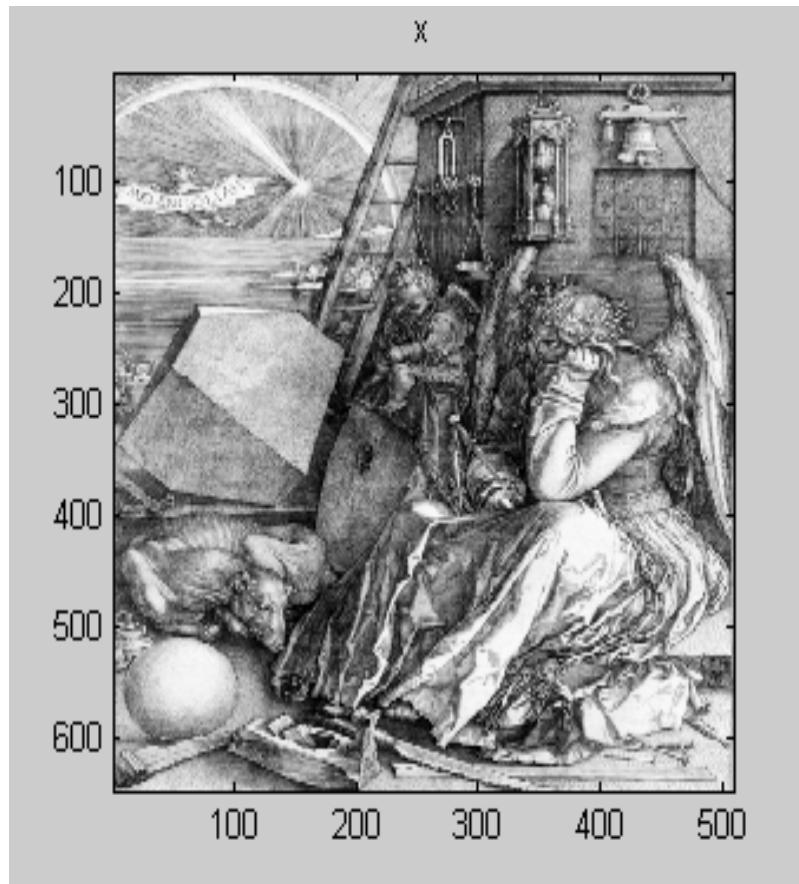
- `image(x);`
- `x1 = double(x)+64;`

# 點運算影像顯示



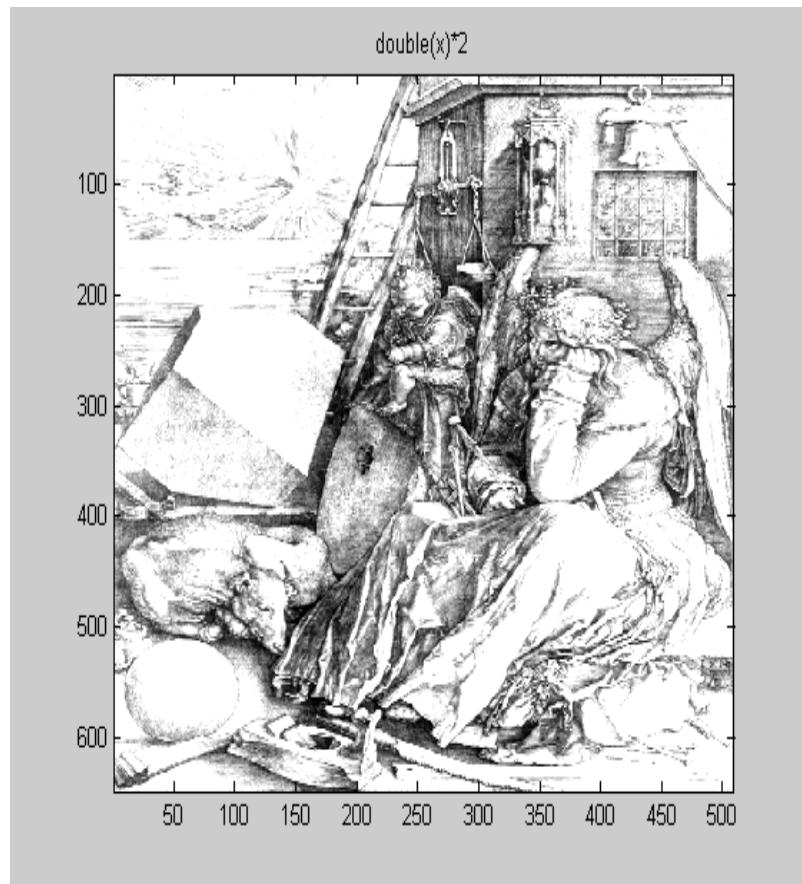
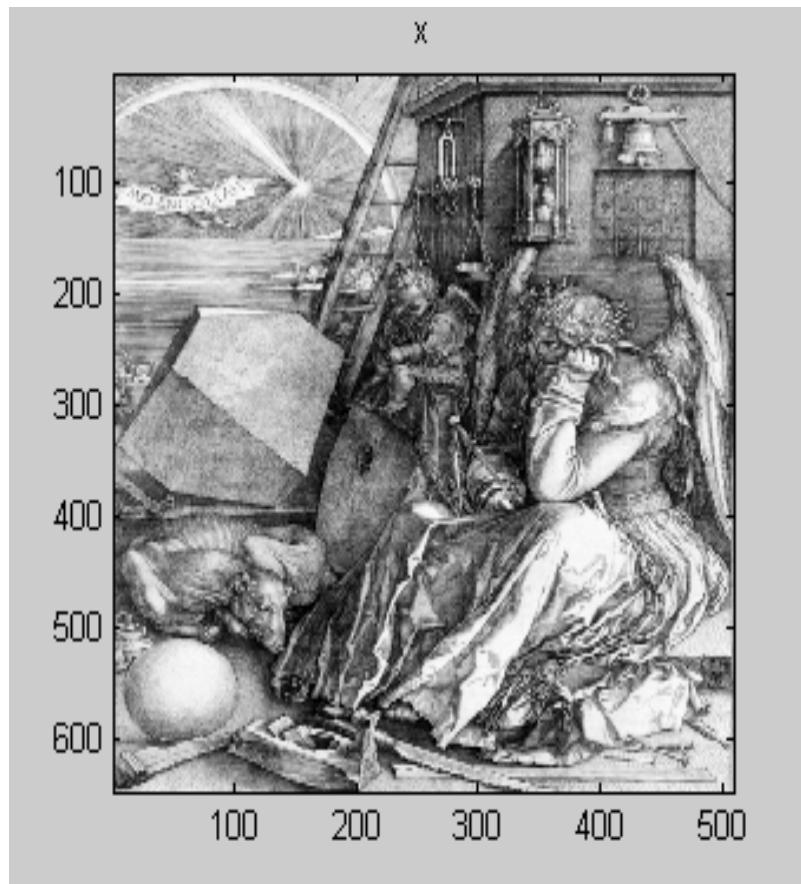
- `image(x);`
- `x1 = double(x)-64;`

# 點運算影像顯示



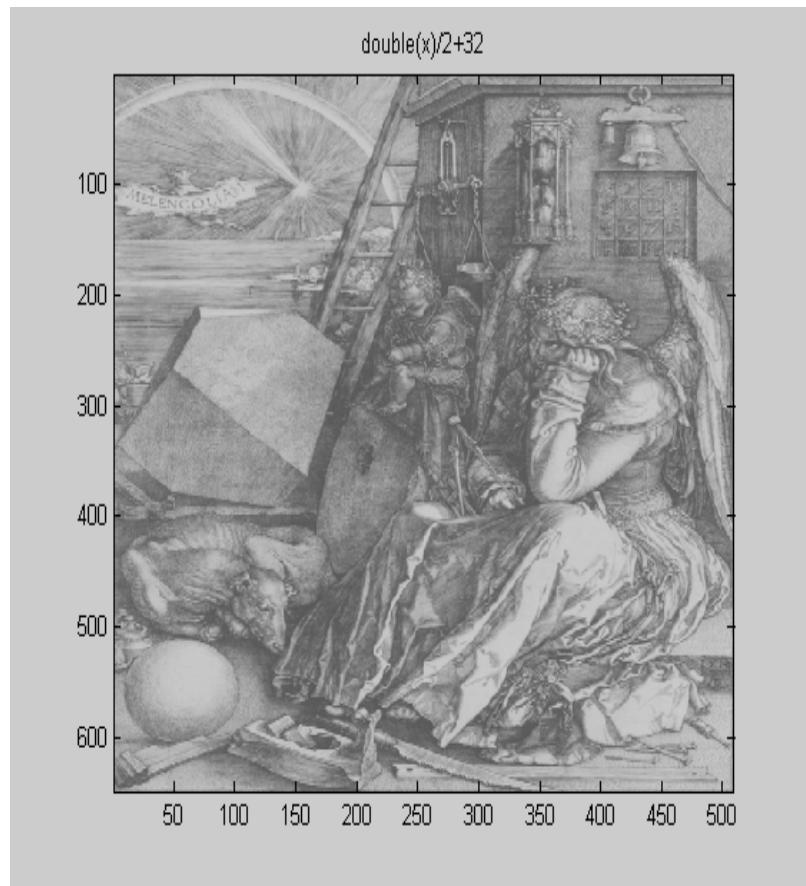
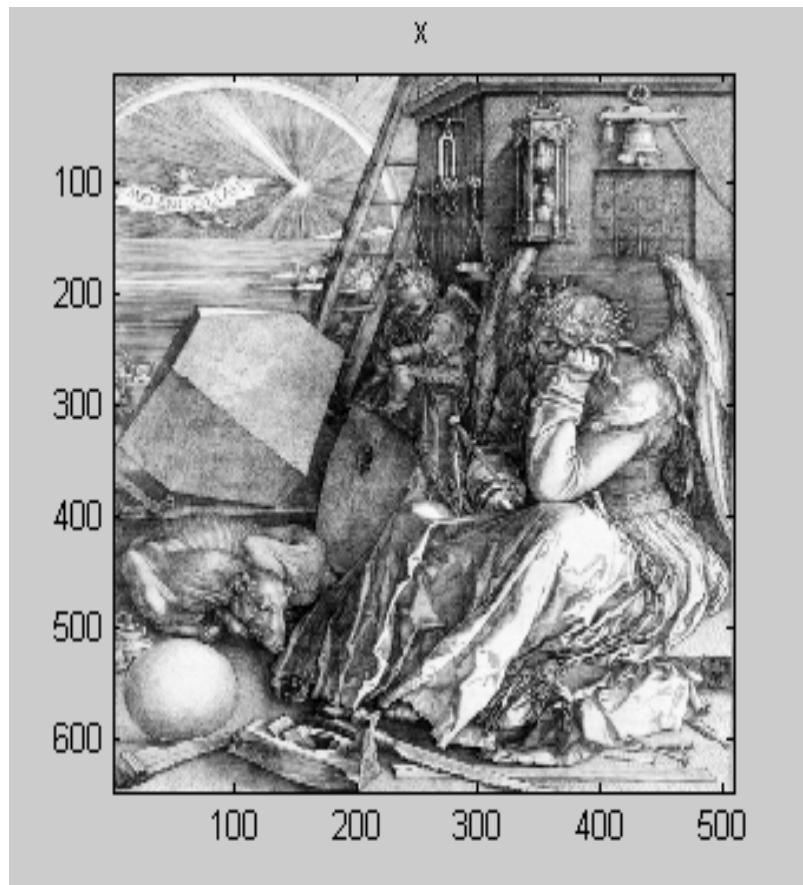
- `image(x);`
- `x1 = double(x)/2;`

# 點運算影像顯示



- `image(x);`
- `x1 = double(x)^2;`

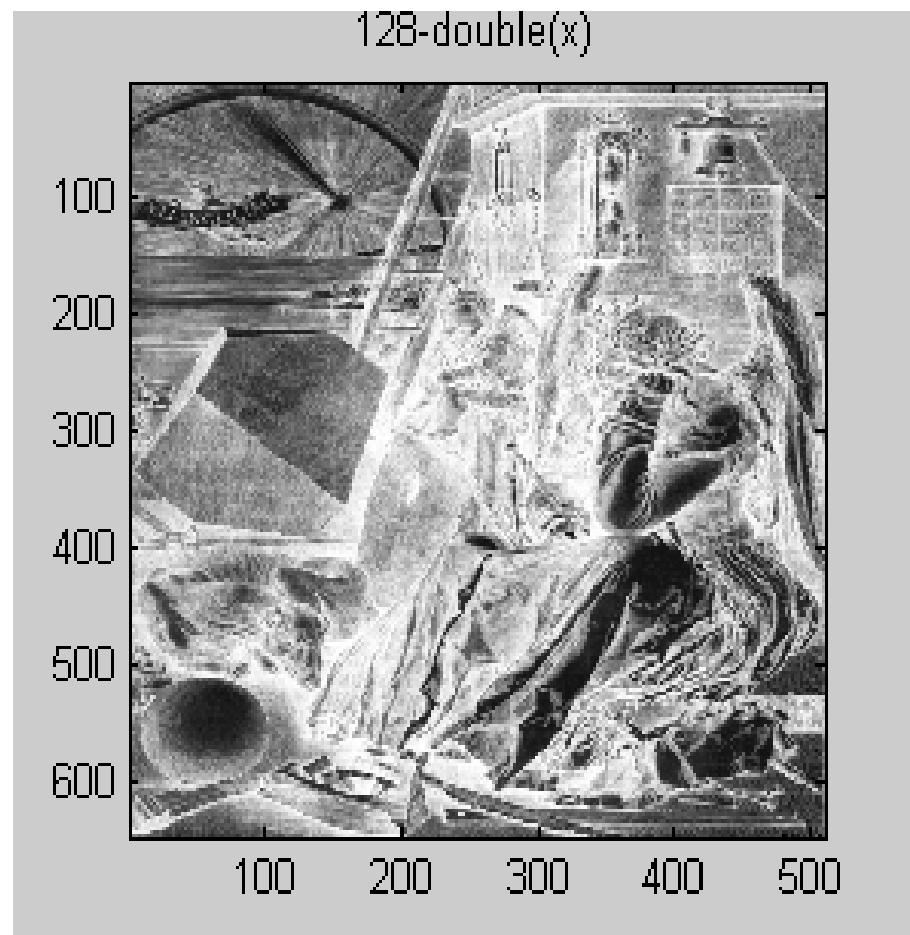
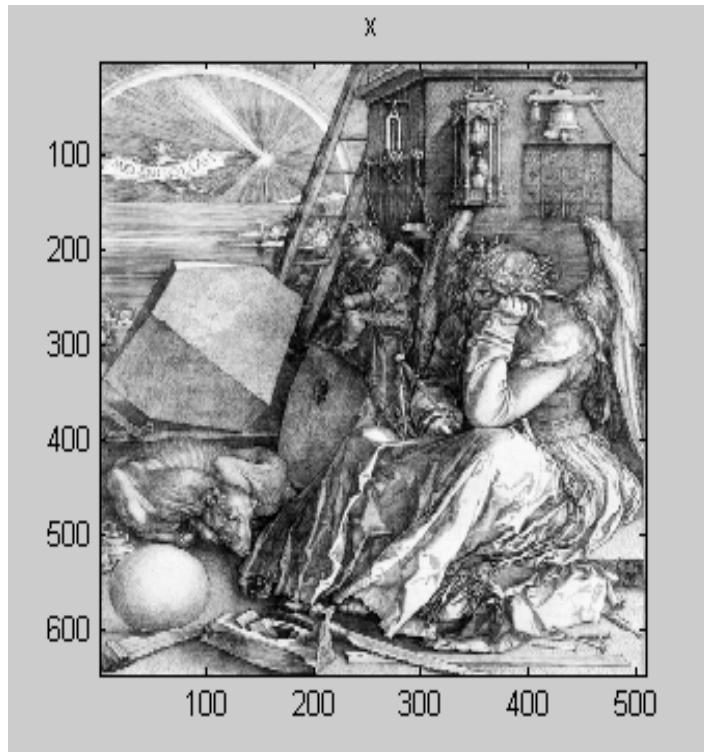
# 點運算影像顯示



- `image(x);`
- `x1 = double(x)/2+32;`

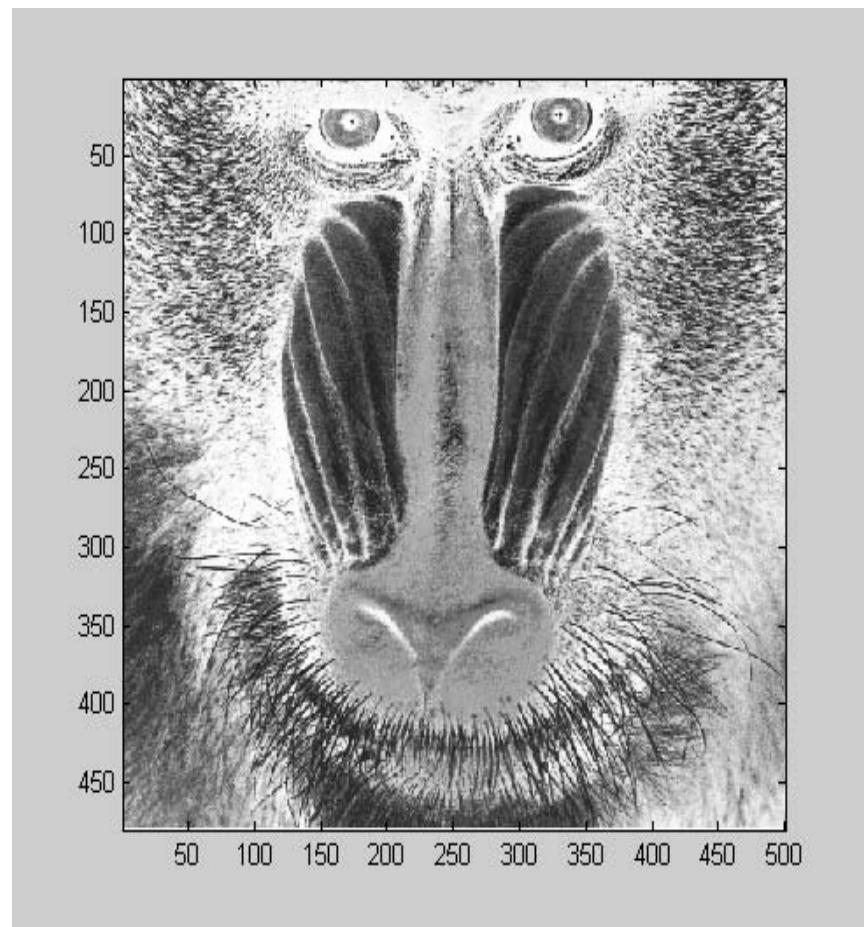
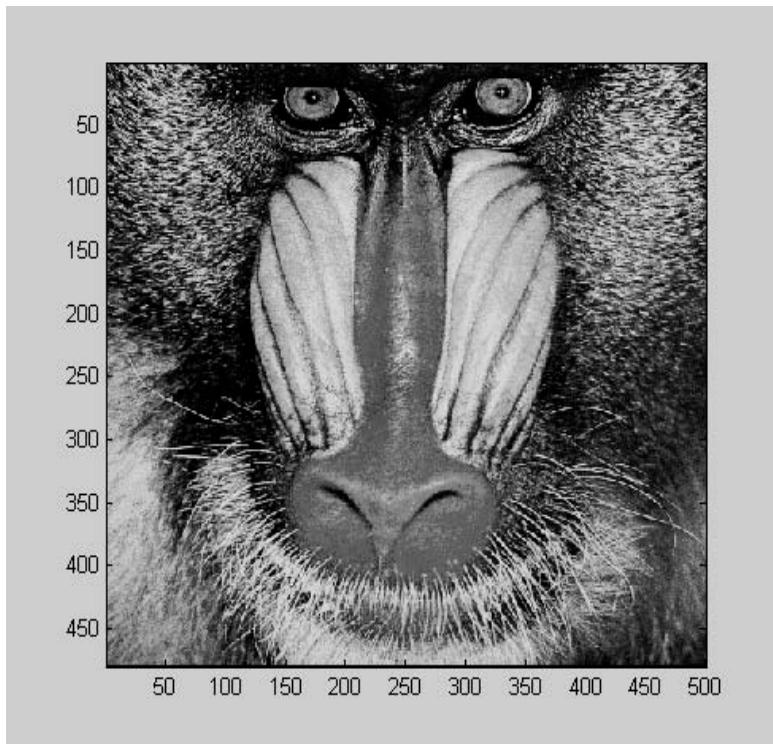
# 補色 complement

- $y = 128 - x;$



# 補色 complement

- $y = 256 - x;$



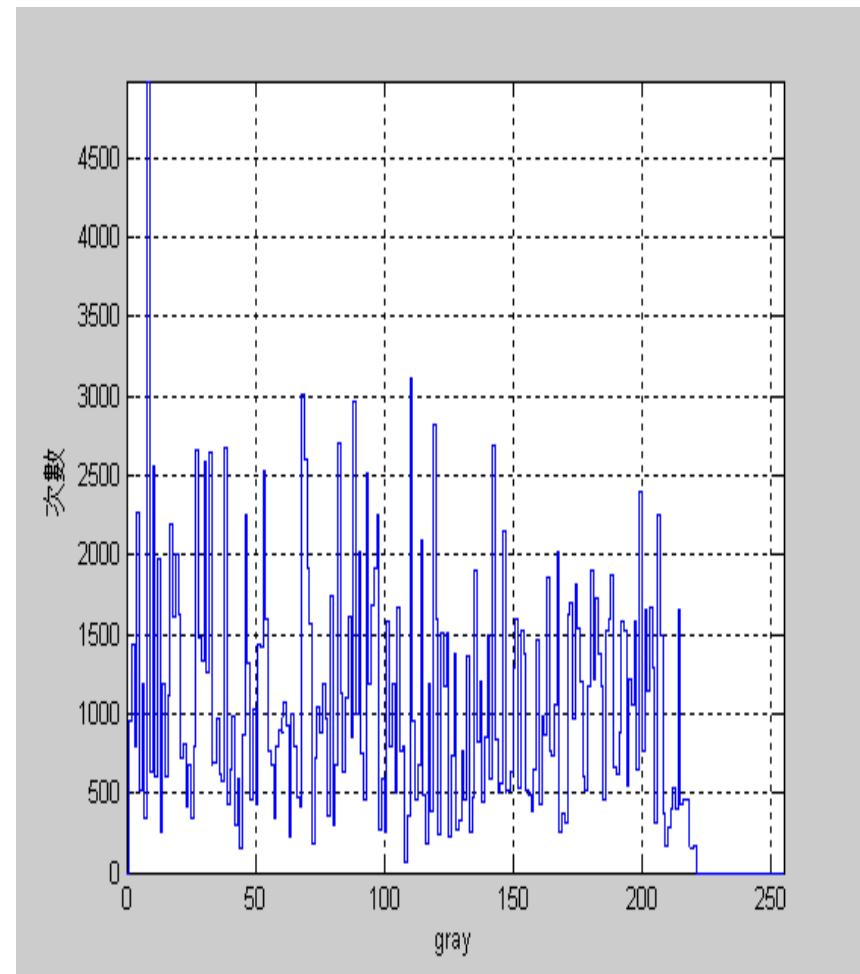
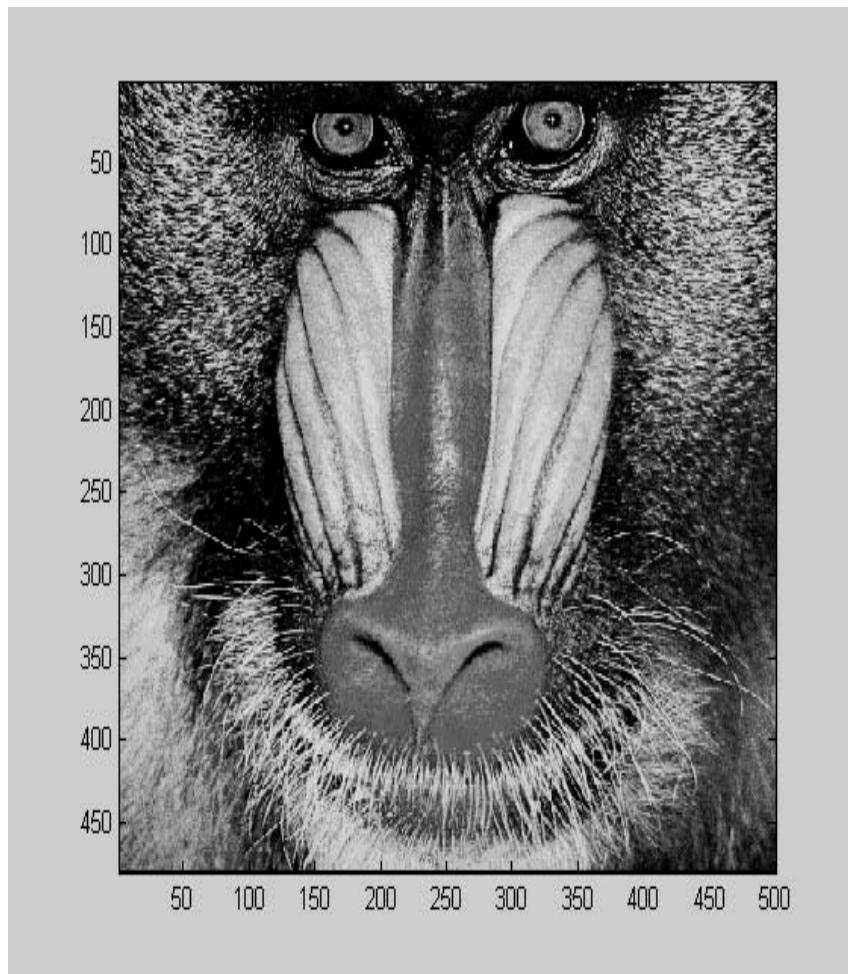
# 直方圖樣(灰階值分佈圖)

- 較暗影像：聚集在數值較低的區域
- 較亮影像：聚集在數值較高的區域
- 對比均勻影像：灰階層次平均分散在所有範圍

# 直方圖

```
■ [mm, nn] = size(X);  
■ for I = 0:1:255;  
■     c(i+1) = 0;  
■     for m = 1:1:mm  
■         for n = 1:1:nn  
■             if (X(m, n) == i)  
■                 c(i+1) = c(i+1)+1;  
■             end  
■         end  
■     end  
■ end  
■ figure(2);  
■ i = 0:1:255;  
■ stairs(i,c);
```

