

# 影像處理

## (Image Processing)

真理大學 資訊工程系 吳汶涓老師

Course 6  
影像幾何



# Outline

- 6.1 數據內插法
- 6.2 影像內插法
- 6.3 一般性內插法
- 6.4 使用空間濾波放大影像
- 6.5 縮小
- 6.6 旋轉
- 6.7 歪像



# 6.1 數據內插法

- 若有4個數值要放大成8個數值，該怎麼做？

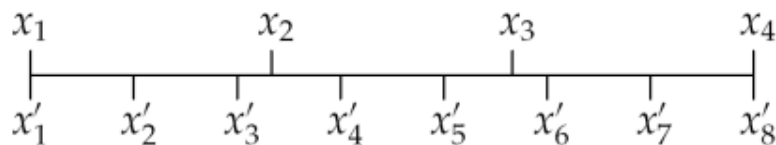
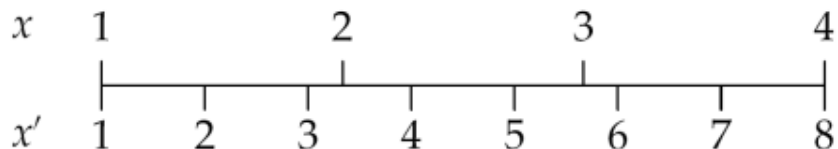


圖 6.1 以 8 個點置換 4 個點

$$\frac{x-1}{4-1} = \frac{x'-1}{8-1}$$



$$x' = \frac{1}{3}(7x - 4)$$
$$x = \frac{1}{7}(3x' + 4)$$

線性關係

圖 6.2 重畫圖 6.1

- 除了第一點與最後一點外，需用已知的鄰近值 $x_j$ 來估算函數值 $x'_i$ ，這種以周圍數值估算函數值的方法稱為**內插法(interpolation)**。
  - **近鄰**內插法(Nearest-neighbor Interpolation)
  - **線性**內插法(Linear Interpolation)

