Image processing in Scilab/Matlab

Programming requirements

- Assignments, the midterm test and the final exam all include Matlab/Scilab programming questions.
- The questions will ask you to achieve a certain image processing by writing a program/function.
- If no additional instruction is given, you are NOT allowed to use image processing functions in the image processing toolboxes (e.g. SIVP and IPCV) apart from some certain ones (imshow, imread, imwrite).
- Practice programming after class **as much as possible!**

Matlab

- Numerical computing environment.
- Based on C++ but much simpler syntax.
- Comprehensive toolboxes for specialized areas (SIVP toolbox).
- Allows matrix manipulations, plotting functions and data, implementation of algorithms, creating of user interfaces and interfacing with programs writing in other languages.
- High efficiency for matrix processing and computing.
- Good graphic presentation and data visualization.

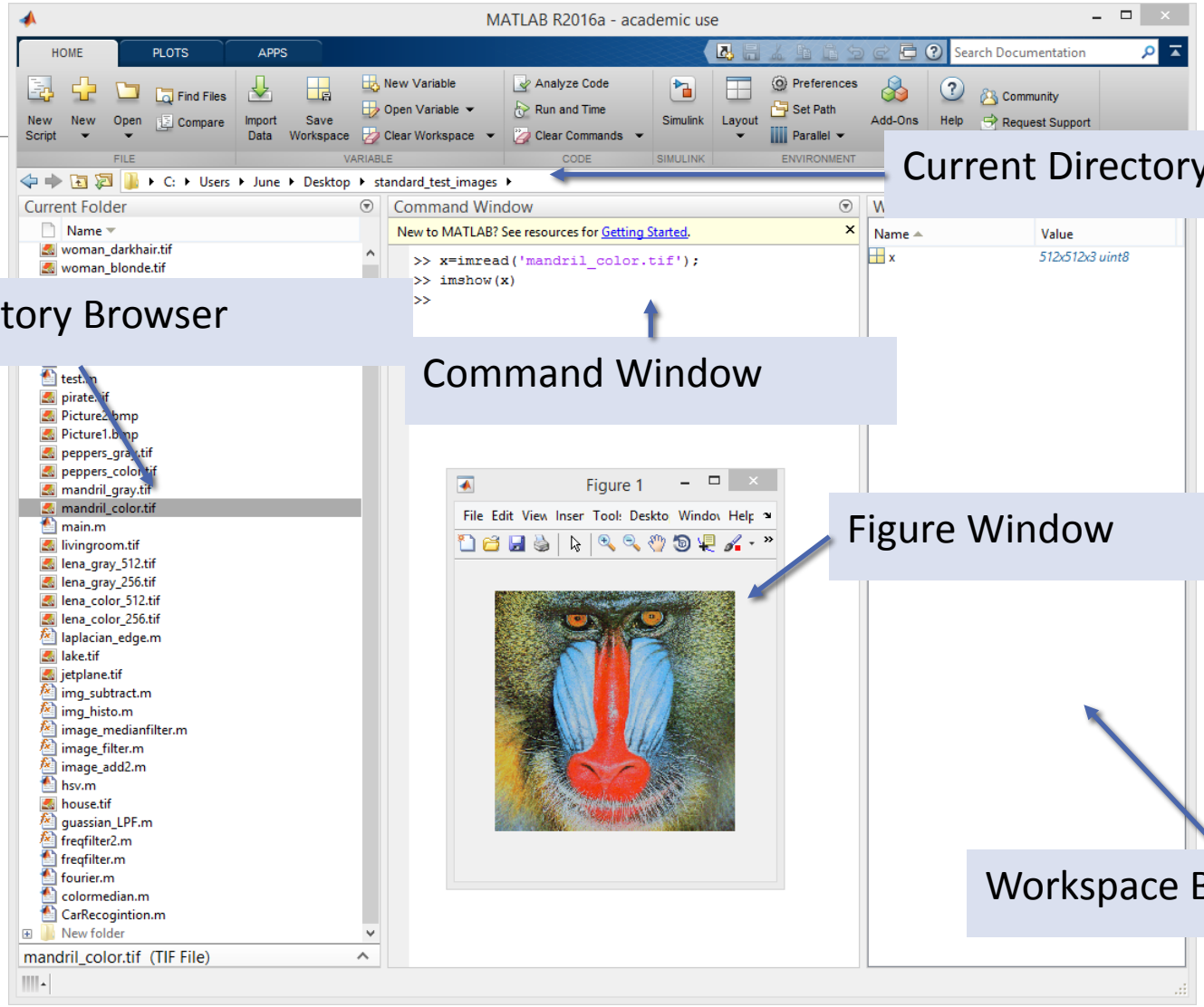
Scilab

- Scilab is an open-source numerical computational package and high level numerically oriented programming language.
- Its syntax is similar to Matlab, including a source code translator for assisting the conversion of code from Matlab to Scilab.
- Scilab has Image Processing and Computer Vision Toolbox (IPCV)
- Scilab is available in the classroom.
- Download Scilab at http://www.scilab.org/

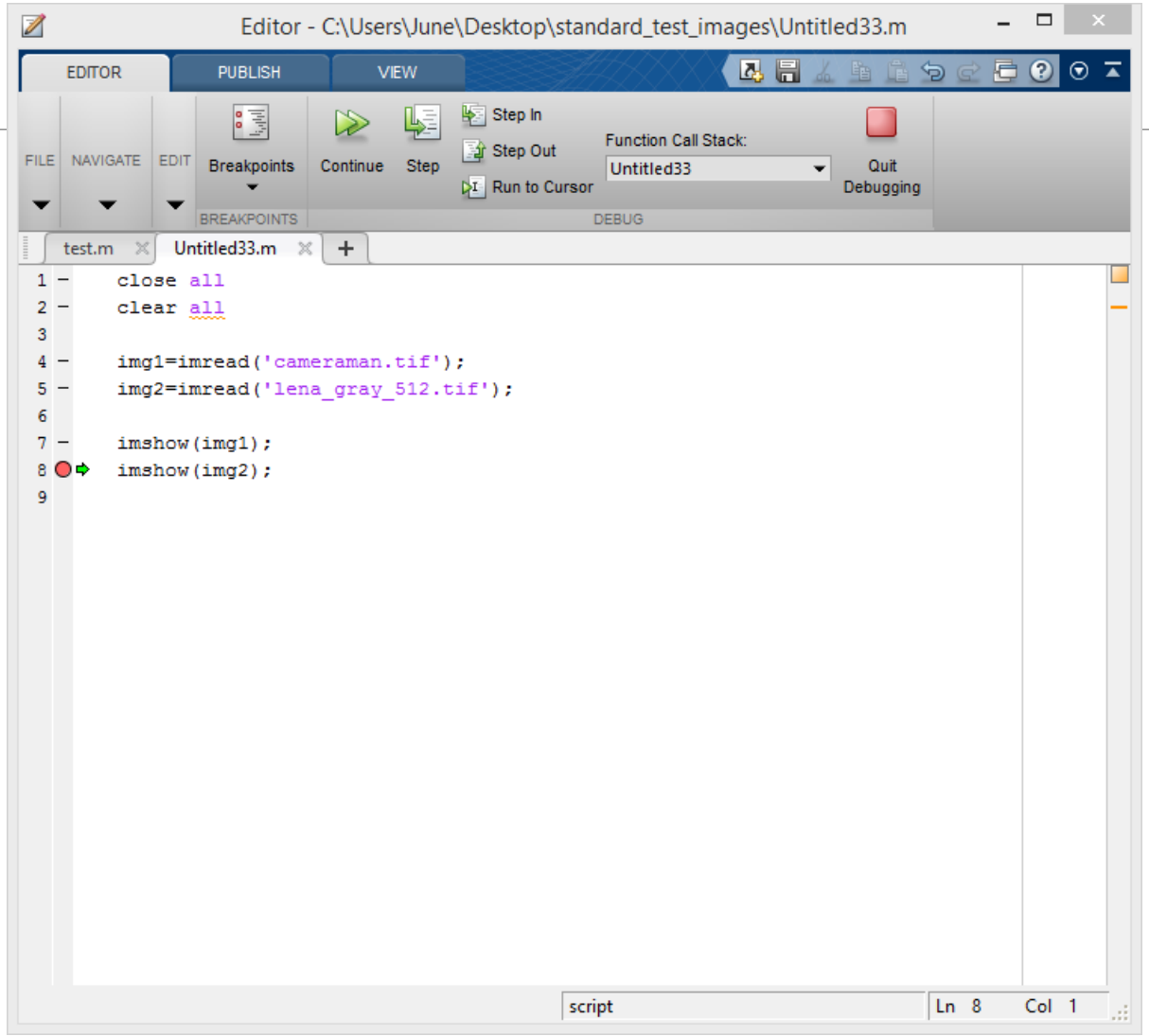
Toolboxes available at https://atoms.scilab.org/

