Enabling Technologies for Sports (5XSF0)
Module 1

Introduction: Digital images in Matlab

Sveta Zinger

(s.zinger@tue.nl)

MATLAB

* MATrix LABoratory
  - High-performance language for technical computing
  - Array – basic data element, does not require dimensioning
  - Toolboxes – application-specific solutions
  - Image Processing Toolbox – for solving digital image processing problems

Matlab desktop

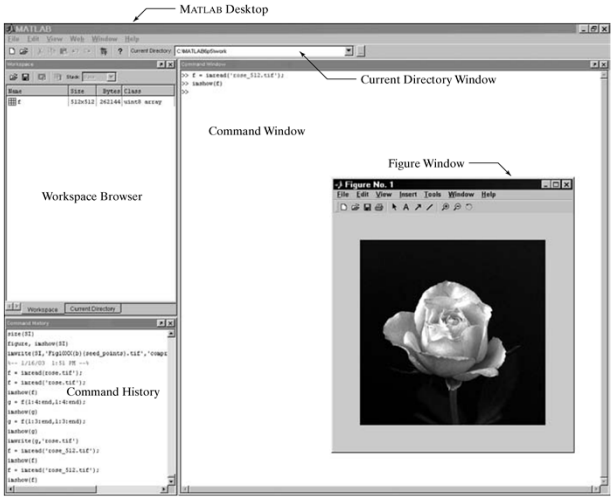


FIGURE 1.1 The MATLAB desktop and its principal components.

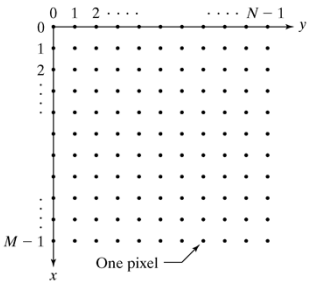
Getting help – (1)

* Help on Matlab
  - Click on the question mark symbol (?) on the desktop toolbar
  - Type `helpbrowser` at the prompt in the command window
* Help on an M-function
  - Types of help information
    - H1 help – a one-line help
    - Help text block – detailed description

Getting help – (2)

* Getting help from command window
  - `help <function_name>`
  - `lookfor <keyword>`
    - displays all H1 lines that contain the keyword
  - `lookfor <keyword> -all`
    - displays H1 lines of all functions that contain the keyword in either the H1 lines or the text block
  - `type <function_name>`
    - displays the text block of the help for the function as well as its code

Digital image representation – (1)



a b

FIGURE 2.1 Coordinate conventions used (a) in many image processing books, and (b) in the Image Processing Toolbox.

Sampling – digitizing the coordinate values
Quantization – digitizing the amplitude (intensity) values

Digital image representation – (2)

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

Example: `f(7,3)` is the element in the 7th row and 3rd column of the matrix (image) `f`.

Reading and displaying images

* **`imread('filename')`**
  - Example: `f=imread('chestxray.jpg');`
  - **`size(f), whos f`** display size and additional information respectively
* **`imshow(f)`**
  - `figure` – creates a new figure window
  - `pixval (impixelinfo)` – shows cursor position and the corresponding intensity

Writing images – (1)

* **`imwrite(f,'filename')`**
  - Option for a JPEG image –
    `imwrite(f,'filename.jpg','quality',q)`
  - Obtaining image file details – `imfinfo filename`
    example: `>> imfinfo chestxray.tif`

```
ans =
        Filename: 'chestxray.tif'
        FileModDate: '18-Mar-2009 14:55:03'
        FileSize: 297030
        Format: 'tif'
        ...
```

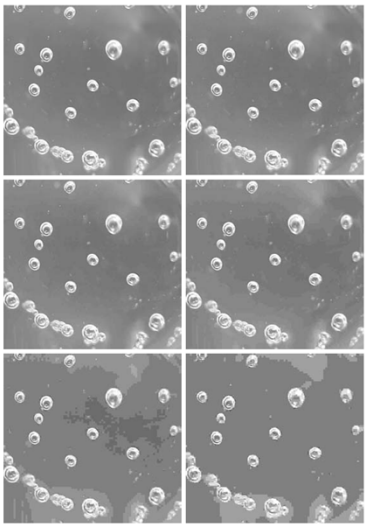

Writing images – (2)

some of the image formats supported by **`imread`** and **`imwrite`**

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`.

Writing images: example



a b
c d
e f

FIGURE 2.4
(a) Original image. (b) through (f) Results of using `jpg` quality values `q = 50, 25, 15, 5,` and 0, respectively. False contouring begins to be barely noticeable for `q = 15` [image (d)] but is quite visible for `q = 5` and `q = 0`.

Array indexing

```
>> v=[1 3 5 7 9]
v =
     1   3   5   7

>> v(3)
ans =
     5

>> v(2:4)
ans =
     3   5   7
```

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

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

>> x=linspace(1,10,4)
x =
     1   4   7   10
```

Matrix indexing

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

>> A(2,3)
ans =
     6
```

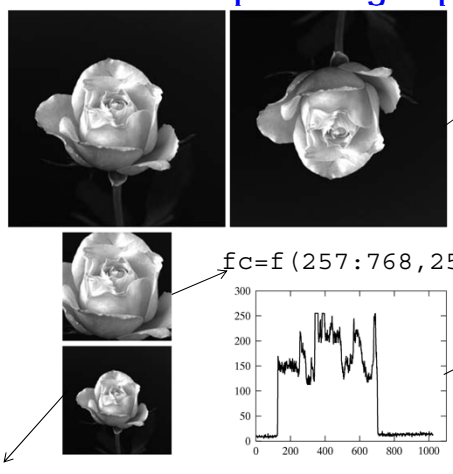
```
>> column3=A(:,3)
column3 =
     3
     6
     9

>> row1=A(1,:)
row1 =
     1   2   3

>> A(2:3,1:2)
ans =
     4   5
     7   8
```


Matrix indexing

Example: image operations



a b
c
d e

FIGURE 2.6 Results obtained using array indexing. (a) Original image. (b) Image flipped vertically. (c) Cropped image. (d) Subsampled image. (e) A horizontal scan line through the middle of the image in (a).

Operators – (1)

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.

Operators – (2)

TABLE 2.6 Relational operators.

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

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.

Flow control

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.

References

- Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, "Digital Image Processing Using Matlab", Pearson Education, 2004
  - Chapter 1
  - Chapter 2