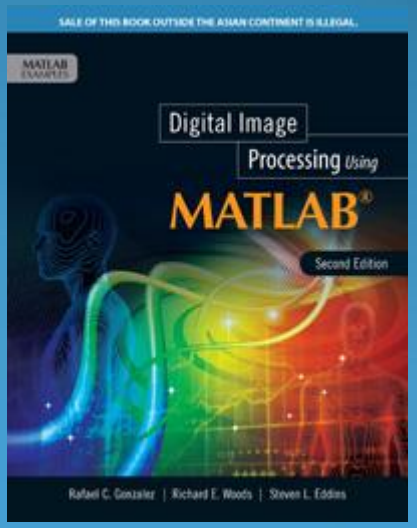
Medical Imaging

Medical Image Processing I

Yi-Li Tseng
FJU-EE
Fall 2017



References:
1. Digital Image Processing Using MATLAB, 2nd ed.: Rafael C. Gonzalez

Course Outline

Date	
09/21	X-ray X射線 (Prof. Paul Wang)
09/28	教師節放假
10/05	X-ray X射線 (Prof. Tseng)
10/12	Computerized Tomography 電腦斷層掃描 (Prof. Tseng)
10/19	Computerized Tomography 電腦斷層掃描 (Prof. Tseng)
10/26	MRI 磁振造影 (Prof. Paul Wang)
11/02	MRI 磁振造影 (Prof. Paul Wang)
11/09	MRI 磁振造影 (Prof. Paul Wang)
11/16	Mid-term Exam
11/23	Biomedical Imaging Processing 生醫影像處理 (Prof. Tseng)
11/30	Biomedical Imaging Processing 生醫影像處理 (Prof. Tseng)
12/07	Biomedical Imaging Processing 生醫影像處理 (Prof. Tseng)
12/14	Ultrasound 超音波 (Prof. Tseng)
12/21	Ultrasound 超音波 (Prof. Tseng)
12/28	Ultrasound 超音波 (Prof. Tseng)
01/04	NIRS (Near-Infrared Spectroscopy) 近紅外光譜系統 (Prof. Tseng)
01/11	生醫影像處理期末報告
01/18	Final Exam

Final Report

- 報告內容
  - Image Introduction
  - Methods (Flow of image processing)
  - Results
  - Discussions
- 每組4人，於指定日期報告
- 口頭報告(10%) 15 min report and 5 min Q&A
- 書面報告(10%) ppt, code, image

VMware View of Matlab

- http://edu.cc.fju.edu.tw/main_control/wsus/view.html

輔仁大學資訊中心
教學資源組

輔仁大學雲端電腦入口網

輔仁大學資訊中心教學資源組
聯絡電話：(02) 2905-3736
E-mail：edu4@mail.fju.edu.tw

資訊中心為使授權數有限制的教學授權軟體能夠充分使用，建置此雲端電腦入口網，讓本校師生可以透過網路，利用Windows電腦、MAC電腦、ubuntu電腦、ARM設備、iPad、iPhone設備與Android設備， 以個人LDAP帳號登入到雲端電腦系統，即可線上使用授權數限制的教學授權軟體，以完成教學與研究之需求。

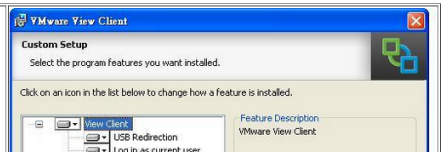
敬請尊重智慧財產權，拒絕非法散佈之行為。

1.什麼是雲端電腦?如何查詢我目前的作業系統版本?請看雲端電腦介紹

2.請下載雲端電腦連線程式

請按這裡下載「Windows 32位元 電腦 安裝程式」	請按這裡下載「Windows 64位元 電腦 安裝程式」	請按這裡下載「MAC 電腦 安裝程式」
請按這裡下載「Ubuntu 電腦 安裝程式」	請按這裡下載「ARM 設備 安裝程式」	請按這裡下載「Android 設備 安裝程式」
iPad、iPhone 請在 Apple AppStore 下載		

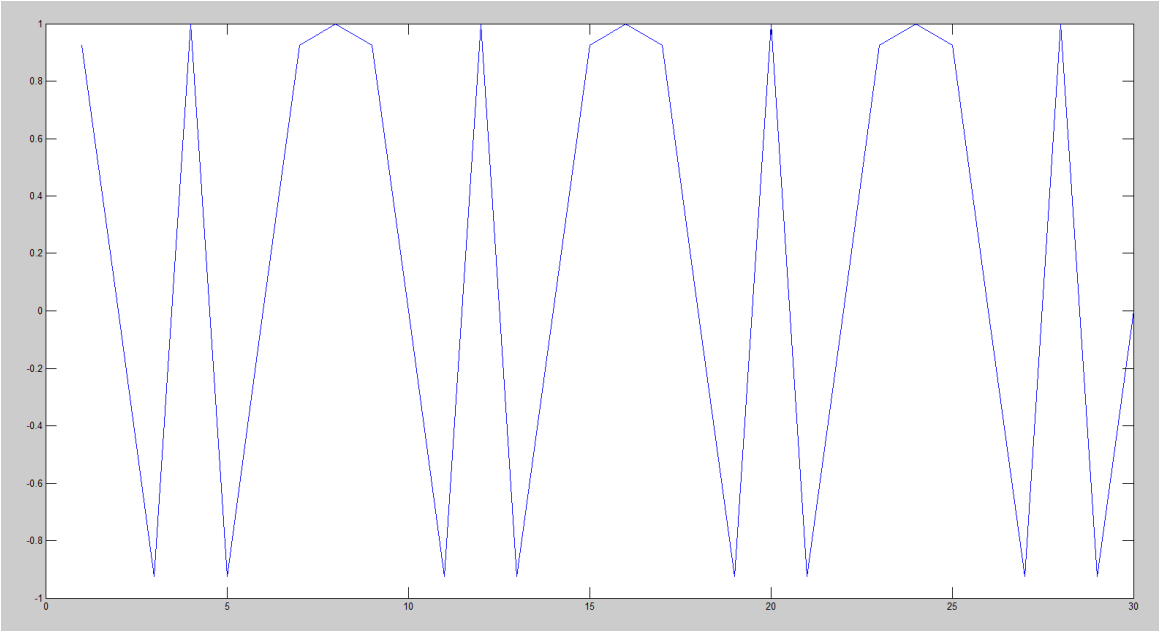
3.安裝雲端電腦連線程式及登入



What can Matlab do?

- Example : Is $\cos(\frac{\pi n^2}{8})$ periodic?

```
>> n=1:1:30;
>> x=cos(pi*n.^2/8);
>> plot(x)
```



Outline

- 什麼是數位影像處理?
- **MATLAB** 背景和影像處理工具箱
- 書中所涵蓋的影像處理的範圍
- 本書網站
- 基本原理

什麼是數位影像處理?

什麼是數位影像處理?

- 影像可定義為二維函數 $f(x, y)$
  - x 和y 是空間座標(spatial coordinates)
  - 在(x, y)座標上f 的振幅叫做影像在該點的**強度**(intensity) 或**灰階**(gray level)
  - 數位影像(digital image): x，y 和f 的振幅是有限的、離散的數量

什麼是數位影像處理?

- 數位影像處理的領域是指藉由數位電腦處理的數位影像
  - 數位影像是由一些有限的元素所組成，每個元素都有特定的位置和值，這些元素指的就是圖像元素(picture elements)、影像元素(image elements)、圖素(pels)和像素(pixels)等。

什麼是數位影像處理?

- 低階(Low-level) 處理 - 基本簡單的處理，例如減少影像雜訊、對比強化和影像銳化等的前置處理
  - 輸入和輸出基本上都是數位影像
- 中階(Mid-level) 處理如影像分割（將影像分隔成區簡化物件的描述來適合電腦的處理，以及個別物件的分類
  - 輸入幾乎都是影像，但輸出是由那些影像所擷取出來的屬性（比如物件的邊緣、輪廓和本體）

什麼是數位影像處理?

- 高階(high-level)處理涉及到辨識物件整體的合理性，如同在影像分析中一樣，通常執行與人類視覺相關的認知功能

MATLAB背景和影像處理工具箱

MATLAB 背景和影像處理工具箱

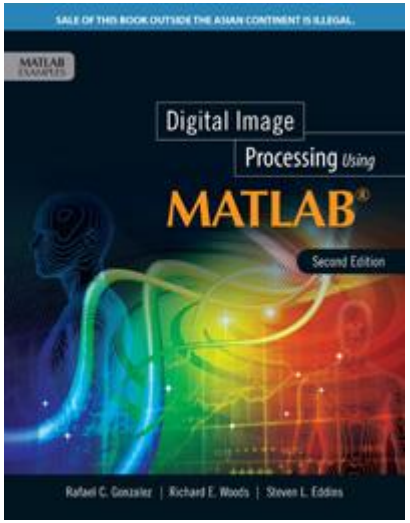
- MATLAB 是一個專業技術計算的高階語言，它整合計算、視覺化和編製程式
- 基本的使用
  - 數學和計算
  - 研製演算法
  - 獲取資料
  - 模型、模擬和原型
  - 資料分析、探勘和視覺化
  - 科學和工程圖學
  - 應用發展，包含建立圖形使用介面

MATLAB 背景和影像處理工具箱

- MATLAB 是Matrix Laboratory的意思
  - 互動式系統，它的基本元素是矩陣
- 影像處理工具箱(image processing toolboxes)
  - MATLAB 函數稱為M 函數(M-functions)或M 檔案(M-files)）的集合
  - 延伸了在MATLAB 環境之下解決數位影像處理問題的能力
  - 輔助工具為信號處理(Signal Processing)、神經網路 (Neural Network)、模糊邏輯(Fuzzy Logic)以及小波(Wavelet)

書中所涵蓋的影像處理的範圍

Reference: Digital Image Processing Using MATLAB, 2nd ed.: Rafael C. Gonzalez



書中所涵蓋的影像處理的範圍

- 第二章：強度轉換和空間濾波
  - 如何使用MATLAB 和影像處理工具箱來實現強度轉換函數
  - 線性和非線性空間濾波器
  - 模糊(fuzzy) 強度轉換和空間濾波器
- 第三章：頻率域的濾波
  - 使用工具箱函數來計算二維的前展 (forward) 和反(inverse) 快速傅立葉轉換(FFT)、傅立葉頻譜、在頻率域實現濾波以及從指定的空間濾波器產生頻率域濾波器的方法

書中所涵蓋的影像處理的範圍

- 第四章：影像復原與重建
  - 傳統的線性復原方法(比如說Wiener 濾波器)，反覆式疊代的非線性方法(比如Richardson-Lucy方法)
  - 盲目反摺積的最大可能性估測
  - 投影的影像重建以及如何使用在電腦斷層
- 第五章： 幾何轉換和影像重合
  - 幾何轉換的基本形式和實現，例如仿射(affine)和投影轉換，內插法也一併討論
  - 不同的影像重合技巧、轉換、重合以及視覺化的範例