# 灰階值分佈圖



# 8-bit影像

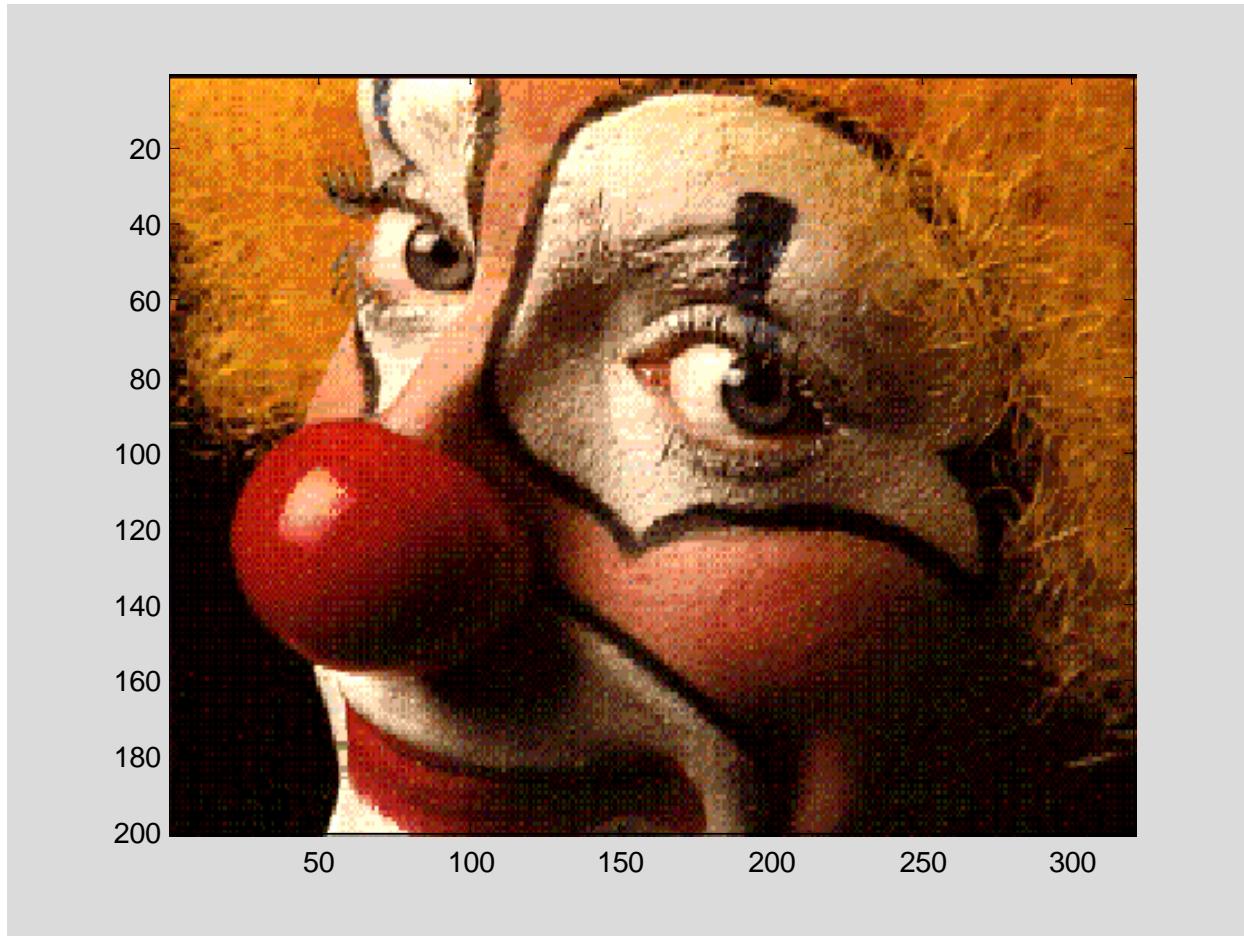
- 在 MATLAB 第 5 版之後，提供了 uint8 的資料型態。
- 由於 uint8 只有 8 個位元，所以能表示的數值範圍為 0 至 255 ( $=2^8-1$ ) 之間的整數。

# 8-bit影像範例

- 由於 8-bit 影像資料的最小值為 0，和一般的雙精準索引影像資料相差 1，因此在兩種資料相互轉換時，要特別小心。例如：
- 範例 19-8：uint8o1.m

```
load clown.mat  
Z8 = uint8(X-1);          % 將 X-1 轉成 uint8 的資料型態  
close all                  % 關掉所有的圖形視窗  
image(Z8)  
colormap(map)
```

# 8-bit影像範例



# 8-bit影像

- 若要將 8-bit 影像轉回雙精準影像，可輸入如下：  
`>> Z64 = double(Z8)+1;`
- uint8 資料型態亦可用於全彩影像資料，此時每一像素的原色（R，G 或 B）範圍為 0 至 255 間的整數，而不再是 0 至 1 的實數。

# 8-bit影像

- 欲將雙精準的全彩影像轉作 uint8 資料型態，可輸入如下：  
**>> RGB8 = uint8(round(RGB64\*255));**
- 其中 RGB64 為雙精準的全彩影像資料，而 RGB8 則是 unit8 的 8-bit 影像資料。反之，若欲進行反轉換，可輸入如下：  
**>> RGB64 = double(RGB8)/255;**
- 關於影像類別及其資料型態的關係，可見下表：

# 影像類別及型態關係表

資料型態		
影像類別	雙精準 ( Double )	uint8
索引影像 (Indexed Images )	影像矩陣大小 : $m \times n$	影像矩陣大小 : $m \times n$
	影像資料範圍 : 介於 [1, k] 的整數	影像資料範圍 : 介於 [0, k-1] 的整數
	色盤矩陣大小 : $k \times 3$	色盤矩陣大小 : $k \times 3$
	色盤資料範圍 : 介於 [0, 1] 的實數	色盤資料範圍 : 介於 [0, 1] 的實數
	影像顯示指令 : image	影像顯示指令 : image ( 註 : k 的值不大於 256 )
強度影像 (Intensity Images)	影像矩陣大小: $m \times n$	影像矩陣大小: $m \times n$
	影像資料範圍: 任意實數(但通常是[0,1])	影像資料範圍 : 介於 [0, 255] 的整數
	色盤矩陣大小 : $k \times 3$	色盤矩陣大小 : $k \times 3$
	色盤資料範圍 : 介於 [0, 1] 的實數	色盤資料範圍 : 介於 [0, 1] 的實數
	影像顯示指令 : imagesc ( 色盤通常是灰階 )	影像顯示指令 : imagesc ( 色盤通常是灰階 )
全彩影像 (Truecolor Images)	影像矩陣大小: $m \times n \times 3$	影像矩陣大小: $m \times n \times 3$
	影像資料範圍 : 介於 [0, 1] 的實數	影像資料範圍 : 介於 [0, 255] 的整數
	影像顯示指令 : image	影像顯示指令 : image

# 影像檔案的讀取與寫入

- `imread` 指令可用於讀取影像檔案。
- `imwrite` 則可用於寫入影像檔案。
- 這兩個指令可以處理的影像格式有下列幾種：

# imread及imwrite支援的格式

影像檔案格式	副檔名	相關字串
微軟視窗的 Bitmap	bmp	'bmp'
階層式資料格式 ( Hierarchical Data Format )	hdf	'hdf'
Joint Photographic Expert Group	jpg 或 jpeg	'jpg' 或 'jpeg'
微軟視窗的 Paintbrush	pcx	'pcx'
可攜式網路圖形 ( Portable Network Graphics )	png	'png'
標記式影像檔案格式 ( Tagged Image File Format )	tiff	'tif' 或 'tiff'
X視窗傾印 ( X Windows Dump )	xwd	'xwd'
圖形交換格式 ( Graphic Interchange Format ) ( 第六版才支援 )	gif	'gif'

# 影像檔案的讀取與寫入

- `imread` 指令可以讀取上述格式的影像檔案，並進行必要之轉換，如下：
  - 對於強度影像，`imread` 將資料以 `uint8` 的矩陣（大小為  $m \times n$ ）傳回。
  - 對於索引影像，`imread` 將資料以 `uint8` 的矩陣（大小為  $m \times n$ ）傳回，並同時傳回一個雙精準的色盤矩陣，其每個元素值介於  $[0,1]$ 。
  - 對於全彩矩陣，`imread` 將資料以 `uint8` 的矩陣（大小為  $m \times n \times 3$ ）傳回。

# 使用imread讀取全彩jpg影像

- imread 可讀出下列全彩影像：

```
RGB = imread('simulinkteam.jpg');  
image(RGB)
```

# 使用imread讀取全彩jpg影像



# 影像檔案寫入範例

- `imwrite` 指令可將資料寫成影像檔如下：

```
load clown.mat  
imwrite(X, map, 'myClown.jpg');  
!start myClown.jpg
```

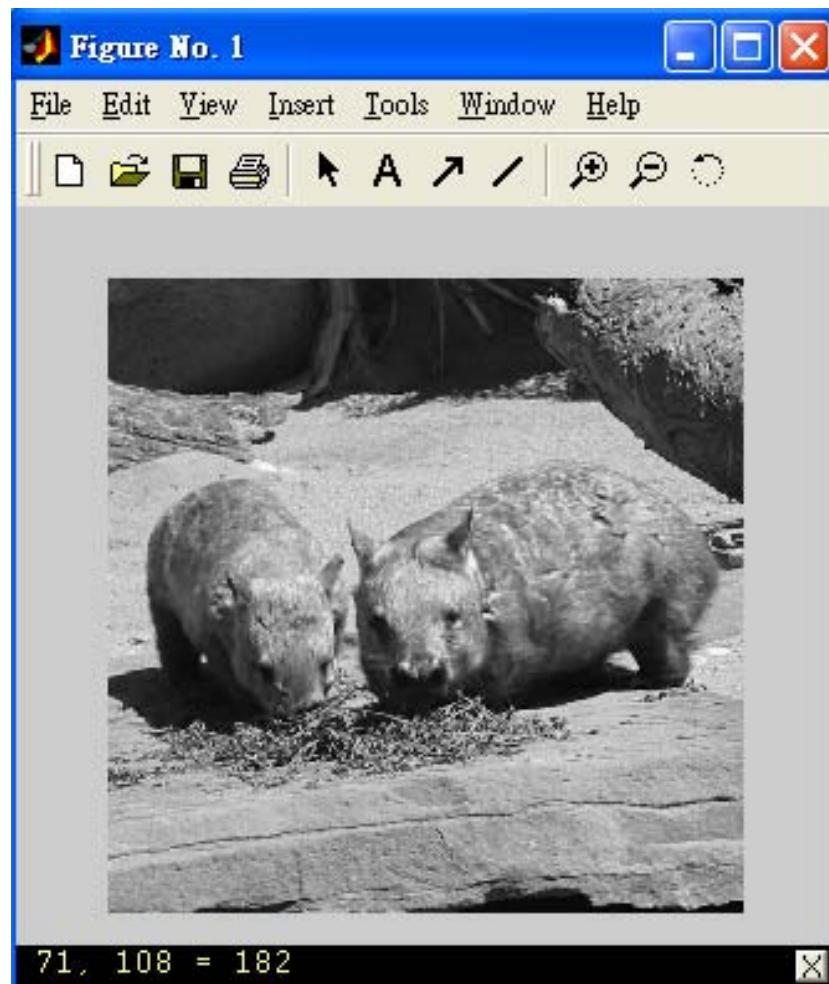
- 上述最後一列敘述將會呼叫 Windows 作業系統下的應用程式來開啟 `myClown.jpg` 檔案。

# 影像檔案的讀取與寫入

- `imfinfo` 指令可用於傳回影像檔案的各項資訊，例如：  
`>> info = imfinfo('simulinkteam.jpg')`
- 對於不同的檔案格式，`imfinfo` 傳回的資訊項目可能有所不同。

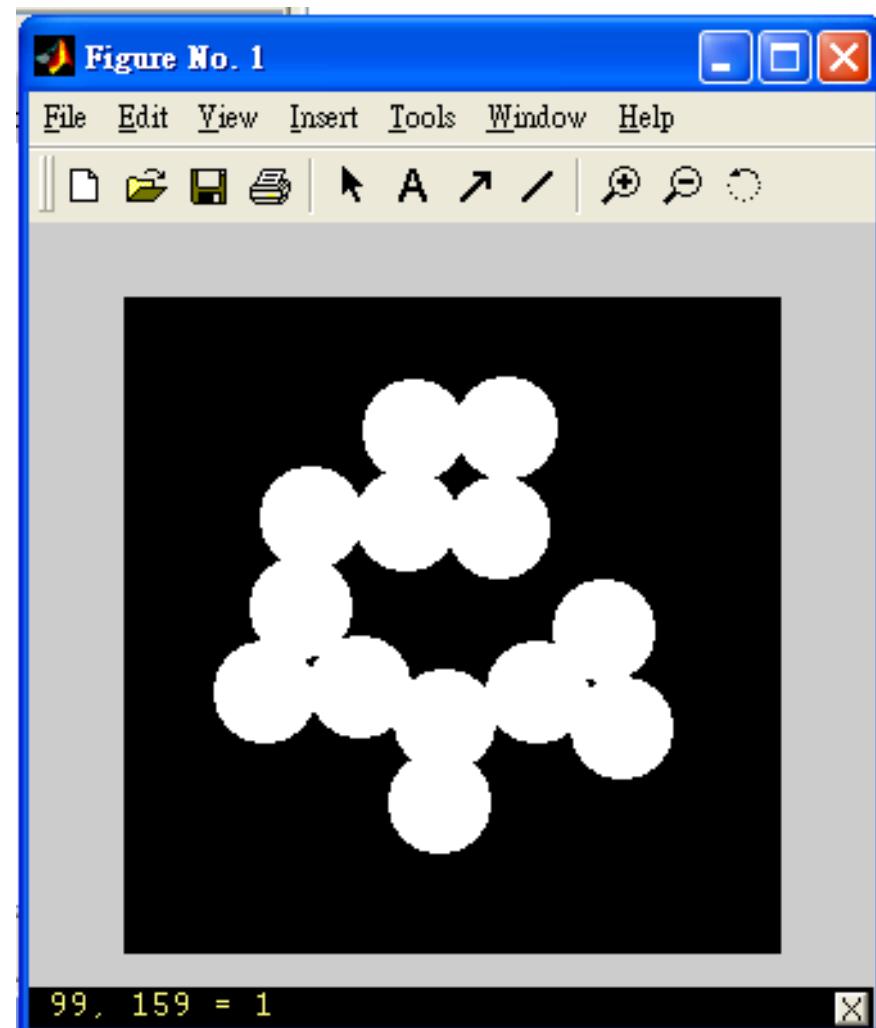
# Grayscale images 灰階影像

- Matlab example:
  - `w=imread('wombats.tif');`
  - `figure, imshow(w), pixval on`
- **figure**: create a window to place graphic object
- **imshow**: display matrix
- Data type of w?
  - `256x256 uint8 (unsigned integer 8 bits)`



# Binary image 二元影像

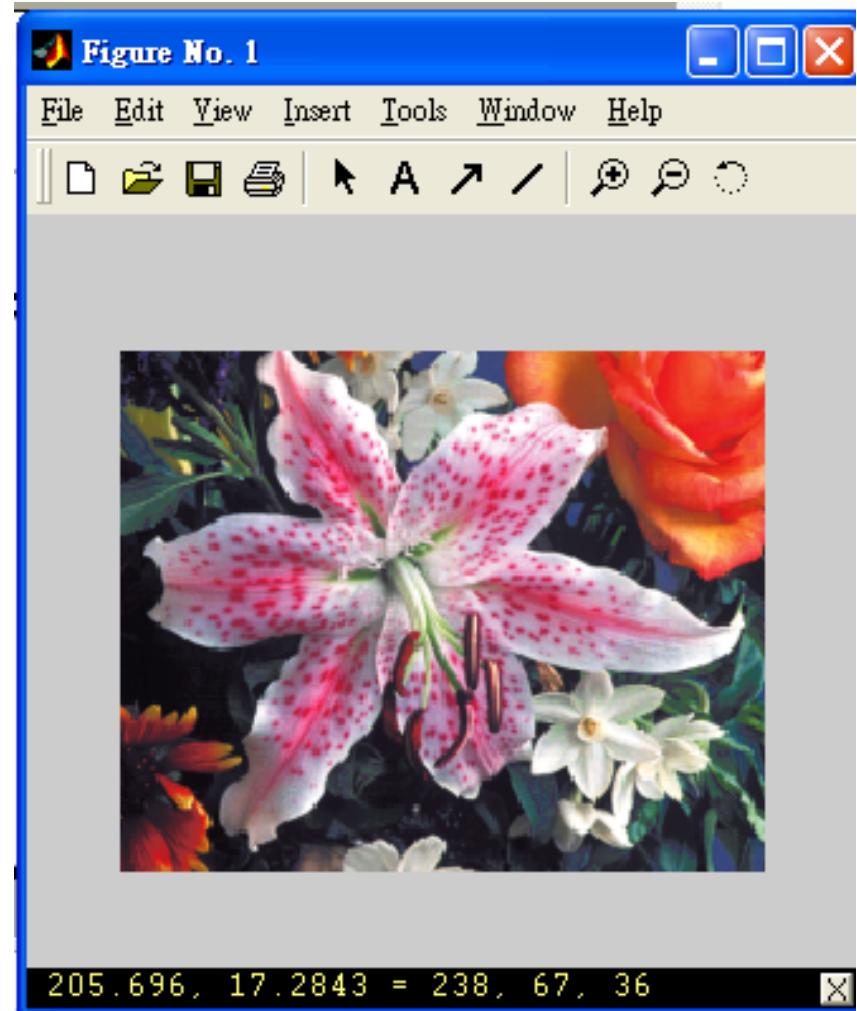
- Matlab example:
  - `w=imread('circles.tif');`
  - `figure, imshow(w), pixval on`
- Data type of w?
  - 256x256 logical
  - Pixel value is 0 or 1



# RGB (true color) images 全彩影像

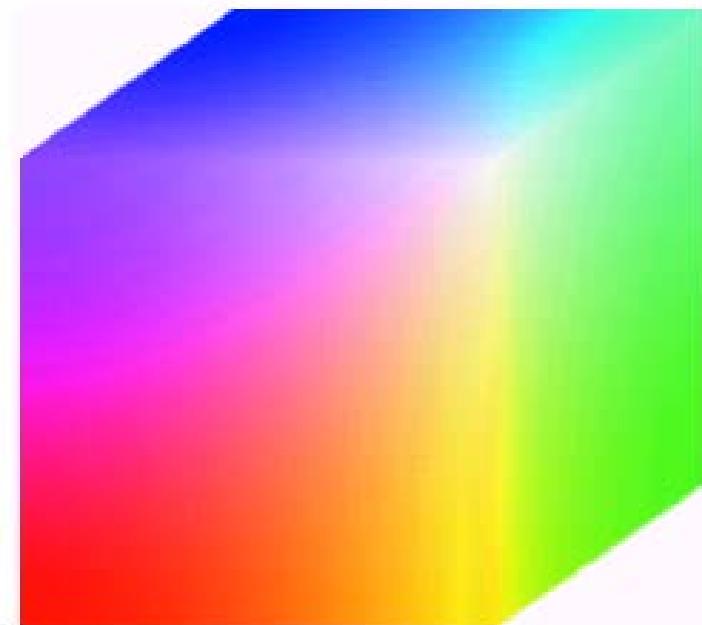
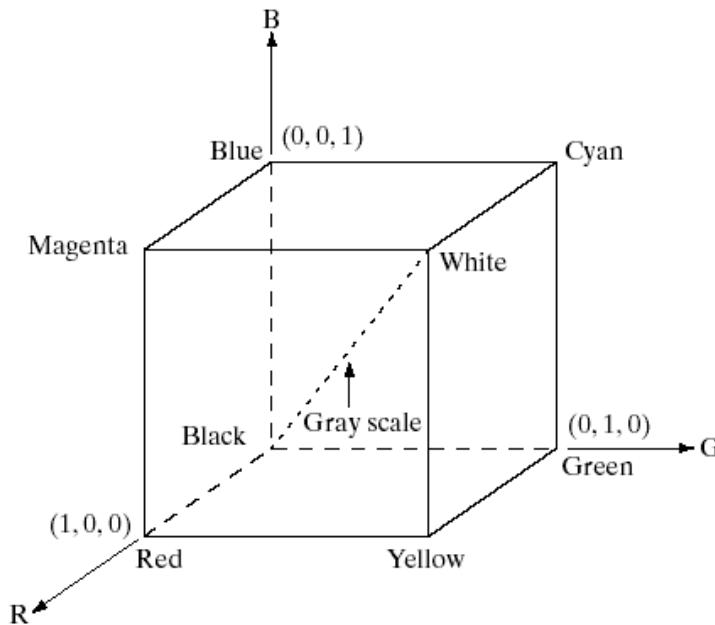
## ■ Matlab example:

- `w=imread('lily.tif');`
- `figure, imshow(w), pixval on`



# Pixel depth

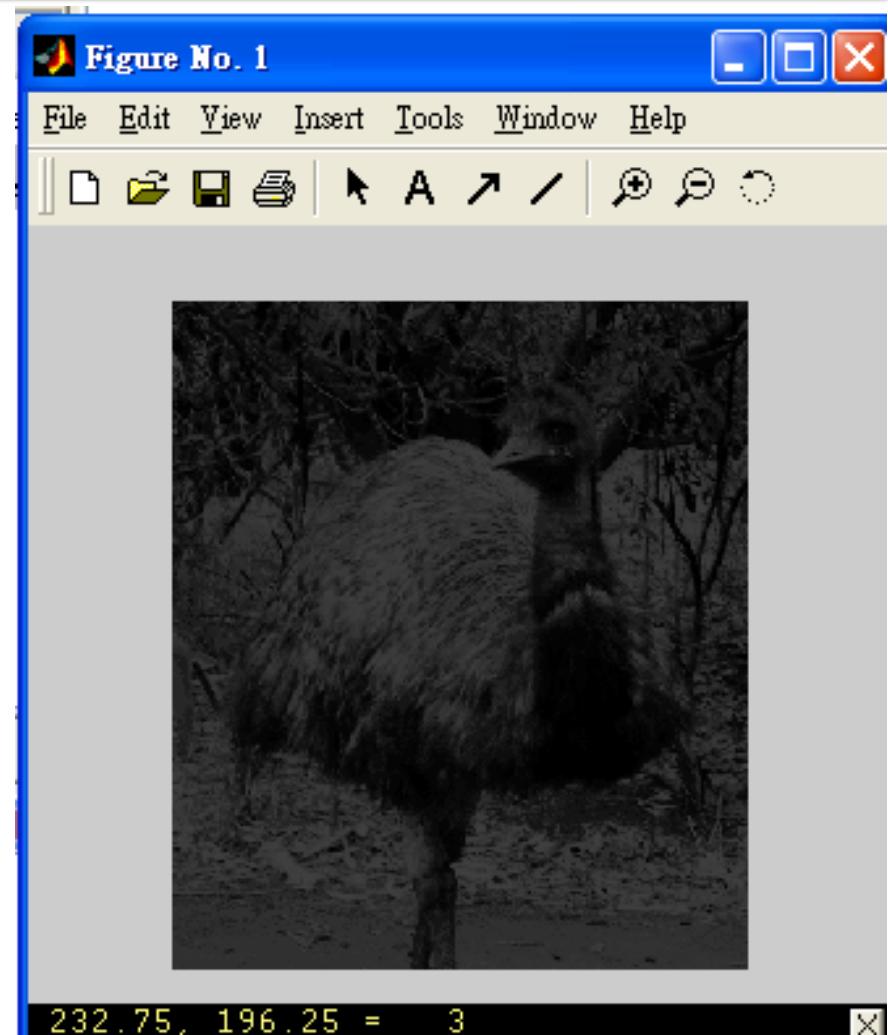
- Pixel depth: the number of bits used to represent each pixel in RGB space
- Full-color image: 24-bit RGB color image
  - $(R, G, B) = (8 \text{ bits}, 8 \text{ bits}, 8 \text{ bits})$



# Indexed color image 彩色索引影像

## ■ Matlab example:

- `w=imread('emu.tif');`
- `figure, imshow(w), pixval on`
- What's wrong?



# Indexed color image



6	10	15	12
5	11	20	10
4	6	10	7

indices

0.1211	0.1211	0.1416
0.1807	0.2549	0.1729
0.2197	0.3447	0.1807
0.1611	0.1768	0.1924
0.2432	0.2471	0.1924
0.2119	0.1963	0.2002
...		

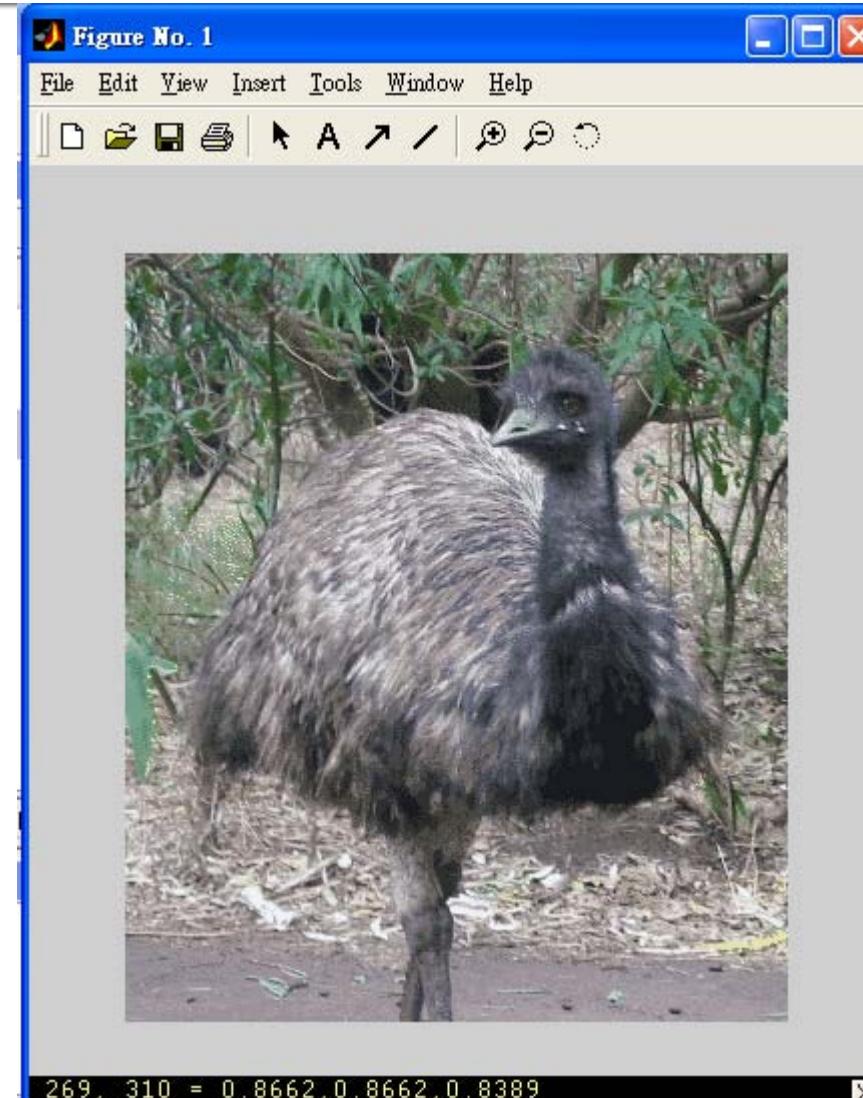
Color map

# Indexed color image

## Matlab example:

- [w,wmap]=imread('emu.tif');
- imshow(w, wmap)

## How do we know it's an indexed image?



# Get information about image

- `imfinfo('emu.tif');`

```
Filename: 'emu.tif'  
FileModDate: '12-Jul-2004 11:40:00'  
FileSize: 119804  
Format: 'tif'  
FormatVersion: []  
Width: 331  
Height: 384  
BitDepth: 8  
ColorType: 'indexed'  
ByteOrder: 'little-endian'  
NewSubfileType: 0  
BitsPerSample: 8  
Colormap: [256x3 double]
```

# Write image matrix to file

- Matlab code
  - `w=imread('wombats.tif');`
  - `imwrite(w, 'wombats.pgm', 'pgm');`
- General form
  - `imwrite(X, map, 'filename', 'format');`

# MATLAB supported image formats

- JPEG: Joint Photographic Experts Group
- TIFF: Tagged Image File Format
- GIF: Graphics Interchange Format
- BMP: Microsoft Bitmap Format
- PNG: Portable Network Graphics
- ...

