Practicum Computational Vision

Practicum 1: Image processing toolbox in Matlab (Part I)

In the first session, we are going to make an introduction to Matlab and the image processing toolbox in MATLAB. This session will not generate any deliverable.

1 Getting Started with MATLAB

To carry out the practicum, first read the introductory Matlab tutorial:

http://www.mathworks.es/help/pdf_doc/matlab/getstart.pdf

1.1 Product Overview:

MATLAB is a high level language that allows to develop algorithms, visualize and analyze data and perform numerical calculations. Its development environment allows easy access to the variables and display of the results.

** See : 1 Quick Start -> Product Description*

1.2 MATLAB Environment

When you start MATLAB, you will see that the application displays several windows. "Command Window" is a command line where you can enter commands. In Workspace all variables created with the command line or by scripts / MATLAB functions are stored.

* Follow the steps in Quick Start -> Desktop Basics

1.3 Functions Lookfor and Help

*See : 1 Quick Start -> Help and Documentation

When information about the functions are necessary in MATLAB, the help function is very useful because it provides information on the function and its parameters. Run on the command line "help max" as an example.

When we have in mind an action but do not know exactly the name of the function in MATLAB, we use the lookfor command. Run " lookfor mean" as an example.

2 Introduction to MATLAB

2.1 Creating vectors and matrices

2.1.1 Create a row vector of 10 items with random values between 0 and 1 and perform the following operations on the vector created:

a) Access to the third position vector.
b) Get the vector formed by the first 5 positions of the original vector.
c) Get the vector formed by the odd positions of the original vector.
d) Sum all elements of the vector.

* See: 1 Quick Start -> Matrices and Arrays
* See: 1 Quick Start -> Array Indexing
* See: 1 Quick Start -> Calling Functions

2.1.2 Create two arrays 'a' and 'b', the first of 2 rows and 3 columns (2x3): [1 2 3; 4 5 6] and the second one of dimension (3x2): [2 4; 5 7; 9 3], and then perform the following operations.
a) Multiply both matrices.
b) Convert the array 'a' into a new array a2 of dimension 3x2 using the function Reshape (use the help to use the function RESHAPE correctly).
c) Add the matrices 'a2' and 'b' and store the result in the variable 'c'.
d) Multiply, element by element, the matrices 'a2' and 'b'.

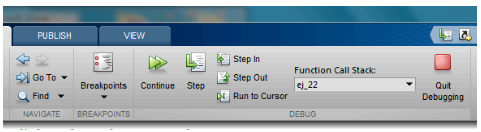
2.2 Scripts and Functions

*See: 1 Quick Start -> Scripts and Programming
*See: 1 Quick Start -> Programming and Functions

a) Create a function in the file divide2.m to divide by two the input value, provided it is greater than 5. Otherwise, keep the input value of the function.
b) Repeat the exercise, using as input argument a row vector instead of a single value. Use a loop to iterate through the vector. Call the function "divideVector2()".
c) MATLAB allows operating on vectors and matrices avoiding the use of loops. Optimize the above algorithm eliminating the loop. Call the function "divideVector2Opt()".
Note: One of the very important features in Matlab is the possibility to work with the whole matrix and vector structures instead of specifying the operations element by element. In http://www.mathworks.es/es/help/matlab/matlab_prog/find-array-elements-that-meet-a-condition.html you can find some more information how to program avoiding loops. As long as possible, always writing the programs without "for" will make the execution much faster!

d) Create a script, where the values 2 and 7 are passed consequently to the function created in a) and the vector x = [4 22 3.5 5.5 7] as an input argument to the functions created in b) and c). Save variables in different results.

2.3 Debug

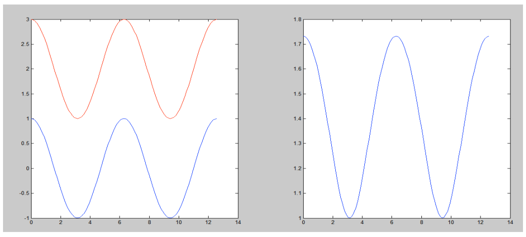


MATLAB includes methods to execute the code line by line and see every step of the value of existing variables.
Open the script divideVector2Opt.m and place a breakpoint on the first line of the code. Then run the code step by step and see how the execution stops at the breakpoint created. Run "continue" to move to the end of the code or to the next breakpoint.