IPCV Toolboxes at http://atoms.scilab.org/toolboxes/IPCV/4.1.2

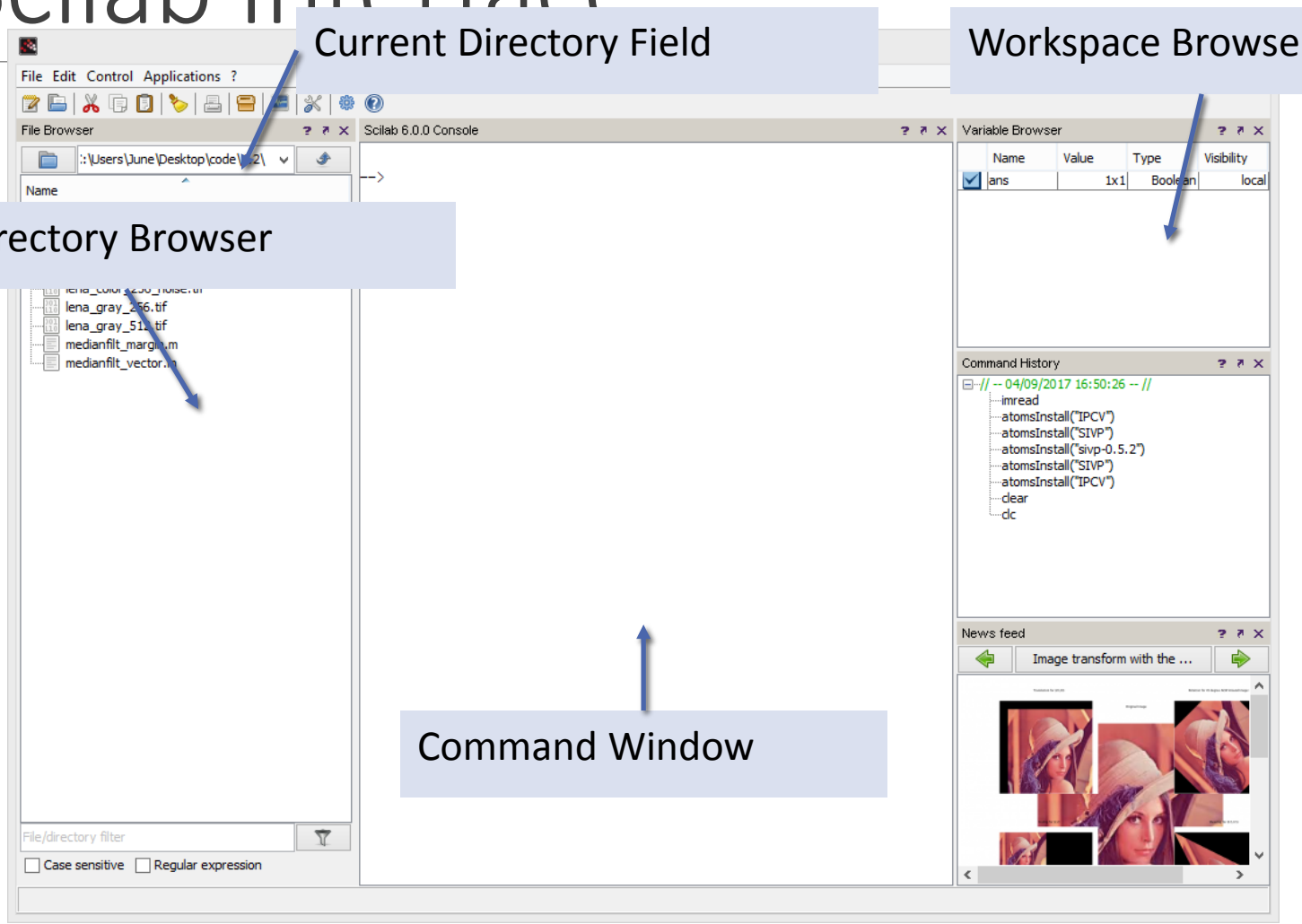
Matlab Interface



Matlab editor



Scilab Interface



Scilab Toolbox Installation

Install toolboxes via ATOMS (AuTomatic mOdules Management for Scilab)

- ATOMS GUI- under Application menu.

or

- ATOMS function `atomsInstall("name_of_the_module")` For example: `-->atomsInstall("IPCV");`

Once a toolbox is installed, the module will be loaded automatically on the next start-up.

More information at: https://wiki.scilab.org/Modules