# Image file formats

- Header information
  - Size of the image
  - Color map, compression method, etc.
- Body
  - Compressed or uncompressed
  - ASCII or binary

# Simple ASCII PGM

- `w=imread('wombats.tif');`
- `imwrite(w, 'wombats.pgm', 'pgm', 'encoding', 'ASCII');`

```
P2 256 256 255
67 64 65 65 63 54 51 48 49 48 45 55 60 52 53 58 52
52 58 61 62 58 52 51 68 65 52 51 54 49 51 55 54 61
54 60 61 57 58 58 61 69 72 71 60 55 61 55 52 46 48
49 58 56 43 42 40 43 48 51 51 54 57 58 52 52 54 50
...
```

header

# BMP File header



Bytes	Information	Description
0–1	Signature	BM in ASCII = 42 4D in hexadecimal.
2–5	FileSize	The size of the file in bytes.
6–9	Reserved	All zeros.
10–13	DataOffset	File offset to the raster data.
14–17	Size	Size of the information header = 40 bytes.
18–21	Width	Width of the image in pixels.
22–25	Height	Height of the image in pixels.
26–27	Planes	Number of image planes (= 1).
28–29	BitCount	Number of bits per pixel: 1: Binary images; two colors, 4: $2^4 = 16$ colors (indexed), 8: $2^8 = 256$ colors (indexed), 16: 16-bit RGB; $2^{16} = 65,536$ colors, 24: 24-bit RGB; $2^{24} = 17,222,216$ colors.
30–33	Compression	Type of compression used: 0: No compression (most common), 1: 8-bit RLE encoding (rarely used), 2: 4-bit RLE encoding (rarely used).
34–37	ImageSize	Size of the image. If compression is 0, then this value may be 0.
38–41	HorizontalRes	The horizontal resolution in pixels per meter.
42–45	VerticalRes	The vertical resolution in pixels per meter.
46–49	ColorsUsed	The number of colors used in the image. If this value is zero, then the number of colors is the maximum obtainable with the bits per pixel, that is, $2^{\text{BitCount}}$ .
50–53	ImportantColors	The number of important colors in the image. If all the colors are important, then this value is set to zero.

# JPEG file header

Bytes	Information	Description
0–1	Start of image marker	Always FF D8.
2–3	Application marker	Always FF E0.
4–5	Length of segment	
5–10	JFIF\ 0	ASCII JFIF.
11–12	JFIF version	In our example above 01 01 or version 1.1.
13	Units	Values are: 0 arbitrary units; 1 pixel/inch; 2 pixels/centimeter.
14–15	Horizontal pixel density	
16–17	Vertical pixel density	
18	Thumbnail width	If this is 0, there is no thumbnail.
19	Thumbnail height	If this is 0, there is no thumbnail.

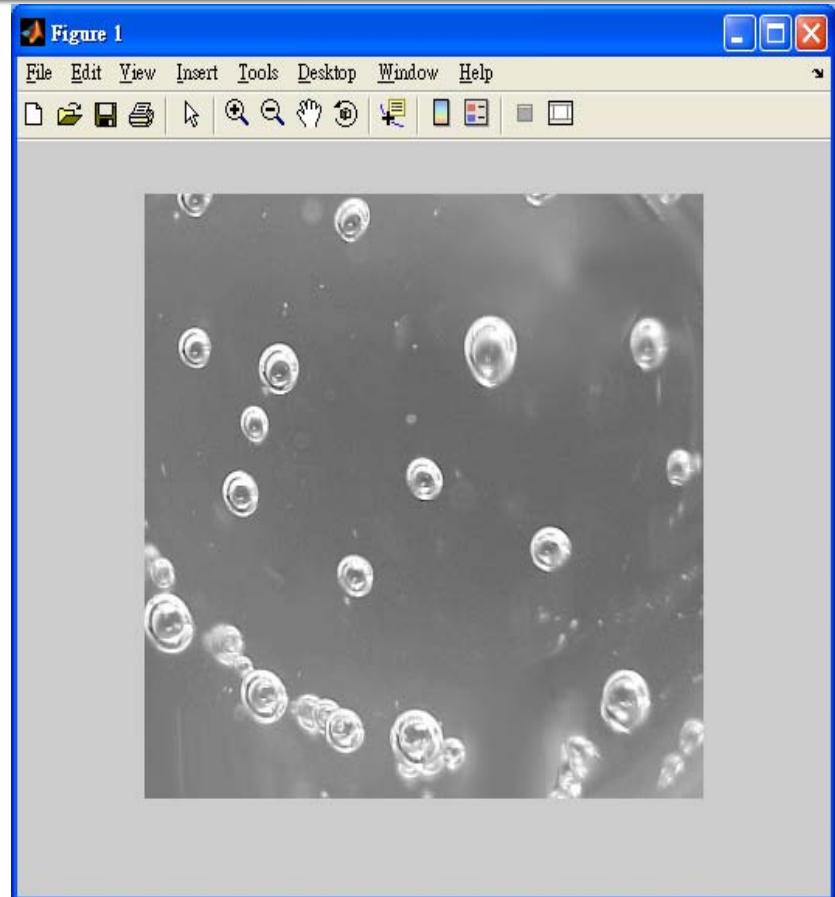
The thumbnail information (stored as 24-bit RGB values) and further information required for decompressing the image data would follow. For further information available see [2], [20].

# Write JPEG image file

- `I=imread('bubbles.tif');`
- `imwrite(I, 'bubbles50.jpg',  
'quality', 50);`

- Check compression ratio

```
K=imfinfo('bubbles50.jpg');  
ratio=(K.Width*K.Height*K.BitDepth/8)/K.FileSize;
```

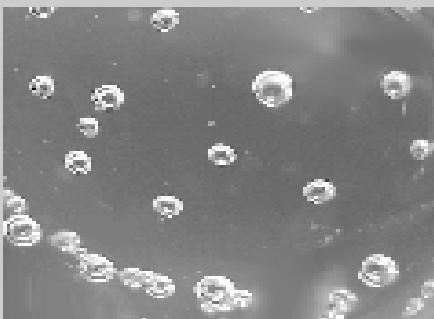


↑ 單位為 Byte

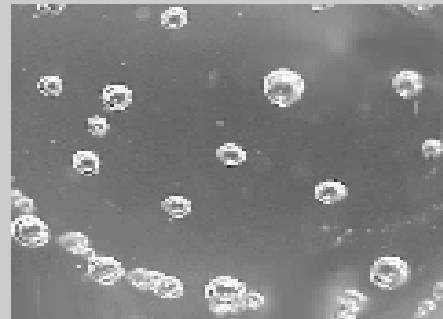
# Exercise

- Write JPEG file with **different quality factor** and compute its compression ratio

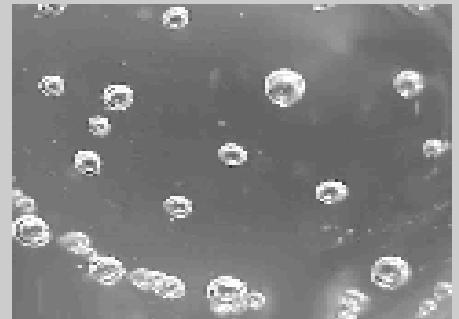
origin



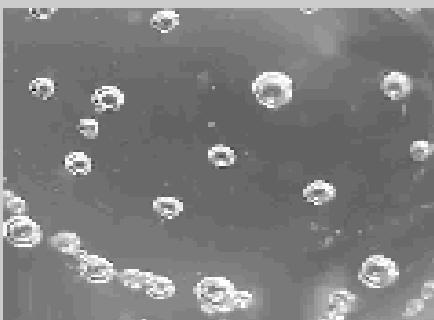
q=50, ratio=24.6841



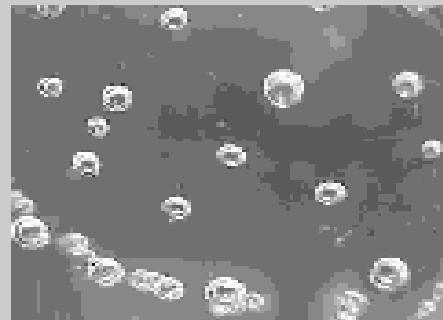
q=25, ratio=37.0945



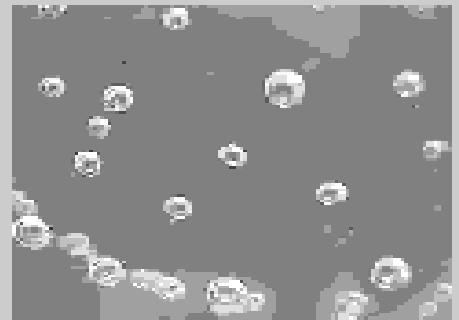
q=15, ratio=45.2508



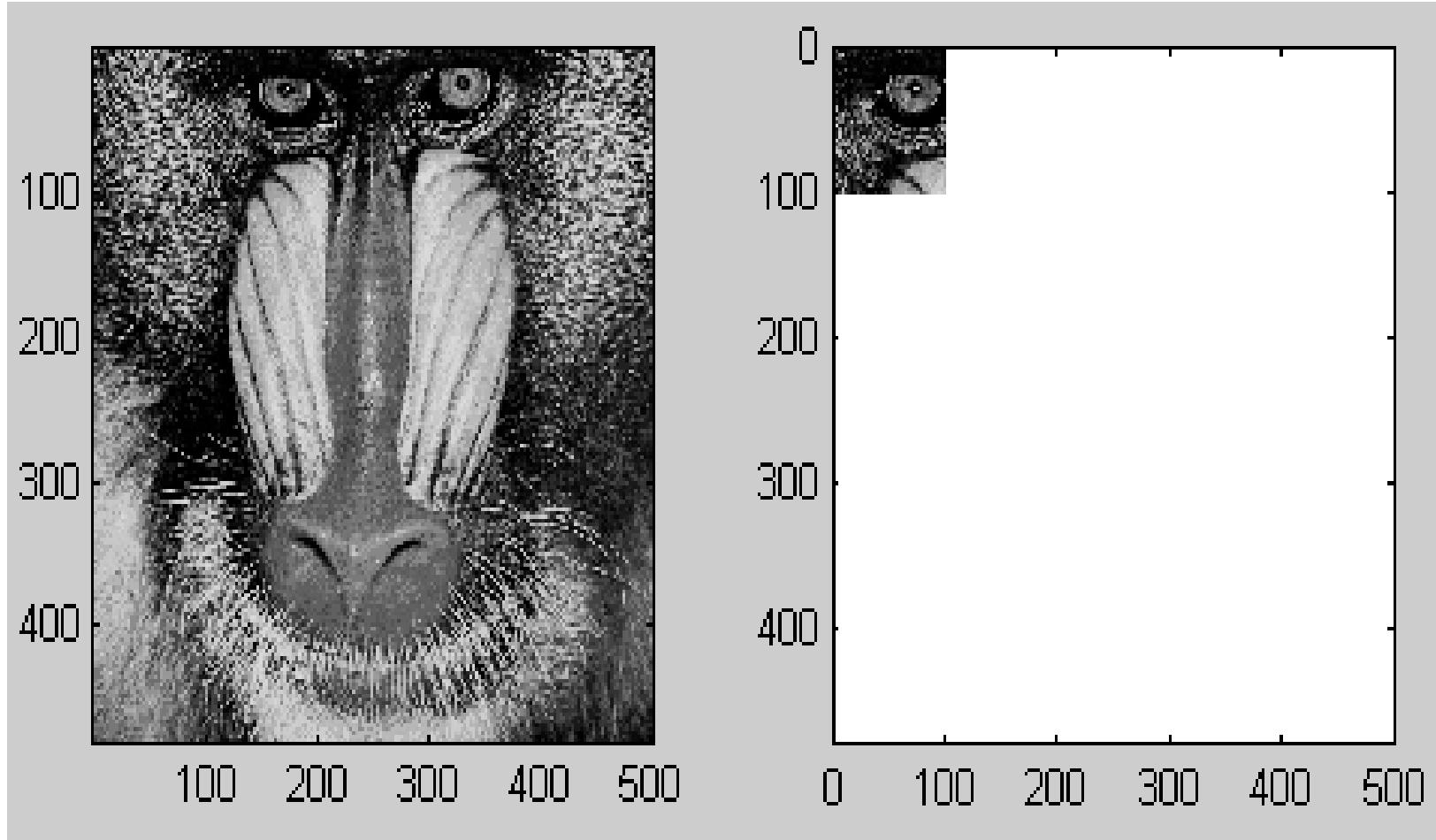
q=5, ratio=62.6721



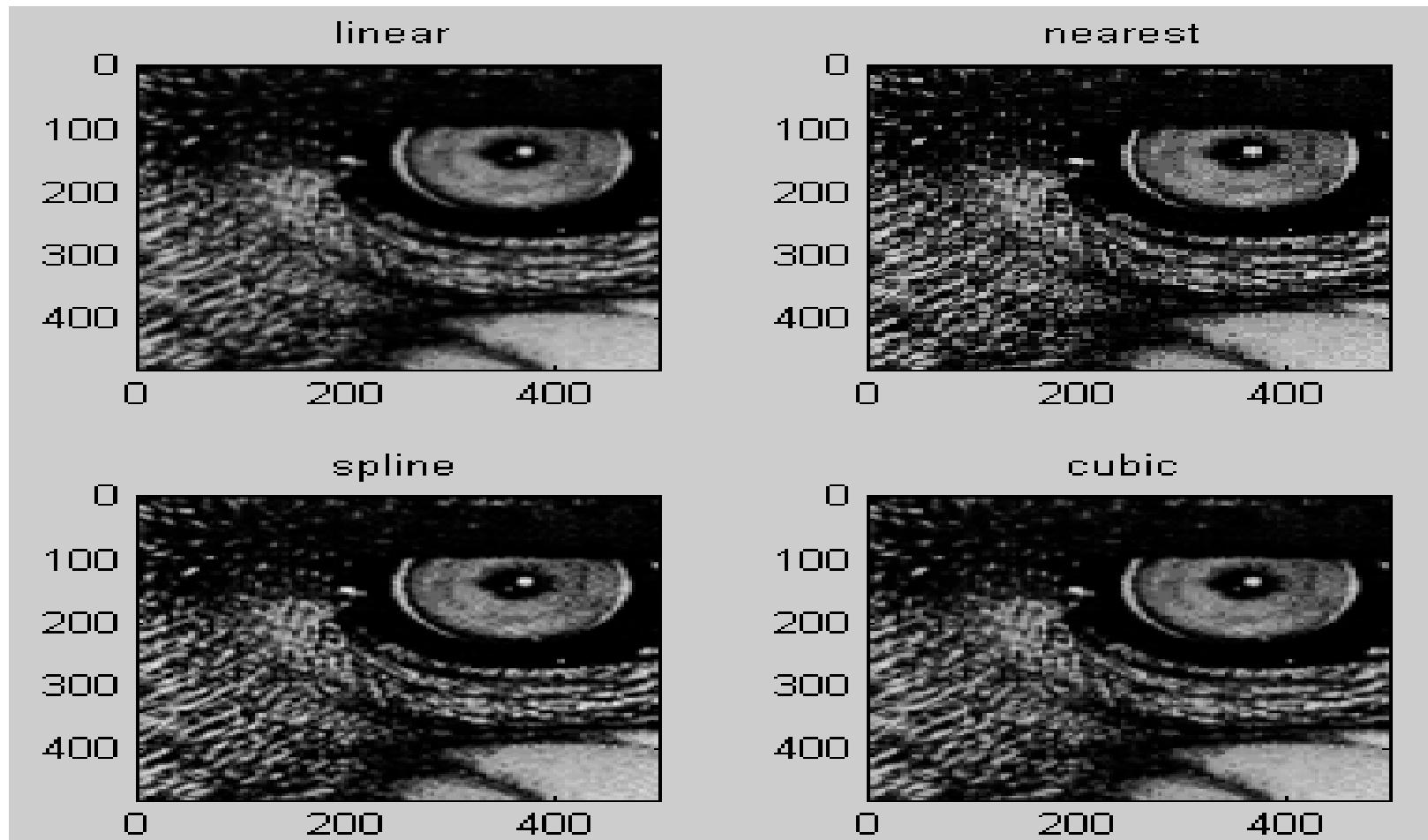
q=0, ratio=72.3259



# 影像放大



# 影像放大5倍



# 影像放大

- subplot(1,2,1);
- [x, map]=imread('mandrill.tif');
- image(x); colormap(gray(256));
- axis square;
- subplot(1,2,2);
- X1 = 1:101; y1 = 100:200;
- xp = x(X1, y1); image(xp);
- axis([0 500 0 480]); axis square;

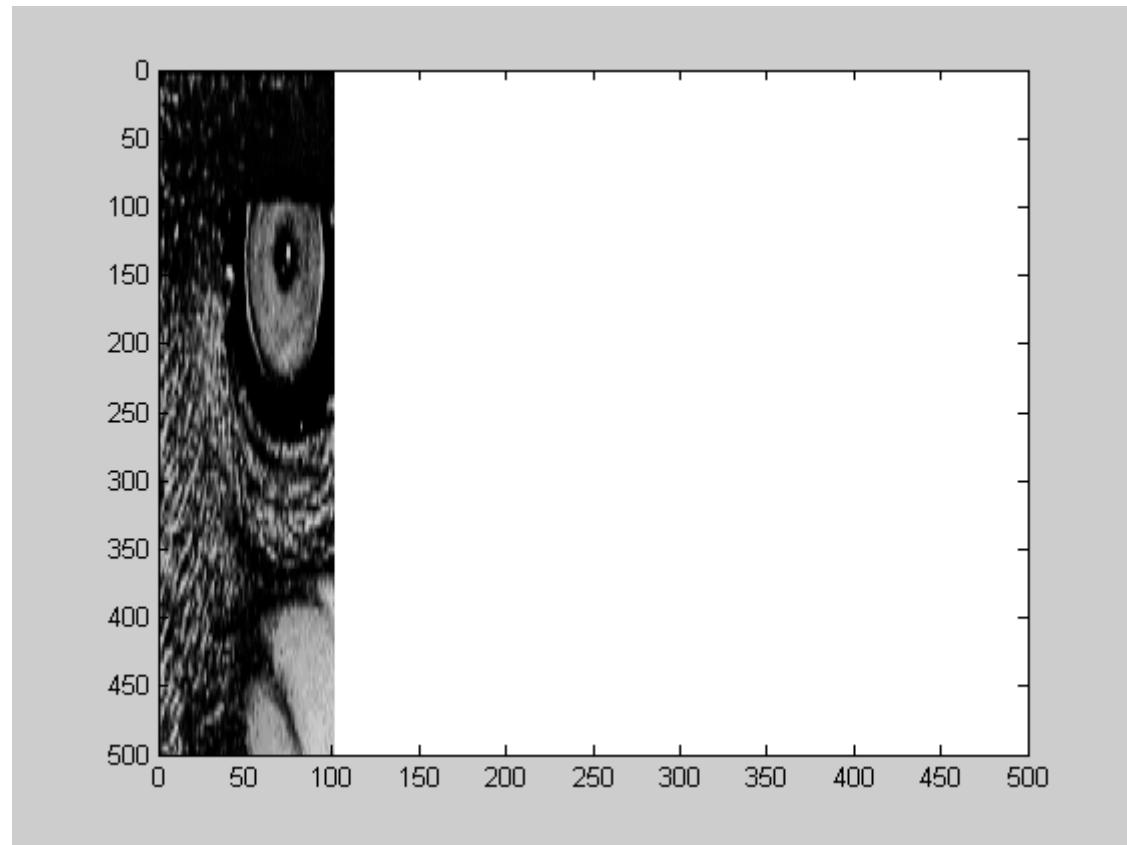
# 影像放大

- `x2 = linspace(1, 101, 500);`
- `imextend = interp1(x1, double(xp), x2, 'linear');`
- `image(imextend');`   `colormap(gray(256));`
- `x3 = 100:200;   x4 = linspace(100, 200, 500);`
- `imextend = interp1(x3, double(imextend'), x4, 'linear');`
- `image(imextend');`   `colormap(gray(256));`
- `axis([0 500 0 480]);   axis square;`
- `title('linear');`

# 影像放大：y方向

- `x2 = linspace(1, 101, 500);`
- `imextend = interp1(x1, double(xp), x2, 'linear');`
- `image(imextend);`  
`colormap(gray(256));`
- `axis([0 500 0 500]);`

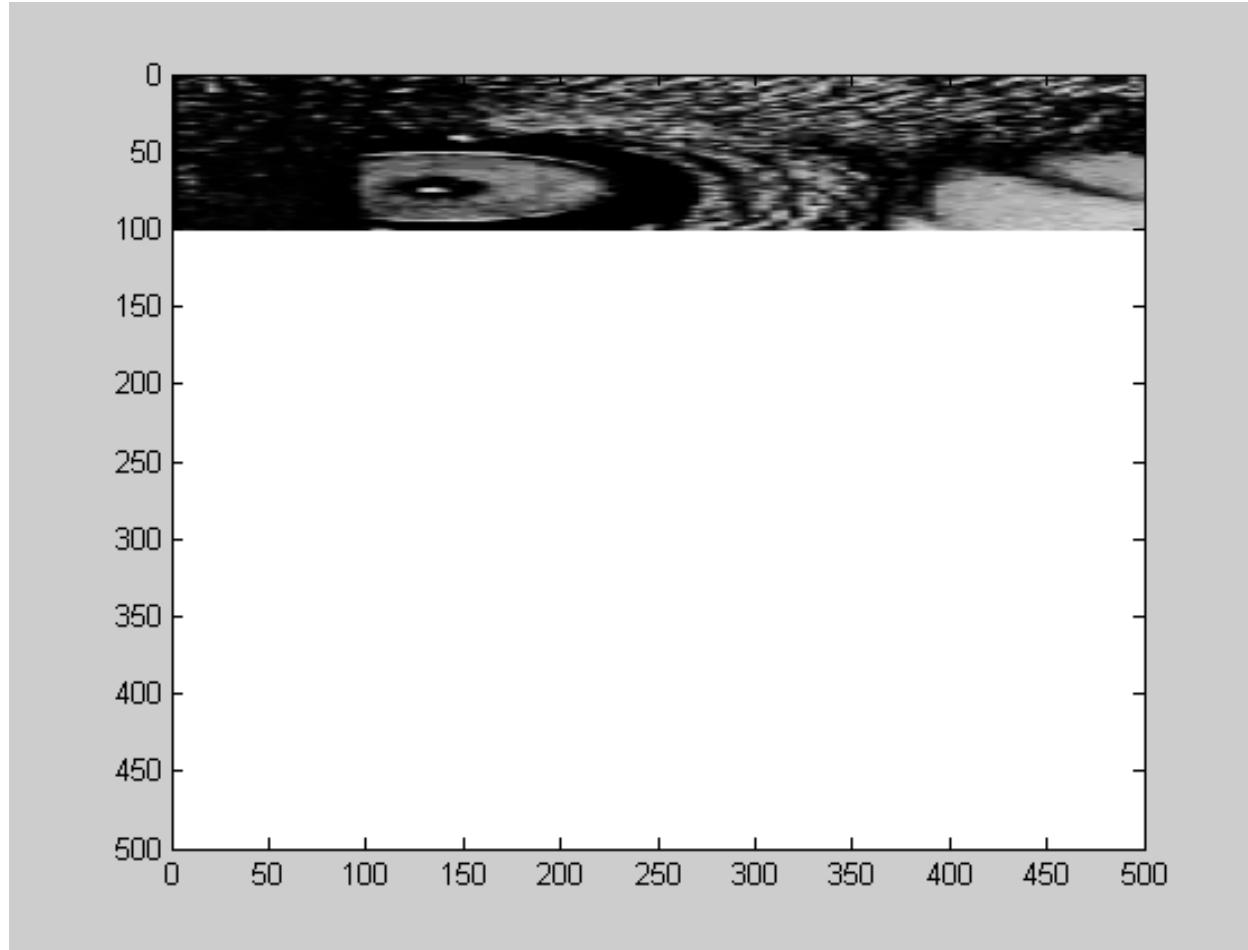
# 影像放大：y方向



# 影像

- `x2 = linspace(1, 101, 500);`
- `imextend = interp1(x1, double(xp), x2, 'linear');`
- `image(imextend');`  
`colormap(gray(256));`
- `axis([0 500 0 500]);`

