Image Analysis and Computer Vision
Lecture 2、Fast Matlab Tutorial

Weiqiang Wang

School of Computer and Control Engineering, UCAS

October 12, 2015

Outline

1. Background on Matlab
2. Matlab Working Environment
3. Matlab and Image Processing Toolbox
4. Matlab programming

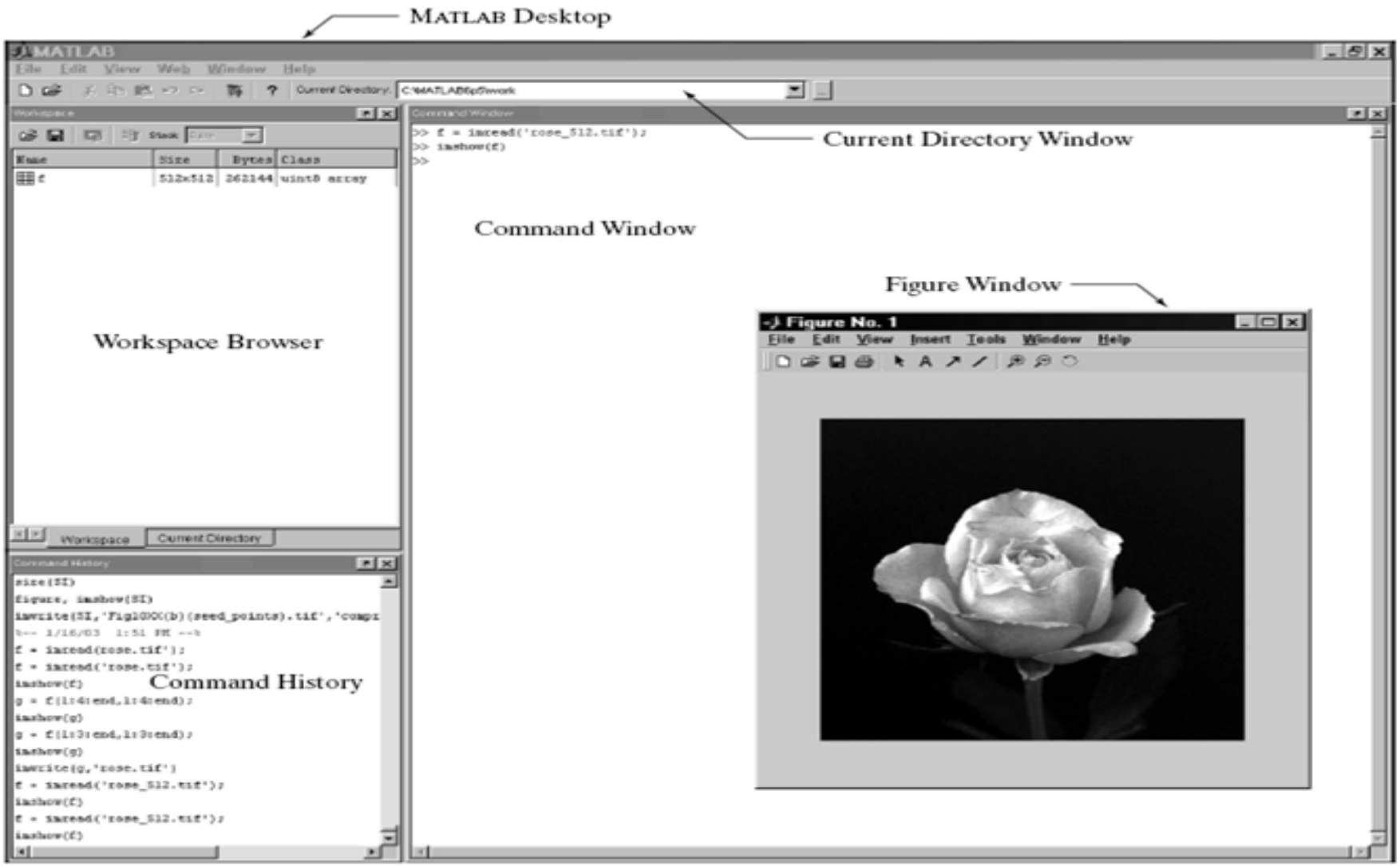
What is Matlab?