書中所涵蓋的影像處理的範圍

- 第六章： 彩色影像處理
  - 偽顏色(pseudocolor) 和全彩(fullcolor) 的影像處理
  - 顏色模式和在其它顏色模式下延伸影像處理工具箱在彩色影像處理 上的功能
  - 彩色邊緣偵測和區域分割上的應用
- 第七章：小波
  - 影像處理工具箱並沒有小波轉換的函數
  - 獨自發展一套小波轉換的函數來實現所有我們討論的小波轉換的概念。

書中所涵蓋的影像處理的範圍

- 第八章：影像壓縮
  - 工具箱沒有任何資料壓縮的函數
  - 我們發展一套可用於影像壓縮的函數
- 第九章：形態學影像處理
  - 使用二元化影像和灰階影像來解釋說明用途廣泛的工具箱函數在形態影像學上的處理

書中所涵蓋的影像處理的範圍

- 第十章：影像分割
  - 工具箱函數在影像分割的使用
  - Hough 轉換、自定的區域成長和臨界函數 (thresholding functions)
- 第十一章：表示和描述
  - 物件表示和描述的新函數，包括鏈碼 (chaincode)和多邊形表示
  - 物件描述，包括傅立葉描述法、紋理和矩不變量(moment invariants)
  - 增補一個可應用在區域特性的函數的集合

本書網站

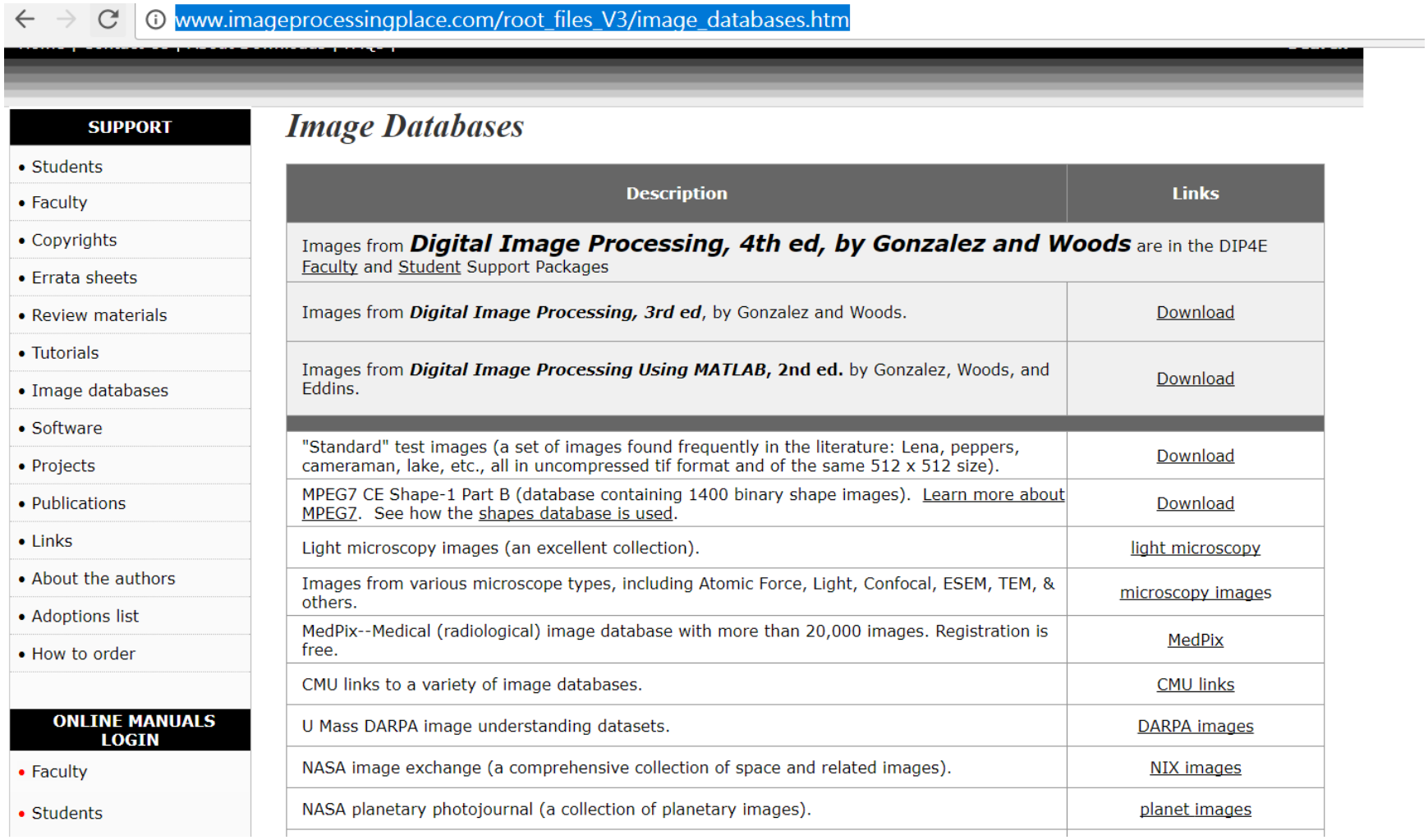
本書網站

www.ImageProcessingPlace.com

- 支援
  - 可下載本書中所有可執行的M 檔案
  - 個別輔導
  - 專題計畫
  - 教材
  - 包括本書中的所有影像到資料庫的連結
  - 教材更新
  - 參考文獻

本書網站

http://www.imageprocessingplace.com/root_files_V3/image_databases.htm



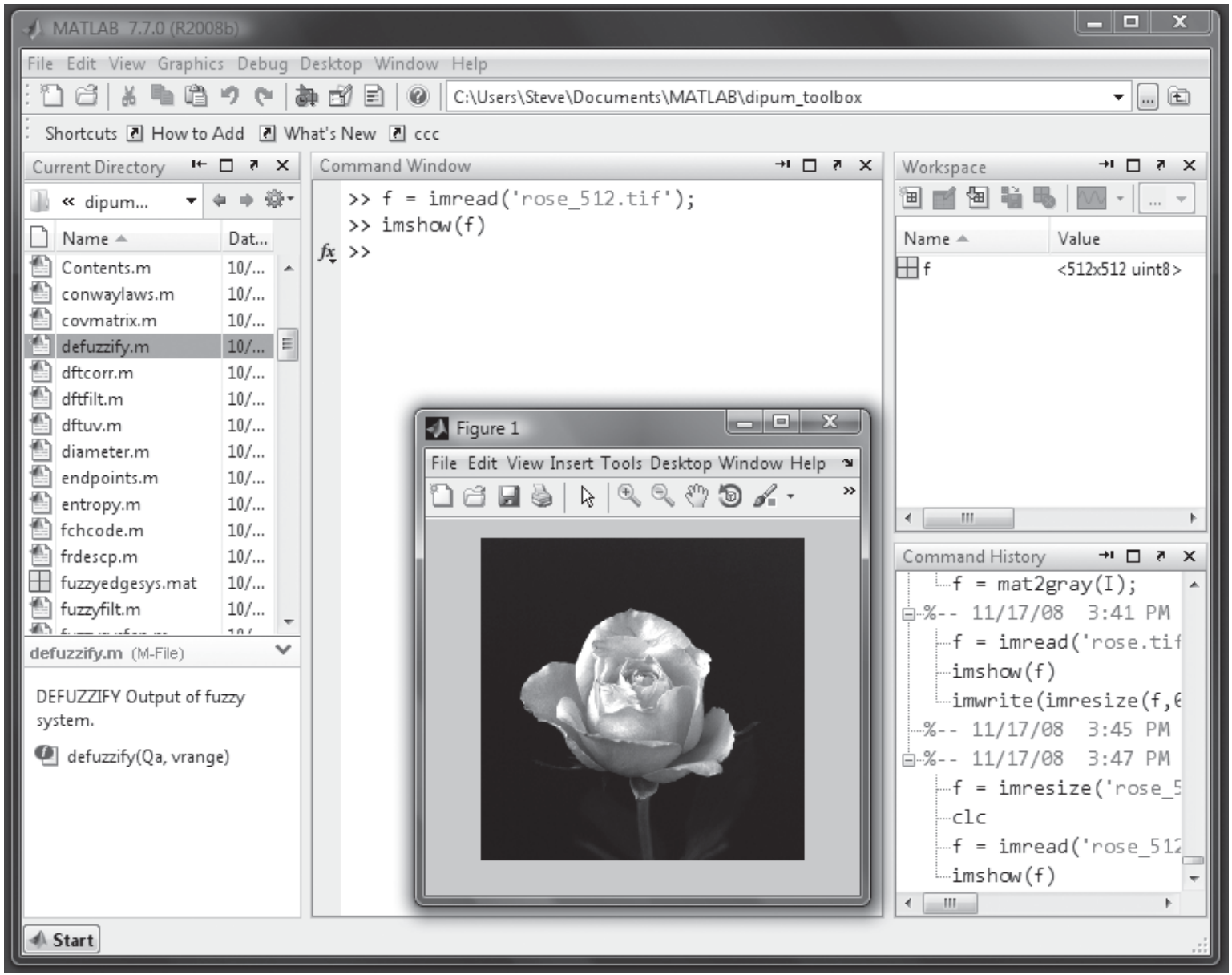
本書網站

基本原理

基本原理 - MATLAB 桌面

- MATLAB桌面(MATLAB Desktop)是主要的工作環境，執行MATLAB 指令、觀看輸出結果、編輯管理檔案及變數、和瀏覽回顧之前工作歷程的圖型工具(graphics tools)

基本原理 - MATLAB 桌面



基本原理 -數位影像的表示

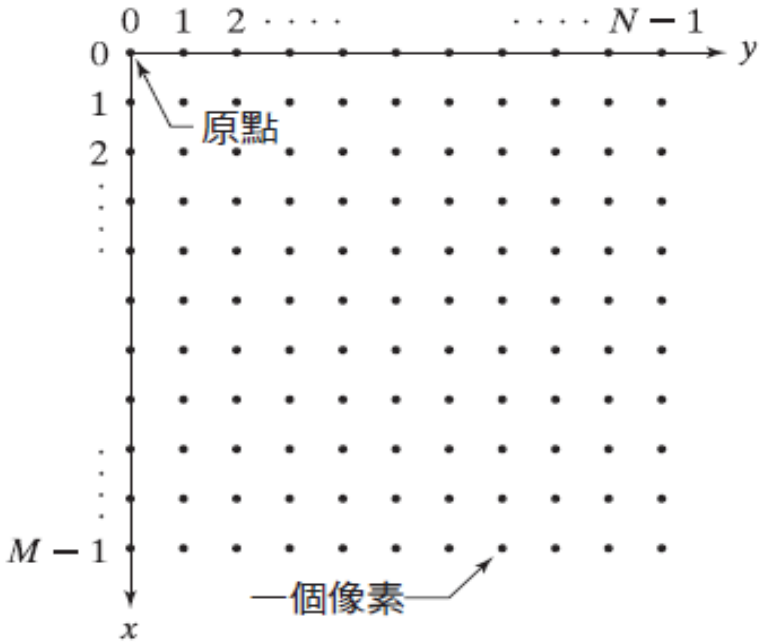
- 影像可定義為二維函數$f(x, y)$， 此處x和y是空間座標(spatial coordinates)，任何一個在(x, y) 座標上f 的振幅叫做影像在該點的**強度**(intensity)
  - **灰階**(gray level) 通常被指為單色 (monochrome) 影像的強度
  - 彩色影像是一些獨立的影像組合而成。 例如，RGB 顏色系統的彩色影像是由三個獨自的單色影像組成的，即紅色(R)、 綠色(G) 和藍色(B) 影像，此三個單色影像稱為成分影像(component images)