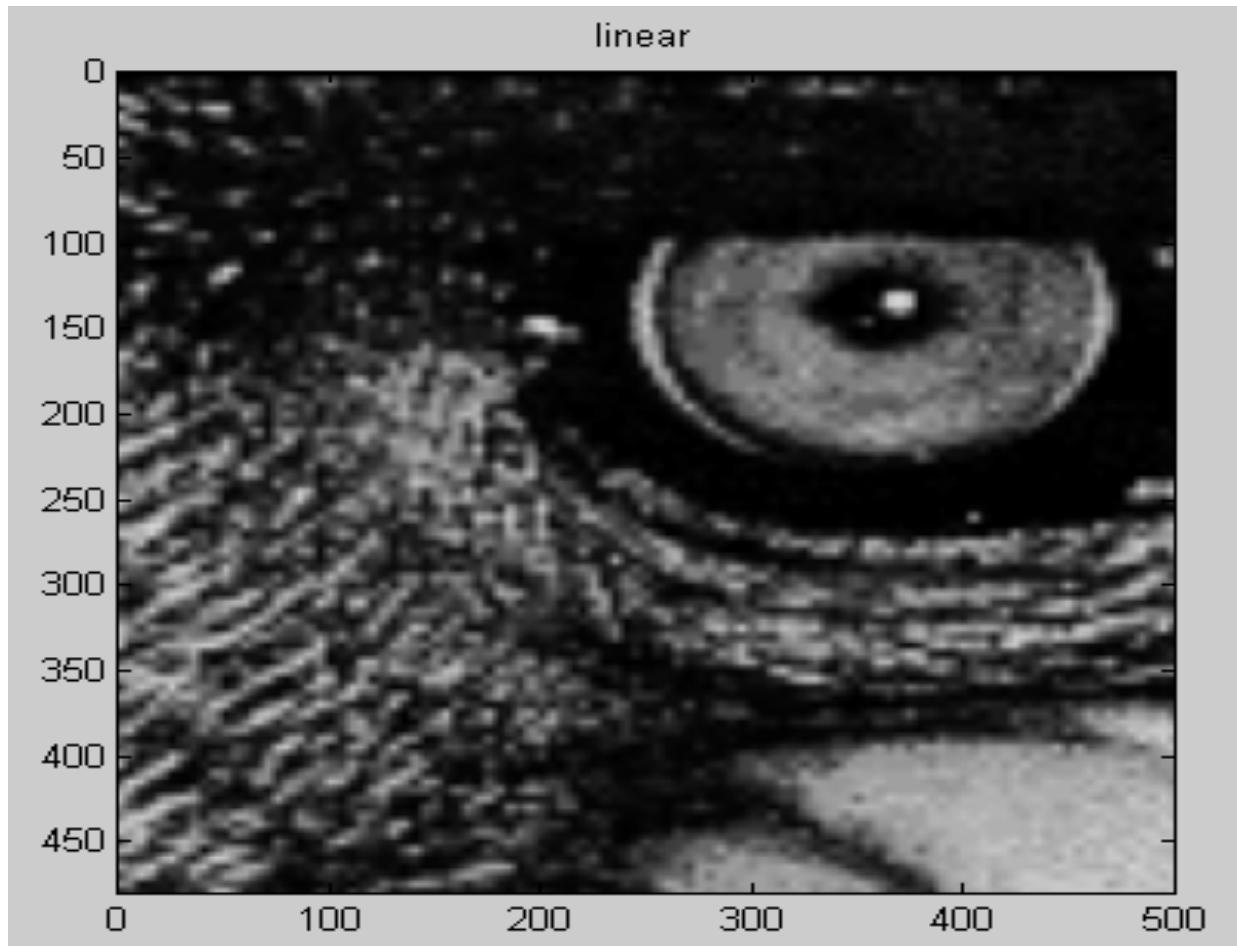
# 影像



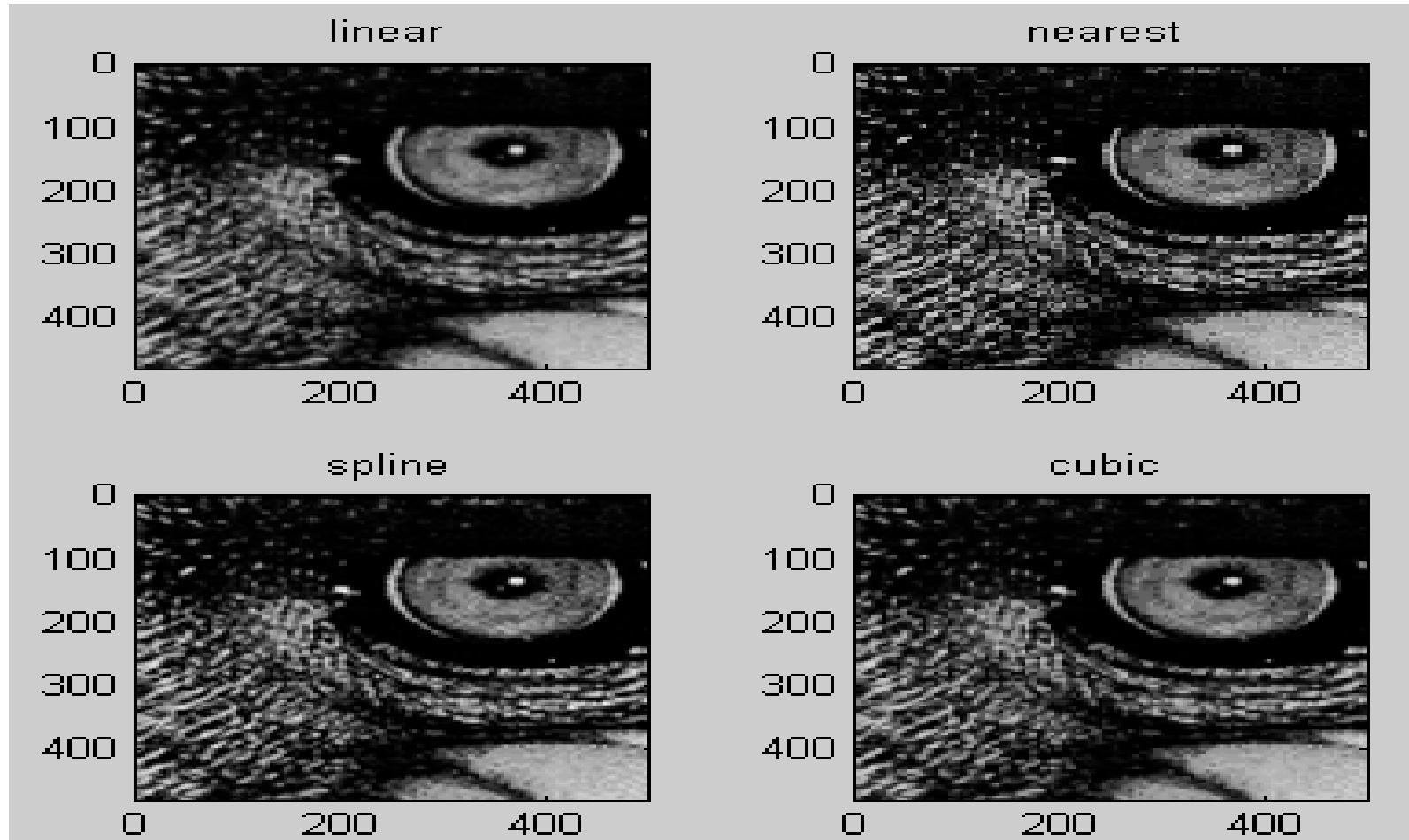
# 影像放大：y方向

- $x_3 = 100:200;$     $x_4 = \text{linspace}(100, 200, 500);$
- $\text{imextend} = \text{interp1}(x_3, \text{double(imextend')},$   
 $x_4, \text{'linear'});$
- $\text{image(imextend');}$     $\text{colormap(gray(256));}$
- $\text{axis([0 500 0 480]);}$     $\text{axis square;}$   
 $\text{title('linear');}$

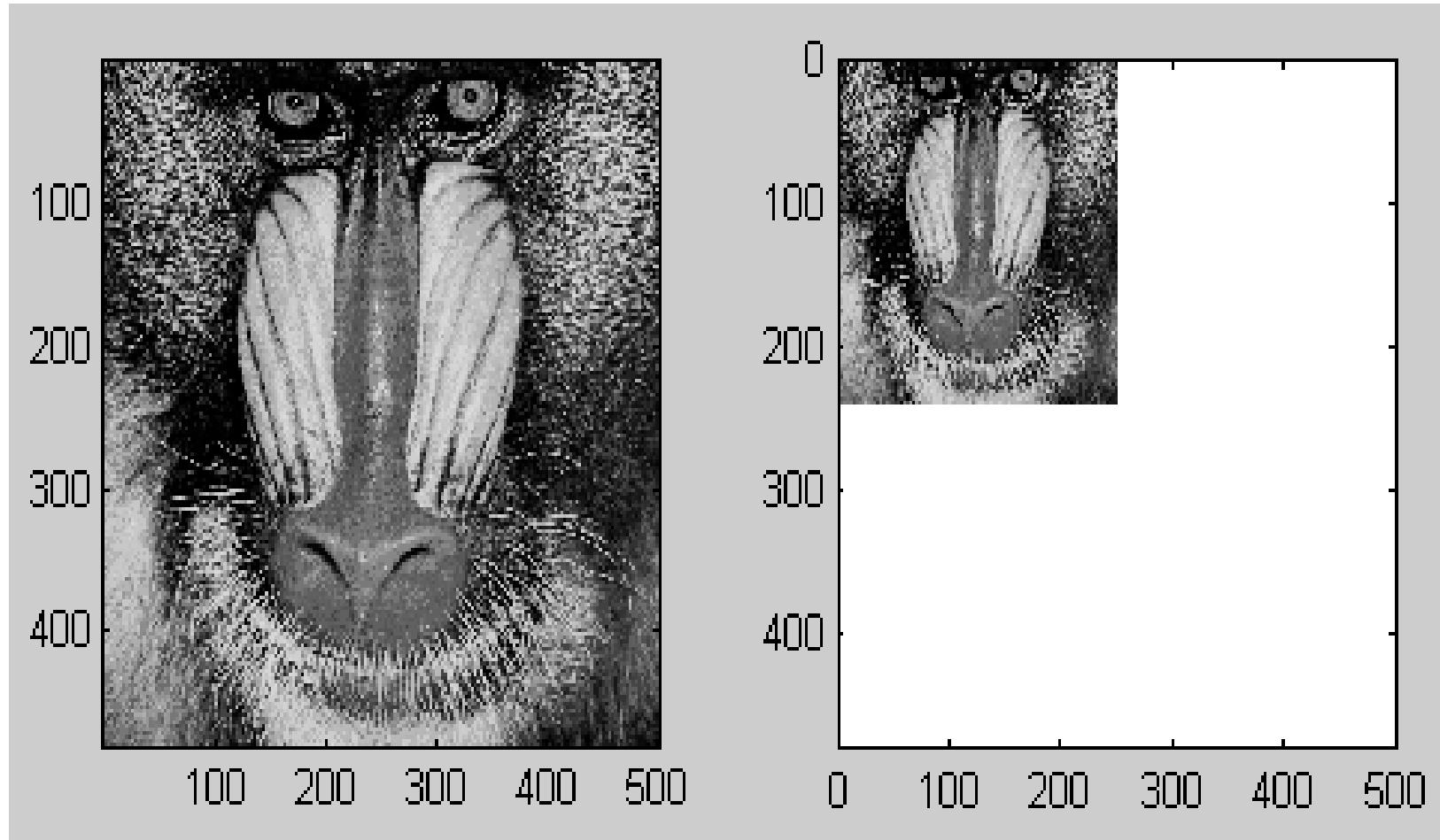
# 影像放大：y方向



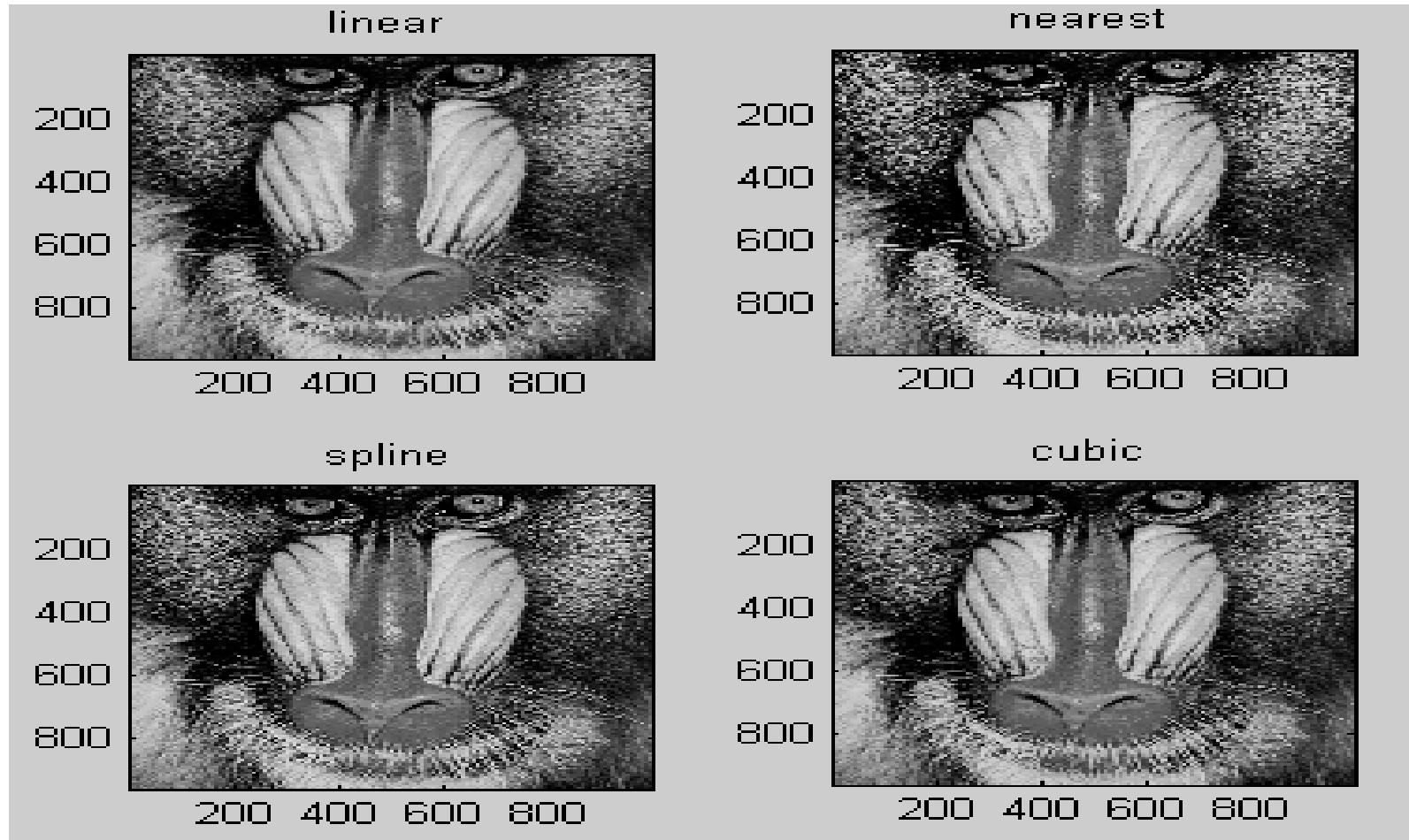
# 影像放大5倍



# 影像縮小



# 影像縮小



# 影像縮小：插值y方向

- `x1 = 1:size(xp,1);`
- `x2 = 1:0.25:size(xp,1);`
- `imextend = interp1(x1, double(xp), x2, 'linear');`
- `image(imextend);`
- `colormap(gray(256));`

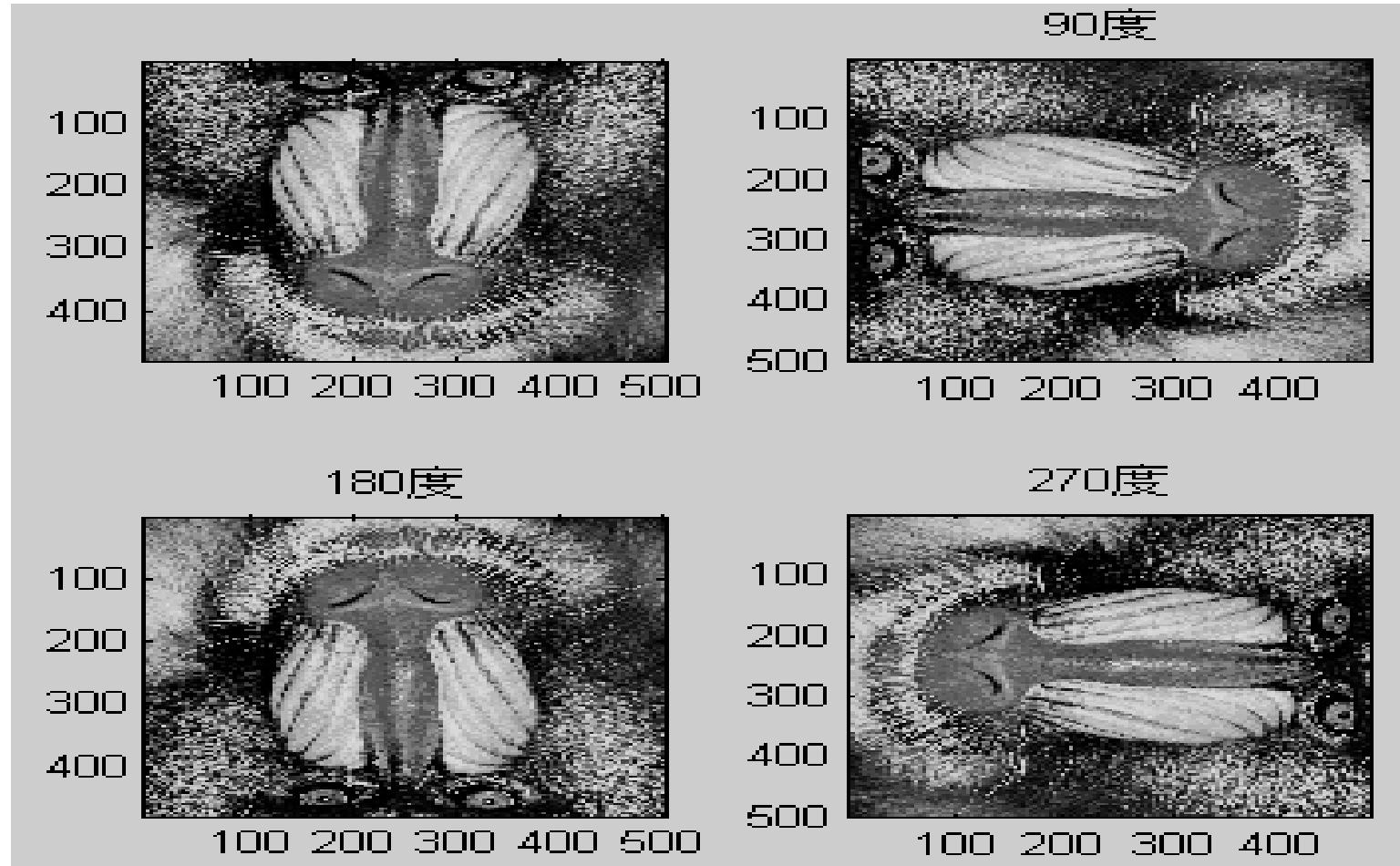
# 影像旋轉

- **flipud**
- **fliplr**

例如

- **90 °** :  $xp = \text{flipud}(x');$
- **180 °** :  $xp = \text{flipud}(x);$
- **270 °** :  $xp = \text{fliplr}(x');$

# 影像旋轉



# 影像濾波器

- 指令:fspecial ('濾波器')
  - average: 平均濾波器
  - sobel: 水平邊緣強化濾波器
  - gaussian: Gaussian低通濾波器
  - log: Laplacian之Gaussian低通濾波器
  - prewitt: Prewitt形式之水平邊緣強化濾波器
  - laplacian: 二維之Laplacian低通濾波器
  - unsharp: 對比強化濾波器
  - motion: 晃動模糊濾波器

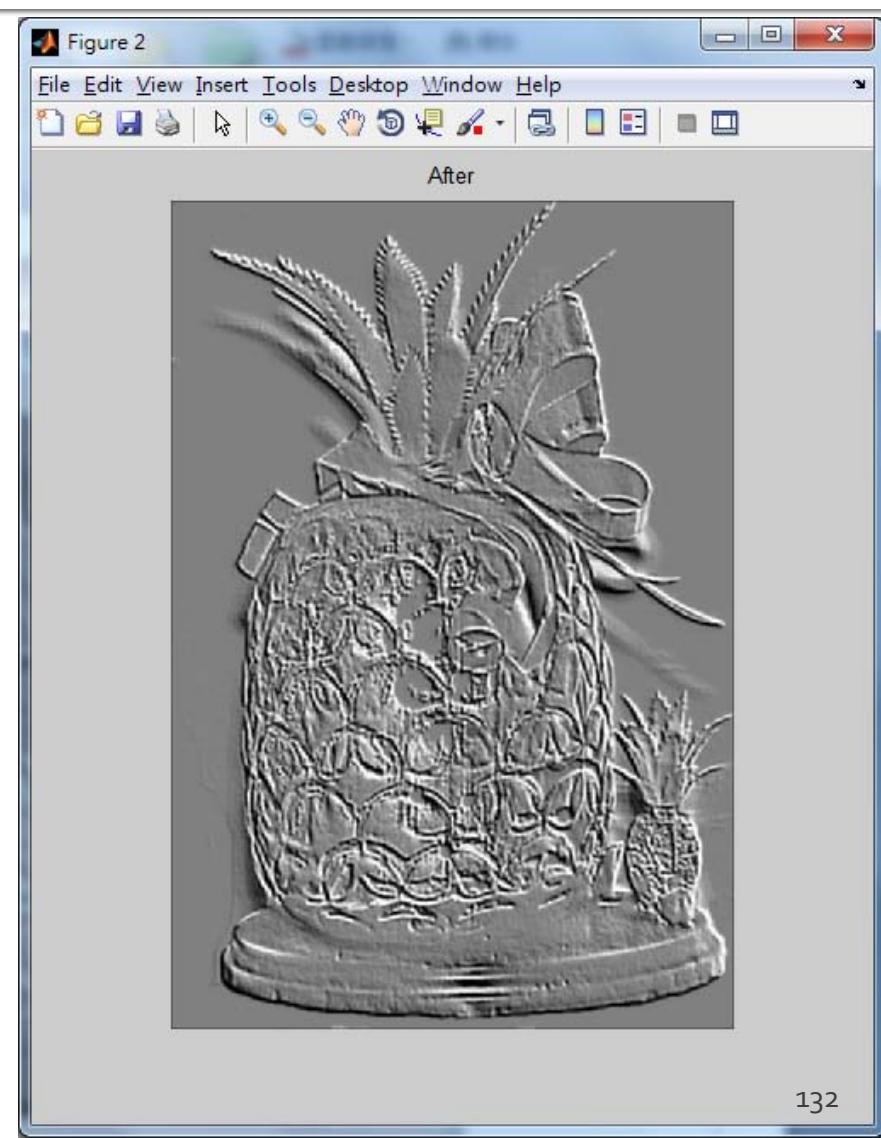
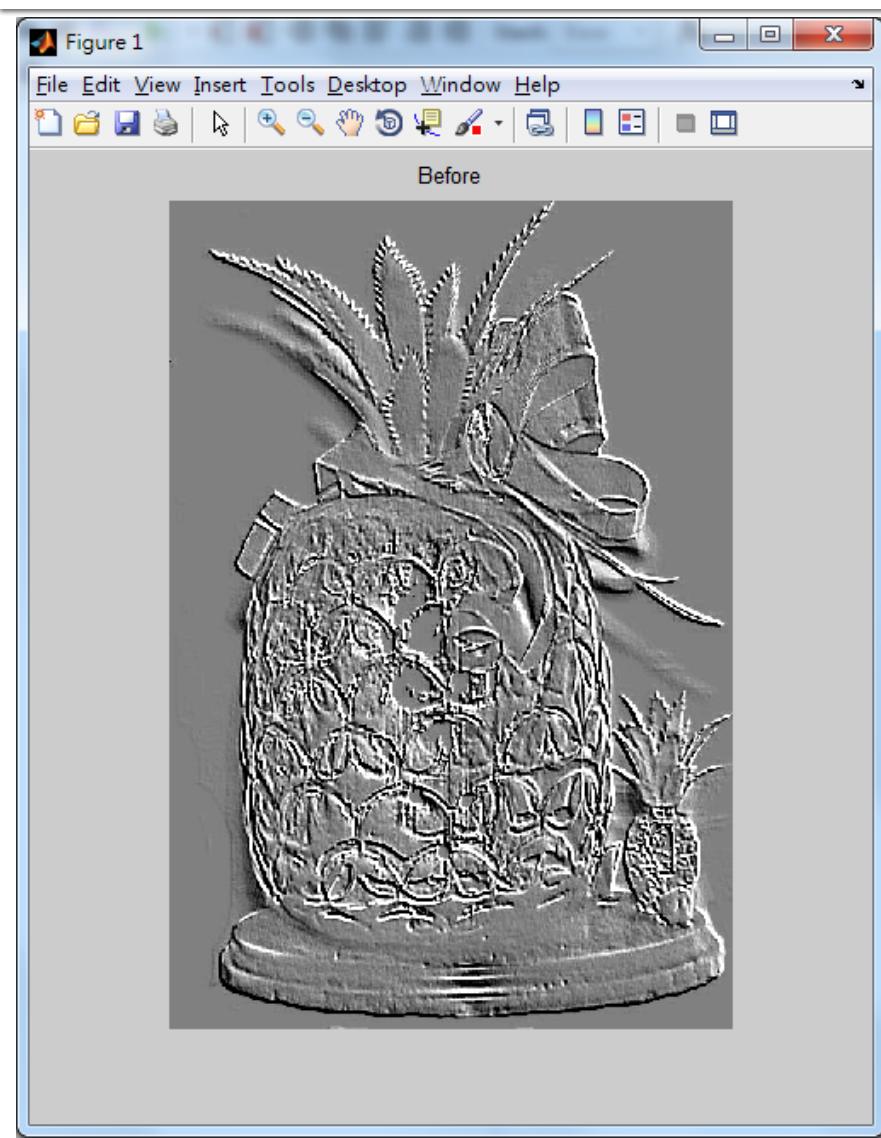
# 影像濾波器

- 指令: `imfilter` (影像檔, 濾波器形式)

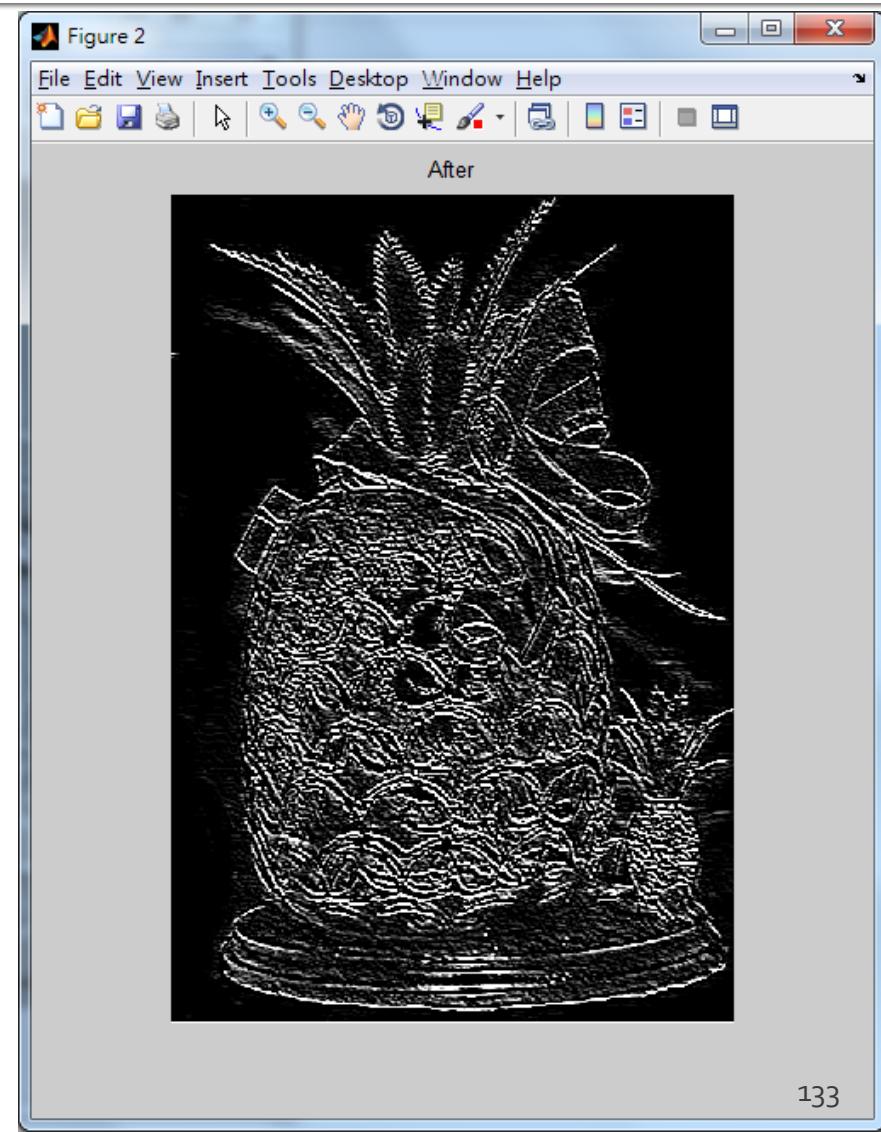
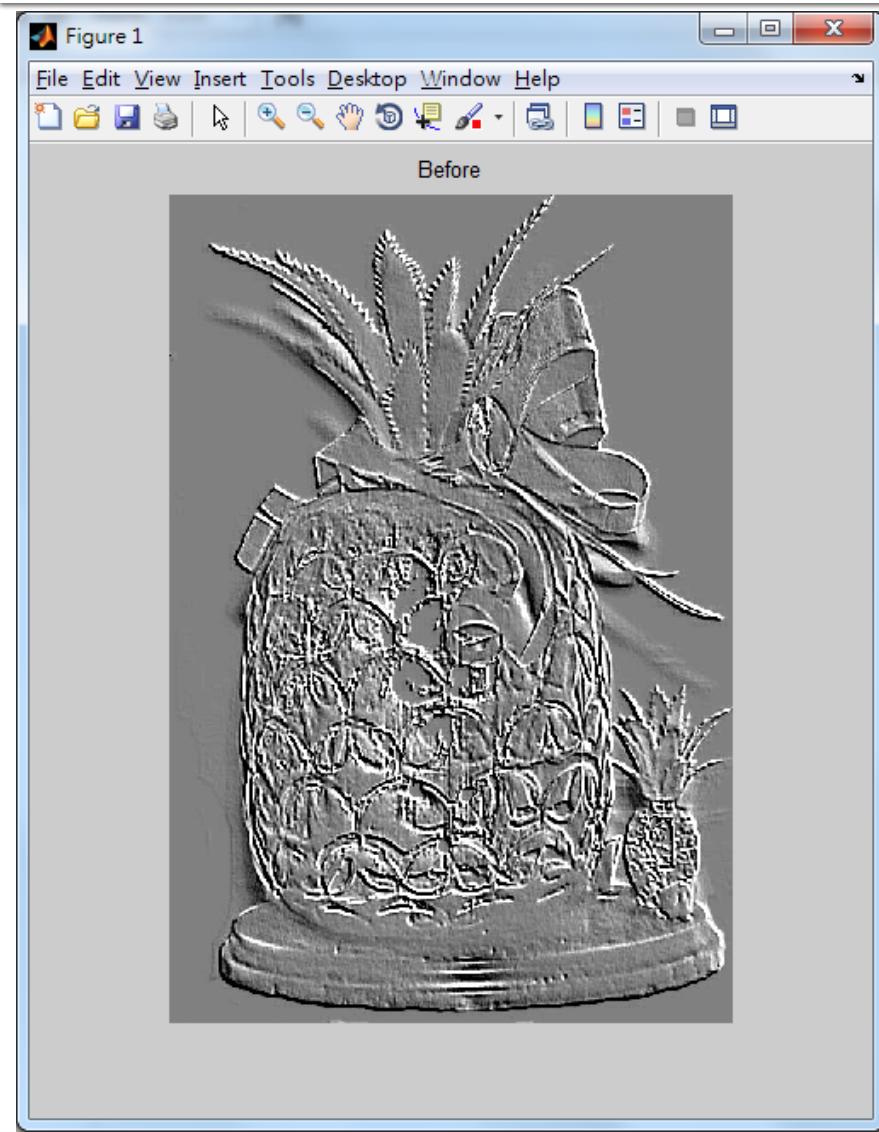
- EX:

- `j = imread('test.bmp');`
    - `h = fspecial('prewitt')`
    - `j1 = imfilter(j, h);`
    - `figure`
    - `imshow(j);`
    - `title('Before')`
    - `figure`
    - `imshow(j1);`
    - `title('After')`

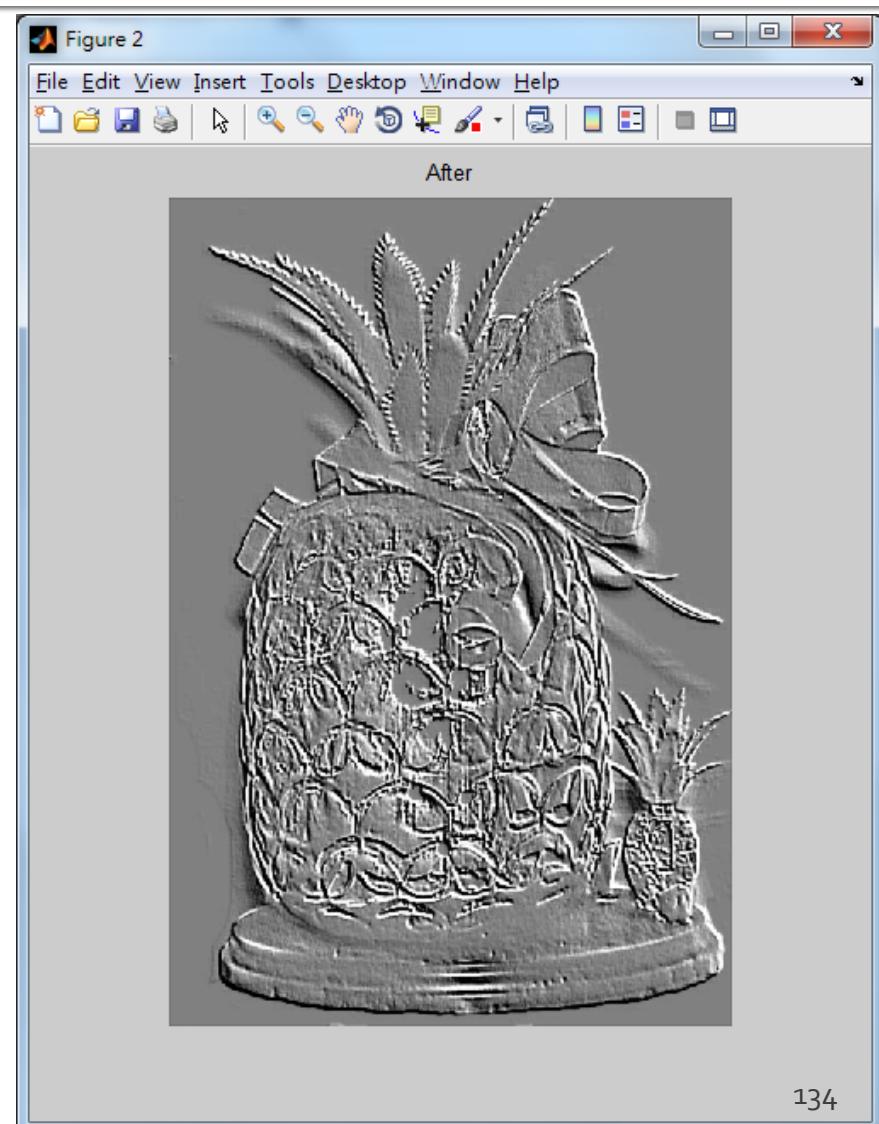
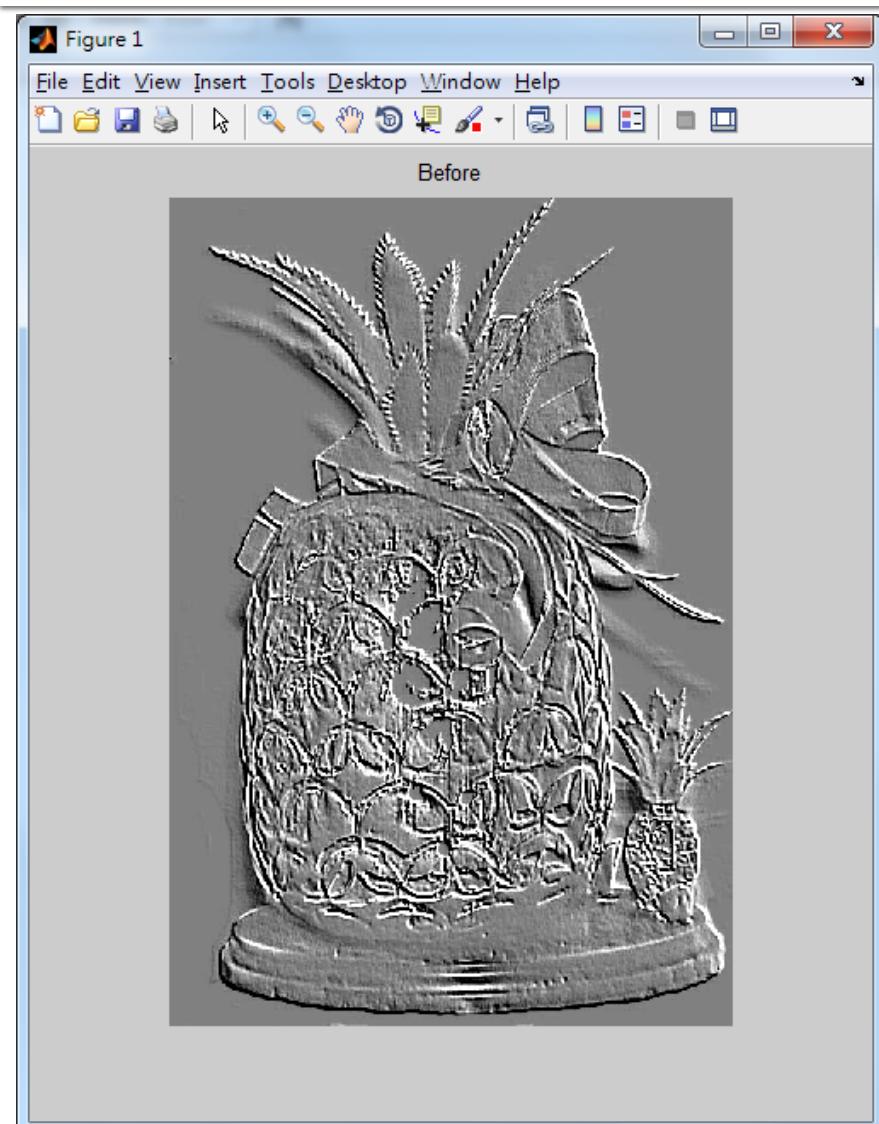
# Average



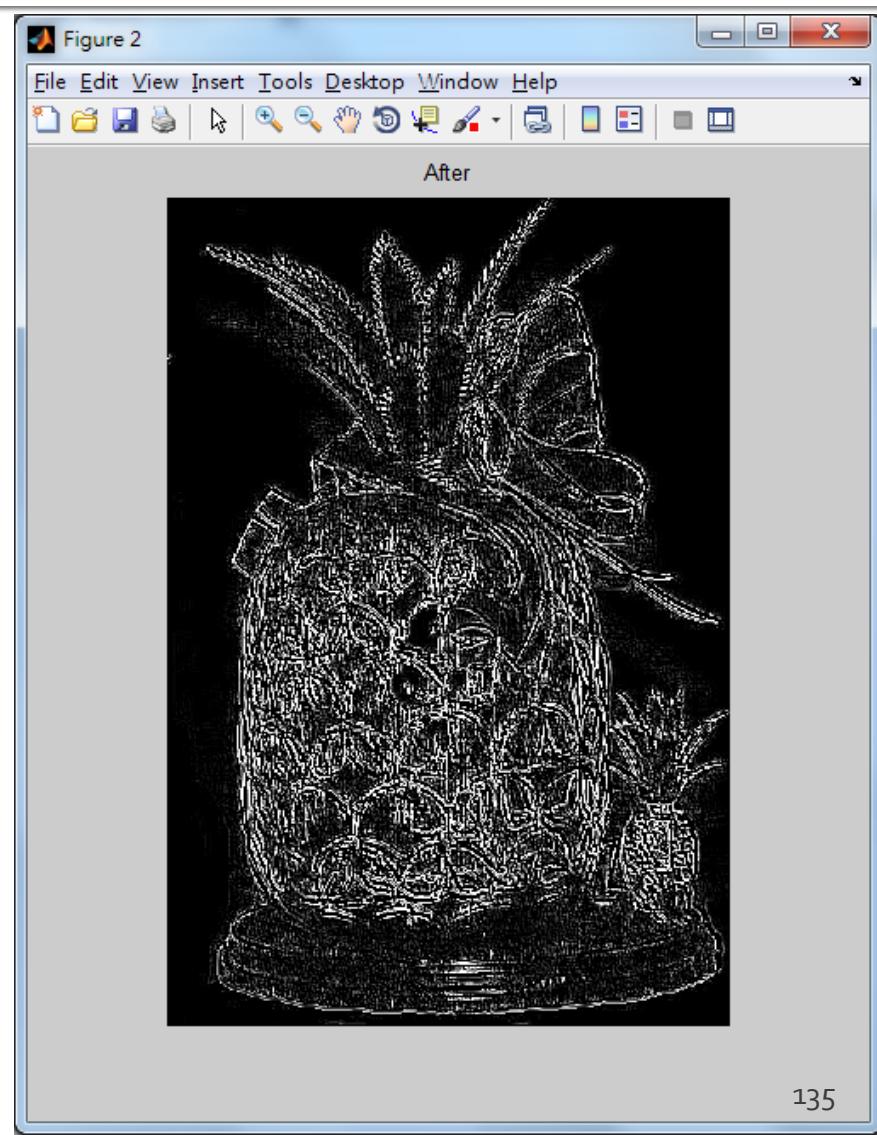
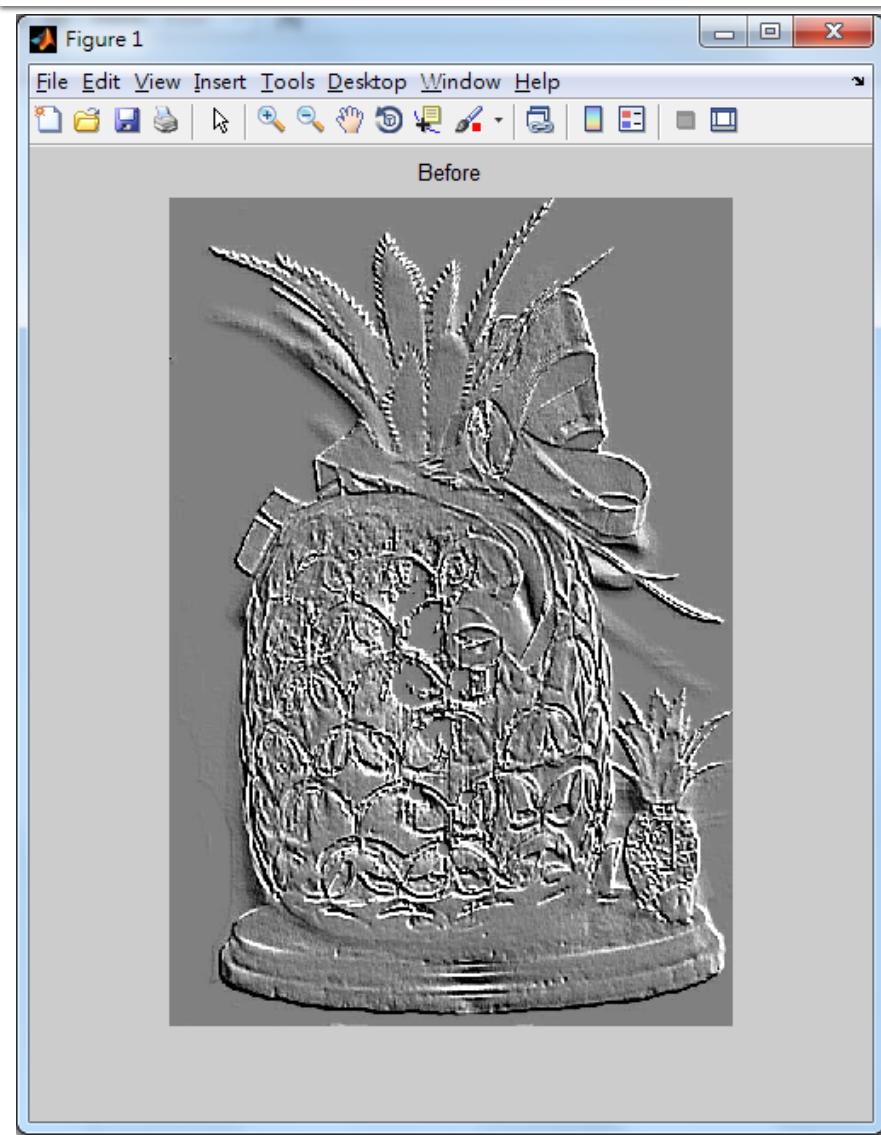
# Sobel



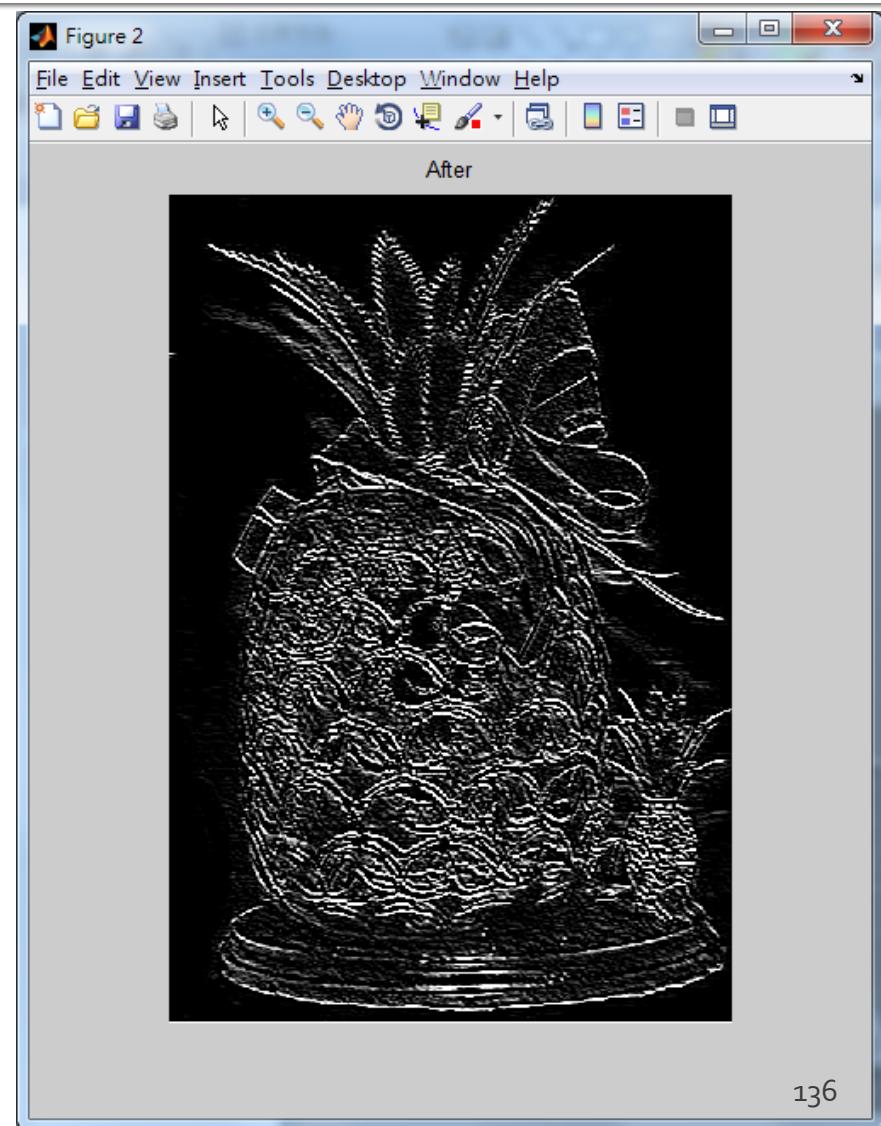
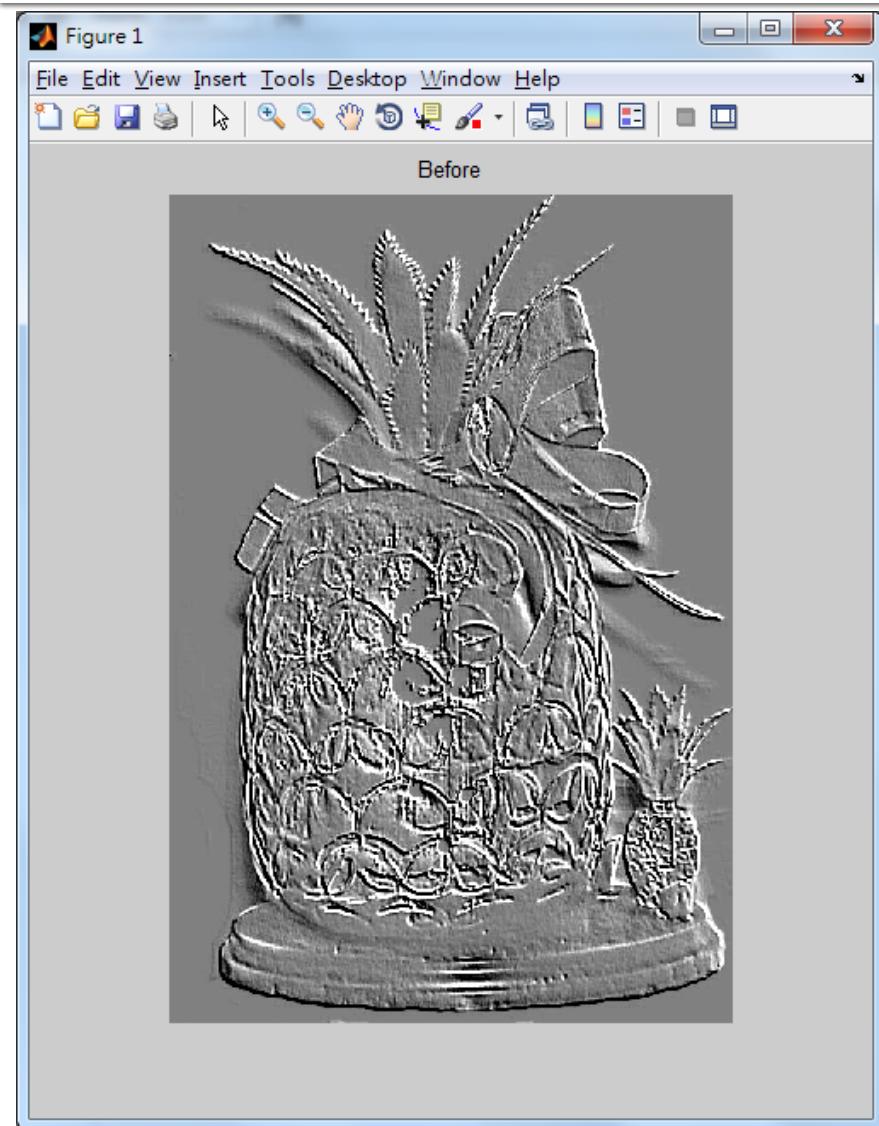
# Gaussian



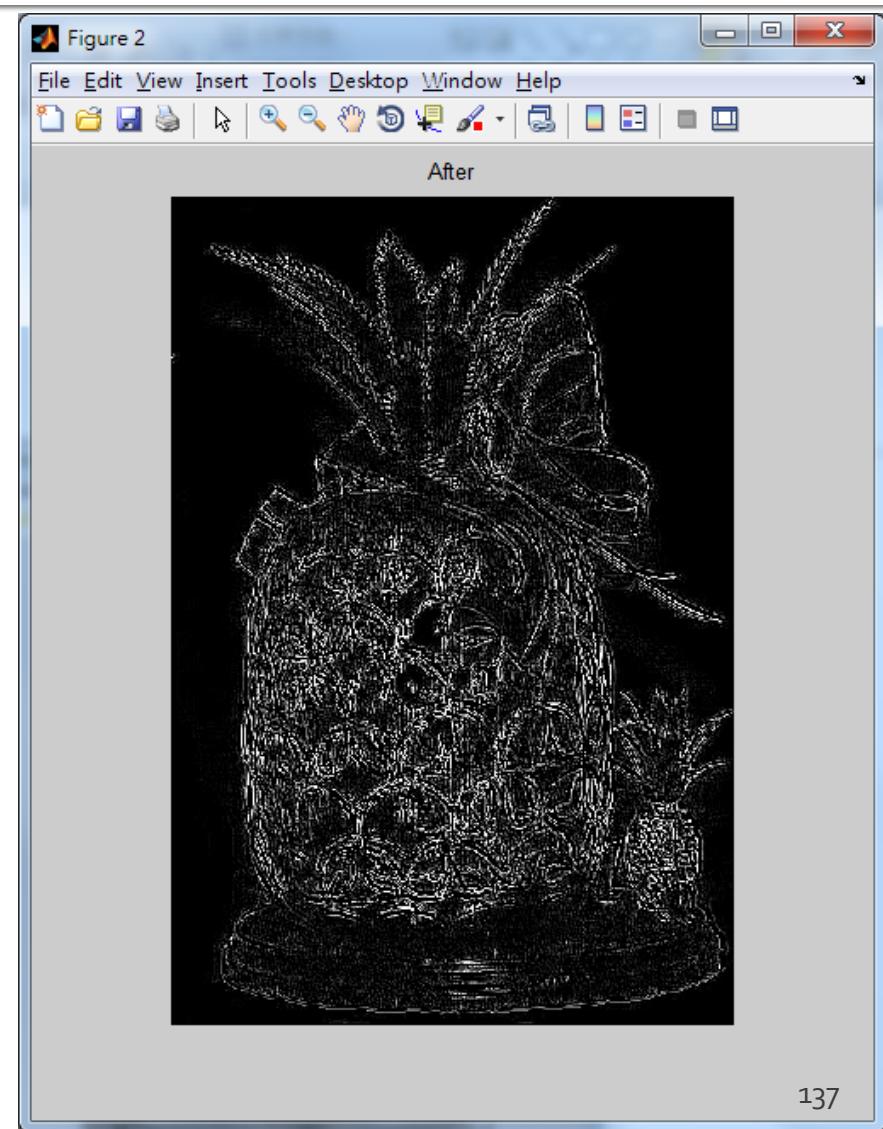
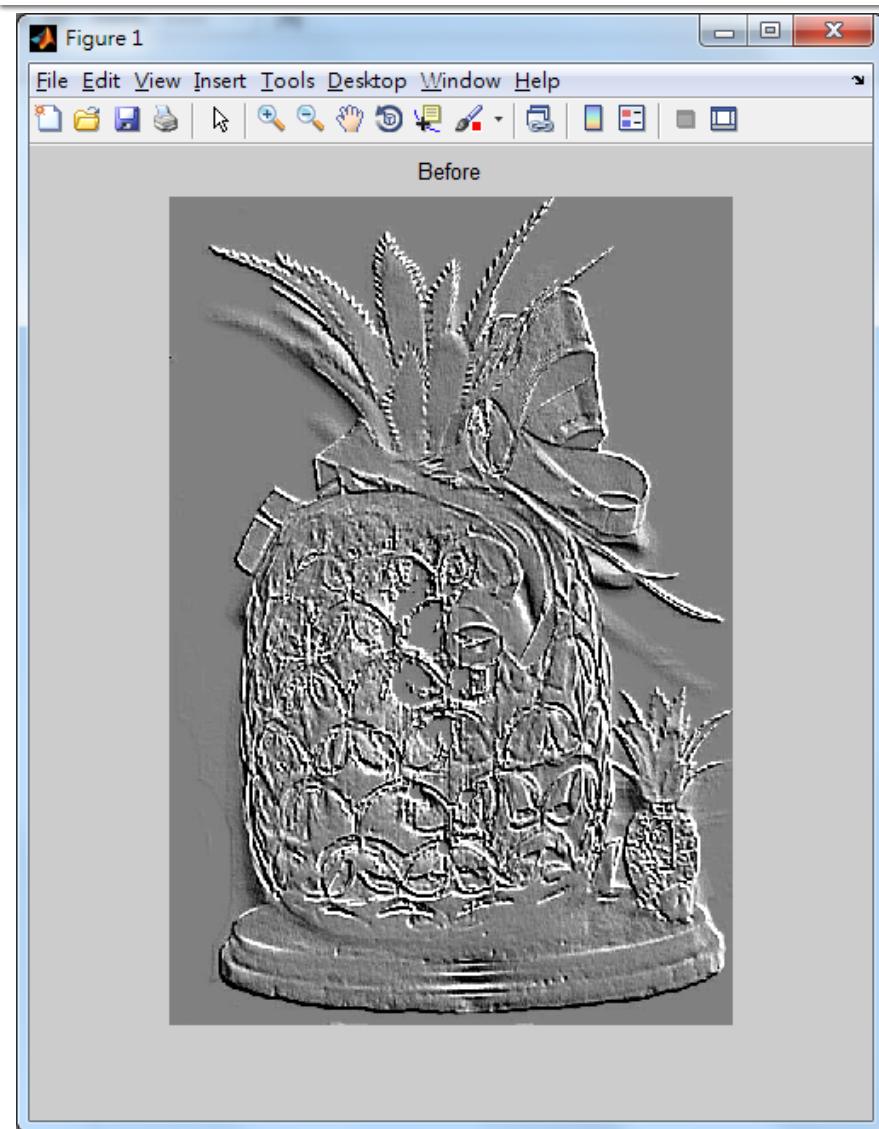
# Log