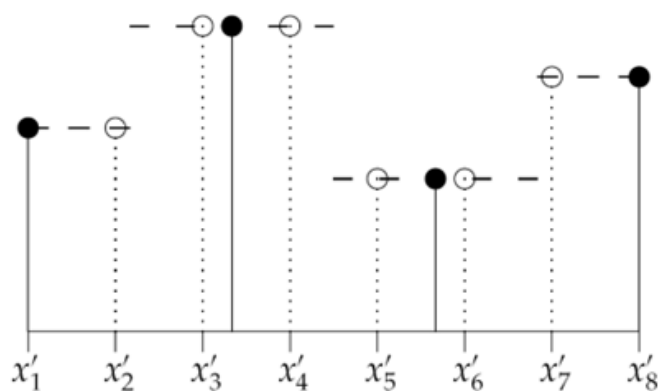


圖 6.4 近鄰內插法

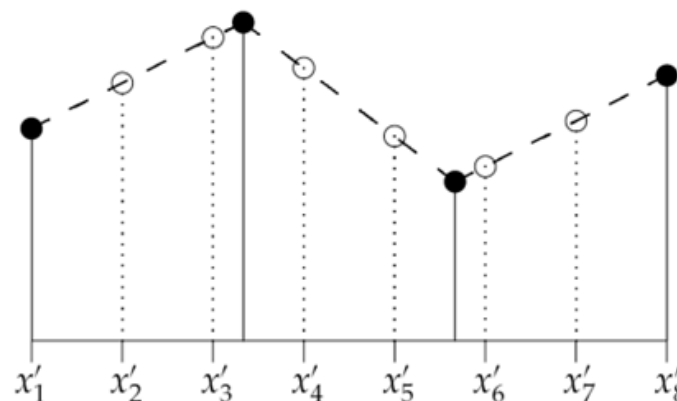


圖 6.5 線性內插法

## ■ 線性內插法(Linear Interpolation)

$$\frac{F - f(x_1)}{\lambda} = \frac{f(x_2) - f(x_1)}{1}$$

$$F = \lambda f(x_2) + (1 - \lambda)f(x_1)$$

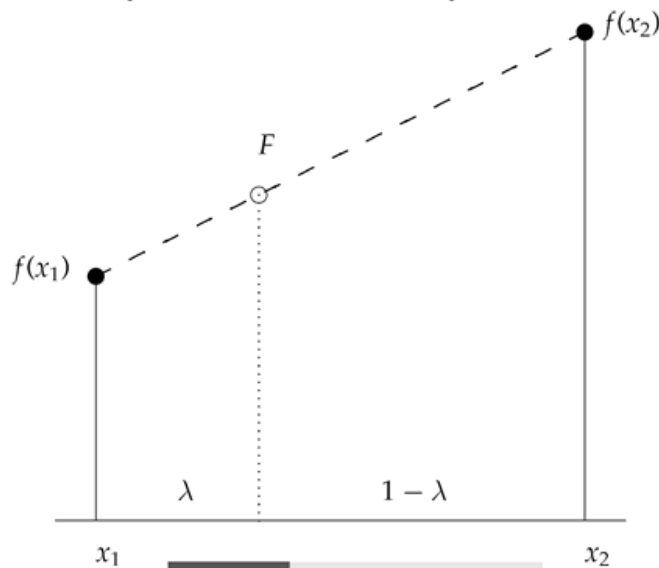


圖 6.6 計算線性內插值

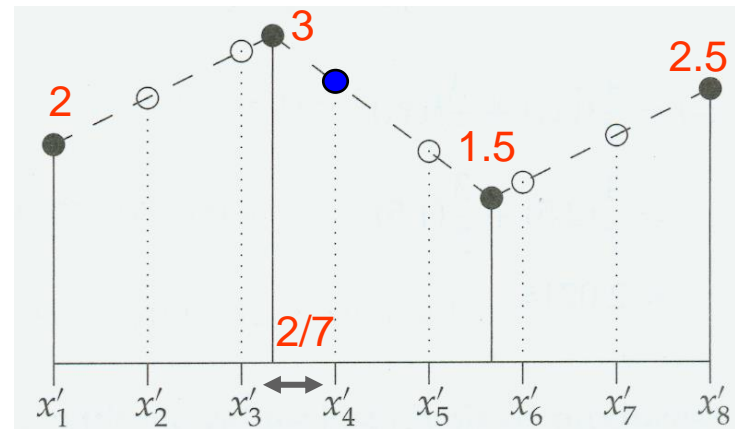
➤ Example:

$$f(x_1) = 2, f(x_2) = 3,$$

$$f(x_3) = 1.5, f(x_4) = 2.5$$

$$\lambda = 2/7 \rightarrow f(x'_4) = ?$$

$$\text{Ans} = 2/7 * (1.5) + 5/7 * (3)$$



## 6.2 影像內插法

- 透過內插法，4×4的影像可以放大為8×8的影像

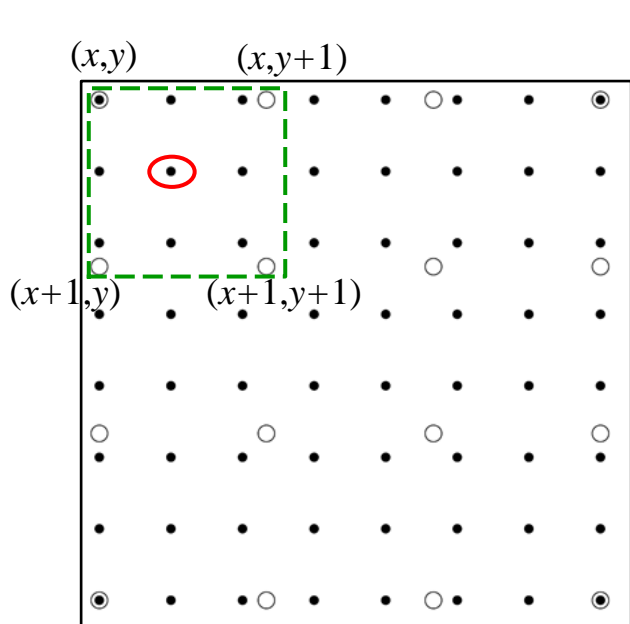


圖 6.7 影像內插

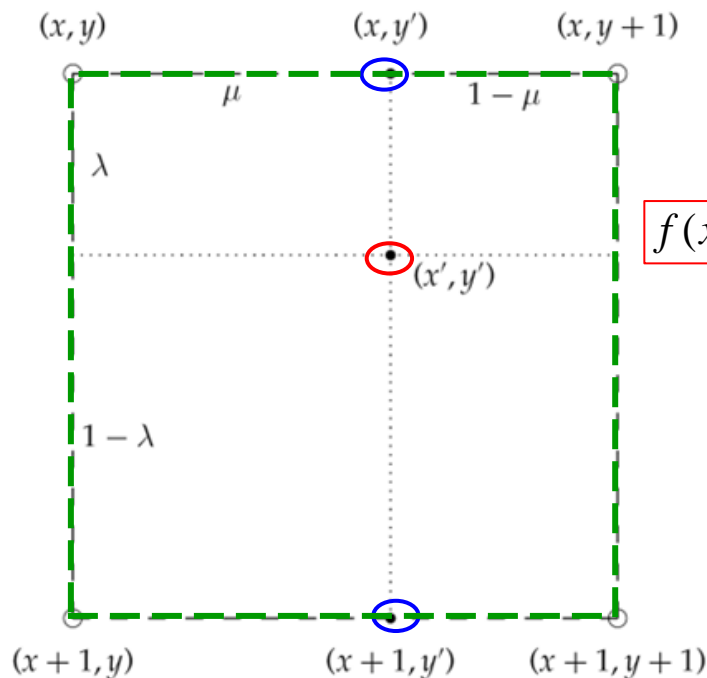


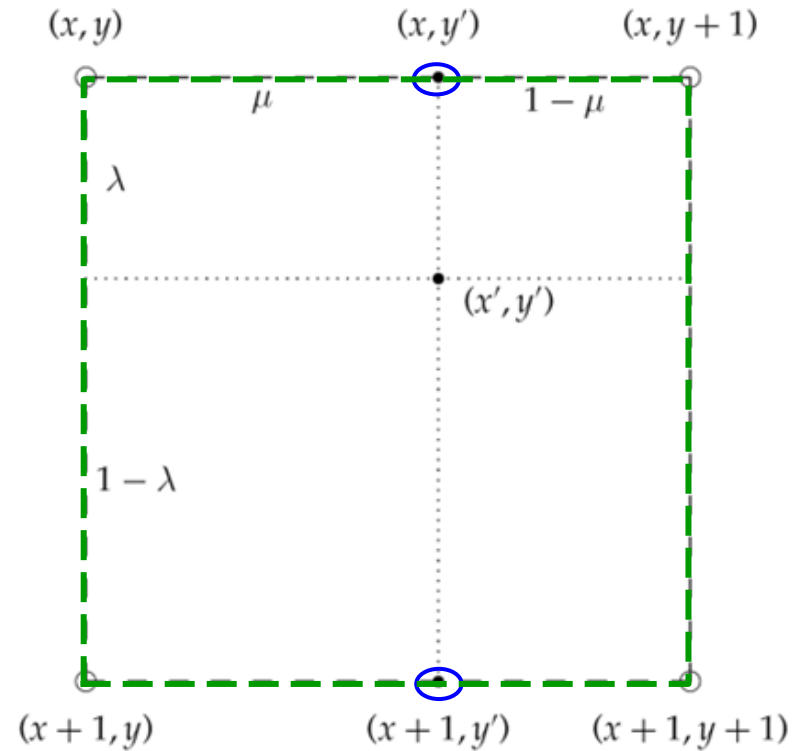
圖 6.8 4 個像素點間的內插

$$f(x, y') \rightarrow f(x', y')$$

(雙線性內插法)

$$f(x', y') = \lambda f(x + 1, y') + (1 - \lambda)f(x, y')$$

- 雙線性內插法  
(Bilinear Interpolation)



$$f(x, y') = \mu f(x, y + 1) + (1 - \mu)f(x, y)$$

$$f(x + 1, y') = \mu f(x + 1, y + 1) + (1 - \mu)f(x + 1, y)$$

➡  $f(x', y') = \lambda \underline{f(x + 1, y')} + (1 - \lambda) \underline{f(x, y')}.$

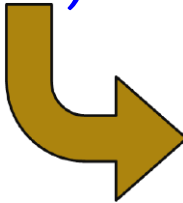
## ■ Matlab的函數imresize可進行影像縮放的功能

- `>>imresize(A,k, 'method')`

- A:影像資料

- k:縮放係數

([m,n]縮放大小，可小於1)



Value	Description
{ 'nearest' }	Nearest-neighbor interpolation
'bilinear'	Bilinear interpolation
'bicubic'	Bicubic interpolation



```
>> c=imread('cameraman.tif');  
>> head=c(33:96,90:153);  
>> imshow(head)  
>> head4n=imresize(head,4,'nearest');imshow(head4n)  
>> head4b=imresize(head,4,'bilinear');imshow(head4b)
```



邊緣鋸齒化，有馬賽克效果



近鄰內插法



雙線性內插法 (平滑但模糊)