基本原理 - 數位影像的表示

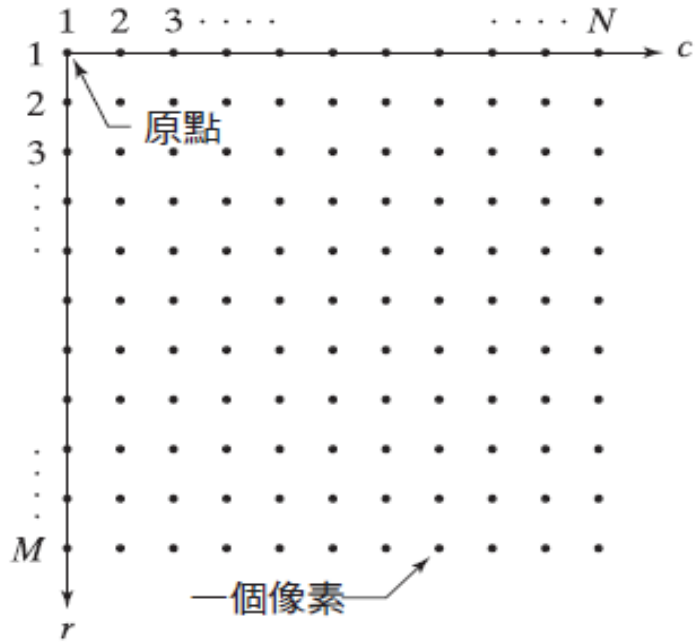
- 將影像轉成數位的形式就需要把座標及振幅數位化，座標數位化稱為**取樣**(sampling)，振幅數位化稱為**量化**(quantization)

基本原理 - 數位影像的表示

- 常用的座標方式
  - 一張影像$f(x, y)$取樣後有M 列N 行，我們說此影像的大小為M x N

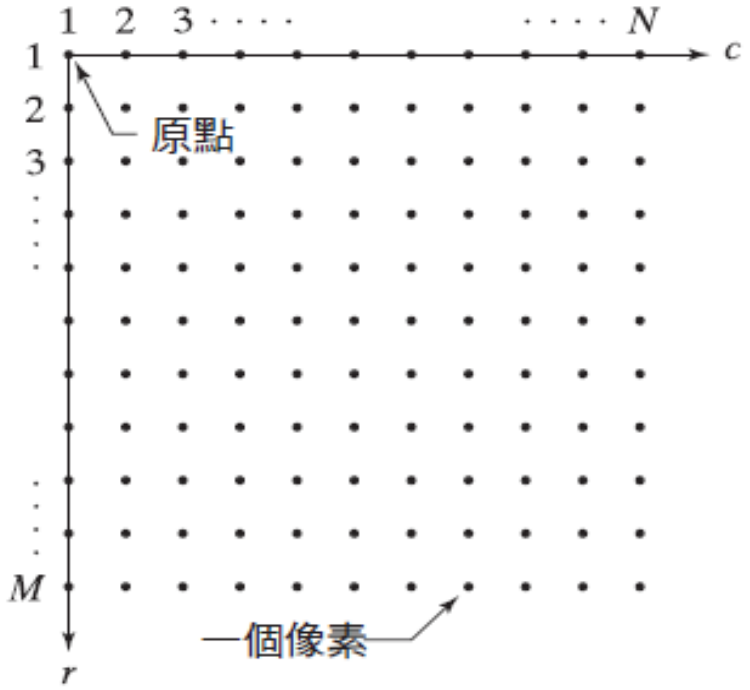


大多數的座標方式



影像處理工具箱
函數的座標方式

基本原理 - 數位影像的表示



- 像素座標(pixel coordinates)

基本原理 - 數位影像的表示

- 將影像視為矩陣

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

  - 陣列的每一個元素稱為影像元素(image element)、圖畫元素(picture element)、像素(pixel) 或圖素(pel)

基本原理 - 數位影像的表示

- MATLAB 的表示

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

  - 一個1 × N 的矩陣稱為列向量(row vector)，M × 1 的矩陣稱為行向量(culumn vector)，1 × 1 的矩陣稱為純量(scalar)

基本原理 - 影像的輸出輸入和顯示

- 函數imread 將影像讀入MATLAB 的環境，它的語法為

```
imread('filename')
```

例如

```
>> f = imread('chestxray.jpg');
```

- 函數imshow 將影像顯示於MATLAB 的桌面，它的語法為

```
imshow(f)
```

其中f是一個影像陣列

基本原理 - 影像的輸出輸入和顯示

- 下面的陳述在命令視窗中讀進影像 rose_512.tif 然後使用imshow 將它顯示

```
>> f = imread('rose_512.tif');
>> imshow(f)
```

- 留住原有的影像並輸出新的影像g，可使用函數 figure

```
>> figure, imshow(g)
```

基本原理 - 影像的輸出輸入和顯示

- 函數imwrite 應用在 tif 影像的一般語法

```
imwrite(g, 'filename.tif', 'compression', 'parameter', ...
                           'resolution', [colres rowres])
```

其中'parameter' 可以有以下的值：

- none:沒有壓縮；
- packbits(非二元化影像的預設）、'lwz'、'deflate'、'jpeg'、'ccitt'（只限二元化影像；預設）、'fax3'（只限二元化影像）和'fax4'
- 陣列[colres rowres]，colres 和rowres 是兩個整數，此兩個整數分別以每單位點數表示在行的解析度和列的解析度（預設值為[72 72]）

基本原理 - 資料與影像類別

表 1.1

名稱	描述
double	雙精確浮點數字，其範圍約 $\pm 10^{308}$（每一元素 8 個位元組）。
single	單精確浮點數字，其範圍約 $\pm 10^{38}$（每一元素 4 個位元組）。
uint 8	沒有正負號的 8 位元整數，範圍在 [0, 255] 之間（每一元素 1 個位元組）。
uint 16	沒有正負號的 16 位元整數，範圍在 [0, 65535] 之間（每一元素 2 個位元組）。
uint 32	沒有正負號的 32 位元整數，範圍在 [0, 4294967295] 之間（每一元素 4 個位元組）。
int 8	有正負號的 8 位元整數，範圍在 [−128, 127] 之間（每一元素 1 個位元組）。
int 16	有正負號的 16 位元整數，範圍在 [−32768, 32767] 之間（每一元素 2 個位元組）。
int 32	有正負號的 32 位元整數，範圍在 [−2147483648, 2147483647] 之間（每一元素 4 個位元組）。
char	字元（每一元素 2 個位元組）。
logical	值為 0 或 1（每一元素 1 個位元組）。

數值 (numeric) 類別

字元(character) 類別

邏輯(logical) 類別

基本原理 - 資料與影像類別

- 工具箱支援四種影像類型：
  - 灰階影像(gray-scale image)
  - 二元化影像(binary image)
  - 索引影像(indexed image)
  - RGB 影像(RGB image)

基本原理 - 資料與影像類別

- 灰階影像(gray-scale image)
  - 用灰色濃淡(shades of gray)表示像素值的矩陣
    - 灰階影像的元素是uint8 或uint16 時，像素值的範圍分別是[0, 255] 或[0, 65535] 的整數
    - 影像類別是double或single，則像素值是浮點數，並且正規化到[0, 1] 的範圍

基本原理 - 資料與影像類別

- 二元化影像(binary image)
  - 由0 和1 組成的邏輯(logical) 陣列
    - 如果資料類別是由0 和1 組成的陣列（諸如uint8的類別），在MATLAB 中將不列入考慮為二元化影像
    - 如果A 是由0 和1 組成的數值陣列，使用下列的陳述建立一個邏輯陣列B

```
B = logical(A)
```

    - 如果A 有0 和1 以外的數值，則函數logical 將非零的數轉為邏輯，而零的數轉為邏輯0，使用邏輯運算和關係式也可以產生邏輯陣列。

基本原理 - 資料與影像類別

- 邏輯陣列可以使用一般的類別轉換函數轉為數值陣列

B = class_name(A)

其中class_name 是im2uint8、im2uint16、im2double、im2single 或mat2gray。

- 工具箱函數mat2gray 將影像轉為類別double 的陣列再調整到[0, 1] 的範圍，使用下面的語法

```
g = mat2gray(A, [Amin, Amax])
```

其中影像g的值在範圍0（黑色）到1（白色），小於Amin 的值用0 取代，大於Amax 的值設為1

-

```
g = mat2gray(A)
```

分別將A 的最小值和最大值設定給Amin 和Amax

基本原理 - 資料與影像類別

- 關於術語的說明
  - 通常我使用class image_type image表示影像
    - class 是表1.1 的一個項目
    - image_type 是影像類型：灰階影像、二元化影像、索引影像或RGB 影像
      例如，“uint8 gray-scale image”是指一張像素值是類別uint8 的灰階影像

基本原理 - M 函數程式規劃

- M 檔案
  - MATLAB的M檔案是執行一連串MATLAB 指令陳述的腳本，或接受引數並產生一個或多個輸出的函數
  - M 檔案可使用文字編輯器來產生，並且以 filename.m 的名稱來儲存，譬如，average.m
  - M 檔案函數的成分有
    - 函數定義列(The function definition line)
    - Hl 列(The Hl line)
    - 輔助說明(Help text)
    - 函數本體(The function body)
    - 註解(Comments)

基本原理 - M 函數程式規劃

- 函數定義列(function definition line)
  - ```
    function [outputs] = name(inputs)
    ```
  - 例如，計算兩個影像的和與乘積（兩個不同的輸出）的函數

```
function [s, p] = sumprod(f, g)
```

其中f 和g 分別為輸入影像，s 為和的影像，而p 為乘積的影像， sumprod 是任意設定的名稱

- 函數名稱必須是以字母為首，其餘的字可以是字母、數字或底線，此外不允許有空格。MATLAB 辨別函數的名稱最長可達63 個字元
```

基本原理 - M 函數程式規劃

- 函數可以在命令提示區被呼叫，例如

```
>> [s, p] = sumprod(f, g);
```

- 或者它們可作為其它函數的元素，如此就成為子函數(subfunction）。

基本原理 - M 函數程式規劃

- H1 列(H1 line)
  - 文字列(text line) 的第一行，它是緊跟在函數定義列(function definition line) 之後的一列註解，
  - 在H1 列和函數定義列之間可以沒有空白列或開頭是空白的。例如

