Let’s start with the Lab:
introduction to OpenCV and Matlab

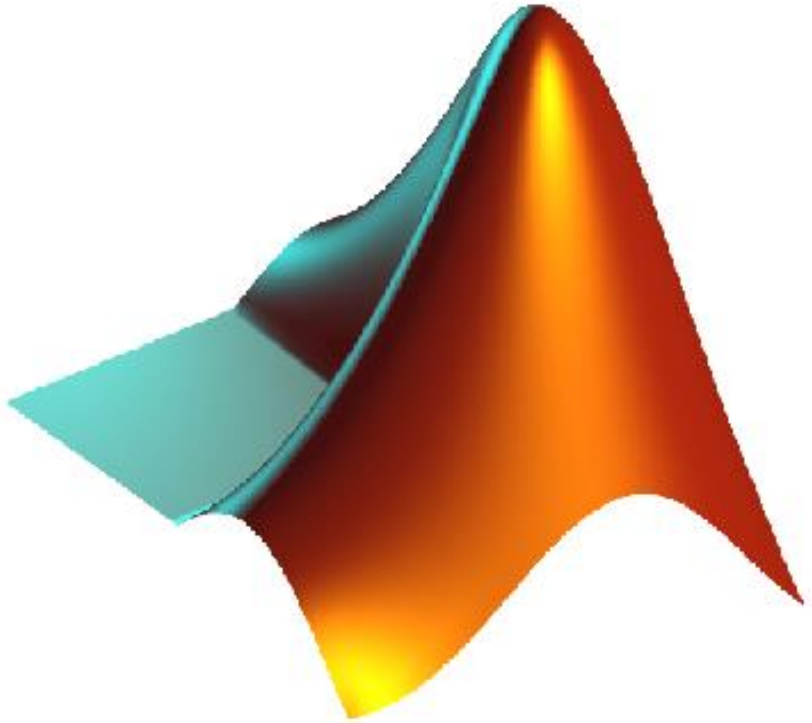
VISIONE ARTIFICIALE

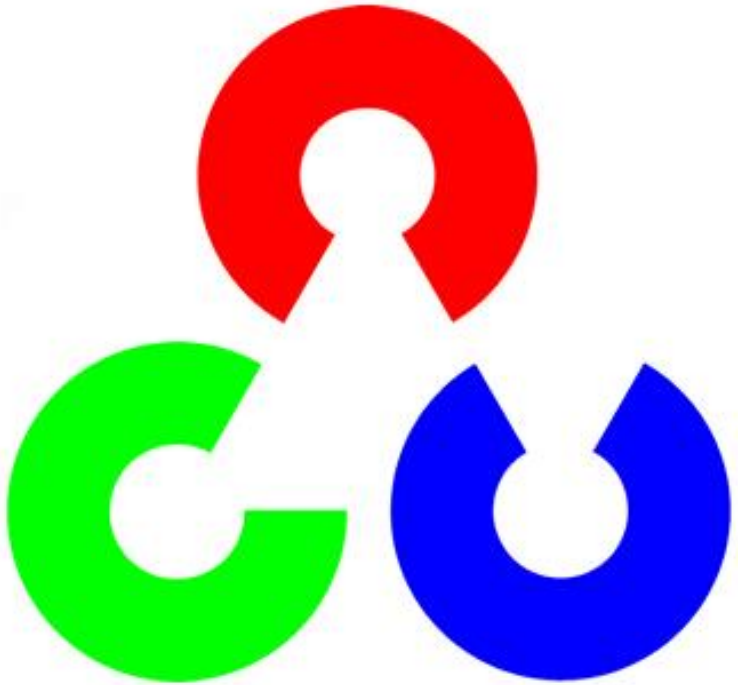
dott. Alessandro Ferrari

Social Q&A

@vs_AR
#askVisionary
www.vision-ary.net

Overview Lesson Two





What is OpenCV?

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras.

It has C++, C, Python, Java and MATLAB interfaces. It supports Windows, Linux, Android and Mac OS. A full-featured CUDA and OpenCL interfaces are being actively developed right now. OpenCV is written natively in C++.

http://itseez.com/ Is the company “behind” last versions of OpenCV.

OpenCV: how to start

To use the OpenCV library there are two options:

Installation by Using the Pre-built Libraries

Installation by Making Your Own Libraries from the Source Files

While the first one is easier to complete, it doesn't take advantage of the most advanced technologies OpenCV integrates.

OpenCV: how to build

Building the OpenCV library from scratch requires the following tools installed beforehand:

- An IDE (preferably) or just a CC++ compiler that will actually make the binary files. Microsoft Visual Studio works quite well, however, you can use any other IDE that has a valid CC++ compiler (Eclipse).