- Matlab stands for matrix laboratory, and it is developed by the MathWorks, Inc. http://www.mathworks.com/
- MATLAB is an interactive system whose basic data element is an array.
- It is a high-performance language for technical computing
  - This allows formulating solutions to many technical computing problems, especially those involving matrix representations
  - in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran
- It integrates computation, visualization, and programming in an easy-to-use environment.
- It is a strong computational tool for university (mathematics, engineering, and science) and Industry (research, development, and analysis).

What is Matlab?

- Typical uses of Matlab include the following:
  - Math and computation
  - Algorithm development
  - Data acquisition
  - Modeling, simulation, and prototyping
  - Data analysis, exploration, and visualization
  - Scientific and engineering graphics
  - Application development, including graphical user interface building
- MATLAB contains a family of toolboxes for various applications.
  - The Image Processing Toolbox (IPT): extend the capability of the MATLAB environment for the solution of digital image processing problems
  - Other toolboxes that sometimes are used to complement IPT are the Signal Processing, Neural Network, Fuzzy Logic, and Wavelet Toolboxes

Matlab Working Environment



Current Directory and Search path

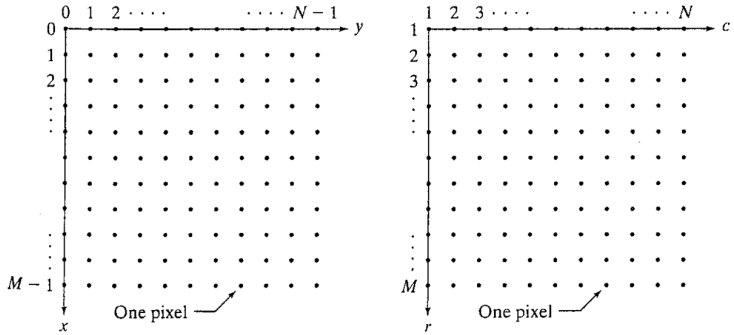
- Any file run in MATLAB must reside in the current directory or in a directory that is on the search path
- By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path
- Add or modify a search path, is to select **Set Path** from the **File** menu on the desktop

Using the MATLAB Editor to Create M-files

- The MATLAB editor is both a text editor specialized for creating M-files and a graphical MATLAB debugger.
- Source files are denoted by the extension .m
- The editor performs some simple checks and also uses color to differentiate between various elements of code
- The file must be in the current directory, or in a directory in the search path.

Digital Image Representation

- Coordinate Conventions



- Images as Matrices

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

- Images as Matrices in Matlab

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

Read an Image

- Image formats

Format Name	Description	Recognized Extensions
TIFF	Tagged Image File Format	`.tif`, `.tiff`
JPEG	Joint Photographic Experts Group	`.jpg`, `.jpeg`
GIF	Graphics Interchange Format†	`.gif`
BMP	Windows Bitmap	`.bmp`
PNG	Portable Network Graphics	`.png`
XWD	X Window Dump	`.xwd`

† `GIF` is supported by `imread`, but not by `imwrite`.

- Read an image using MATLAB IPT

```
f=imread('bird.jpg');
```

  - Imread reads the file from the current directory and, if fails, it tries to find the file in the search path
  - Please notice the semicolon(;). If erase it, the command window will show the image' s matrix value

Obtain the Information of an Image

1. size(f)

   ans = 230 352 3

2. whos f

   Name Size Bytes Class
   f 230x352x3 242880 uint8 array
   Grand total is 242880 elements using 242880 bytes

3. imfinfo('bird.jpg') show more info than whos

   ans = Filename: 'bird.jpg'
   FileModDate: '25-Sep-2002 19:00:16'
   FileSize: 7028
   Format: 'jpg'
   FormatVersion: ''
   Width: 352
   Height: 230
   BitDepth: 24
   ColorType: 'truecolor'
   FormatSignature: ''
   NumberOfSamples: 3
   CodingMethod: 'Huffman'
   CodingProcess: 'Sequential'
   Comment: {'ACD Systems Digital Imaging'}