    %計算兩個影像的和與乘積
  - 當使用者在MATLAB 提示區鍵入下列指令時，H1 列是頭一列所出現的文字

```
>> help function_name
```

  - 鍵入lookfor keyword， 便會顯示包含字串 keyword 的所有H1列，這一列提供關於M 檔案重要的摘要資訊

基本原理 - M 函數程式規劃

- 輔助說明(Help text) 是在H1 列之後的文字區塊，兩者之間沒有任何的空白列
  - 輔助說明是用來提供函數註解及在螢幕上的援助，當使用者在提示處鍵入help function_name時，MATLAB 會顯示介於函數定義列和非註解的第一列（可執行或空白）之間的所有註解，援助系統忽略輔助說明區塊之後的任何註解
- 函數本體(function body) 包括所有執行運算和輸出引數指定值的MATLAB 程式碼

基本原理 - M 函數程式規劃

- 註解(comment)
  - 不是H1 列或輔助說明而以符號“%”開頭的列可以視為函數的註解，但不被視為輔助說明區塊的一部份，此外允許在一列程式碼的末端附加註解
  - 輔助說明是用來提供函數註解及在螢幕上的援助，當使用者在提示處鍵入help function_name時，MATLAB 會顯示介於函數定義列和非註解的第一列（可執行或空白）之間的所有註解，援助系統忽略輔助說明區塊之後的任何註解

基本原理 -算術運算子(Arithmetic Operators)

- MATLAB 有兩種不同的算術運算
  - 由線性代數所定義的矩陣算術運算 (matrix arithmetic operations)
  - 由一個元素一個元素執行的陣列算術運算 (array arithmetic operations)，此運算亦可用於多維陣列

基本原理 -算術運算子(Arithmetic Operators)

表 1.2 陣列和矩陣算術運算子，字元 a 和 b 是純量

運算子	名稱	註解及範例
+	陣列和矩陣加法	a+b，A+B 或 a+A
-	陣列和矩陣減法	a-b，A-B，A-a 或 a-A
.*	陣列乘法	Cv=A.*B，C(I,J)=A(I,J)*B(I,J)
*	矩陣乘法	A*B：標準矩陣乘法；a*A：純量 a 乘上 A 的所有元素
./	陣列右除法[註]	C=A./B，C(I,J)=A(I,J)/B(I,J)
.\	陣列左除法[註]	C=A.\B，C(I,J)=(B(I,J)/A(I,J)
/	矩陣右除法	A/B 是計算 A*inv(B)
\	矩陣左除法	A\B 是計算 inv(A)*B
.^	陣列次方	如果 C=A.^B，則 C(I,J)=A(I,J)^B(I,J)
^	矩陣次方	有關此運算子的討論見 help
.'	向量和矩陣轉置	A.'，標準向量和矩陣轉置
'	向量和矩陣複數共軛轉置	A'，標準向量和矩陣共軛轉置；當 A 是為實數時，A.'= A'
+	單一加	+A 與 0+A 一樣
-	單一減	-A 與 0-A 或 -1*A 一樣
:	冒 號	在本節稍後討論

註：如果分母為 0，則 MATLAB 回傳無限大（記為 Inf）。如果分母與分子都為 0，則 MATLAB 回傳 NaN（即不是一個數）。

基本原理 -算術運算子(Arithmetic Operators)

- 陣列運算 和矩陣運算之間的差異

$$A = \begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \quad 和 \quad B = \begin{bmatrix} b1 & b2 \\ b3 & b4 \end{bmatrix}$$

則 A 和 B 的**陣列乘積 (array product)** 是

$$A.*B = \begin{bmatrix} a1b1 & a2b2 \\ a3b3 & a4b4 \end{bmatrix}$$

然而**矩陣乘積 (matrix product)** 為：

$$A*B = \begin{bmatrix} a1b1+a2b3 & a1b2+a2b4 \\ a3b1+a4b3 & a3b2+a4b4 \end{bmatrix}$$

大部分在影像的算術、關係式和邏輯運算都是陣列運算。

基本原理 -關係運算子(Relational Operators)

- MATLAB 的關係運算子是陣列運算子，也就是說它們在相同的維度之下比較陣列相對應的元素

運算子	名稱
<	小於
<=	小於或等於
>	大於
>=	大於或等於
==	等於
~=	不等於

  - 當位置上的元素與指定的關係滿足時用1 表示，否則以0 表示

基本原理 - 邏輯運算子(Logical Operators)

- 對邏輯與數值資料作運算
  - 邏輯的測試是將邏輯1 或非零的數值設為true，而邏輯0 或數值0 設為false
  - 運算子& 和|可以在陣列上運算，它們在輸入的陣列相對應的元素上分別計算AND和OR。
  - 運算子&&和||只能針對純量運算，它們主要是用在各種形式的if、while和for迴圈

運算子	名稱
&	陣列元素逐一做 AND
|	陣列元素逐一做 OR
~	陣列元素逐一和純量 NOT
&&	純量 AND
||	純量 OR

基本原理 -流程控制(Flow Control)

指令陳述	描述
if	if（必要時連同 else 與 elseif）根據指定的邏輯條件執行指令陳述。
for	對一組指令陳述執行一指定的次數。
while	依據特定的邏輯條件，執行不定次數的指令陳述。
break	停止 for 或 while 迴圈的執行。
continue	將控制權傳給 for 或 while 迴圈的下一個迴圈，跳過迴圈主體中剩下的指令陳述。
switch	switch 連同 case 和 otherwise，依據特定的值或字串執行不同組群的指令陳述。
return	回到呼叫函數繼續執行。
try…catch	在執行期間檢測到錯誤時，改變流程控制。

基本原理 - 陣列索引(Array Indexing)

- MATLAB 支援許多功能強大的索引技術來簡化陣列操作及改善程

```
>> v = [1 3 5 7 9]
v =
    1 3 5 7 9
>> v(2)
ans =
    3
```

使用**轉置運算子 (transpose operator)** (.') 將列(行)向量轉成行(列)向量:

```
>> w = v.'
w =
    1
    3
    5
    7
    9
```

基本原理 - 陣列索引(Array Indexing)

存取一區塊 (block) 的元素時，使用 MATLAB 的冒號 (:)，例如，要讀取向量 v 的前三個元素，我們可以寫

```
>> v(1:3)
ans =
    1    3    5
```

同樣地，讀取第三個到最後一個元素可以寫：

```
>> v(3:end)
ans =
    5    7    9
```

其中 end 表示向量的最後一個元素。我們也可以將一個向量當作另一個向量的索引，比如

```
>> v([1 4 5])
ans =
    1    7    9
```

基本原理 - 陣列索引(Array Indexing)

除此之外，索引並不限於是連續的元素，例如

```
>> v(1:2:end)
ans =
     1     5     9
```

其中 `1:2:end` 是從 1 開始，每次索引加 2，直到最後一個元素。

在 MATLAB 中可以在中括號內將一序列的列向量以分號隔開來表示一個矩陣，例如

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

產生一個 3×3 的矩陣

```
A =
     1     2     3
     4     5     6
     7     8     9
```

基本原理 - 陣列索引(Array Indexing)

選取矩陣中的元素的方式和在向量中選取是一樣的，但是需要兩個索引：一個選定列位置，另一個選定行位置。我們也可以使用冒號當作索引來選取整列、整行或矩陣：

```
>> A(2,:)
ans =
     4     5     6

>> sum(A(:))
ans =
     45
```

函數 sum 計算引數的行的總和，單獨一個冒號將 A 轉成一個行向量，再將和傳給 sum。

基本原理 - 陣列索引(Array Indexing)

索引的另一種形式是**邏輯索引 (logical indexing)**，邏輯索引的表示法為 A(D)，其中 A 是一個陣列而 D 是一個與 A 大小一樣的邏輯陣列。A(D) 的意思是在 A 中將與 D 相對位置為 1 的元素抽取出來，例如

```
>> D = logical([1 0 0; 0 0 1; 0 0 0])
D =
     1     0     0
     0     0     1
     0     0     0

>> A(D)
ans =
     1
     6
```

線性索引是使用單獨一個下標來索引一個矩陣或較高維度的陣列，對一個 $M \times N$ 的矩陣(見 1.7.5 節)，元素 (r, c) 可以透過 $r + M(c - 1)$ 來存取，因此 A(2,3) 可以由 A([8]) 或 A(8) 來選取。

基本原理 -函數處理權、基層陣列與結構

- 函數處理權(function handle)是一個MATLAB的資料型態
  - 它包含用來引用函數的資訊
  - 一個使用處理權的主要好處是在呼叫其它函數@時可以將函數處理權當作引數來傳遞
  - 函數處理權可以在重覆運算時改善執行效能，並且可以傳給其它函數，它們可以存在資料結構或檔案中以供後續使用

基本原理 -函數處理權、基層陣列與結構

- 函數處理權有兩種不同的型態，但都由函數處理權運算子@ 來建立
  - 第一種函數處理權稱為**命名**（named 或稱簡單(simple)）函數處理權(function handle)，它是在運算子@之後緊接函數名稱

```
>> f = @sin
f =
    @sin
```

函數 sin 可以由呼叫函數處理權 f 來直接被呼叫：

```
>> f(pi/4)
ans =
    0.7071

>> sin(pi/4)
ans =
    0.7071
```

基本原理 -函數處理權、基層陣列與結構

- 第二種函數處理權稱為**匿名**函數處理權(anonymous functionhandle)，由MATLAB 的表示式取代函數名稱，建立一個匿名函數處理權的通式是：

```
@(input-argument-list) expression
```

例如，下列的匿名函數處理權將輸入平方：

```
>> g = @(x) x.^2;
```

而下列的匿名函數處理權計算兩個變數平方和的平方根：

```
>> r = @(x, y) sqrt(x.^2 + y.^2);
```

匿名函數處理權被呼叫的方式與命名函數處理權一樣。

基本原理 -函數處理權、基層陣列與結構

- 基層陣列(cell array) 提供一個將混合的物件（比方說數值、字元、矩陣或其它基層陣列）組合在一個變數之下的方法

  例如，

  假設要處理(1) 大小為512 x 512 類別uint8 的影像f；(2) 188 x 2陣列方式的二維座標b；

  和(3) 兩個字元名稱的基層陣列：

  ```
  char_array={'area','centroid'}
  ```

  這三種不同的資料個體可以使用基層陣列整合在一個變數C 中

  ```
  C = {f, b, char_array}
  ```

基本原理 -函數處理權、基層陣列與結構

在提示區鍵入 C 將輸出下面的結果：

```
>> C
C =
    [512x512 uint8]    [188x2 double]    {1x2 cell}
```

- 要看基層陣列內某一元素的完整內容，可以指出該元素在基層陣列的位置，例如，要看 char_array 的內容可以鍵入

```
>> C{3}
ans =
    'area' 'centroid'
```

基本原理 -函數處理權、基層陣列與結構

- 如果使用小括號取代大括號將產生變數的描述：

```
>> C(3)
ans =
    {1x2 cell}
```

- 基層陣列包含引數的**複製品**(copies) 而不是引數的**指標**(pointers)。因此，前面基層陣列的例子C的引數內容不隨該引數內容的改變而改變