Installation using a Downloaded Toolbox

```
--> atomsInstall("IPCV");
Scanning repository http://atoms.scilab.org/6.1 ... Done

at line   237 of function atomsDownload ( E:\scilab-6.1.0\modules\atoms\macros\atoms_internals\atomsDownload.sci line 252 )
at line   314 of function atomsInstall  ( E:\scilab-6.1.0\modules\atoms\macros\atomsInstall.sci line 330 )

getmd5: The file E:\scilab-6.1.0\contrib\IPCV\IPCV-4.1.2-win64-61-bin.zip does not exist.

-->
```

Step2:

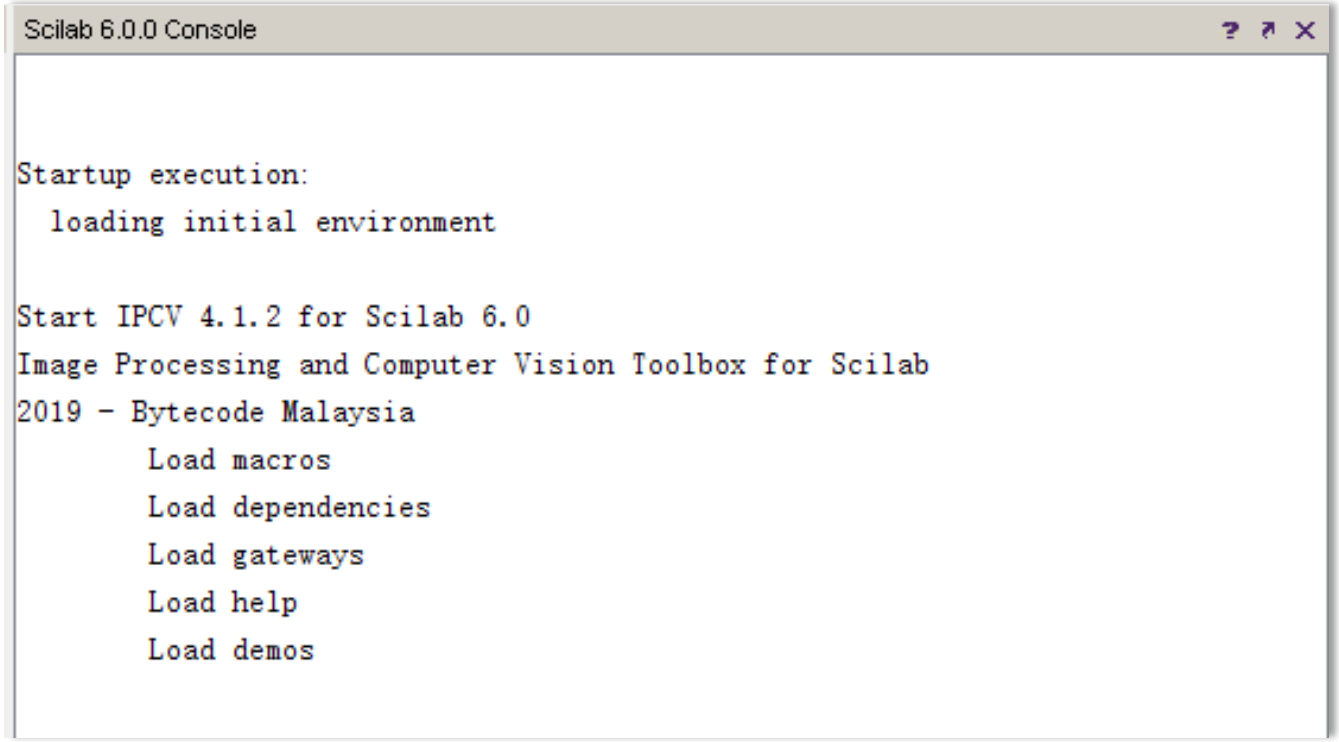
Put IPCV-4.1.2-win64-61-bin.zip in the folder above