圖 6.10 進行放大

## 6.3 一般性內插法

■ 近鄰與雙線性內插法其實是一般性內插法的特例。

- $R(u)$  為定義的內插法函數
- 以內插法求  $f(x')$  之值

$$f(x') = R(-\lambda)f(x_1) + R(1-\lambda)f(x_2)$$

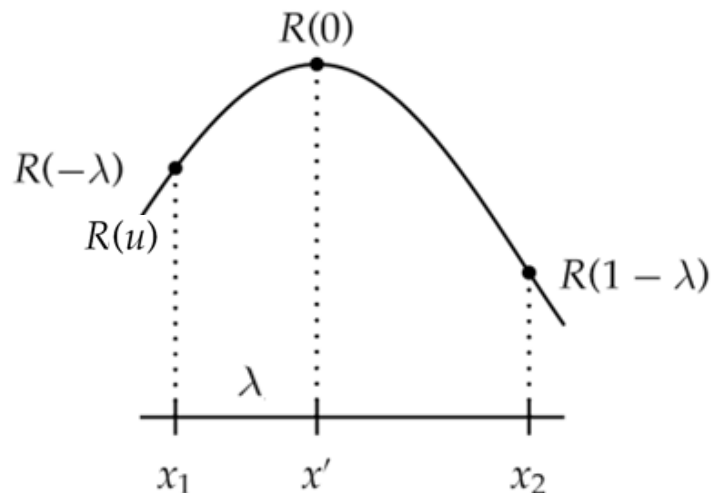
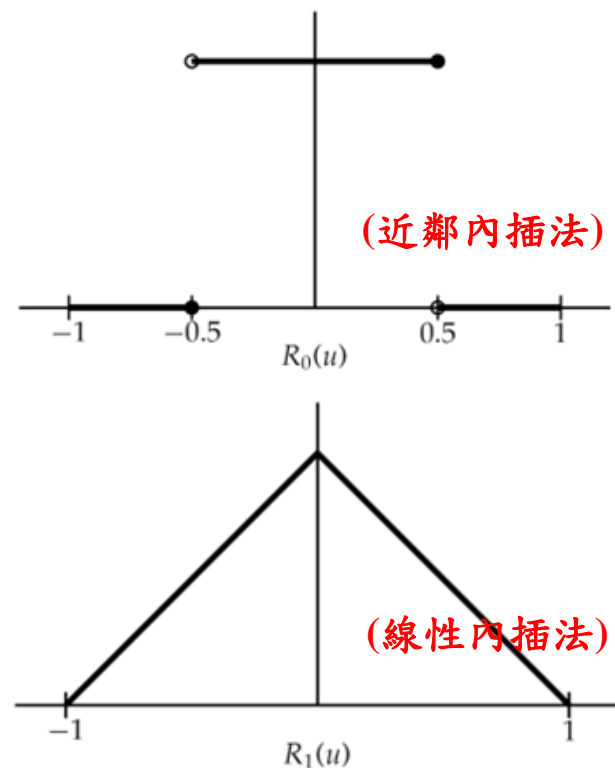


圖 6.11 執行一般內插函數



- 另一類似函數是**立方內插法**(cubic interpolation)

$$R_3(u) = \begin{cases} 1.5|u|^3 - 2.5|u|^2 + 1 & \text{若 } |u| \leq 1 \\ -0.5|u|^3 + 2.5|u|^2 - 4|u| + 2 & \text{若 } 1 < |u| \leq 2 \end{cases}$$

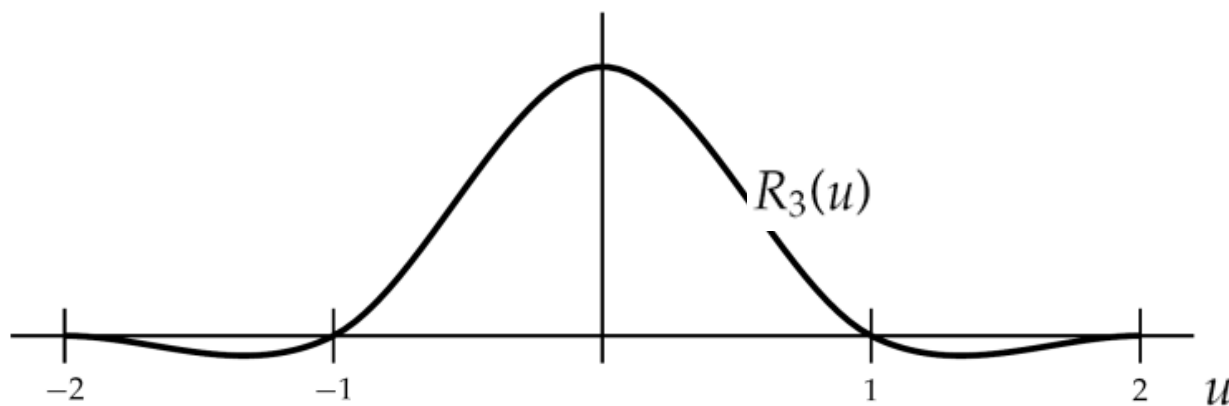


圖 6.13 立方內插函數  $R_3(u)$

- 除了使用 $x'$ 兩邊的 $x_2$ 與 $x_3$ 外，需考慮更遠的 $x$ 來進行內插計算。對影像執行此種內插，必須使用點 $(x,y)$ 周圍的16個已知數值。

$$f(x') = R_3(-1 - \lambda)f(x_1) + R_3(-\lambda)f(x_2) + R_3(1 - \lambda)f(x_3) + R_4(2 - \lambda)f(x_4)$$