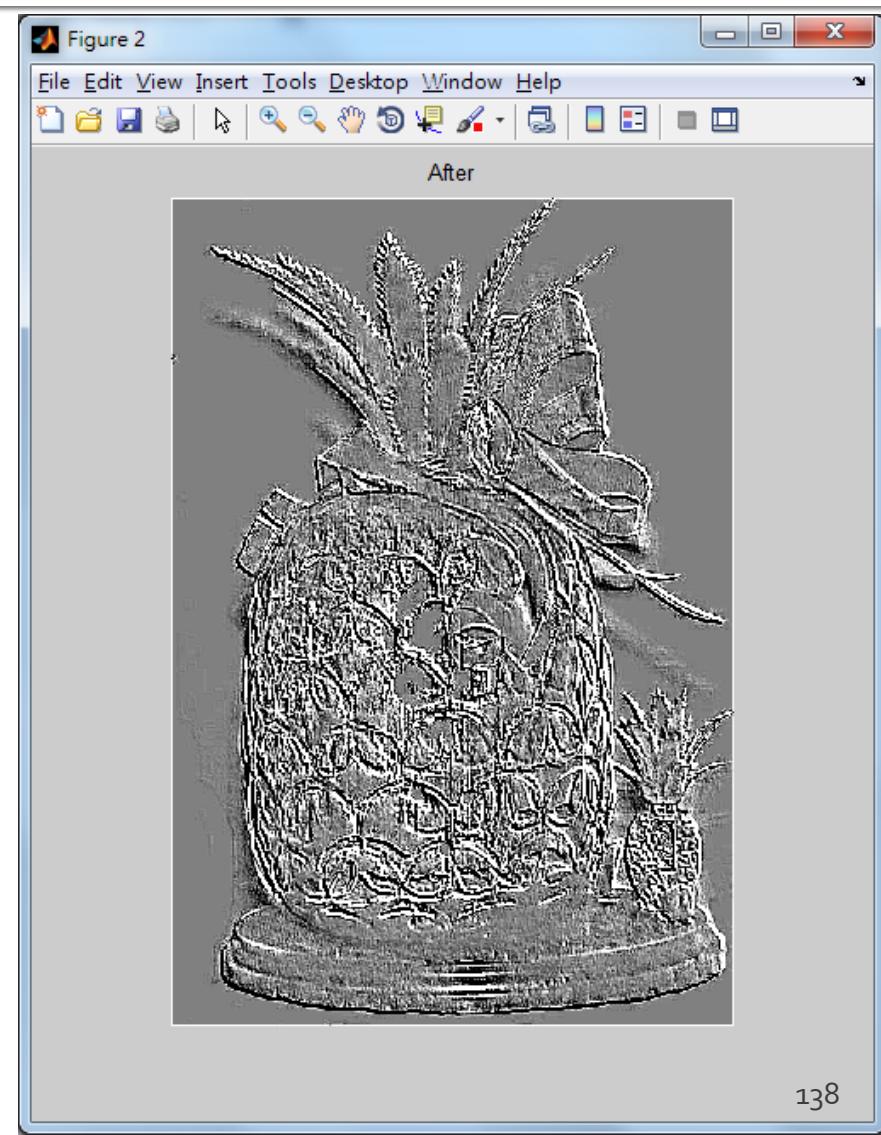
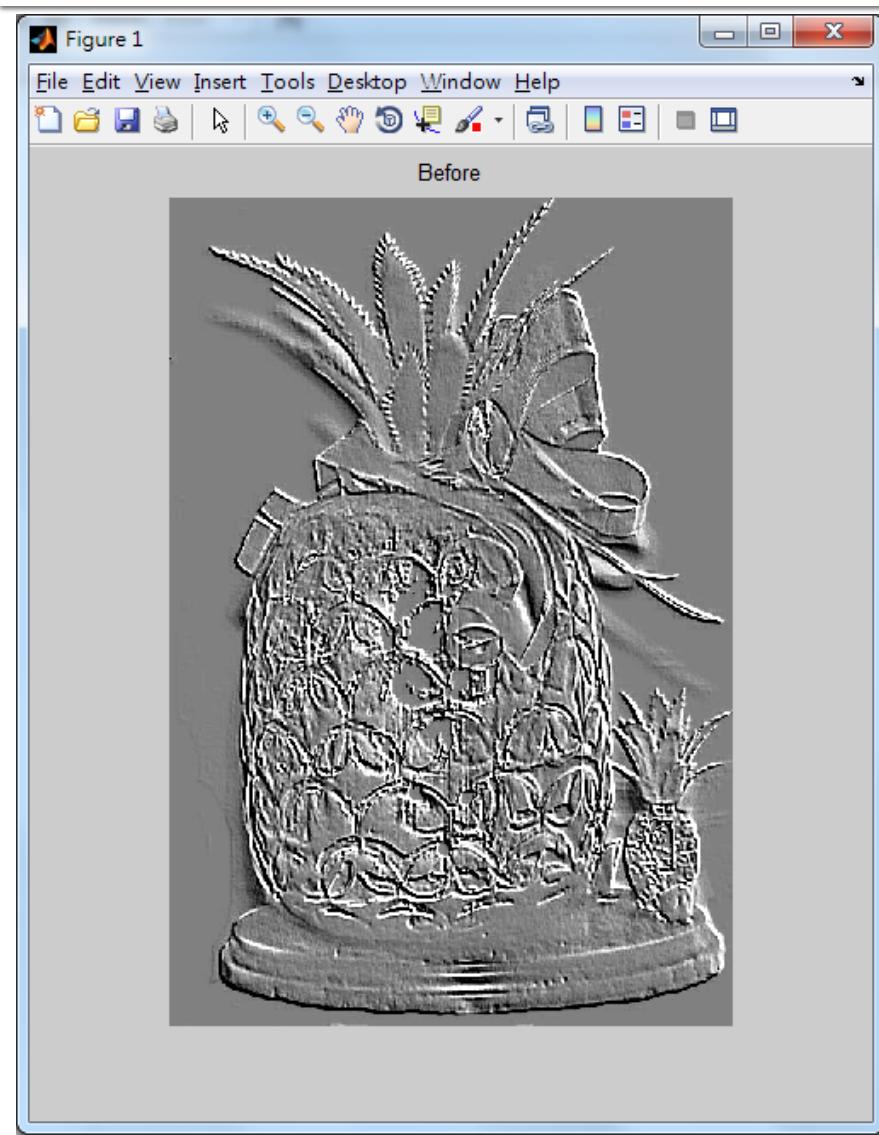
# Prewitt



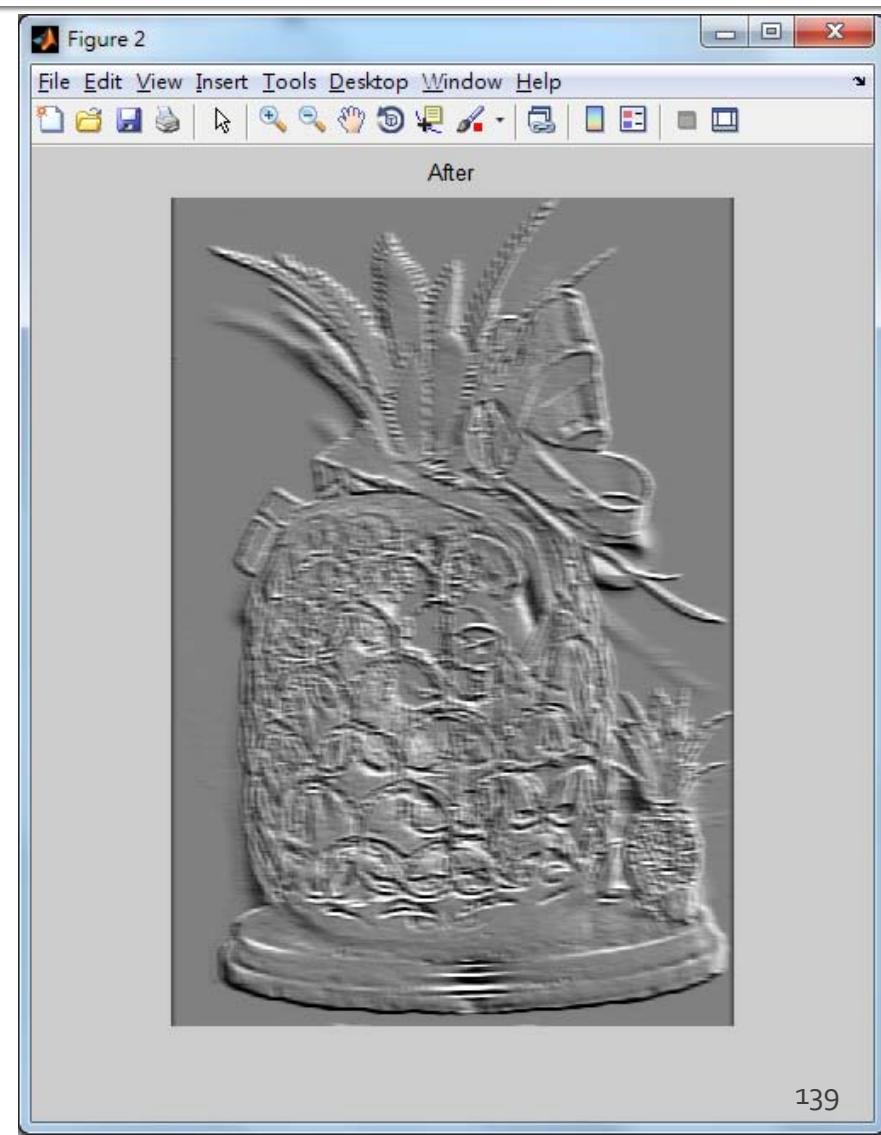
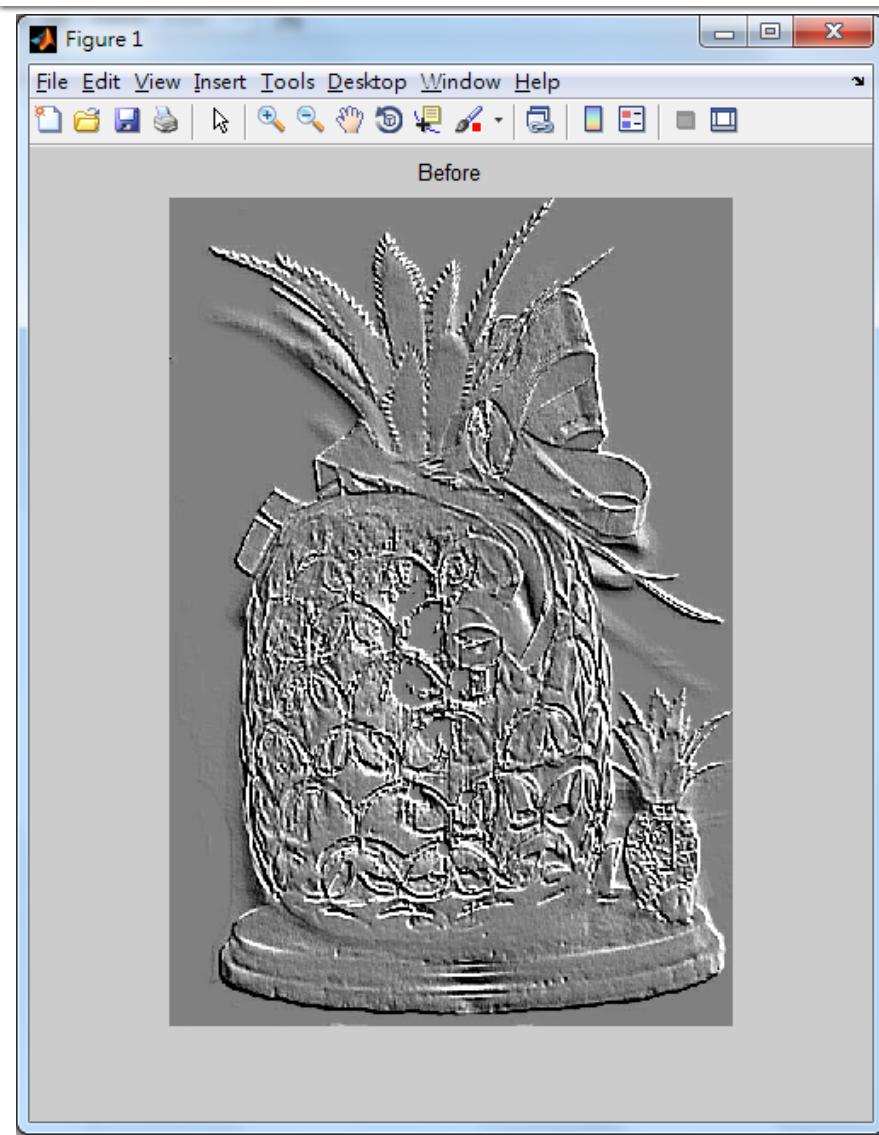
# Laplocian



# Unsharp



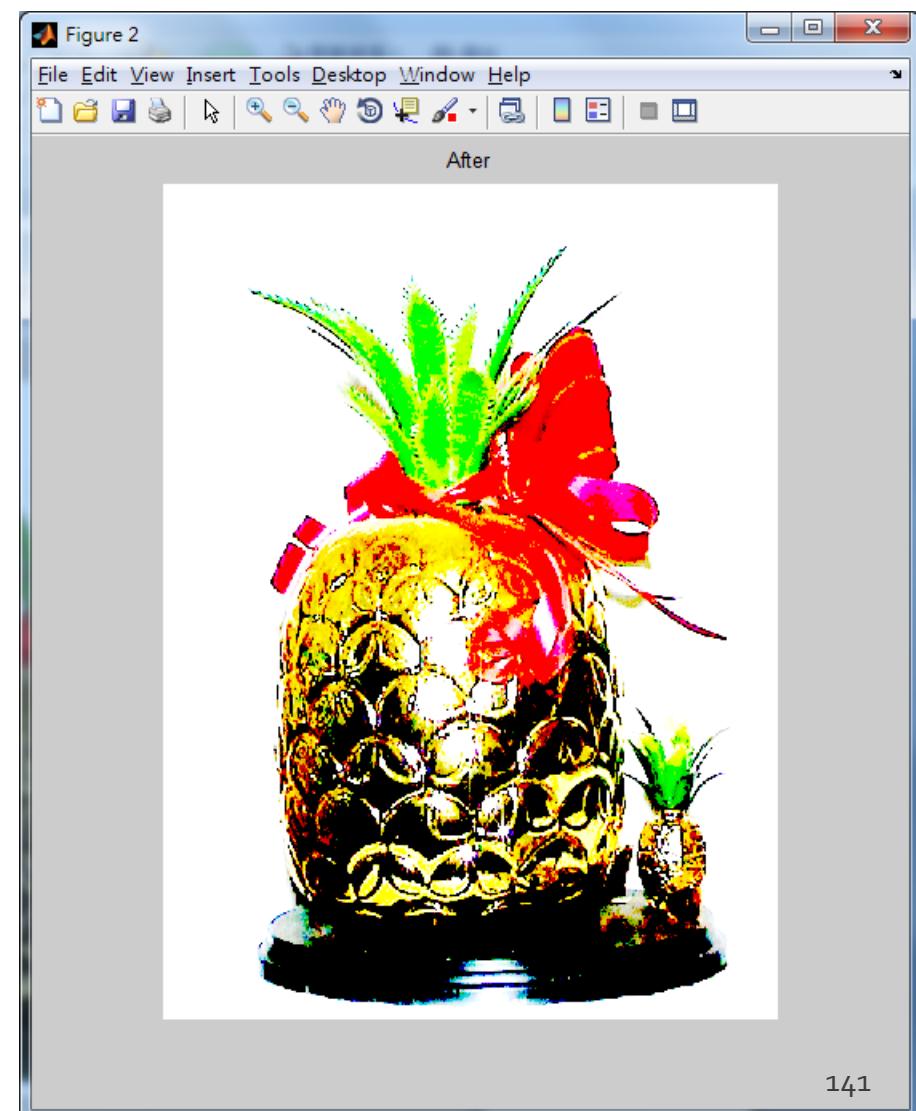
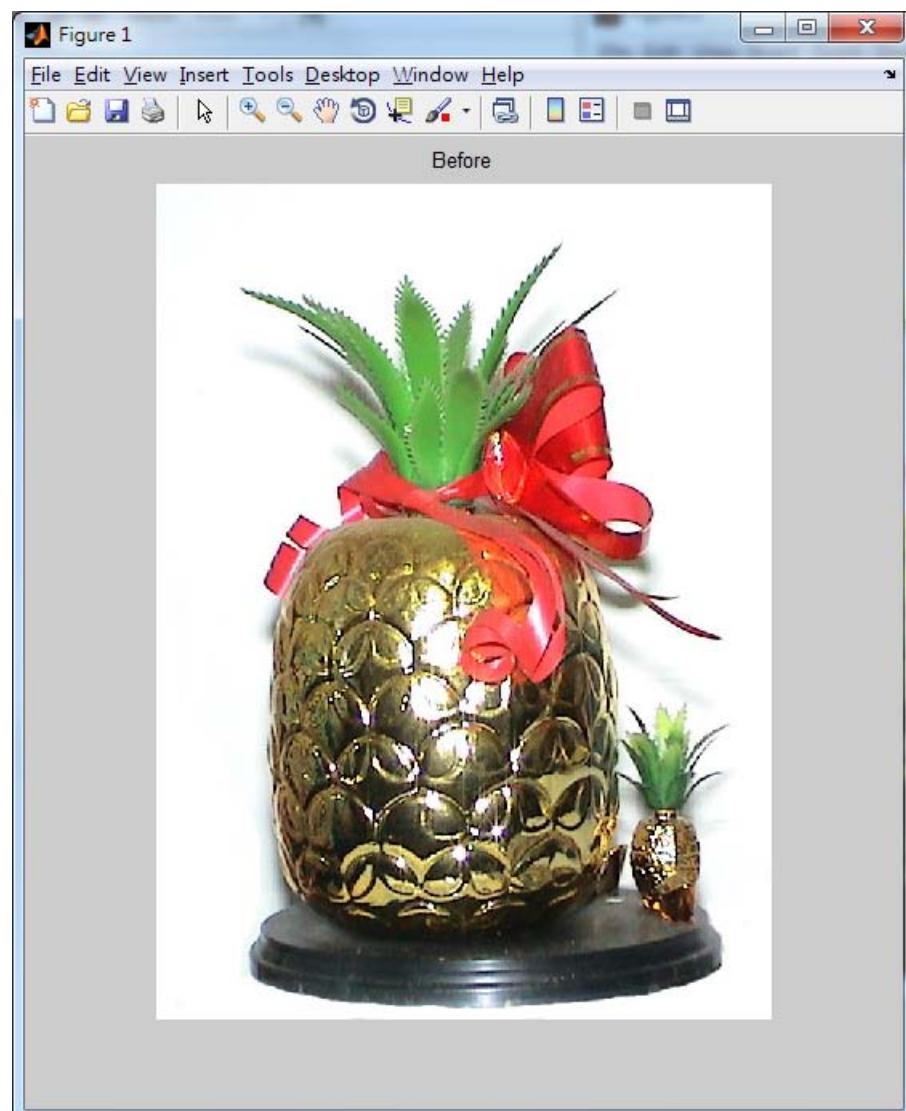
# Motion



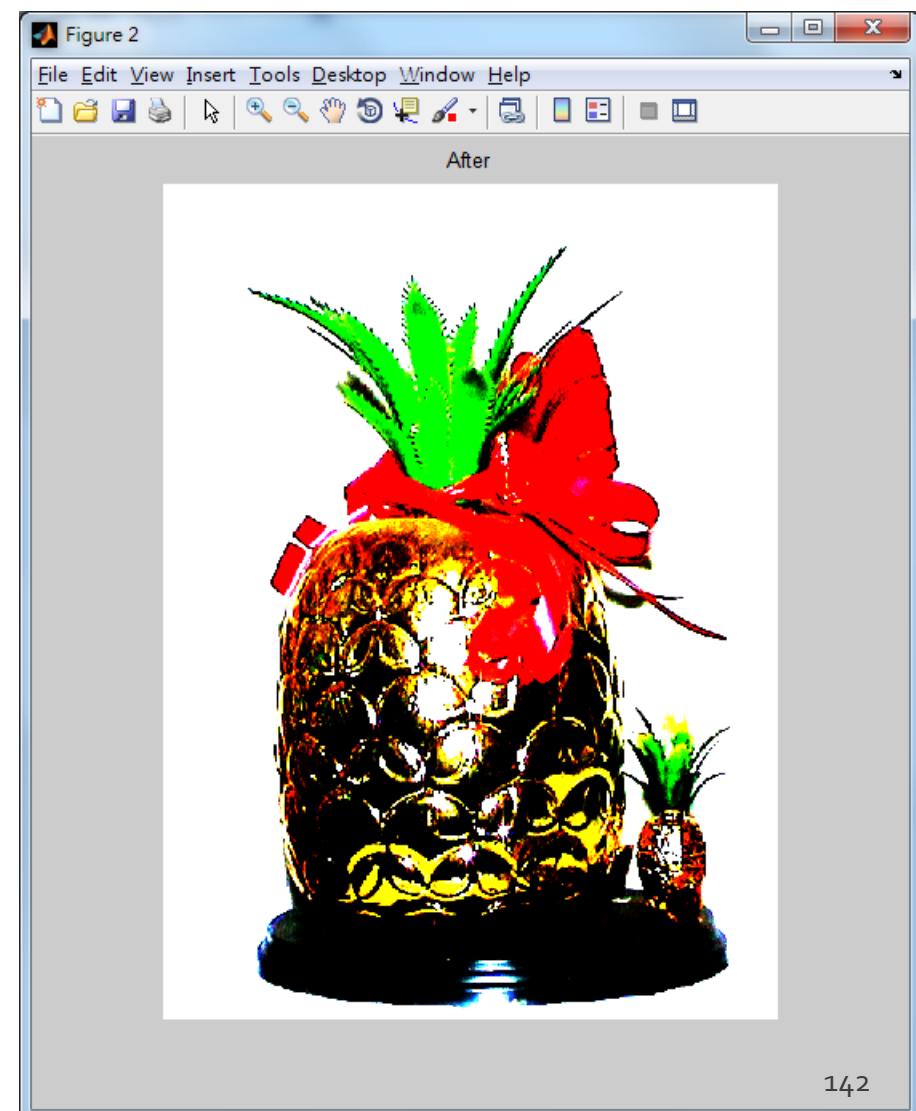
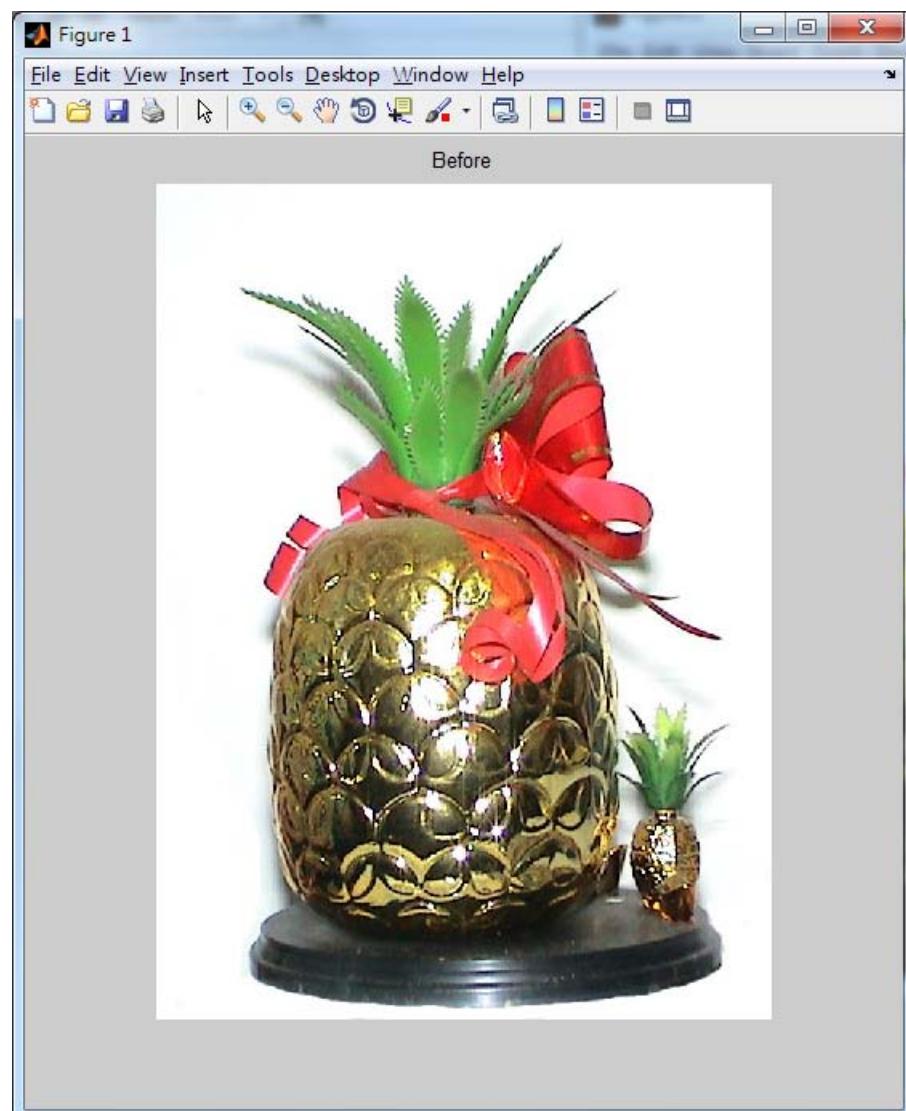
# 影像加強

- 指令: imadjust (影像檔,[低輸入 高輸入],[低輸出 高輸出],gamma)
  - [低輸入 高輸入]: 限定影像色度擷取的範圍
  - Gamma: >1變暗; 越小越亮
  - EX:
    - j=imread('pineapple.jpg');
    - j1=imadjust(j,[0.4 0.6],[],0.15);
    - figure
    - imshow(j);
    - title('Before')
    - figure
    - imshow(j1);
    - title('After')