Step3:

Re-run `atomsInstall("IPCV");`

Scilab Toolbox Installation

Once a toolbox is installed, the module will be loaded automatically on the next start of Scilab.



Scilab Help

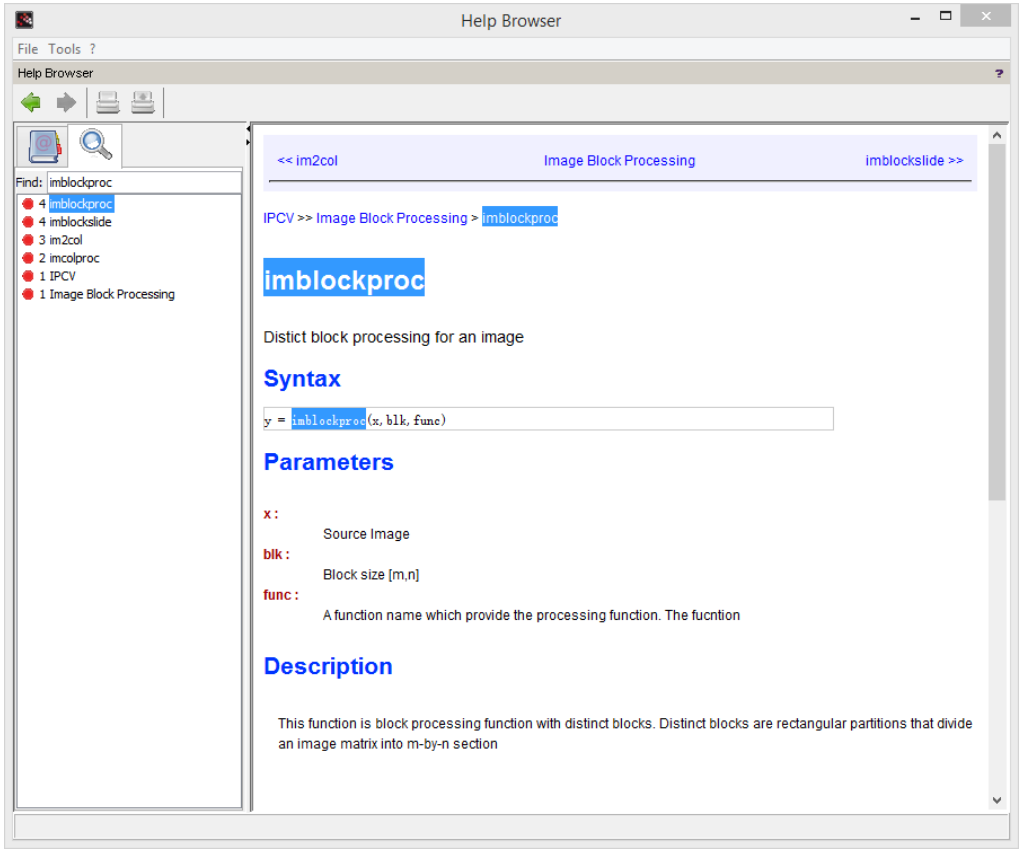

Help Browser
File Tools ?
Help Browser
Find: imblockproc
4 imblockproc
4 imblockslide
3 im2col
2 imcolproc
1 IPCV
1 Image Block Processing
<< im2col
Image Block Processing
imblockslide >>
IPCV >> Image Block Processing > imblockproc
imblockproc
Distict block processing for an image
Syntax
y = imblockproc(x, blk, func)
Parameters
x :
Source Image
blk :
Block size [m,n]
func :
A function name which provide the processing function. The fucntion
Description
This function is block processing function with distinct blocks. Distinct blocks are rectangular partitions that divide an image matrix into m-by-n section