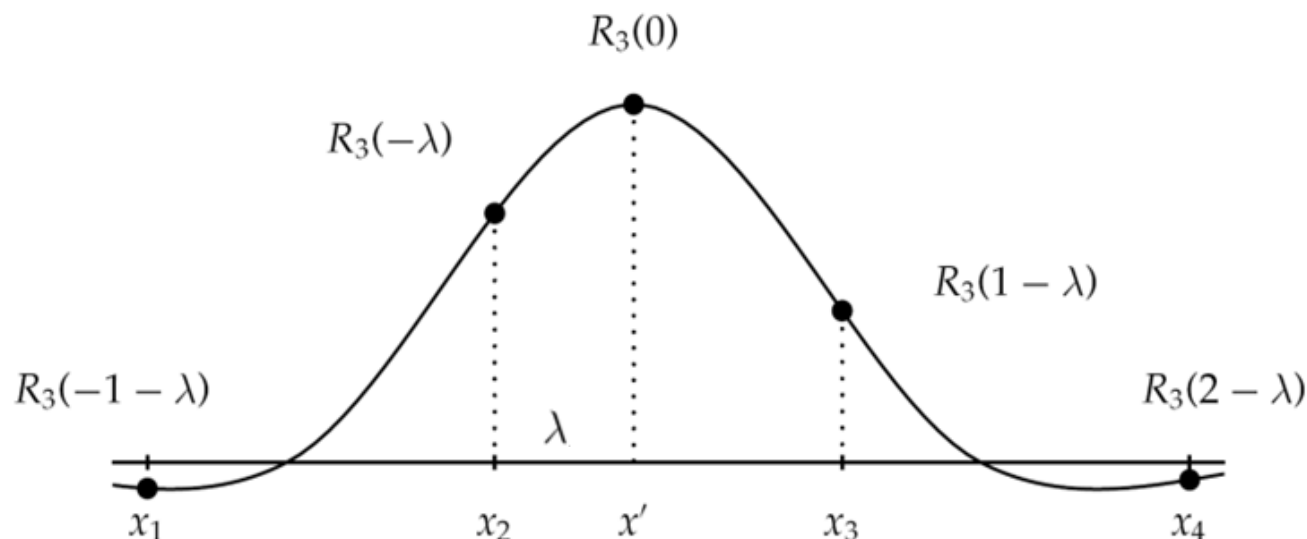


圖 6.14 用  $R_3(u)$  進行內插法

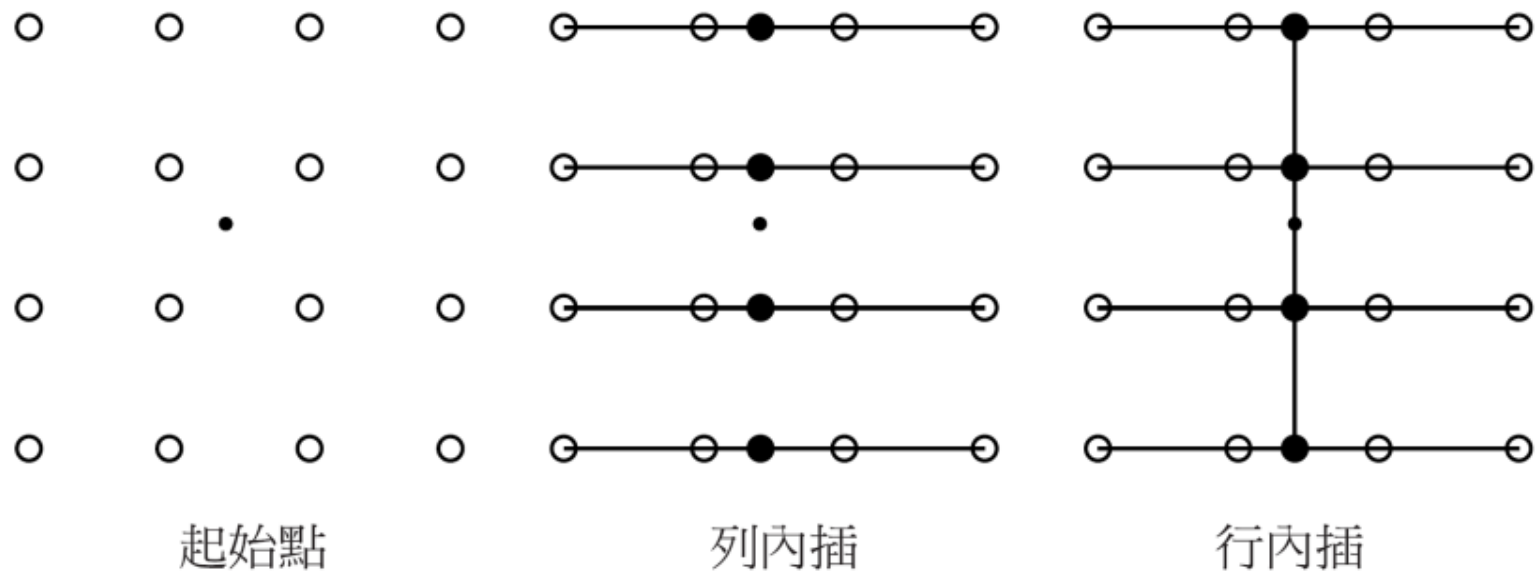


圖 6.15 如何執行立方內插法

注意：雙立方內插法是指從行與列兩個方向對影像執行立方內插法。

```
>> head4c=imresize(head,4,'bicubic');imshow(head4c)
```