- CMake, which is a tool to make the project files (for your chosen IDE) from the OpenCV source files. It will also allow an easy configuration of the OpenCV build files, in order to make binary files that fits exactly to your needs (we will see how to exploit it to make OpenCV faster).

- Git to acquire the OpenCV source files (TortoiseGit). Alternatively, you can just download an archived version of the source files from Sourceforge

OpenCV: «must have» extras

- Intel © Threading Building Blocks (*TBB*) is used inside OpenCV for parallel code. Using this will make sure that OpenCV will take advantage of all the cores you have. We will use it to speed the training up.
- Intel © Integrated Performance Primitives (*IPP*) used to improve the performance of algorithm (i.e color conversion, Haar training, DFT functions, etc.). It isn't a free service, a sub-version is free and automatically downloaded when you configure the makefile with Cmake GUI.
- CUDA Toolkit will allow you to use the power lying inside your GPU. This will drastically improve performance for some algorithms (ie. the HOG descriptor). It's a work-in-progress feature, be careful.
- OpenNI Framework contains a set of open source APIs that provide support for natural interaction with devices via methods such as voice command recognition, hand gestures and body motion tracking. We do not focus on NUI, but it's a "must hand" lib if you want to challenge with natural interaction.

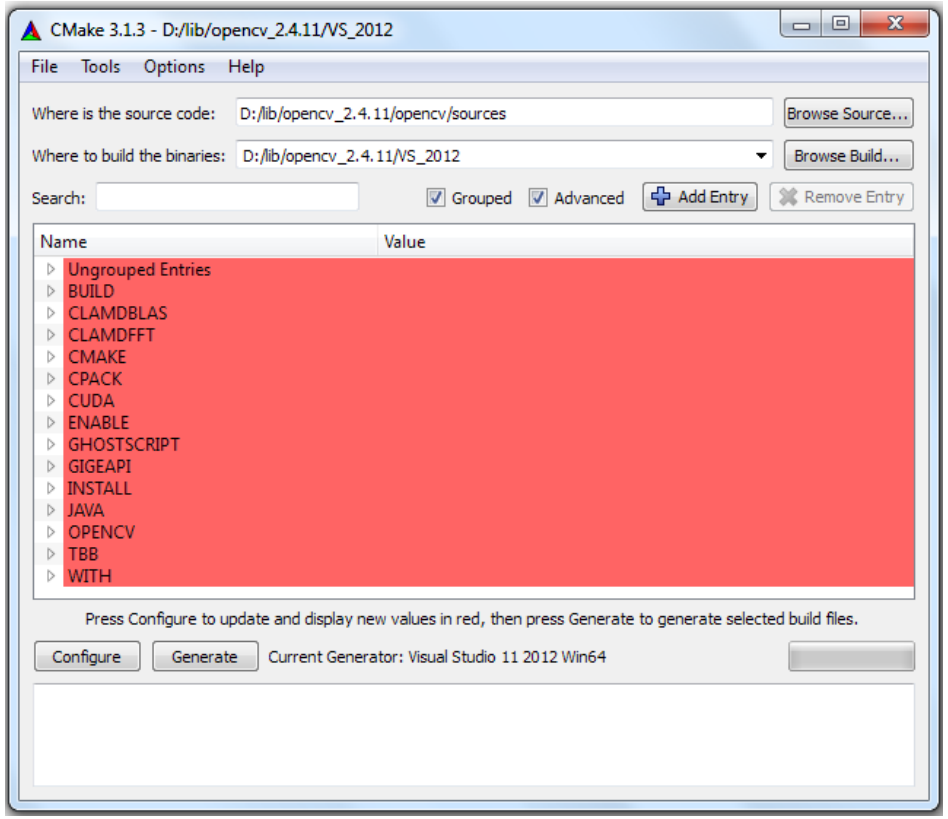
TODO: Assignment #01

GOALS:

- Having a laptop with OpenCV ready to use.

TODO list (next lab lesson):

- Browse for the OpenCV sources
- Configure the OpenCV flavour your desire
- Generate the MakeFile for the IDE you use
- Open The IDE, build OpenCV (Debug and Release)
- Let's start browsing OpenCV solution.
- Run «facedetect», «edge» and «contours».