# 影像加強 (Gamma=0.15)



# 影像加強 (Gamma=3)



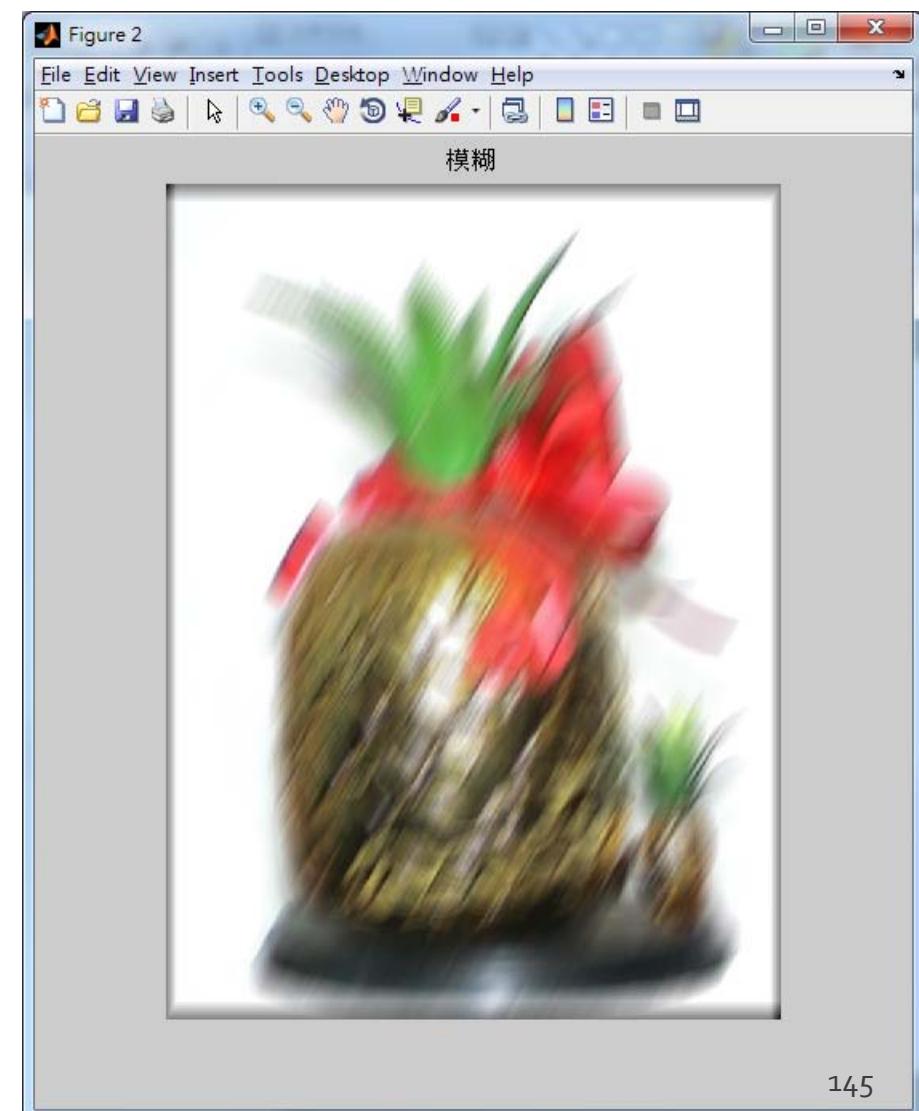
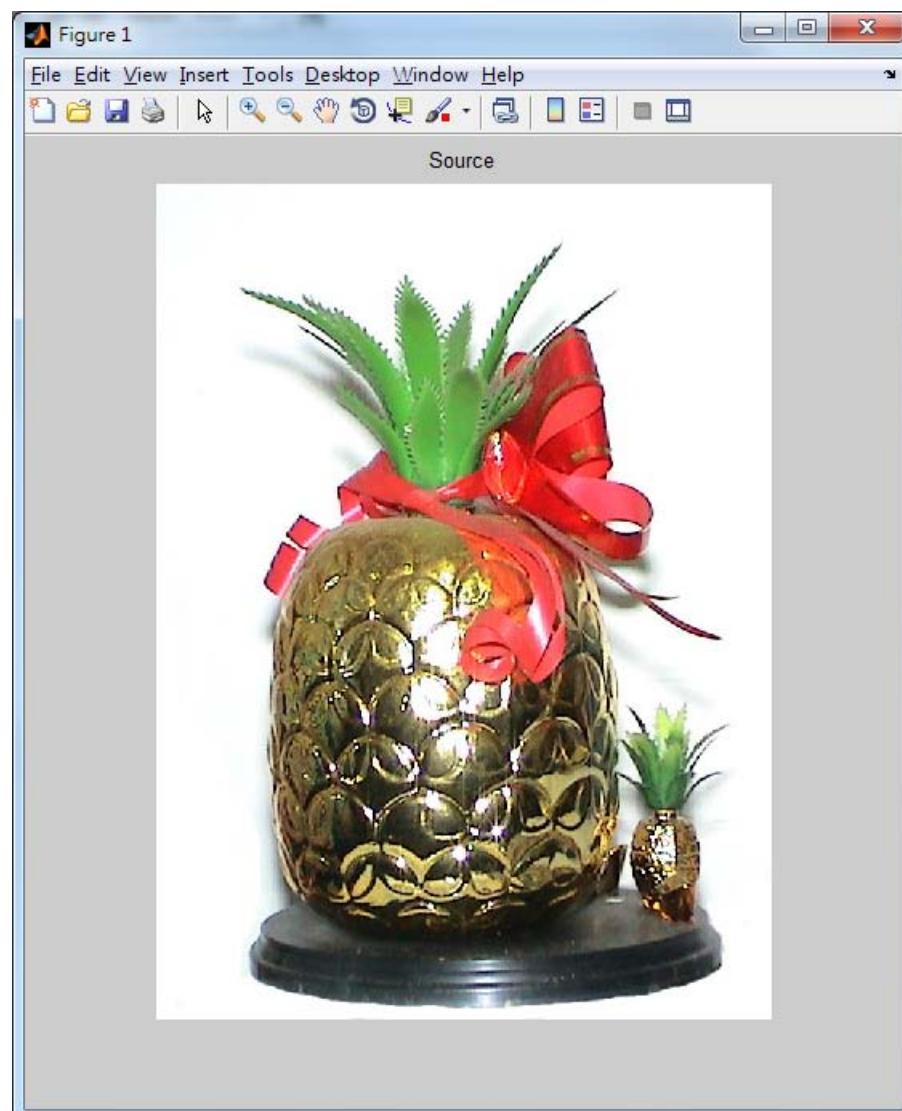
# 影像移動模糊及還原

- 指令：
  - Deconvwnr: 使用Wiener濾波器
  - Deconvreg: 使用一般濾波器
  - Deconvlucy: 使用Lucy-Richardson濾波器
  - Deconvblind: 使用Blind濾波器
- 影像模糊一定是和**長度**和**角度**有關

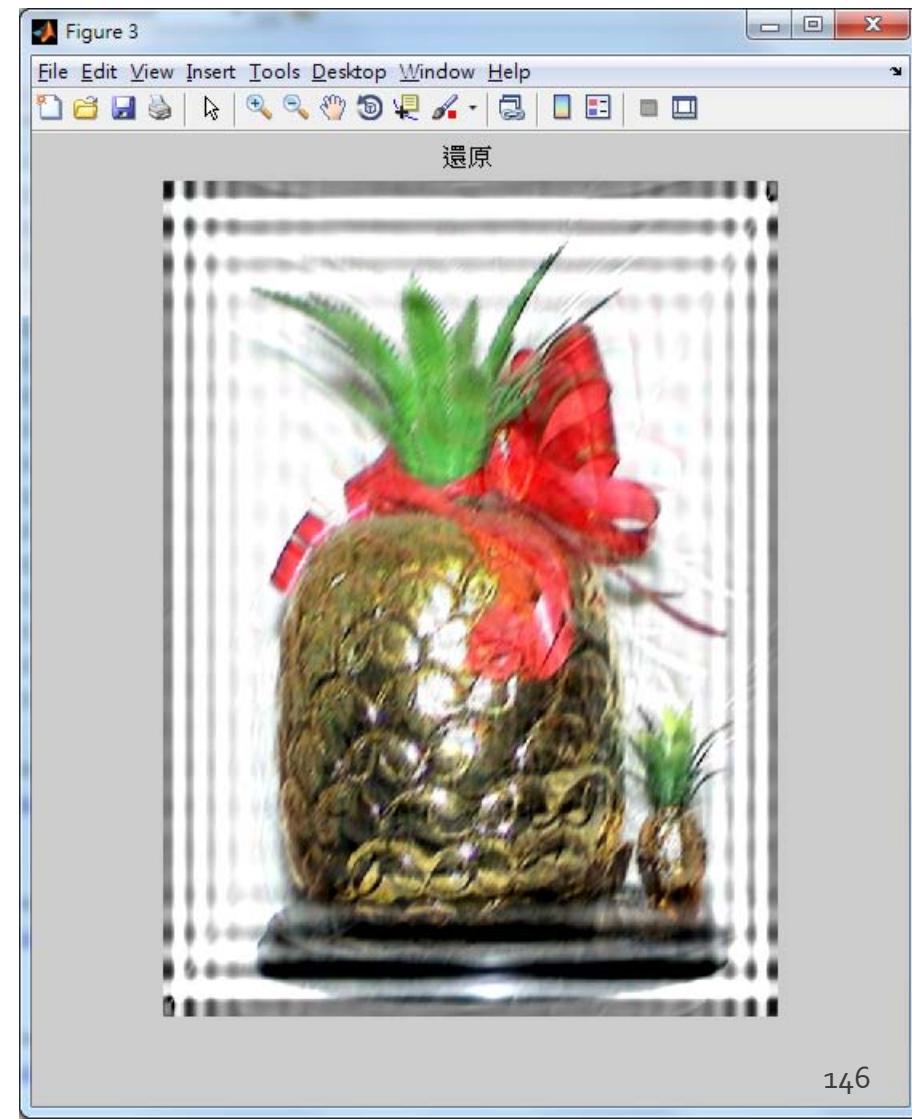
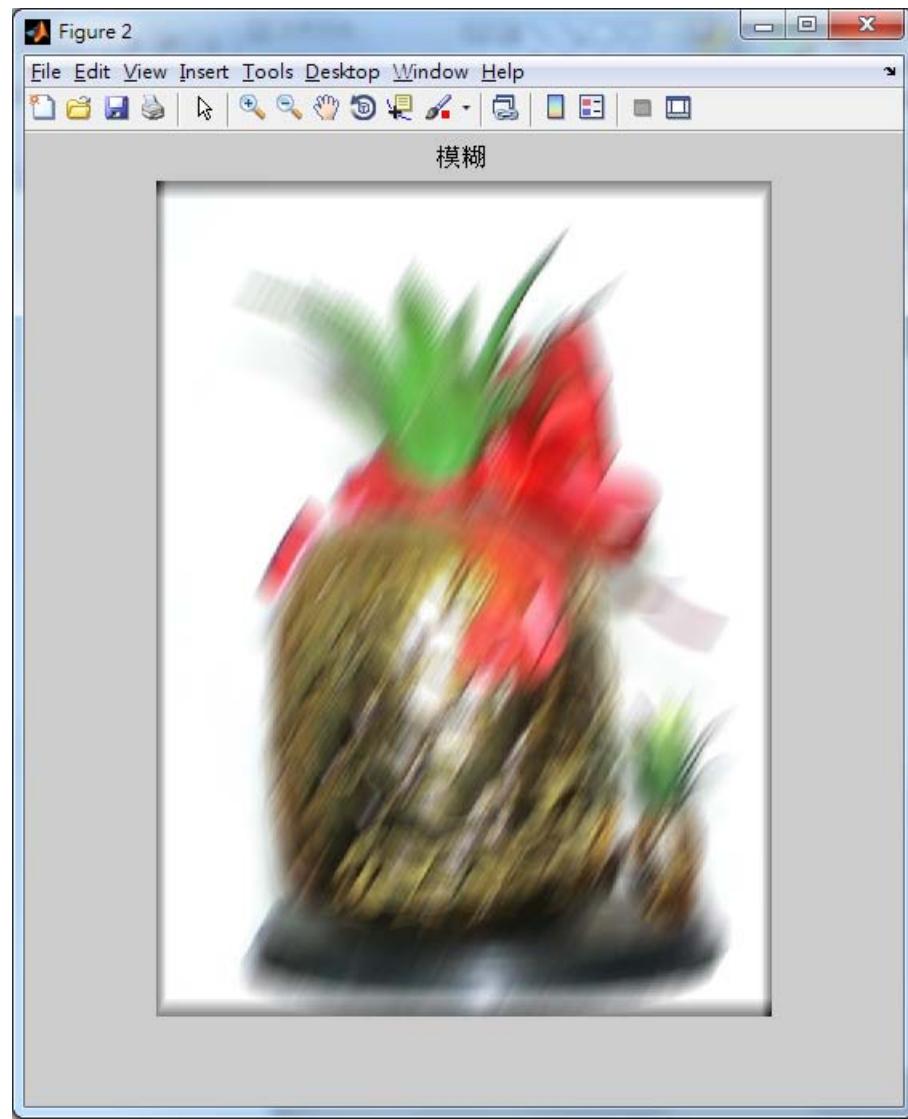
# 影像移動模糊及還原

```
■ j=imread('pineapple.jpg');
■ h = fspecial('motion',40,60)
■ j1=imfilter(j,h);
■ j2 = Deconvblind(j1,h);
■ figure
■ imshow(j);
■ title('Source')
■ figure
■ imshow(j1);
■ title('模糊')
■ figure
■ imshow(j2);
■ title('還原')
```

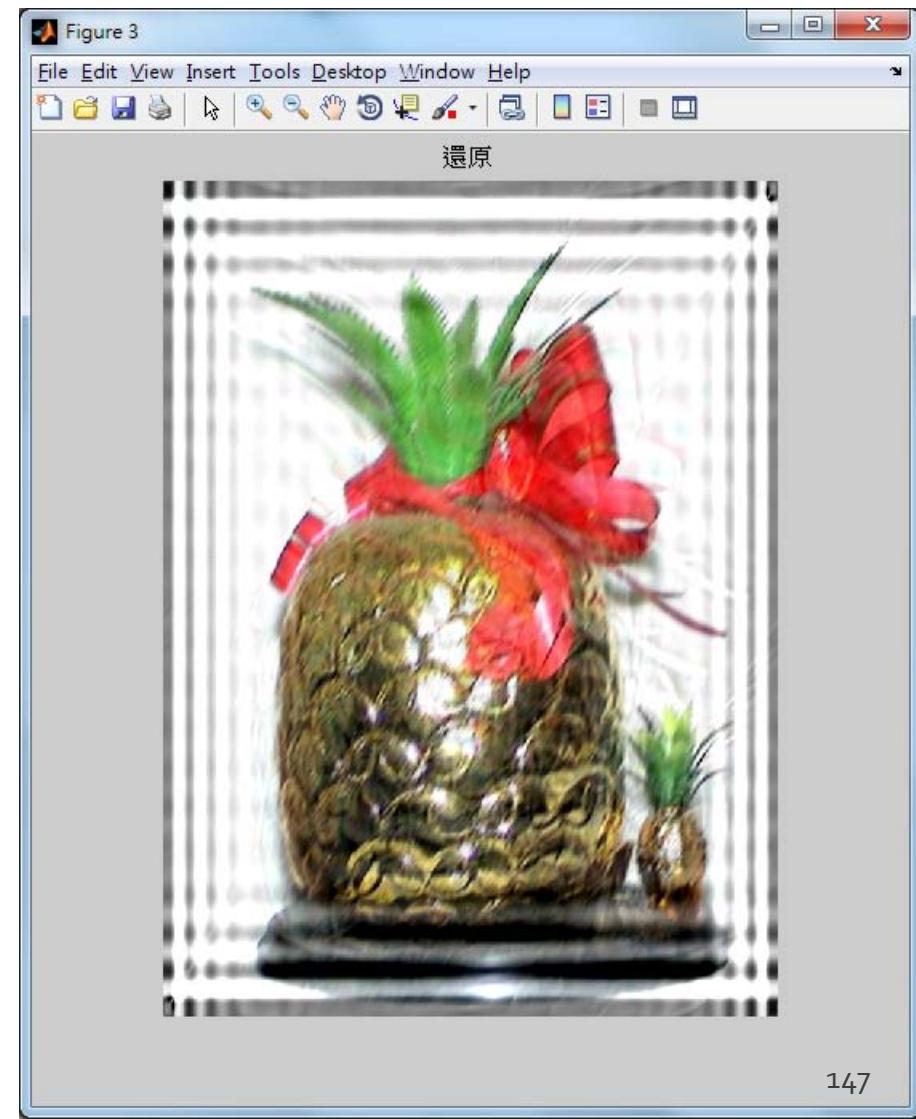
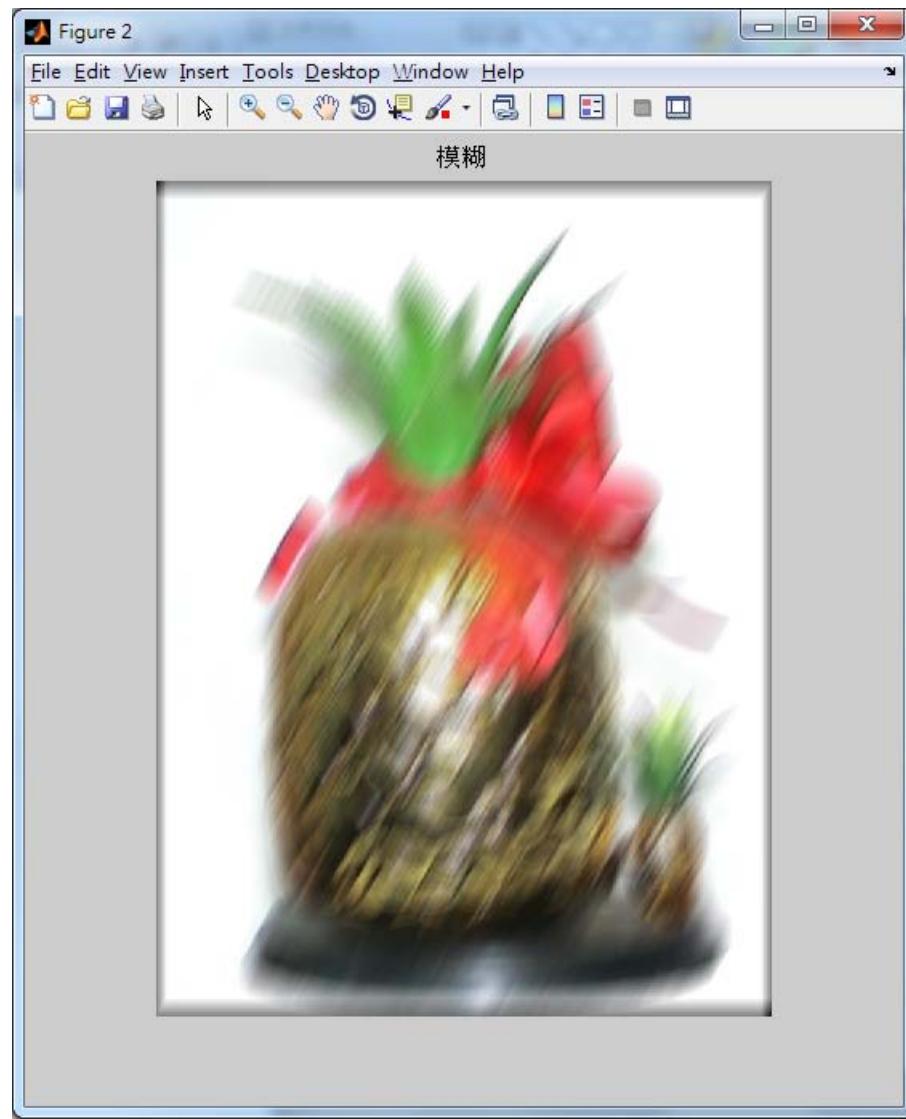
# 影像移動模糊及還原(模糊)



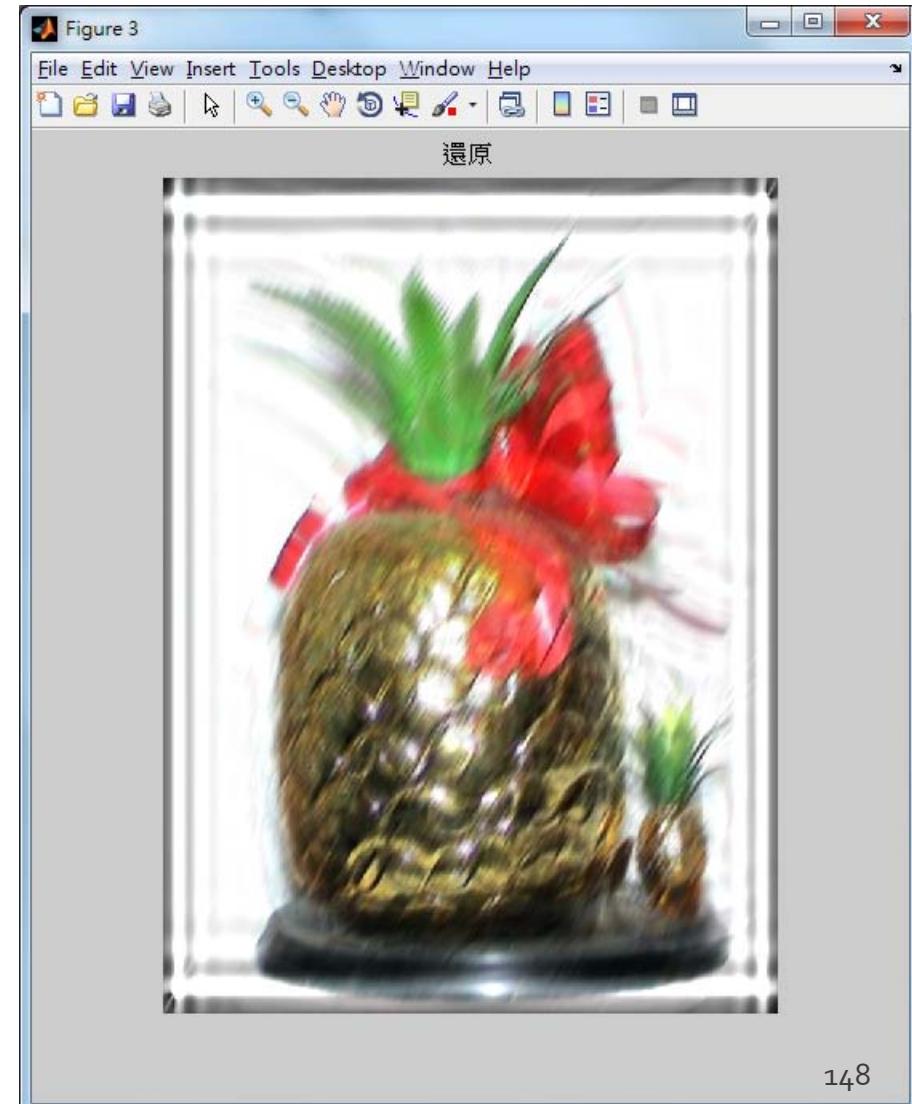
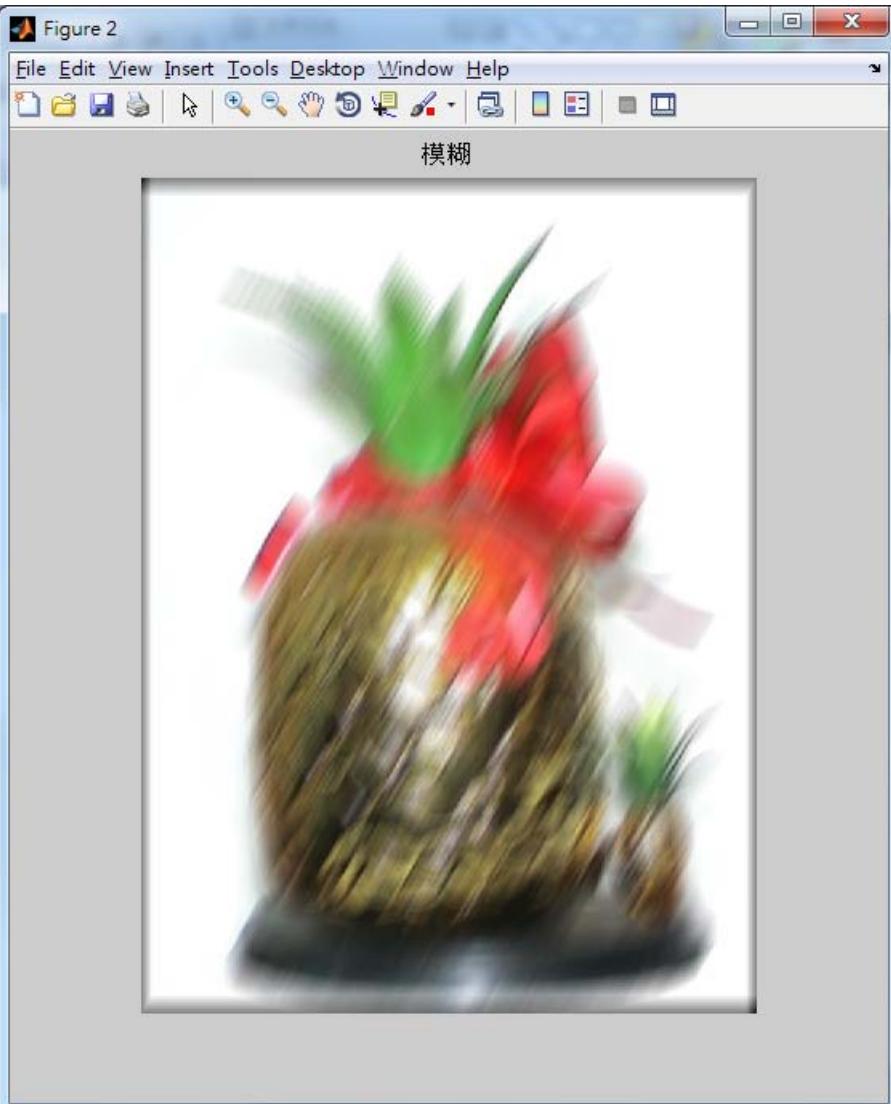
# 影像移動模糊及還原(Wiener濾波器)