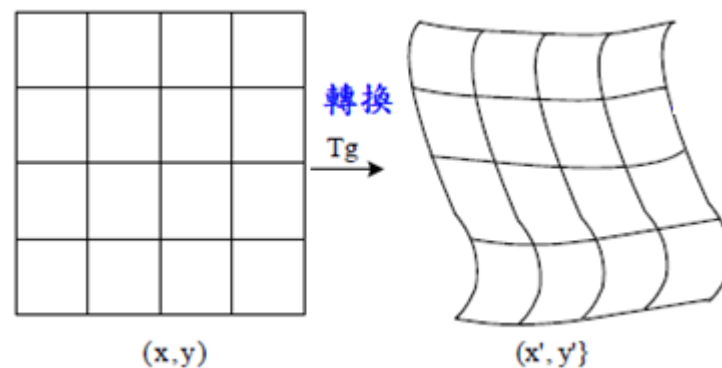


圖 6.16 使用雙立方內插法放大

➤ 應用: 魚眼特效



Fish-eye lens distortion



## 6.4 使用空間濾波放大影像

- 若只是想將影像放大為2的次方倍，可以使用**線性濾波器**。

➤ 例如：

```
>> m=magic(4)
```

```
m = 16     2     3    13
      5    11    10     8
      9     7     6    12
      4    14    15     1
```

➡ 對影像執行**零交錯**  
(zero-interleaved)

```
>> m2=zeroint(m)
```

```
m2 = 16     0     2     0     3     0    13     0
      0     0     0     0     0     0     0     0
      5     0    11     0    10     0     8     0
      0     0     0     0     0     0     0     0
      9     0     7     0     6     0    12     0
      0     0     0     0     0     0     0     0
      4     0    14     0    15     0     1     0
      0     0     0     0     0     0     0     0
```

$$m_2(i,j) = \begin{cases} m((i+1)/2, (j+1)/2) & \text{若 } i \text{ 與 } j \text{ 均為奇數} \\ 0 & \text{其他條件} \end{cases}$$

接著，執行空間濾波器

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

近鄰內插法

$$\frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

雙線性內插法

$$\frac{1}{64} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

雙立方內插法

```
>> c=imread('cameraman.tif');  
>> head=c(33:96, 90:153);  
>> hz=zerosint(head);  
>> f1=filter2([1 1 0; 1 1 0; 0 0 0], hz);  
>> f2=filter2([1 2 1; 2 4 2; 1 2 1]/4, hz);  
>> imshow(hz), figure, imshow(f1/256), figure, imshow(f2/256)
```



零交錯



近鄰內插



雙線性內插

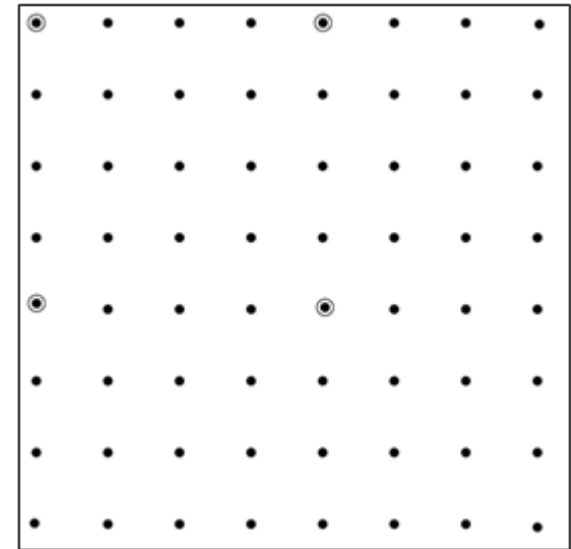


雙立方內插



## 6.5 縮小

- 將影像變小也稱為**影像最小化** (minimization)，其中一個方法就是取走間格的像素，此舉稱為**次取樣(subsampling)**。
    - 例如:縮小影像為原先的1/16，像素座標i與j必須為4的倍數
- ```
>> c=imread('cameraman.tif');  
>> c1=imresize(c,0.25);  
>> imshow(c1);
```



❑ 但，對於影像高頻部分，效果不太好。

```
>> t=zeros(1024,1024);  
>> for i=1:1024  
    for j=1:1024  
        t(i,j)=((255.5)^2<(i-512).^2+(j-512).^2)&((i-512)...  
            ^2+(j-512).^2<(256.5)^2);  
    end  
end  
>> t=~t;
```

```
>> tr=imresize(t,0.25);
```

```
>> trc=imresize(t,0.25,'bicubic');
```