OpenCV: «Taxonomy»

OpenCV has a wide community of programmers, researcher and supporters. Every question you have, it is (probably) an already closed topic. Before climbing the hill, let's start reading tutorials:

http://docs.opencv.org/doc/tutorials/tutorials.html

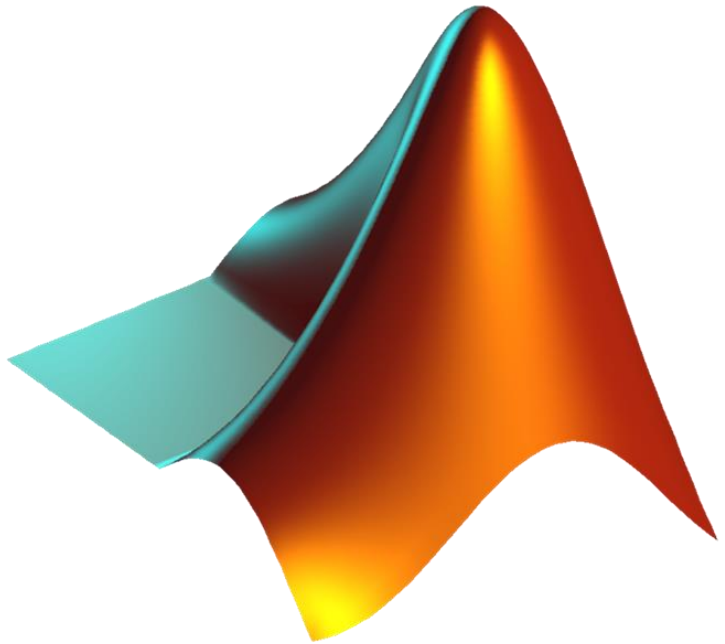
The OpenCV «engine» is designed by modules, the most relevant (for us):

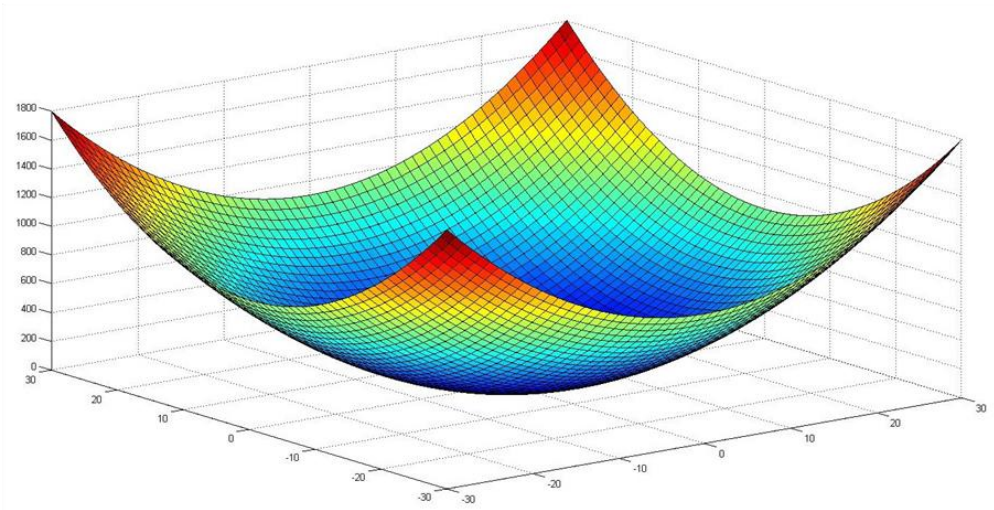
- ImageProc - image processing manipulation (linear filtering, morphology, Sobel, Canny, etc.)
- Feature2D - points detectors, descriptors and matching framework (Harris, Features, etc.)
- Objdetect - feature for detection of rigid objects (Haar, LBP, HOG and cascades).

What is Matlab?

- A high-level language and interactive environment to work across several disciplines including DSP and image processing, communications, control systems, finance, etc.
- A special purpose programming language (oriented to the numerical computation)
- An Integrated Development Environment (IDE).
- A wide set of integrated scientific libs (like Maple, FFTW, LAPACK, … and a wide set of Toolboxes)
- A powerful tool that allow to write multi-language coded applications (Matlab, Java, C, C++, etc.)
- A GUI toolkit and a GUI Design Environment (GUIDE).
- An easy «bridge» to the coolest HW (RaspberryPi and Pi2, Arduino, Kinect, etc.).

MATLAB & UNIMI





http://www.unimi.it/ateneo/80207.htm

Matlab: doing what?

What we can do with Matlab:

- Prototype Algorithms | Systems | Applications
- We can write quickly our prototypes using an high-level scripting language
- We can easily share our research tool and knowledge writing self-explaining code

BUT we CANNOT obtain an optimized real-time applications like ones written in C or C++

Matlab: approaching the philosophy

Matlab allows to write various types of libs for our applications to modularize the code:

- Toolboxes: packages of functions and classes organized per topic
- Java packages: Java is completely integrated in the matlab language
- DLL, COM and OCX modules can be easily integrated in our code
- The MEX interface allow to write C and C++ code that interacts with Matlab (efficiency / code obfuscation)

Matlab: into the language

The official tutorial to start with:

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

- It works interactively using the command prompt (inspecting data using directly from the interface).
- Gives syntax and a set of data structures and tools to manage multidimensional vectors (like matrices, etc.) and multiclass data containers (called cells).
- Generating 2D and 3D graphs of the data (with a wide range of graph types and visualization options).
- Using GUI tools given from Matlab and from the Toolboxes (great for image processing)
- Allow to manage indexes in a sophisticated manner.