Show or Write an Image

- imshow(f)
- imshow(f, G)
- imshow(f,[low,high])
- imshow(f,[])

An demo example 2.1

- imwrite(f,'filename')
- imwrite(f,'filename','ext')
- imwrite(f,'filename','quality',q)

  The quality q ranges from 0 to 100, the lower the number , the quality is worse, but the compression is higher

Image Types

- Intensity images/ grayscale image
  - An intensity image is a data matrix whose values have been scaled to represent intensities.
  - When the elements of an intensity image are of class uint8 or uint16, they have integer values in the range [0,255] or [0,65535].
  - If the image is of class double, the values are floating-piont numbers. Values of scaled, class double intensity images are in the range [0,1] by convention
- Binary images
  - A binary image is a logical array of 0s and 1s.
  - A numeric array is converted to binary using function logical.
  - If A contains elements other than 0s and 1s, use of the logical function converts all nonzero quantities to logical 1s and all entries with value 0 to logical 0s
- Indexed images
- RGB images

Indexed Image and RGB Image

- Image data can be either indexed or true color.
- An indexed image stores colors as an array of indices into the figure colormap.
- A true color image does not use a colormap; instead, the color values for each pixel are stored directly as RGB triplets.
- In MATLAB, the CData property of a truecolor image object is a three-dimensional (m-by-n-by-3) array. This array consists of three m-by-n matrices concatenated along the third dimension.

Image Type	Double-Precision Data (double Array)	8-Bit Data (uint8 Array) 16-Bit Data (uint16 Array)
Indexed (colormap)	Image is stored as a two-dimensional (m-by-n) array of integers in the range [1, length(colormap)]; colormap is an m-by-3 array of floating-point values in the range [0, 1].	Image is stored as a two-dimensional (m-by-n) array of integers in the range [0, 255] (uint8) or [0, 65535] (uint16); colormap is an m-by-3 array of floating-point values in the range [0, 1].
True color (RGB)	Image is stored as a three-dimensional (m-by-n-by-3) array of floating-point values in the range [0, 1].	Image is stored as a three-dimensional (m-by-n-by-3) array of integers in the range [0, 255] (uint8) or [0, 65535] (uint16).

Data Types

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range $[0, 255]$ (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range $[0, 65535]$ (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range $[0, 4294967295]$ (4 bytes per element).
`int8`	Signed 8-bit integers in the range $[-128, 127]$ (1 byte per element).
`int16`	Signed 16-bit integers in the range $[-32768, 32767]$ (2 bytes per element).
`int32`	Signed 32-bit integers in the range $[-2147483648, 2147483647]$ (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

Converting between Data Types and Image Types

- In general, we refer to an image as being a "data_image_type" image
- For instance, a statement discussing an "unit8 intensity image" is simply referring to an intensity image whose pixels are of data class unit8.
- Converting between data classes is straightforward. The general syntax is

$$B = \text{data\_class\_name}(A)$$

where data class-name is one of the names in the first column of data class table.
For example, suppose that A is an array of class uint8. A double precision array, B is generated by the command $B = \text{double}(A)$.
This conversion is used routinely throughout the book because MATLAB expects operands in numerical computations to be double precision, floating point numbers

Converting between Data Classes

- For any data types (uint8, logical...), conversion can be easily completed without precision loss.
- For an array of class double, any values outside the range of the destination class will be replaced by the corresponding boundary values.
- If converting a double matrix to an integer matrix, numbers in between are converted to integers by discarding their fractional parts
- If converting a double matrix to an logical matrix, the logical 1s are generated where the input array had nonzero values and logical 0s in places where the input array contained 0s.

Converting between Data Types and Image Types (cont.)