(a) 近鄰縮小



(b) 雙立方內插法縮小

## 6.6 旋轉

### ■ 影像旋轉(Rotation)

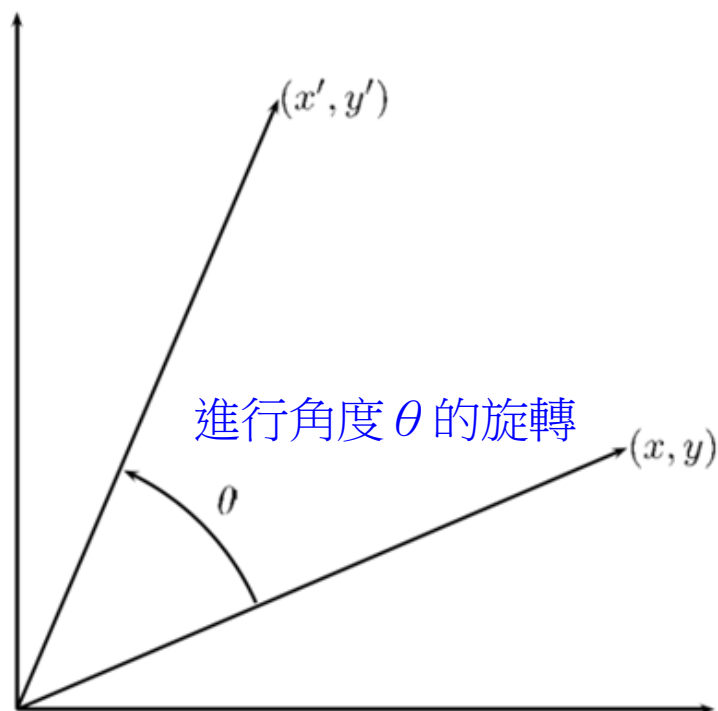


圖 6.20 將像素點旋轉  $\theta$  度

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$



重點在於如何保證旋轉之後，  
像素仍在原始網格上

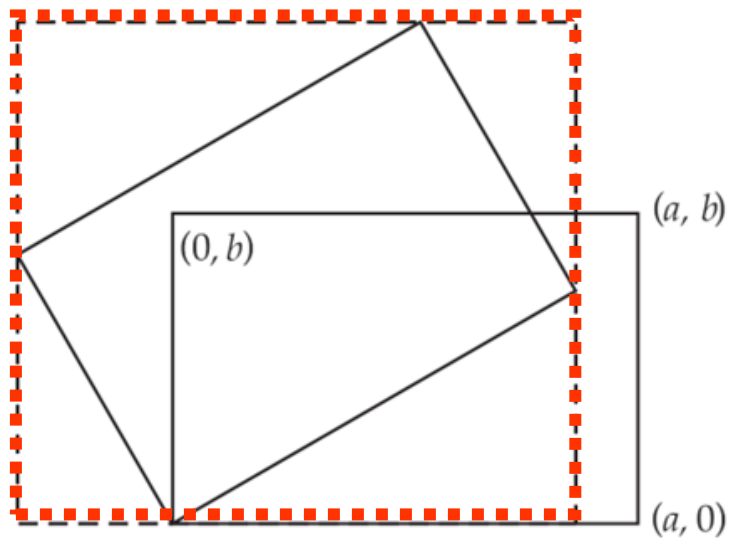


圖 6.22 旋轉後影像周圍的方形區域

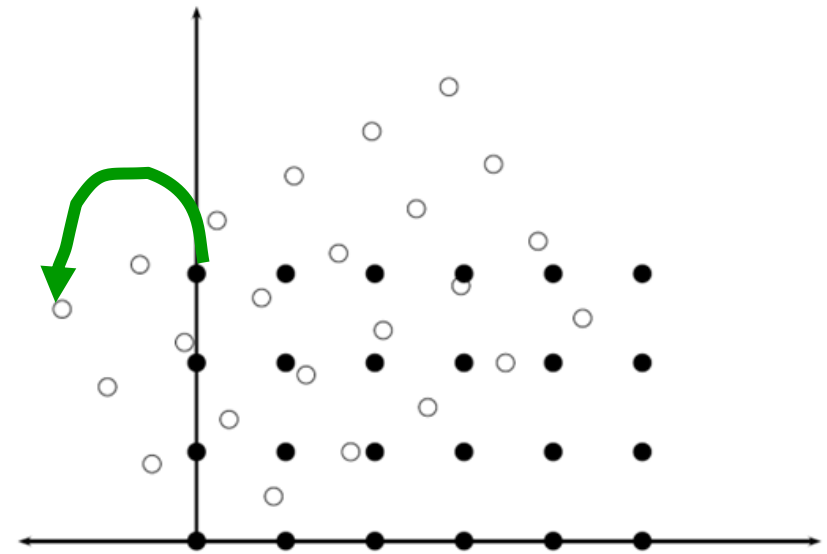


圖 6.21 旋轉方形

| Value         |
|---------------|
| { 'nearest' } |
| 'bilinear'    |
| 'bicubic'     |