基本原理 -函數處理權、基層陣列與結構

- 結構(structure) 類似基層陣列
  - 它是將不同的資料整合在單一變數內，不像基層陣列是由數字來定引數的位置，結構是用使用者定義的名稱來定位置，這個稱為**欄位**(field)
    例如，假設f 是一張輸入影像，我們寫

```
function s = image_stats(f)
s.dm = size(f);
s.AI = mean2(f);
s.AIrows = mean(f, 2);
s.AIcols = mean(f, 1);
```

- s是一個結構
- 欄位
  (1) dm（1×2 的向量）
  (2) AI（純量）
  (3) Airows（M×1 的向量
  (4) Aicols（1×N 的向量），其中M 和N分別是影像的列數和行數

要注意的是結構和欄位是用點來區隔，欄位可以隨意的命名，但開頭一定要非數值的字元

基本原理 - 程式碼最佳化(Code Optimization)

- MATLAB 是被設計來作為陣列運算的程式語言， 兩個重要的程式碼最佳化：
  事先配置陣列(preallocating arrays) 和
  向量化迴圈(vectorizing loops)

- 事先配置陣列
  - 在進入for 迴圈之前，起始陣列計算陣列的元素
    假設我們要建立一個MATLAB 的函數來計算

$$f(x) = \sin(x/100\pi)$$

其中x = 0, 1, 2, ..., M – 1，下列是函數的第一種版本：

基本原理 - 程式碼最佳化(Code Optimization)

```
function y = sinfun1(M)
x = 0:M - 1;
for k = 1:numel(x)
   y(k) = sin(x(k) / (100*pi));
end
```

numel(x) 回傳陣列x 的元素個數

M = 5 的輸出為

```
>> sinfun1(5)
ans =
    0    0.0032    0.0064    0.0095    0.0127
```

MATLAB 函數 tic 和 toc 可以用來測量函數執行的時間，我們呼叫 tic，然後呼叫 sinfun1，最後再呼叫 toc：

```
>> tic; sinfun1(100); toc
Elapsed time is 0.001205 seconds.
```

如果這三個指令分別在分開的三列鍵入，則執行時間將包含後面兩列鍵入的時間

基本原理 - 程式碼最佳化(Code Optimization)

- 前一段的時間函數呼叫可能在時間測量上會有很大的差異，特別是在命令提示區的時候。
  例如，重覆前面的呼叫會有不同的結果：

```
>> tic; sinfun1(100); toc
Elapsed time is 0.001197 seconds.
```

函數timeit在測量函數呼叫的時間是可重複的，timeit的語法為

```
s = timeit(f)
```

其中f是要被量測時間的函數處理權，s是量測的時間（以秒計），函數處理權f被呼叫是沒有輸入引數的

基本原理 - 程式碼最佳化(Code Optimization)

- 當M =100 時，我們使用timeit 來量測sinfun1的時間：

```
>> M = 100;
>> f = @() sinfun1(M);
>> timeit(f)

ans =

    8.2718e-005
```

- 我們繼續使用timeit來測量sinfun1的時間，當M=500、1000、1500、...、20000：

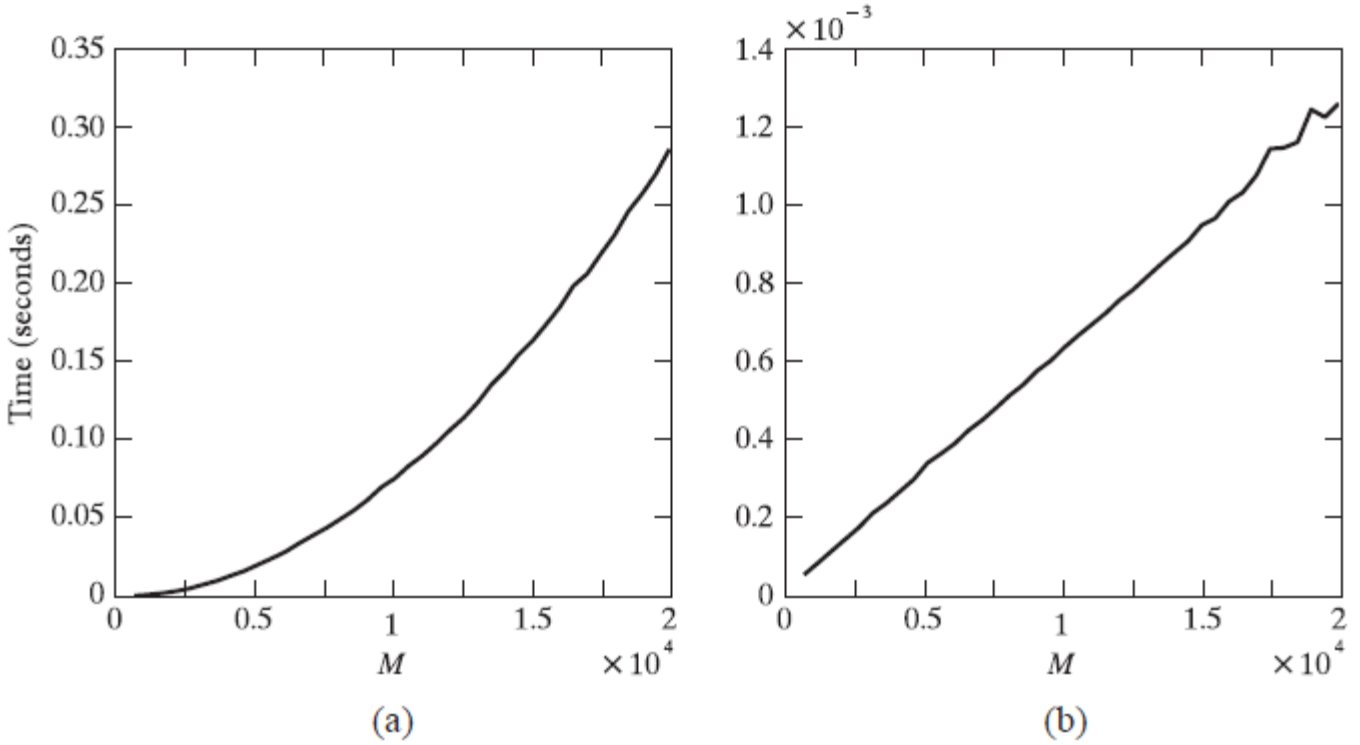
```
M = 500:500:20000;
for k = 1:numel(M)
   f = @() sinfun1(M(k));
   t(k) = timeit(f);
end
```

基本原理 - 程式碼最佳化(Code Optimization)

- 我們期望計算sinfun1(M) 的時間與M 的大小形成比例關係，但是圖1.3(a)卻顯示它是以M^2 的函數在成長
  - 原因是sinfun1.m 的輸出變數y 每一迴圈以一個元素來成長
  - MATLAB 可以自動處理這種陣列隱含的成長，但是每次陣列成長時，它必須事先配置新的記憶體空間，並且複製先前的陣列元素，這樣經常做記憶體配置和複製相當耗時，甚至比計算sin 本身還花更多的時間

基本原理 - 程式碼最佳化(Code Optimization)

圖1.3



sinfun1 以M為函數的近似執行時間

sinfun2的近似執行時間，小誤差是在由記憶體分頁間隔變化引起的

基本原理 - 程式碼最佳化(Code Optimization)

- MATLAB 編輯器對這種執行效能的問題提出解決方案，它會對sinfun1.m 提出通報：

'y'可能在迴圈內增長，為了速度考慮事先配置。

- 事先配置y意指在迴圈開始之前起始所期望的輸出大小，通常事先配置是使用一個對函數zeros的呼叫
- sinfun2.m就是使用事先配置：

```
function y = sinfun2(M)
x = 0:M-1;
y = zeros(1, numel(x));
for k = 1:numel(x)
   y(k) = sin(x(k) / (100*pi));
end
```

基本原理 - 程式碼最佳化(Code Optimization)

比較 sinfun1(20000) 和 sinfun2(20000) 的時間：

```
>> timeit(@() sinfun1(20000))
ans =
    0.2852

>> timeit(@() sinfun2(20000))
ans =
    0.0013
```

- 使用事先配置的時間約快了220 倍，圖1.3(b) 顯示了sinfun2 的時間與M 的大小形成比例關係（注意圖1.3(a) 與(b) 的時間尺度不一樣）。

基本原理 - 程式碼最佳化(Code Optimization)

- MATLAB 的向量化(vectorization) 迴圈是將矩陣/ 向量運算子、索引技術、MATLAB 或工具箱函數組合起來消除整個迴圈的技巧，
  - 我們第三個版本sinfun3 就是利用sin 可以用陣列當輸入而不以純量當輸入這一事實來設計。因此sinfun3 不使用迴圈計算：

```
function y = sinfun3(M)
x = 0:M-1;
y = sin(x ./ (100*pi));
```

基本原理 - 程式碼最佳化(Code Optimization)

範例1.1 向量化的說明及函數meshgrid的介紹

根據下面的方程式寫兩個MATLAB 函數來作影像合成：

$$f(x,y) = A\sin(u_0 x + v_0 y)$$

第一個函數 twodsin1 使用兩個巢式 for 迴圈來計算 f：

```
function f = twodsin1(A, u0, v0, M, N)
f = zeros(M, N);
for c = 1:N
    v0y = v0 * (c - 1);
    for r = 1:M
        u0x = u0 * (r - 1);
        f(r, c) = A*sin(u0x + v0y);
    end
end
```

基本原理 - 程式碼最佳化(Code Optimization)

觀察在 for 迴圈前面的事先配置的步驟 f=zeros(M,N)，我們使用 timeit 來測量此函數建立一個大小為 512 × 512 的正弦曲線影像的時間：

```
>> timeit(@() twodsin1(1, 1/(4*pi), 1/(4*pi), 512, 512))
ans =
    0.0471
```

沒有事先配置將花1.9826 秒的時間，而此函數慢了將近42 倍。

基本原理 - 程式碼最佳化(Code Optimization)

我們使用 imshow 的自動範圍語法([])來顯示此影像（圖 1.4）：

```
>> f = twodsin1(1, 1/(4*pi), 1/(4*pi), 512, 512);
>> imshow(f, [ ])
```