- The toolbox provides specific functions that perform the scaling necessary to convert between image classes and types.
- Functions in IPT for converting between image classes and types

Name	Converts Input to:	Valid Input Image Data Classes
`im2uint8`	`uint8`	`logical`, `uint8`, `uint16`, and `double`
`im2uint16`	`uint16`	`logical`, `uint8`, `uint16`, and `double`
`mat2gray`	`double` (in range $[0, 1]$)	`double`
`im2double`	`double`	`logical`, `uint8`, `uint16`, and `double`
`im2bw`	`logical`	`uint8`, `uint16`, and `double`

Converting between Data Types and Image Types (cont.)

- Function im2uint8 detects the data class of the input and performs all the necessary scaling for the toolbox to recognize the data as valid image data.
- For example, consider the following 2×2 image f of class double, which could be the result of an intermediate computation

```
f=
    -0.5   0.5
    0.75   1.5
G=im2uint8(f)
G=
       0   128
     191   255
```

Converting between Data Types and Image Types (cont.)

- Converting an arbitrary array of class double to an array of class double scaled to the range [0,1] can be accomplished by using function mat2gray whose basic syntax is

  G = mat2gray(A,[Amin,Amax])

  where image g has values in the range 0 (black) to 1 (white).The specified parameters Amin and Amax are such that values less than(including equal to) Amin in A become 0 in g,and values greater than(including equal to) Amax in A correspond to 1 in g.

  G = mat2gray(A)

  sets the values of Amin and Amax to the actual minimum and maximum values in A

Converting between Data Types and Image Types (cont.)

- For example

```
A=[128,300;-12,66.98];
G=mat2gray(A,[0,255])
 G=
    0.5020   1.0000
         0   0.2627
G=mat2gray(A)
G=
    0.4487   1.0000
         0   0.2531
```

Indexing

- Array Indexing

  MATLAB supports a number of powerful indexing schemes that simplify array manipulation and improve the efficiency of programs.

  - Vector Indexing
  - Matrix Indexing
  - Useful functions

- Vector indexing

  an array of dimension $1 \times N$ is called a row vector.

  - v=[1 2 3 4]
  - w=v'
  - w(1:end)
  - zeros(1,4)

Indexing(cont.)

- Matrix indexing

  Matrices can be represented conveniently in MATLAB as a sequence of row vectors enclosed by square brackets and separated by semicolons.

  - A=[1 2 3;4 5 6]
  - A=rand(4,4)
  - A=magic(5)
  - A=5*ones(3,3)
  - A(1,2)
  - A(:,2:4)
  - sum(A) is different from sum(A(:))
  - mean(A) and mean(A(:))
  - max(A) and max(A(:)) or max(max(A))
  - min(A) and min(A(:)) or min(min(A))

Useful functions

- Reshape a 3-by-4 matrix into a 2-by-6 matrix.

```
A=
     1  4  7  10
     2  5  8  11
     3  6  9  12
B = reshape(A,2,6)
B =
     1  3  5  7  9  11
     2  4  6  8  10  12
```

Introduction to M-Function Programming

- Operators
  - +,-,*, .* ,./,/,\,^,.^,
  - xor,any,all
  - is series functions, for example iscell,ischar
  - imadd,imsubtract,immultiply,imdivide
  - .....

Operator	Name	MATLAB Function	Comments and Examples
+	Array and matrix addition	plus(A, B)	a + b, A + B, or a + A.
–	Array and matrix subtraction	minus(A, B)	a – b, A – B, A – a, or a – A.
.*	Array multiplication	times(A, B)	C = A.*B, C(I, J) = A(I, J)*B(I, J).
*	Matrix multiplication	mtimes(A, B)	A*B, standard matrix multiplication, or a*A, multiplication of a scalar times all elements of A.
./	Array right division	rdivide(A, B)	C = A./B, C(I, J) = A(I, J)/B(I, J).
.\	Array left division	ldivide(A, B)	C = A.\B, C(I, J) = B(I, J)/A(I, J).
/	Matrix right division	mrdivide(A, B)	A/B is roughly the same as A*inv(B), depending on computational accuracy.
\	Matrix left division	mldivide(A, B)	A\B is roughly the same as inv(A)*B, depending on computational accuracy.
.^	Array power	power(A, B)	If C = A.^B, then C(I, J) = A(I, J)^B(I, J).
^	Matrix power	mpower(A, B)	See online help for a discussion of this operator.
.'	Vector and matrix transpose	transpose(A)	A.'. Standard vector and matrix transpose.
'	Vector and matrix complex conjugate transpose	ctranspose(A)	A'. Standard vector and matrix conjugate transpose. When A is real A.' = A'.
+	Unary plus	uplus (A)	+A is the same as 0 + A.
–	Unary minus	uminus (A)	–A is the same as 0 – A or –1*A.
:	Colon		Discussed in Section 2.8.