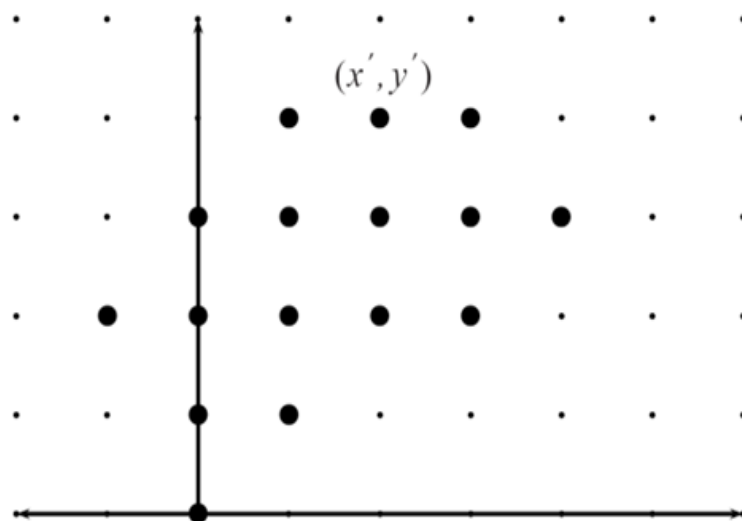


圖 6.23 旋轉後仍在座標上的像素點

$(x', y')$  的灰階值可透過周圍的灰階值，以內插法找出來

`flipud(c);` → 90度  
`fliplr(c);` → 左右倒轉

➤ `imrotate(image, angle, 'method')`

```
>> cr=imrotate(c,60);
>> imshow(cr)
>> crc=imrotate(c,60,'bicubic');
>> imshow(crc)
```



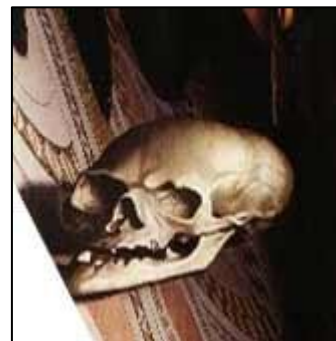
(a) 近鄰內插法



(b) 雙立方內插法

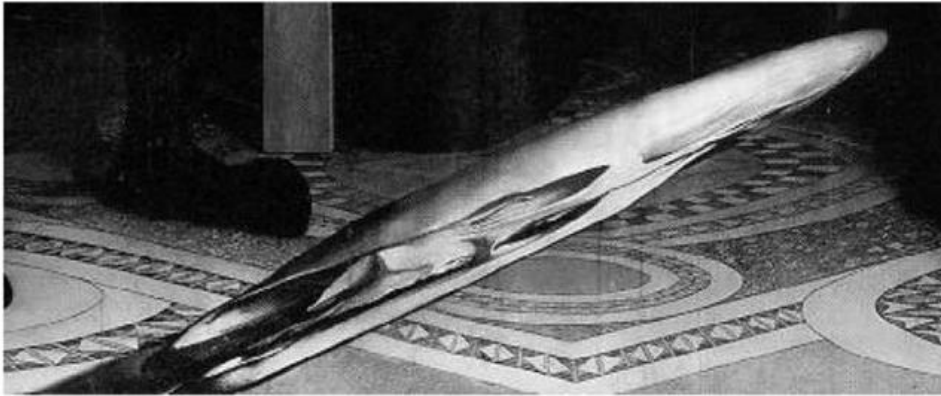
## 6.7 歪像

- 歪像：指延展或扭曲物體形狀
  - 為了藝術或戲劇效果
  - 霍爾班的<使節> 1533年



```
>> a=imread('AMBASSADORS.JPG');  
>> a=rgb2gray(a);
```

```
>> skull=a(566:743,157:586);
```



```
>> skull2=imresize(imrotate(skull,-22,'bicubic'),[500,150],'bicubic');
```

```
>> imshow(skull2(200:350,:))
```



圖 6.28 校正後的骷顱頭



# 練習

- 取出影像的頭部地方，放至4倍大，使用各種參數執行`imresize`，同時執行零交錯後使用三種濾波器。請比較濾波後結果與`imresize`的輸出結果，有任何差異嗎？

```
>> p=imread('pout.tif');  
>> ph=histeq(p);  
>> head=ph(10:129, 60:179);
```

- 若影像放大 $k$ 倍，再縮小 $k$ 倍。結果會和原始影像相同嗎？若否，理由是？