第二個函數我們使用 MATLAB 中非常有用的函數 meshgrid 來向量化，meshgrid 的語法為

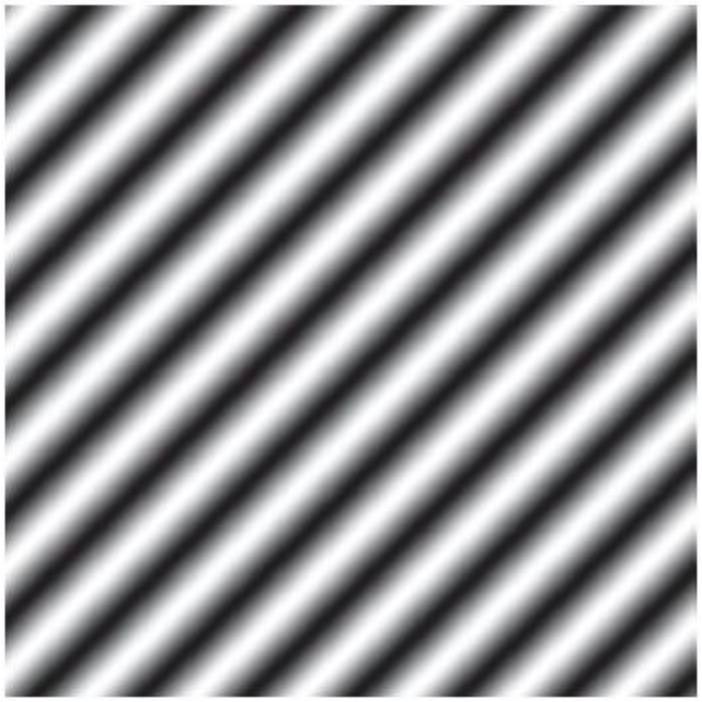
```
[C, R] = meshgrid(c, r)
```

c 和 r 分別為水平座標向量（行）和垂直座標向量（列）的輸入引數，函數 meshgrid 將座標向量轉為兩個陣列 C 和 R，這兩個陣列可以用來計算兩個變數的函數。例如，使用下面的指令來計算 $z = x + y$，其中 x 是 1 到 3 的整數，y 是 10 到 14 的整數：

基本原理 - 程式碼最佳化(Code Optimization)

```
>> [X, Y] = meshgrid(1:3, 10:14)
X =
     1     2     3
     1     2     3
     1     2     3
     1     2     3
     1     2     3
```

圖1.4
範例1.1 的正弦曲線函數影像



基本原理 - 程式碼最佳化(Code Optimization)

```
Y =
     10    10    10
     11    11    11
     12    12    12
     13    13    13
     14    14    14

>> Z = X + Y
Z =
     11    12    13
     12    13    14
     13    14    15
     14    15    16
     15    16    17
```

基本原理 - 程式碼最佳化(Code Optimization)

最後，我們使用 meshgrid 來重寫不具有迴圈的二維正弦函數：

```
function f = twodsin2(A, u0, v0, M, N)
r = 0:M - 1; % 行座標。
c = 0:N - 1; % 列座標。
[C, R] = meshgrid(c, r);
f = A*sin(u0*R + v0*C);
```

使用 timeit 來測量它的速度：

```
>> timeit(@() twodsin2(1, 1/(4*pi), 1/(4*pi), 512, 512))
ans =
    0.0126
```

向量化的函數大約可減少 50% 的時間。

基本原理 - 程式碼最佳化(Code Optimization)

- 對很多熟悉矩陣和向量的使用者來說向量化程式碼往往比以迴圈為基礎的程式碼容易讀，看下面的例子來比較twodsin2中的

```
f = A*sin(u0*R + v0*C);
```

與twodsin1 的

```
for c = 1:N
   v0y = v0*(c - 1);
   for r = 1:M
      u0x = u0 * (r - 1);
      f(r, c) = A*sin(u0x + v0y);
    end
end
```

很明顯地第一個比較簡明，但第二個才是確實進行的方式。

第二章

強度轉換和空間濾波

Outline

- 背景
- 強度轉換函數
- 直方圖處理和繪圖函數
- 空間濾波
- 影像處理工具箱標準的空間濾波器
- 使用模糊技巧的強度轉換和空間濾波

背景

背景

- 空間域處理方式可表示成

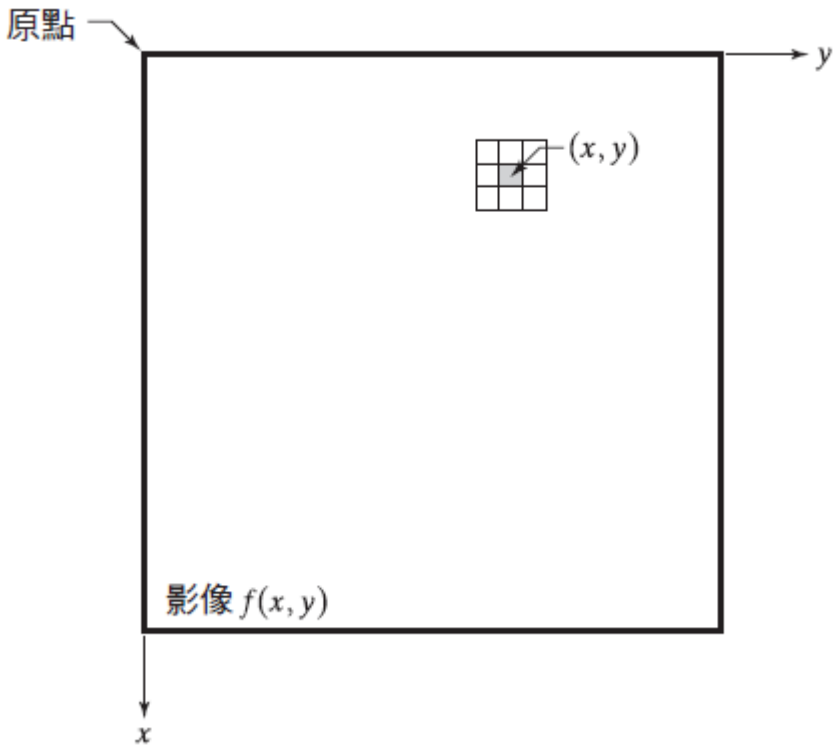
$$g(x, y) = T[f(x, y)]$$

其中$f(x, y)$是指輸入的影像，$g(x, y)$是指輸出的影像，而T是一個對f作轉換的運算子（或稱為轉換式），它定義在(x, y)附近的點

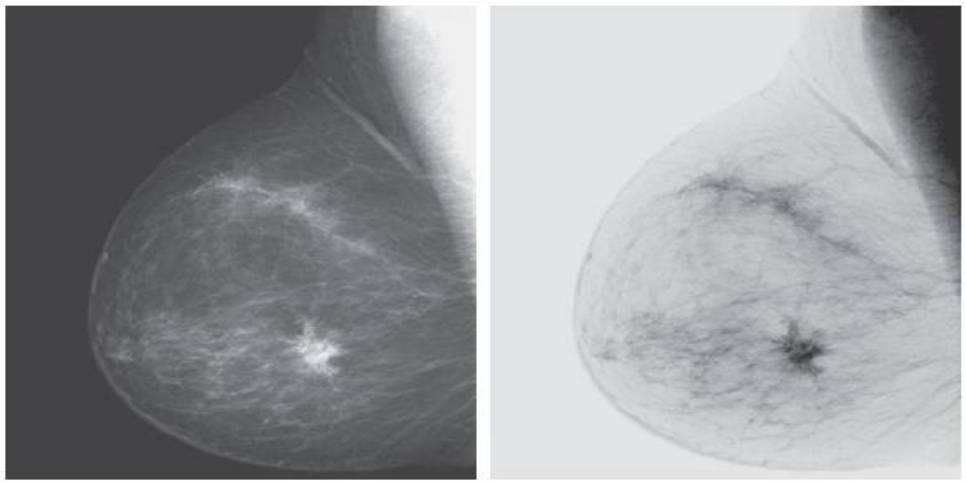
  - T可以操作於一張影像也可以操作於一組影像
  - 定義關於點(x, y)的空間域鄰近區域(spatial neighborhoods)的主要方法是使用一個以(x, y)為中心的正方型或長方形區域

背景

圖**2.1** 在影像中以(*x*,*y*)為中心及其鄰域，它的大小為3×3



強度轉換函數



強度轉換函數

- 強度(intensity) 或灰階轉換函數(gray-level transformation function)
  - 當處理單色（如灰階）影像時，這兩種專有名詞“強度”與“灰階轉換函數”可互相交替使用
  - x 和y 是空間座標(spatial coordinates)
  - 輸出值僅與點的強度值有關，不是與點的鄰近區域有關，所以強度轉換函數通常以簡化形式寫為

$$s = T(r)$$

其中r代表f 在(x, y)的強度，s 是g 在(x, y) 的強度

強度轉換函數 – 函數imadjust和stretchlim

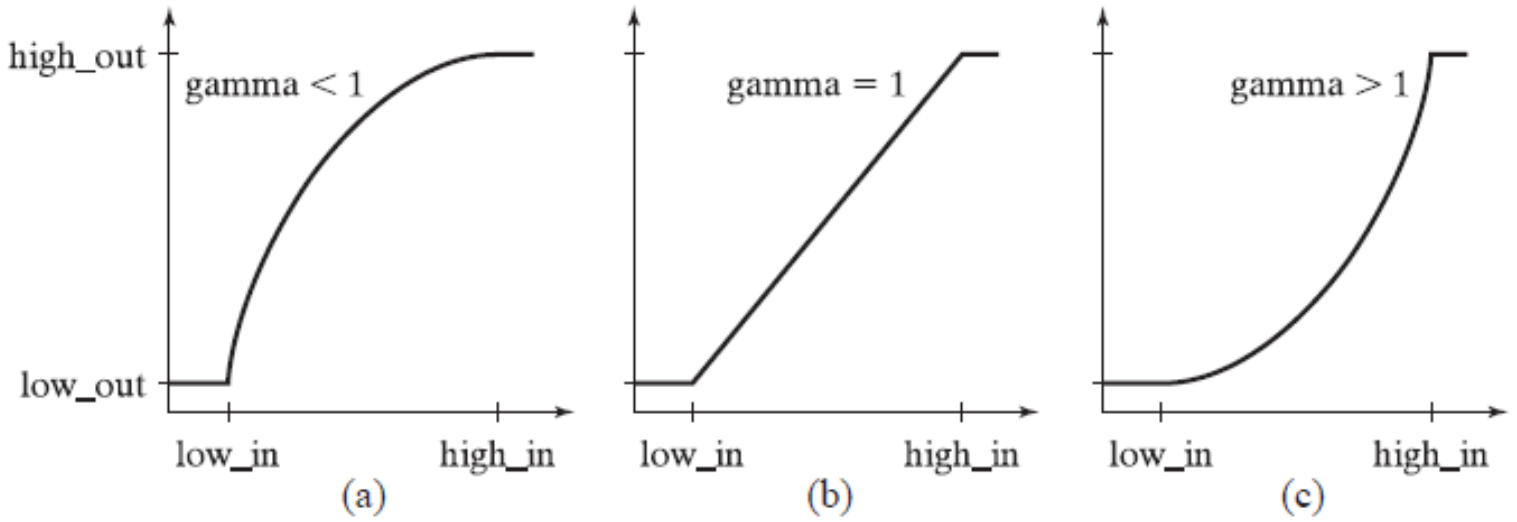
- 對灰階影像強度轉換而言，函數 imadjust 是基本的影像處理工具箱函數，它的基本語法為

```
g = imadjust(f, [low_in high_in], [low_out high_out], gamma)
```