Matlab: first example

Let’s write some code:

```
%To create two-dimensional line plots, use the plot function. For example, plot the value of the sine function from 0 to 2*pi:
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y);

%You can label the axes and add a title:
xlabel('x'); ylabel('sin(x)');  title('Plot of the Sine Function');

%To add plots to an existing figure, use hold
hold on y2 = cos(x); plot(x,y2,':') legend('sin','cos')
```

Matlab: into the numbers

Data types are defined from a set of standard classes. RTTI (run time type identification) it’s a mechanism that exposes information about an object's data type at runtime:

```
>> class(1)
ans = double
```

```
>> class(int32(1))
ans = int32
```

By default numbers are in double precision. Other formats are provided: uint8, uint16, uint32, uint64, int8, int16, int32, int64, single, double, etc.

Matlab: Vectors and Matrices

Vectors, matrices and multidimensional vectors can be managed easily using:

- Array constructors like zeros, ones, rand, etc.
- Structured indexes (but simple to use) that allow indexing and slicing
- Native algebraic operations on vectors and matrices (matrix multiplication, system resolution, algebraic operators (+ - * /) both for scalars and element by element operations, etc.)
- Manipulation functions for squeeze, dimensions removal, dimension composition, etc.

There exists different tools for the different types of arrays (numerical, logical, images)

Speed Matlab up

- Writing good sources allows you to make your application faster and (sometimes) near to the real-time purposes. In many cases writing good code is not enough due to the large CPU-load many algorithms requires. In this cases «parallel» programming could be necessary.

- Parallel Computing Toolbox lets you solve computationally and data-intensive problems using multicore processors, GPUs, and computer clusters. High-level constructs—parallel for-loops, special arrays, and parallelized numerical algorithms—let you parallelize MATLAB applications without CUDA.

- The toolbox lets you use the full processing power of multicore desktops by executing applications on workers (MATLAB computational engines) that run locally. Watch out the RAM!

```
%Starting parallel pool (parpool)
parpool

%Parallel pool using the 'local' profile is shutting down.
delete(gcp);
```

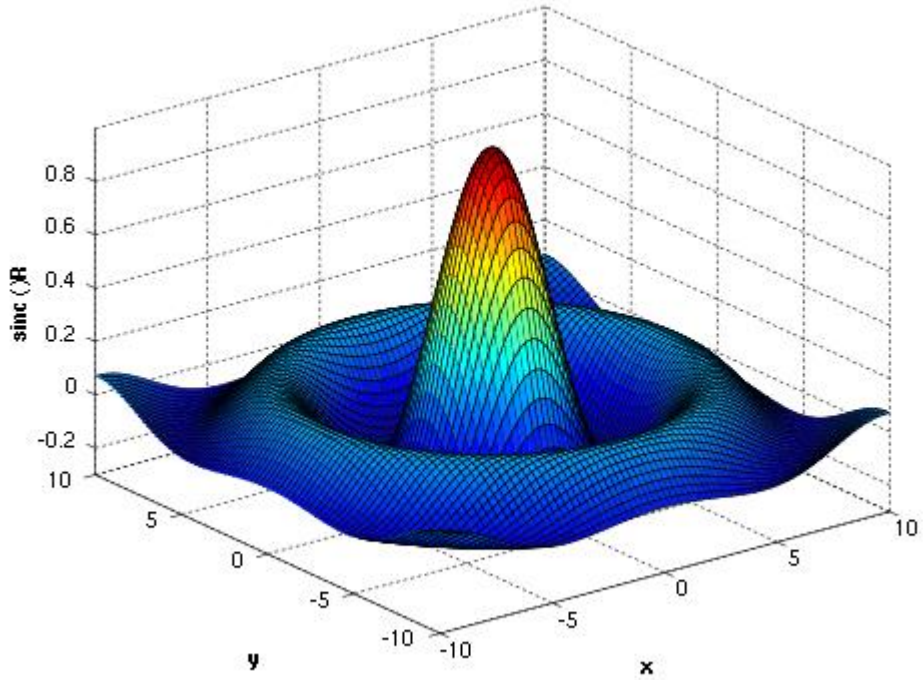
TODO: Assignment #02

GOALS:

- Having a laptop with Matlab ready to use.
- Do again the exercises we are going to see.

TODO list (next lab lesson):

- Download Matlab from academic source
- Install Matlab, CV toolboxes.
- Run exercises from the second part of the lesson.



Social Q&A

@vs_AR
#askVisionary
www.vision-ary.net