Function	Description
`imadd`	Adds two images; or adds a constant to an image.
`imsubtract`	Subtracts two images; or subtracts a constant from an image.
`immultiply`	Multiplies two images, where the multiplication is carried out between pairs of corresponding image elements; or multiplies a constant times an image.
`imdivide`	Divides two images, where the division is carried out between pairs of corresponding image elements; or divides an image by a constant.
`imabsdiff`	Computes the absolute difference between two images.
`imcomplement`	Complements an image. See Section 3.2.1.
`imlincomb`	Computes a linear combination of two or more images. See Section 5.3.1 for an example.

Operator	Name
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

TABLE 2.6
Relational operators.

Operator	Name
&	AND
|	OR
~	NOT

TABLE 2.7
Logical operators.

Function	Comments
`xor` (exclusive OR)	The `xor` function returns a 1 only if both operands are logically different; otherwise `xor` returns a 0.
`all`	The `all` function returns a 1 if all the elements in a vector are nonzero; otherwise `all` returns a 0. This function operates columnwise on matrices.
`any`	The `any` function returns a 1 if any of the elements in a vector is nonzero; otherwise `any` returns a 0. This function operates columnwise on matrices.

TABLE 2.8
Logical functions.

Function	Description
`iscell(C)`	True if `C` is a cell array.
`iscellstr(s)`	True if `s` is a cell array of strings.
`ischar(s)`	True if `s` is a character string.
`isempty(A)`	True if `A` is the empty array, `[ ]`.
`isequal(A, B)`	True if `A` and `B` have identical elements and dimensions.
`isfield(S, 'name')`	True if `'name'` is a field of structure `S`.
`isfinite(A)`	True in the locations of array `A` that are finite.
`isinf(A)`	True in the locations of array `A` that are infinite.
`isletter(A)`	True in the locations of `A` that are letters of the alphabet.
`islogical(A)`	True if `A` is a logical array.
`ismember(A, B)`	True in locations where elements of `A` are also in `B`.
`isnan(A)`	True in the locations of `A` that are `NaN`s (see Table 2.10 for a definition of `NaN`).
`isnumeric(A)`	True if `A` is a numeric array.
`isprime(A)`	True in locations of `A` that are prime numbers.
`isreal(A)`	True if the elements of `A` have no imaginary parts.
`isspace(A)`	True at locations where the elements of `A` are whitespace characters.
`issparse(A)`	True if `A` is a sparse matrix.
`isstruct(S)`	True if `S` is a structure.

TABLE 2.9 Some functions that return a logical 1 or a logical 0 depending on whether the value or condition in their arguments are `true` or `false`. See online help for a complete list.

Function	Value Returned
`ans`	Most recent answer (variable). If no output variable is assigned to an expression, MATLAB automatically stores the result in `ans`.
`eps`	Floating-point relative accuracy. This is the distance between 1.0 and the next largest number representable using double-precision floating point.
`i` (or `j`)	Imaginary unit, as in `1 + 2i`.
`NaN` or `nan`	Stands for Not-a-Number (e.g., `0/0`).
`pi`	3.14159265358979
`realmax`	The largest floating-point number that your computer can represent.
`realmin`	The smallest floating-point number that your computer can represent.
`computer`	Your computer type.
`version`	MATLAB version string.

TABLE 2.10
Some important variables and constants.

Introduction to M-Function Programming(cont.)

- Flow control
  - If,for,while,break,continue,switch,return,try...catch
  - ...