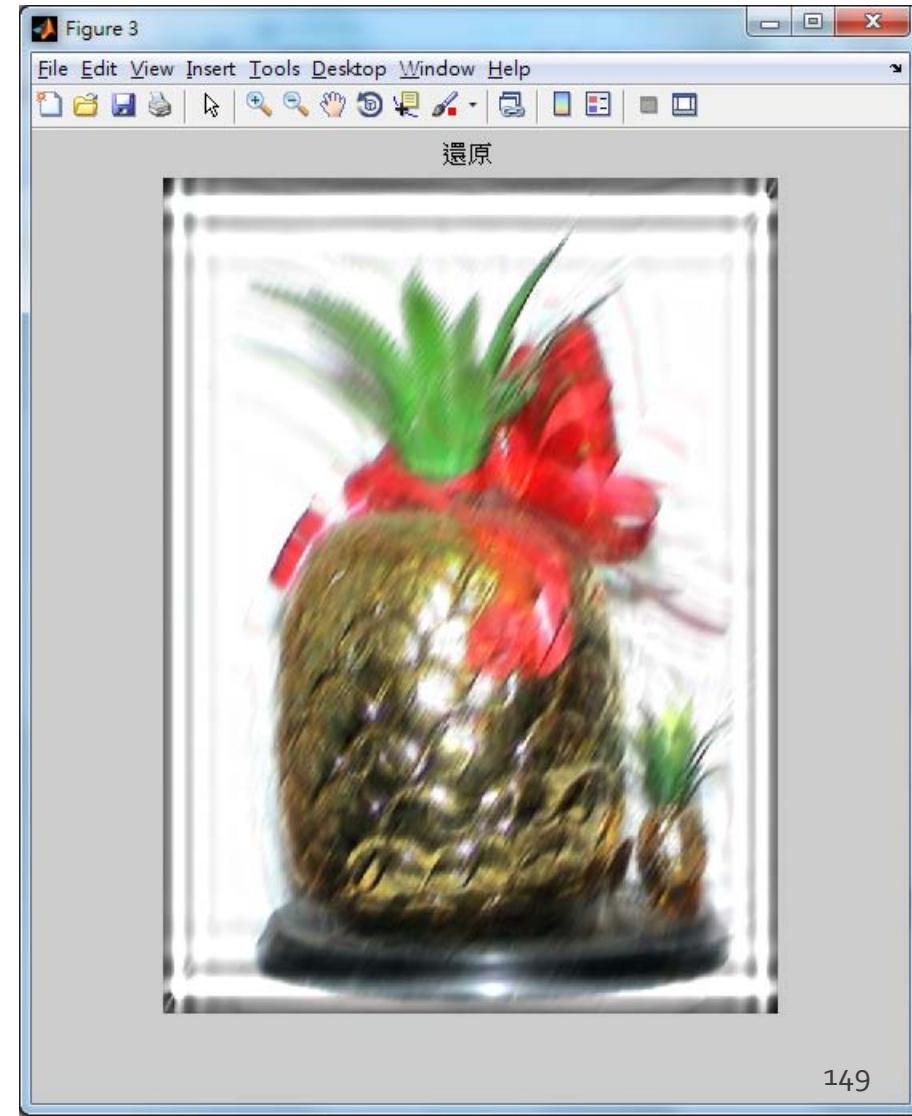
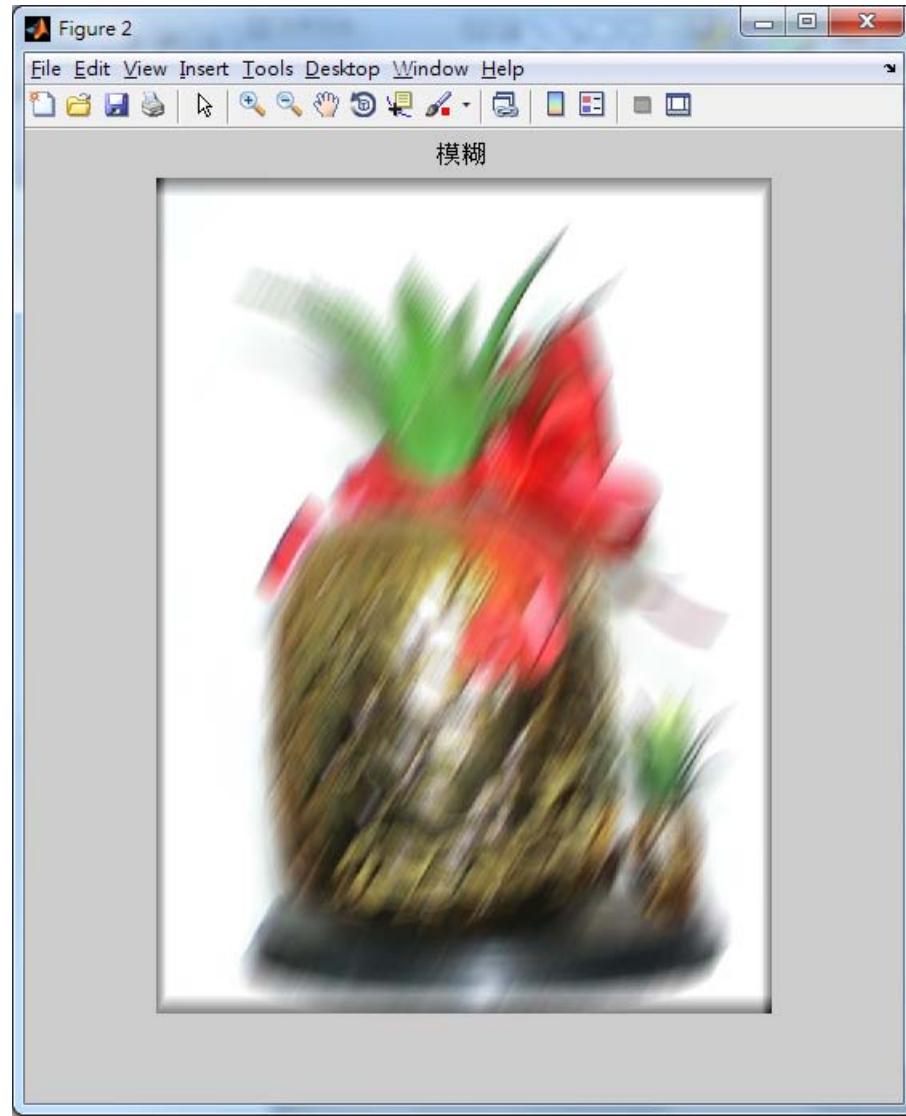
# 影像移動模糊及還原(一般濾波器)



# 影像移動模糊及還原(Lucy-Richardson濾波器)



# 影像移動模糊及還原(Blind濾波器)



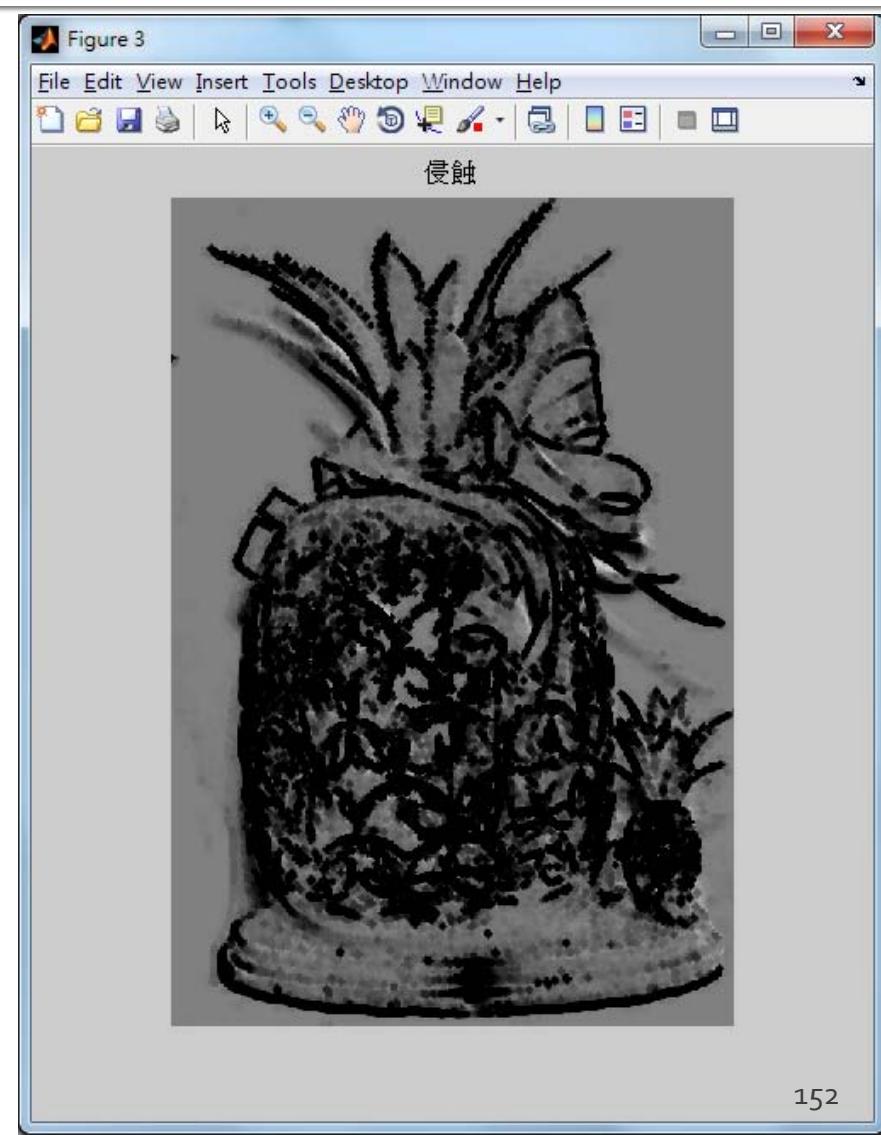
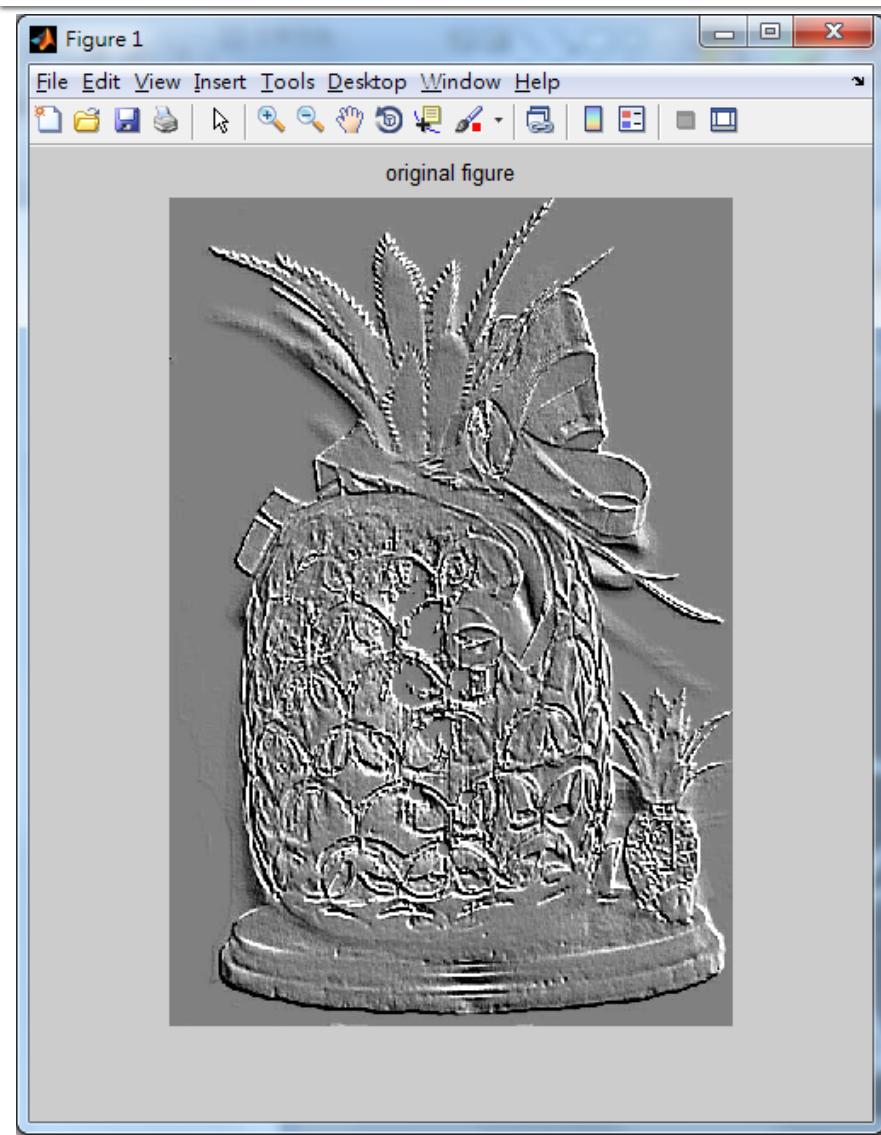
# 影像的擴張與侵蝕

- 結構元素指令: `strel` (影像檔, '結構元素', 參數)
  - `strel` (影像檔, 'square', 寬度)
  - `strel` (影像檔, 'diamond', 寬度)
  - `strel` (影像檔, 'disk', 半徑, 圓度)
  - `strel` (影像檔, 'arbitrary', 寬度)
  - `strel` (影像檔, 'line', 長度, 角度)
  - `strel` (影像檔, 'rectangle', [高 寬])
- 擴張指令: `imdilate` (影像檔, H)
- 侵蝕指令: `imerode` (影像檔, H)
  - H為結構元素設定好的物件

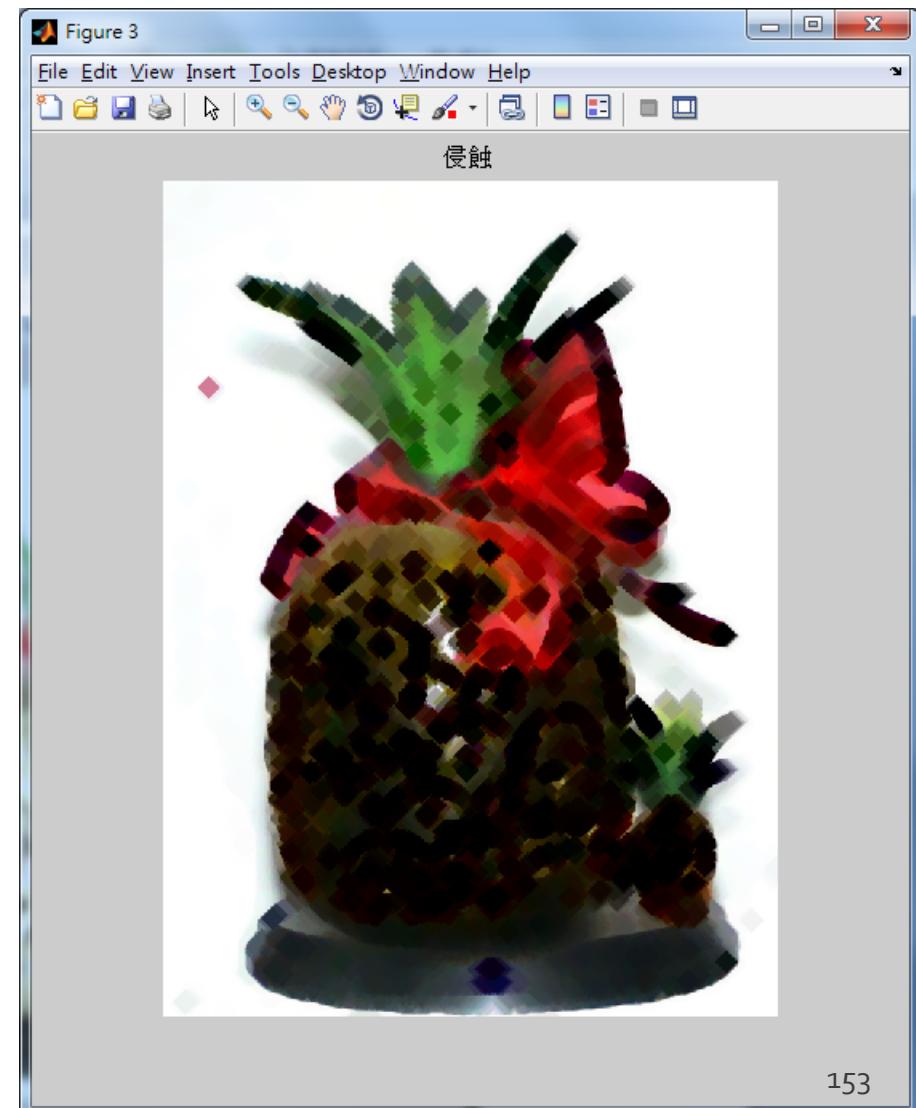
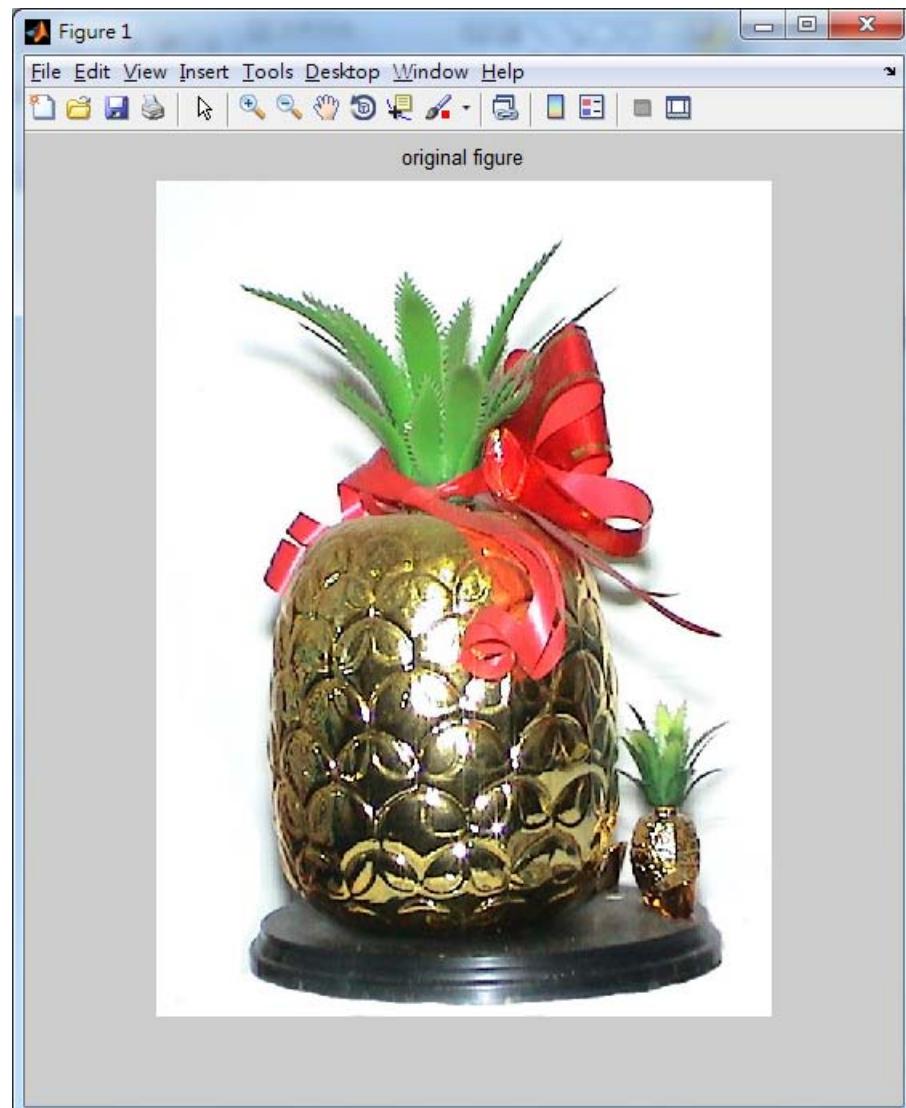
# 影像的擴張與侵蝕

- j=imread('pineapple\_gray.bmp');
- k=strel('diamond',3);
- h1=imdilate(j,k);
- h2=imerode(j,k);
- figure
- imshow(j);
- title('original figure')
- figure
- imshow(h1);
- title('dilated figure')
- figure
- imshow(h2);
- title('eroded figure')

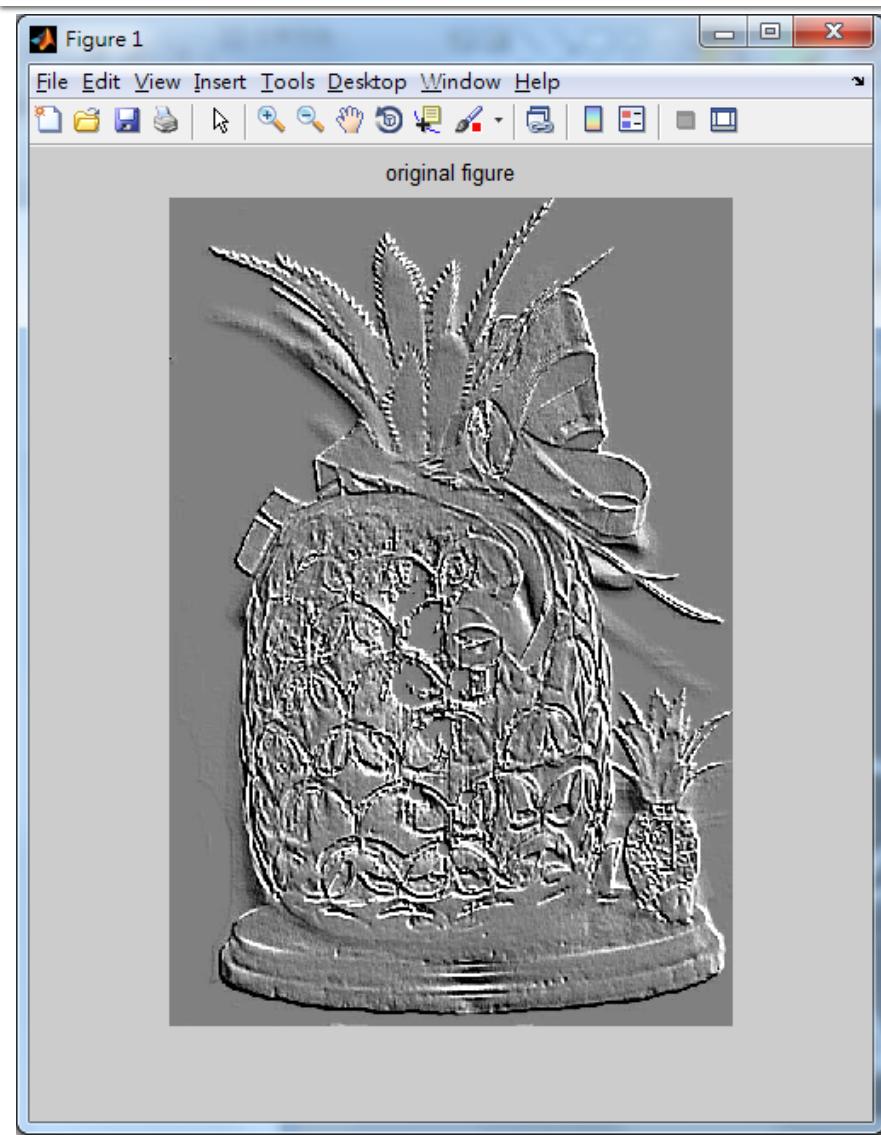
# 影像的擴張與侵蝕(侵蝕)



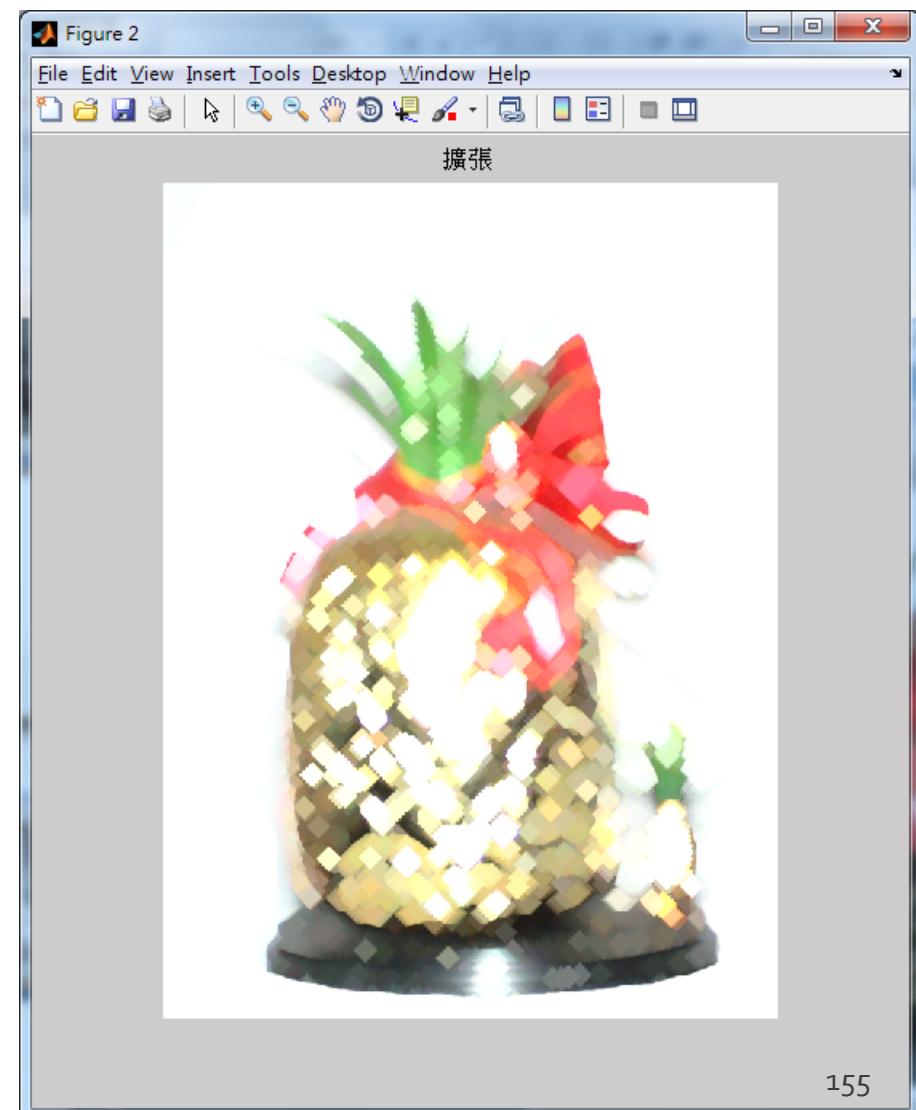
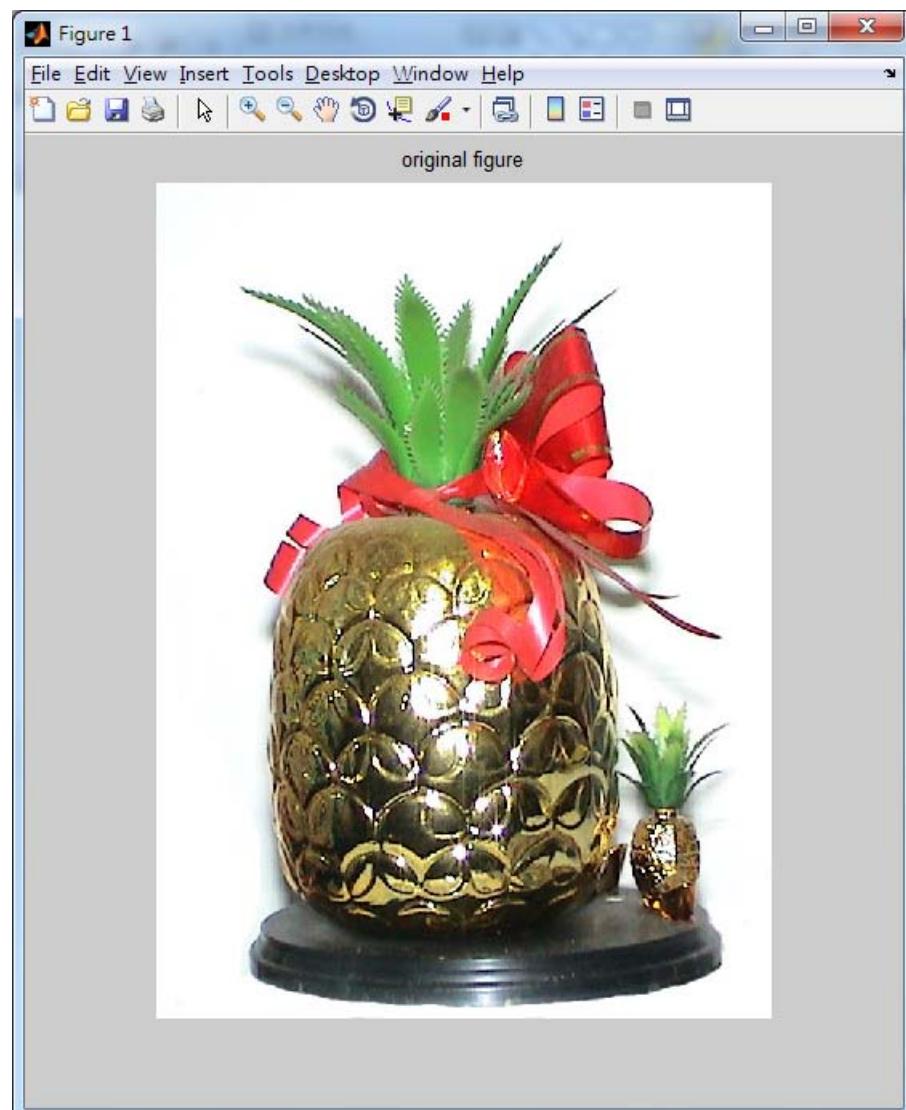
# 影像的擴張與侵蝕(侵蝕)



# 影像的擴張與侵蝕(擴張)



# 影像的擴張與侵蝕(擴張)



# Matlab GUI

image\_process



Show (gray)	Prewitt filter	Enhancement 1	LR filter
Show (color)	Laplacian filter	Enhancement 2	Blind filter
Sobel filter	Unsharp filter	模糊	Erode filter
Gaussian filter	Motion filter	Normal filter	Dilate filter
Average filter	Log filter	Wiener filter	End