Image in Matlab/Scilab

- Image is regarded as a matrix defined by:
  - The number of rows.
  - The number of columns.
  - The data type
- Grayscale image
  - 2D matrix
  - Data type: uint8 or uint16
  - Value range: [0,255] or [0 65535]
- Binary image-a logical array of 0s and 1s.
  - 2D matrix
  - Data type : logical
  - Value range: 0 or 1
- Colour image
  - 3D matrix
  - Data type: uint8 or uint16
  - Value range: [0,255] or [0 65535]

Image O/I and display

imread

- `Im=imread('filename.tif/bmp/jpg/tiff/gif');`

imwrite

- `imwrite(Im, 'filename.jpg','quality',q)`
- `q` is an integer between 0-100.

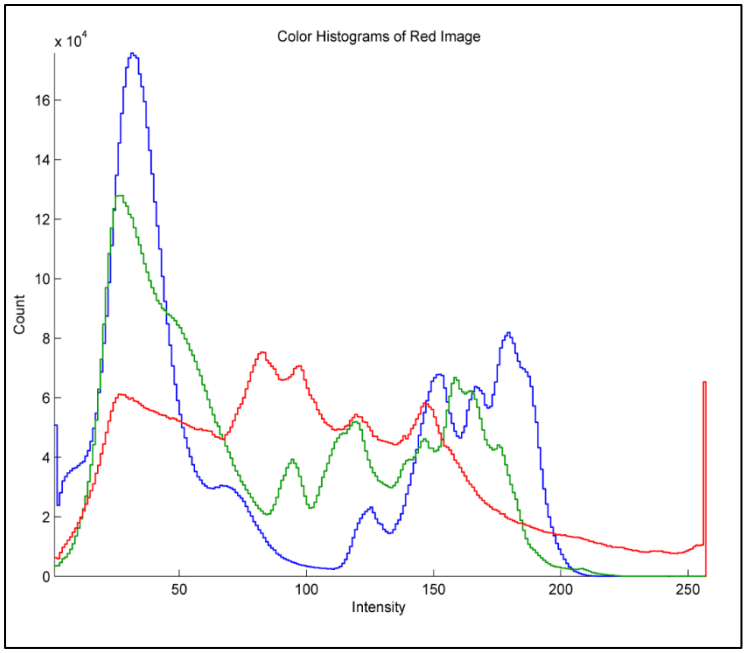
imshow

- `imshow(Im);`
- `figure, imshow(Im);`
- `subplot(2,3,1); imshow(Im); title('after masking');`