Statement	Description
`if`	`if`, together with `else` and `elseif`, executes a group of statements based on a specified logical condition.
`for`	Executes a group of statements a fixed (specified) number of times.
`while`	Executes a group of statements an indefinite number of times, based on a specified logical condition.
`break`	Terminates execution of a `for` or `while` loop.
`continue`	Passes control to the next iteration of a `for` or `while` loop, skipping any remaining statements in the body of the loop.
`switch`	`switch`, together with `case` and `otherwise`, executes different groups of statements, depending on a specified value or string.
`return`	Causes execution to return to the invoking function.
`try...catch`	Changes flow control if an error is detected during execution.

TABLE 2.11 Flow control statements.

Code Optimization

- MATLAB is a programming language specifically designed for array operations. Taking advantage of this fact whenever possible can result in significant increases in computational speed.
  - vectorizing loops
  - preallocating arrays

Vectorizing loops

- Vectorizing simply means converting for and while loops to equivalent vector or matrix operations
- As will become evident shortly, vectorization can result not only in significant gains in computational speed, but it also helps improve code readability
- For example $f(x) = Asin(x/2pi)$
  - for x= 1:M

    f(x) =A* sin((x-1)/(2*pi));

    end
  - more efficient code by as follows:

    x=0:M-1;

    f= A*sin(x/(2*pi));

Vectorizing loops(cont.)

- Similarly, using the meshgrid function,we can optimize the calculation of $f(x,y) = Asin(ux + vy)$
  - [C,R]=meshgrid(c,r)
  - This function transforms the domain specified by row vectors c and r into arrays C and R that can be used for the evaluation of functions of two variables and 3-D surface plots (note that columns are listed first in both the input and output of meshgrid).
  - An Example

```
c=[0,1] r=[0 1 2]
>>[C R]=meshgrid(c,r)
C =
     0  1
     0  1
     0  1
R =
     0  0
     1  1
     2  2
```

Vectorizing loops(cont.)

- Useful functions

```
h= R.^2+C.^2
  h =
       0   1
       1   2
       4   5
```

Preallocating arrays

- When working with numeric or logical arrays,preallocation simply consists of creating arrays of 0s with the proper dimension.
- For example, if we are working with two images, f and g, of size 1024 $\times$ 1024 pixels, preallocation consists of the statements

  f=zeros(1024);g=zeros(1024);
- Preallocation helps reduce memory fragmentation when working with large arrays. Memory can become fragmented due to dynamic memory allocation and deallocation. The net result is that there may be sufficient physical memory available during computation, but not enough contiguous memory to hold a large variable. Preallocation helps prevent this by allowing MATLAB to reserve sufficient memory for large data constructs at the beginning of a computation.

Interactive I/O

- disp(argument)

  If argument is an array, disp displays its contents. If argument is a text string, disp displays the characters in the string.

  For example,

```
A=[1 2; 3 4];
disp(A);
sc= 'Digital image processing' ;
disp(sc);
```

Interactive I/O(cont.)

- t=input(‘message’)

  t=input(‘message’ , ’ s’)

  For example,

  ```
  t=input('Enter your data:','s')
  class(t)
  size(t)
  n=str2num(t)
  size(n)
  class(n)
  ```

Interactive I/O(cont.)

- >>t='12.6, x2y, z';
  >>[a,b,c]=strread(t,'%f%q%q','delimiter',',')
  a =
  12.6000
  b =
  'x2y'
  c =
  'z'

Interactive I/O(cont.)

- strcmp(s1,s2)

  returns 1 if strings S1 and S2 are the same and 0 otherwise.

  For example,
  strcmp(b,'x2y')
  strcmp(b,'z')

Cell arrays and structure

- Cell arrays

  a cell array is a multidimensional array whose elements are copies of other arrays.

  For example, c={'gauss'; [1 0;0 1],3}

- Structures

  Structures are similar to cell arrays, in the sense that they allow grouping of a collection of dissimilar data into a single variable.

  For example,

  ```
  S.char_string='gucas';
  S.matrix=[1 0;0 1];
  S.scalar=3;
  ```