2.4 Data Visualization

* See: 1 Quick Start -> Workspace Variables
* See: Quick Start 1 -> 2-D and 3-D Plots

a) Create the function y = <u>miCos</u> (x). Tip: Create a vector of 100 positions with values between 0 and 4 * pi to save the variable 'x' using the linspace function.
b) Use the MATLAB Workspace to see the value of the elements of the vectors created. What value does appear at position 31 of the vector 'y'?
c) Create a graph (use: plot function) to display the cosine function created above.
d) Apply an offset of +2 in each of the positions of y and store the result in a new variable y1.
e) Show the graphs functions 'y' and 'y1' in the same window drawing them with different colors (hint: plot, hold on, hold off,).
f) Create a new vector 'y2', where each position contains the square root of the corresponding position in 'y1'. Tip: If you do not find the square root function, use the lookfor command.
g) Show in a figure, one beside the other the graphics functions 'y' and 'y2' (hint: you can use the Matlab function subplot).



3. Introduction to imageprocessingtoolbox

The MATLAB imageprocessingtoolbox allows to open images, manipulate them, convert the space of colors, perform distinct types of filtering, etc.. In this first practice, we will work on a series of exercises to familiarize with the image management in the Matlab environment. Review:
In the Image Processing toolbox (doc images), review:

- Import, Export, and Conversion
- Display and Exploration

  In Matlab: Study Example 1 of the ImageProcessing toolbox.. How much type of images can we have in Matlab?

Additional material:
http://en.wikibooks.org/wiki/MATLAB_Programming/Image_Processing_Toolbox

1.1 Image construction

The images in MATLAB are matrices:
a) Create an image of size 256x256 of black color and only one channel, each pixel should be of type 8-bit unsigned.
b) Obtain the information about the kind of image created and its size. Help: Consult functions: class, size and display.
c) Visualize the image and save it (imwrite()) as void_img.jpg (Fig. 1 (a))
d) Create a 256x256 sized image with a white square inside, of a type one channel 8-bit unsigned, similar to the image in Figure 1 (b). Propose two ways for it without using loops.
e) Visualize the image and save it as square_img.jpg.

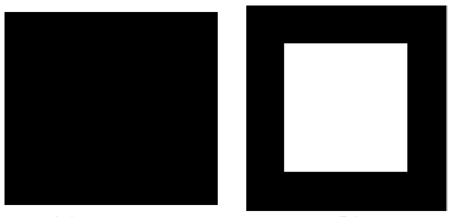

(a) (b)

Figure 1 Constructing images from matrix

1.2 Contrast image change

a) Open the image friends.jpg and display it.
b) Find the minimal and maximal gray value of its pixels.
c) Manipulate the image so that the minimum value is 0. Display it.
d) Manipulate the image in order to maximize its contrast (Figure 3 (c)) that is the minimum value to be 0 and the maximum to be 255.

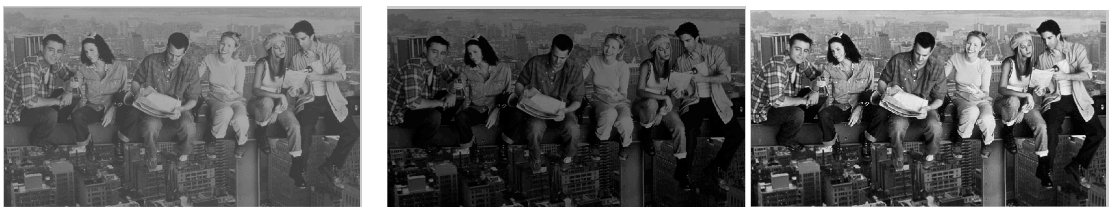

(a) (b) (c)

Figure 2 Manipulating grey level values of images