Plot curves

plot

- `plot(0:255, hr, 'r', 0:255, hg, 'g' 0:255, hb, 'b');`



https://www.mathworks.com/help/matlab/ref/plot.html?s_tid=gn_loc_drop

Plot curves

plot

Table 2.3: Attributes for `plot`

Symbol	Color	Symbol	Line Style	Symbol	Marker
k	Black	–	Solid	+	Plus sign
r	Red	−−	Dashed	o	Circle
b	Blue	:	Dotted	*	Asterisk
g	Green	−.	Dash-dot	.	Point
c	Cyan	none	No line	×	Cross
m	Magenta			*s*	Square
y	Yellow			*d*	Diamond

https://www.mathworks.com/help/matlab/ref/plot.html?s_tid=gn_loc_drop

Matrix generating functions

`zeros(M,N)`

generates an MxN matrix of 0s of class double

`ones(M,N)`

generates an MxN matrix of 1s of class double

`true(M,N)`

generates an MxN matrix of 1s of class logical

`false(M,N)`

generates an MxN matrix of 0s of class logical

`rand(M,N)`

generates an MxN matrix whose entries are uniformly distributed random numbers in the interval [0,1]

Vectorization

- Matlab/Scilab matrices are stored in an internal data structure which can be managed at the interpreter level.
- Most basic arithmetic operations are performed by a compiled, optimized, source code.
- Most matrix algorithms do not require to use loops. a Matlab/Scilab script which performs the same operations with loops is typically from 10 to 100 times slower.
- For higher efficiency, matrix algorithm should be used as often as possible.

Arithmetic operation functions

Oper.	Name	Comments
+	Array and matrix Addition	a +b, A + B, or a + A
-	Array and matrix subtract	a –b, A – B, or a – A
.*	Array multiplication	C=A.*B; C(I,J)=A(I,J)*B(I,J)
*	Matrix multiplication	A*B, std. matrix multiplicat.
./	Array right division	C=A./B; C(I,J)=A(I,J)/B(I,J)
.\	Array left division	C=A.\B; C(I,J)=B(I,j)/A(I,J)
/	Matrix right division	A/B; A*inv(B)
\	Matrix left division	A\B; inv(A)*B
.^	Array power	C(I,J)=A(I,J)^B(I,J)
^	Matrix power	
.’	Vector & matrix transpose	A.’
‘	Vector & matrix complex Conjugate transpose	A’
+	Unary plus	+A is same as 0+A.
-	Unary minus	-A is same as 0-A or -1*A

- Here A and B are matrices or arrays and a and b are scalars. All operands can be real or complex.

.* vs *

.* –dot product

- .* is element-wise multiplication.
- $C(i,j) = A(i,j) \times B(i,j)$
- A and B should have the same size. (A:M×N, then B: M×N)

* –is matrix multiplication.

$$C(i,j) = \sum_{k=1}^{N} A(i,k) \times B(k,j)$$

- the column number of A should be the same as the row number of B, if A*B. (A:M×N, B:N×K and C:M×K)

When one of matrices is a scalar, * and .* give the same result.

Matrix with ‘:’

Syntax	Access
`A`	the whole matrix
`A(:,:)`	the whole matrix
`A(i:j,k)`	the elements at rows from i to j, at column k
`A(i,j:k)`	the elements at row i, at columns from j to k
`A(i,:)`	the row i of matrix
`A(:,j)`	the column j of matrix
`A(end,end)`	the elements at the last row, at last column. * Matlab uses ‘`end`’ and Scilab uses ‘`$`’.

Flow control operator: If...else/elseif

The if statement allows to perform a statement if a condition is satisfied.

```
i = 5;
if i == 1
   disp (" Hello !");
elseif  i == 2
   disp (" Goodbye !");
elseif  i == 3
   disp (" Tchao !");
else
   disp ("Au Revoir !");
end
```

Flow control operator: for

The for statement allows to perform loops, i.e. allows to perform a given action several times.

```
for i = 1:2:6
   disp (i);
end
```

Flow control operator: while

The while statement allows to perform a loop while a boolean expression is true.

```
s = 0;
i = 1;
while i <= 10
   s = s + i;
   i = i + 1;
end
```

Defining a function

Matlab uses '`function`' and '`end`' to define a new function. (Scilab uses '`function`' and '`endfunction`'.)

```
function y=myfunction(x)

y=2*x;

end
```

the header of the function

the body of the function

The script which stores the function should share the name with the function.

If several functions are stored in a script, only the one with the same name as the script is public. The rest are private.