  - 此函數將影像f 的強度值對映到影像g，使得在 low_in 和high_in 之間的值對映到low_out 和 high_out 之間
  - 輸入影像的類別可以是uint8、uint16、int16、single 或double，輸出影像與輸入影像的類別是一樣的
  - 所有對函數 imadjust的輸入除了f 和gamma 之外，都被指定在0 到 1 之間

強度轉換函數

- 參數gamma 是用來指定f 的強度值對映到g 的曲線形狀



強度轉換函數

- 範例 2.1 函數imadjust的使用
  - 圖2.3(a) 是數位乳腺造影的灰階影像f ，它的負片顯示在圖2.3(b)，是使用下列指令得到的

```
>> g1 = imadjust(f, [0 1], [1 0]);
```

  - 圖2.3(c) 是使用下面指令得到的結果

```
>> g2 = imadjust(f, [0.5 0.75], [0 1]);
```

    - 此指令將原為0.5 到0.75 的灰階擴展到整個[0,1] 的範圍
  - 最後使用指令

```
>> g3 = imadjust(f, [ ], [ ], 2);
```

強度轉換函數

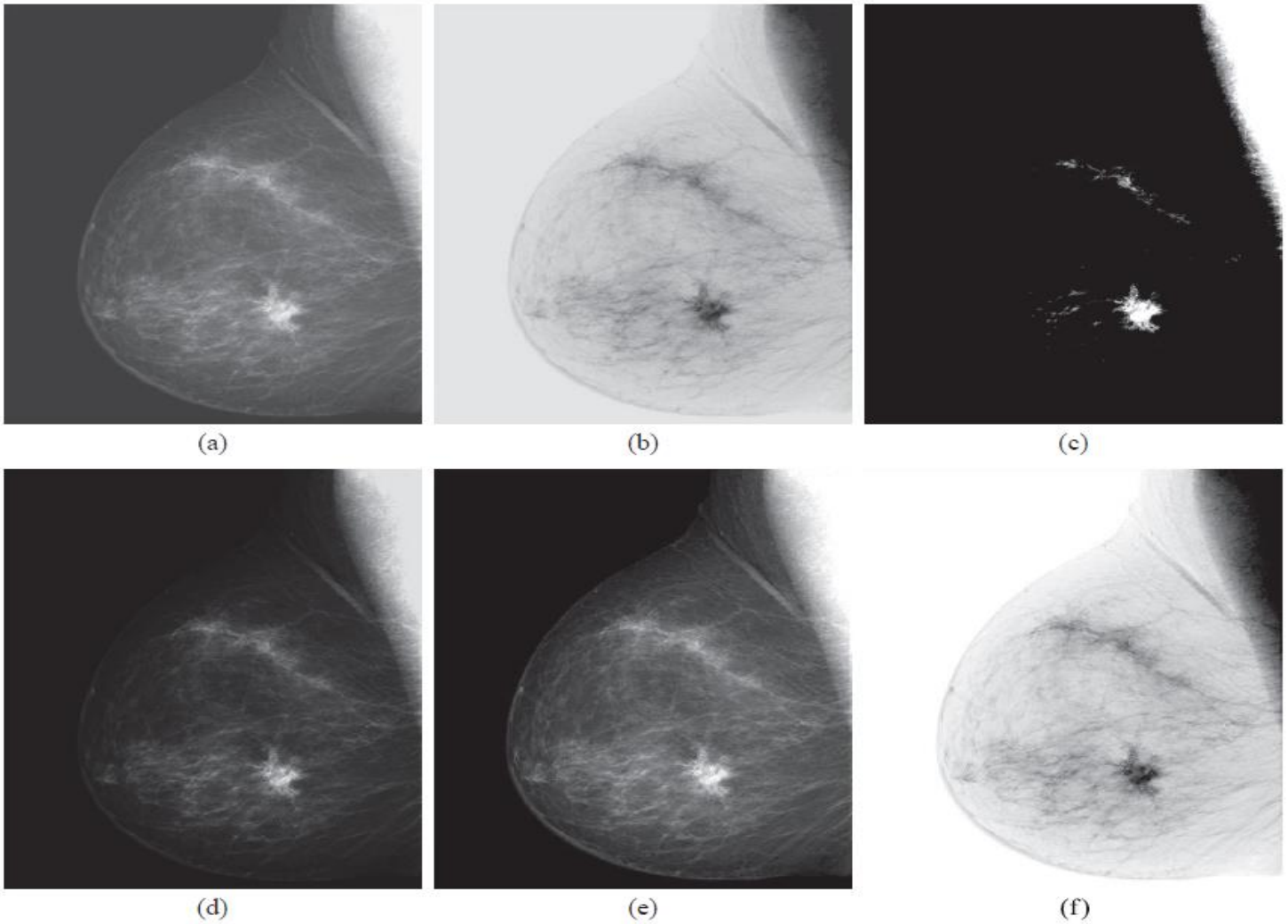


圖 2.3

(a) 原始數位乳腺造影；(b) 負片；(c) 使用 [0.5, 0.75] 擴展強度的結果；(d) 使用 gamma = 2 強化的結果；(e) 和 (f) 使用函數 stretchlim 對函數 imadjust 的自動輸入（原始影像採自 G. E. Medical 系統）

強度轉換函數–對數與對比延展轉換

- 對於動態範圍操作而言，對數(logarithmic)與對比延展(contraststretching)轉換是基本工具。對數轉換使用下面算式來實現

```
g = c*log(1 + f)
```

其中c 是常數而f 是浮點數

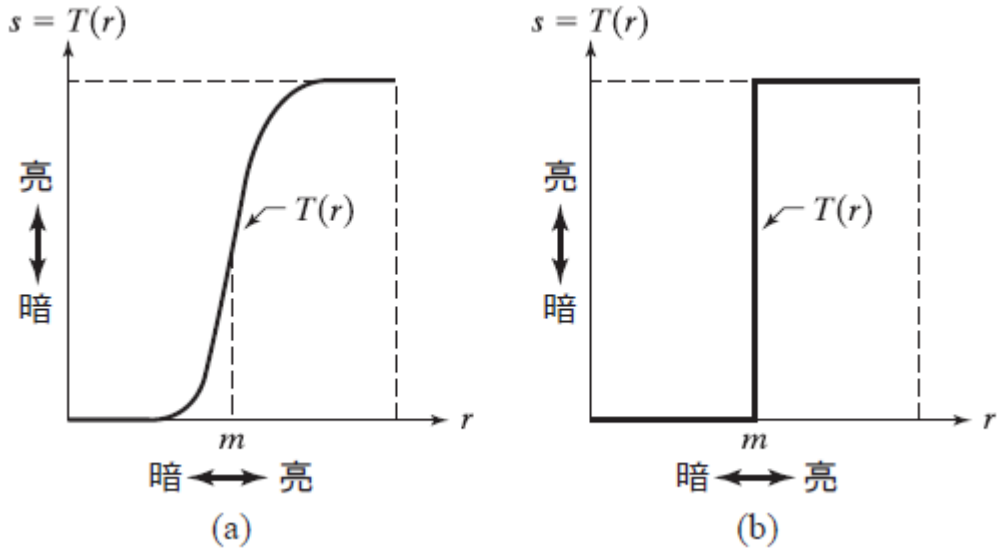
- 當執行log 轉換時，將壓縮值的結果回復為整個顯示的範圍通常是需要的。譬如以8 位元而言，在MATLAB 中最簡單的方法是使用下面的指令

```
>> gs = im2uint8(mat2gray(g));
```

- 使用mat2gray 產生介於[0, 1] 的值，而使用im2uint8 則把這些值轉換為[0, 255] 之間，此將影像轉換為uint8 的格式

強度轉換函數–對數與對比延展轉換

圖2.4 (a) 對比延伸轉換函數；(b) 臨界函數



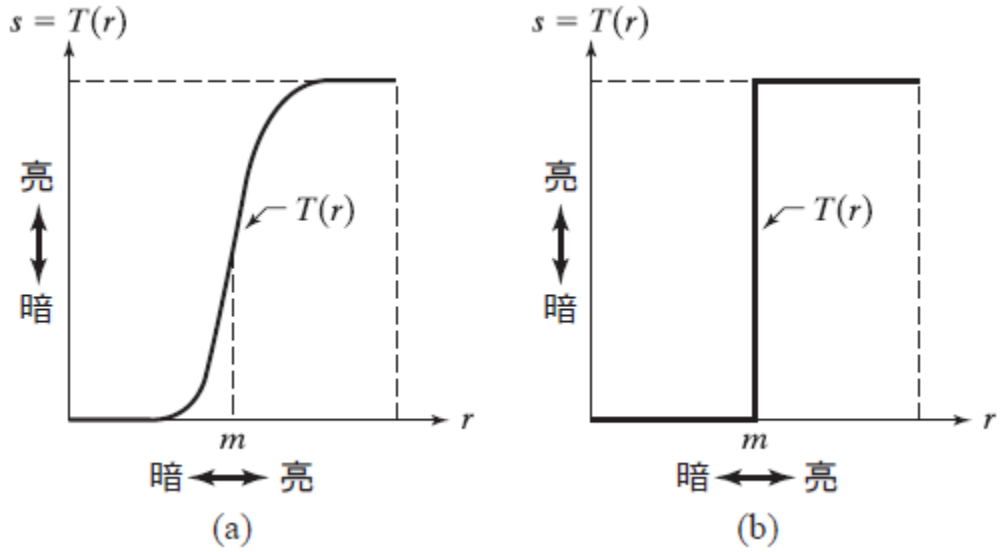
$$s = T(r) = \frac{1}{1+(m/r)^E}$$

其中r 指的是輸入影像的強度，s 指的是輸入影像相對應的強度值，而E 是控制函數的斜率

強度轉換函數–對數與對比延展轉換

- 對比延伸(contrast-stretching) 轉換函數

圖2.4 (a) 對比延伸轉換函數；(b) 臨界函數



$$s = T(r) = \frac{1}{1+(m/r)^E}$$

其中r 指的是輸入影像的強度，s 指的是輸入影像相對應的強度值，而E 是控制函數的斜率

強度轉換函數–對數與對比延展轉換

- 對浮點影像在MATLAB 實作的方程式如下

```
g = 1./(1 + (m./f).^E)
```

  - 因為g的極限值是1，當處理此類型的轉換時，其輸出值不能超過[0,1]的範圍。在圖2.4(a) 所示的曲線是以E = 20 得到的

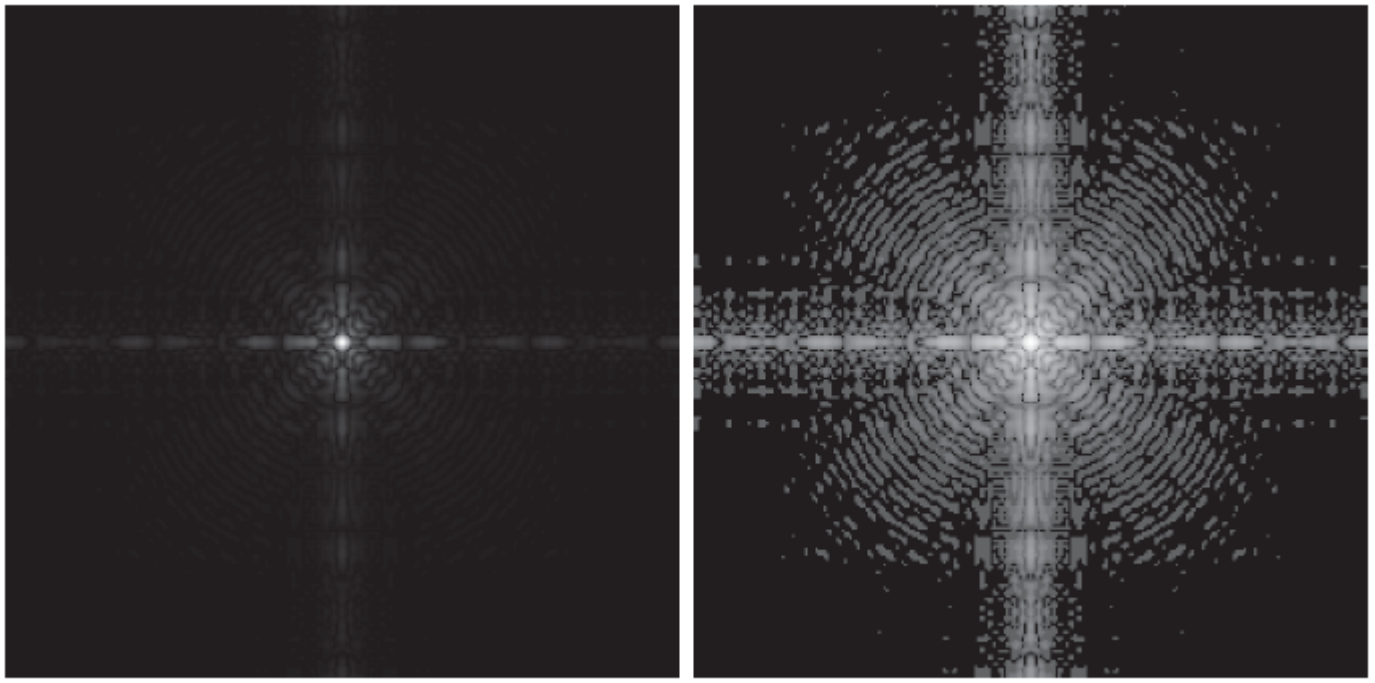
強度轉換函數–對數與對比延展轉換

- 範例 2.2 使用log 轉換降低動態範圍
  - 圖2.5(a) 是一個範圍介於0 到106 的傅立葉頻譜，顯示在線性比例調整為8 位元的系統上。
    圖2.5(b) 是用下列指令得到的結果

```
>> g = im2uint8(mat2gray(log(1 + double(f))));
>> imshow(g)
```

強度轉換函數–對數與對比延展轉換

圖2.5 (a) 傅立葉頻譜；(b) 使用log 轉換的結果



(a) (b)

強度轉換函數–指定任意的強度轉換

- 使用特定轉換函數來轉換影像的強度
  - 函數interp1
    - 令T 表示一個含有函數轉換值的行向量。以一張8 位元影像來說，T (i + 1) 表示輸入影像強度為i 的輸出值，i = 0, 1, 2, ..., 255。

```
g = interp1(z, T, f)
```

其中f是輸入影像，g 是輸出影像z是與T相同長度的行向量，有以下的格式：

```
z = linspace(0, 1, numel(T))';
```

強度轉換函數-強度轉換的一些公用M函數

- 處理各種數量的輸入及（或）輸出
  - 函數nargin

```
n = nargin
```

    - 將會回傳輸入到M 函數的引數個數
  - 函數nargout

```
n = nargout
```

    - 在提示區執行下列的M 函數：

```
>> T = testhv(4, 5);
```

在這個函數主體內使用nargin 將回傳2，而使用nargout 將回傳1。

強度轉換函數-強度轉換的一些公用M函數

- 函數nargchk 可以用在M 函數的主體來檢測被傳遞的引數個數是否正確，它的語法是

```
msg = nargchk(low, high, number)
```

  - 如果number 小於low， 此函數將回傳訊息Not enough input arguments
    如果number 大於high， 則回傳 Too many input arguments
    如果number 介於low 和high（含）之間，nargchk回傳一個空矩陣
  - 如果不正確的引數個數被輸入時，函數nargchk常可經由函數error 來停止執行，函數nargin 決定輸入的引數個數

強度轉換函數-強度轉換的一些公用M函數

- 能夠將輸入和（或）輸出的引數個數當作變數來撰寫函數是有用的，對於此，我們使用變數varargin 和varargout ，例如

```
function [m, n] = testhv3(varargin)
```

接受一個輸入個數的變數給函數testhv3.m

- 從函數testhv4 回傳一個輸出個數的變數

```
function [varargout] = testhv4(m, n, p)
```

- 如果函數testhv3 有一固定的輸入引數x，之後接一個輸入引數的變數，則

```
function [m, n] = testhv3(x, varargin)
```

當函數被呼叫時，將可由使用者引發varargin 啟動第二個輸入引數

強度轉換函數-強度轉換的一些公用M函數

- 當varargin 被當作是函數的輸入引數時，MATLAB 設定它為一個基層矩陣
- 一個重要的概念是呼叫此函數時可包含一組混合的輸入。例如，假定函數testhv3 的程式碼能夠處理，則一個有混合輸入且完全可接受的語法為

```
>> [m, n] = testhv3(f, [0  0.5  1.5], A, 'label');
```

其中f 為一張影像，下一個引數是長度3 的列向量，A 是一個矩陣，而'label' 是一個字元字串

強度轉換函數-強度轉換的一些公用M函數

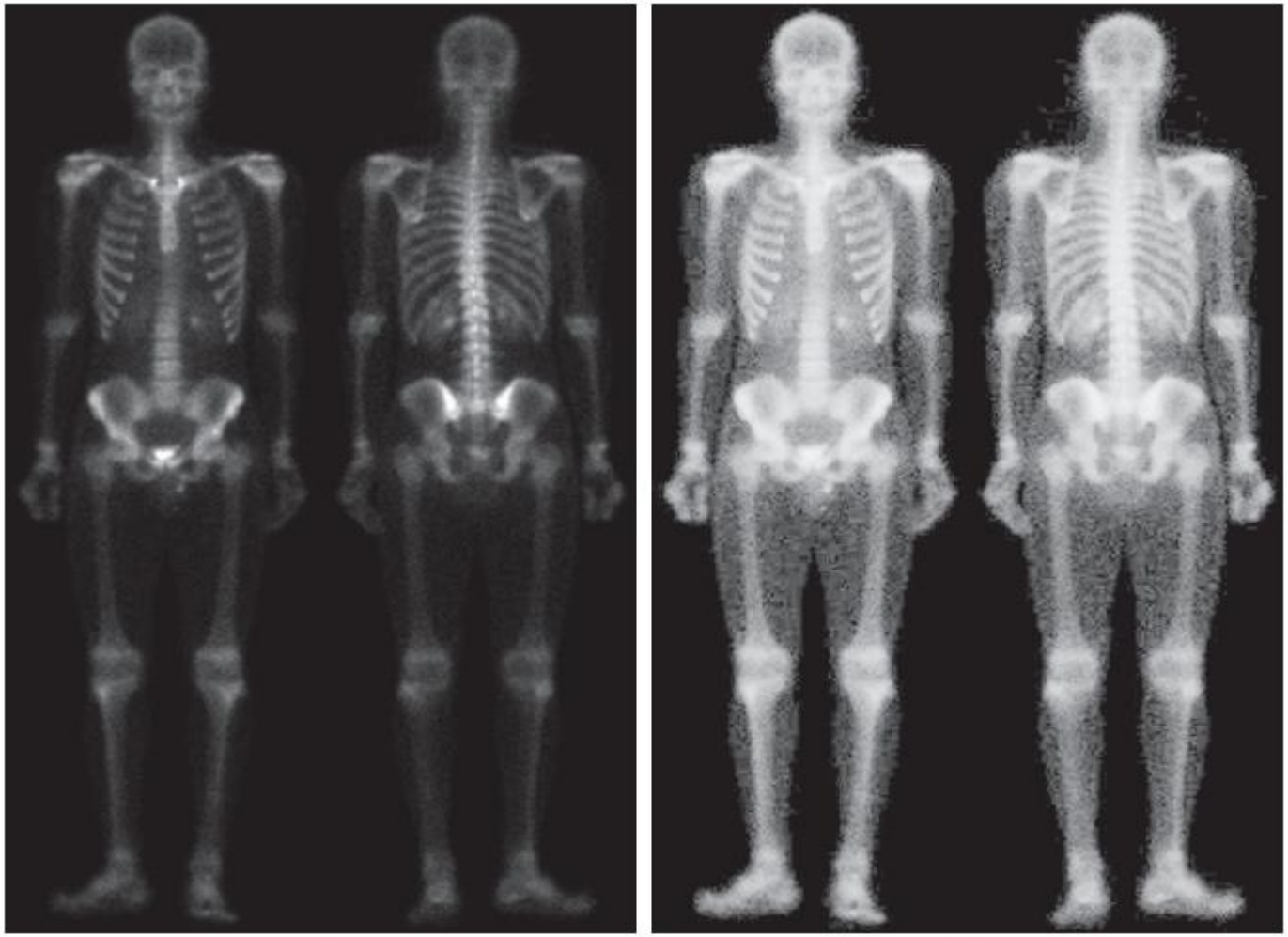
- 範例 2.3 函數intrans的說明
  - 圖2.6(a) 來說明intrans，這是一張用對比延展來強化骨骼結構的理想影像，它的結果（圖2.6(b)）是呼叫intrans 得到的：

```
>> g = intrans(f, 'stretch', mean2(tofloat(f)), 0.9);
>> figure, imshow(g)
```

    - 函數mean2 直接在函數呼叫內計算f 的平均值，此平均值給m 使用。依據輸入m 的要求，為了調整影像f 的值到[0, 1] 的範圍（平均值當然也在此範圍內），使用tofloat 將f 轉為浮點，而E 的值被交互決定

強度轉換函數–強度轉換的一些公用M函數

圖2.6 (a) 骨骼掃描影像；(b) 使用對比延展轉換的影像強化



(a) (b)

強度轉換函數-強度轉換的一些公用M函數

- 強度調整的M 函數
  - 通常像素值會從負值到正值，當我們想要以8 位元或16 位元儲存或觀看影像時就變成一個議題
    - 因為一般都會將影像調整為[0, 255] 或[0, 65535]
  - 函數gscale

```
g = gscale(f, method, low, high)
```

其中f 是要調整的影像，method 的值可以有'full8'(預設)、'full16' 和 'minmax'

- 'full8' 將輸出調到[0, 255], low 和high 被忽略
- 'full16' 將輸出調到[0, 65535], low 和high 被忽略
- 'minmax'，則low 和high 必須在[0, 1] 之間，而輸出被調到[low, high]