OpenCV
A short introduction to OpenCV libraries

Marco Moltisanti

Image Processing Lab
Dipartimento di Matematica e Informatica
Università degli Studi di Catania

moltisanti@dmi.unict.it

March 13, 2013
OpenCV (Open Source Computer Vision) is a library of programming functions containing all the standard algorithms for Computer Vision;
OpenCV (Open Source Computer Vision) is a library of programming functions containing all the standard algorithms for Computer Vision;
Currently, functions are implemented in C/C++, Python and Java;
What?

- OpenCV (**Open Source Computer Vision**) is a library of programming functions containing all the standard algorithms for Computer Vision;
- Currently, functions are implemented in C/C++, Python and Java;
- OpenCV libraries can run on Windows, Linux, Android and Mac systems;
OpenCV (Open Source Computer Vision) is a library of programming functions containing all the standard algorithms for Computer Vision;
Currently, functions are implemented in C/C++, Python and Java;
OpenCV libraries can run on Windows, Linux, Android and Mac systems;
Latest version is 2.4.4.
Where?

- Download;
Where?

- Download;
- Documentation;
Where?

- Download;
- Documentation;
- Tutorials.
Other Resources

- Book: *Learning OpenCV: Computer Vision with the OpenCV Library*;
- Wiki;
- C/C++ Cheatsheet (PDF);
Main Packages

**core** Basic functionalities and data structures;
Main Packages

**core** Basic functionalities and data structures;

**imgproc** Image processing functions (blurring, histograms, registration, tracking, detection);
Main Packages

- **core** Basic functionalities and data structures;
- **imgproc** Image processing functions (blurring, histograms, registration, tracking, detection);
- **highgui** High-level Graphical User Interface;
Main Packages

- **core**: Basic functionalities and data structures;
- **imgproc**: Image processing functions (blurring, histograms, registration, tracking, detection);
- **highgui**: High-level Graphical User Interface;
- **calib3d**: Camera calibration and 3D Reconstruction;
Main Packages

- **core**: Basic functionalities and data structures;
- **imgproc**: Image processing functions (blurring, histograms, registration, tracking, detection);
- **highgui**: High-level Graphical User Interface;
- **calib3d**: Camera calibration and 3D Reconstruction;
- **features2d**: Features detection and description;
Library structure

Main Packages

- **core**: Basic functionalities and data structures;
- **imgproc**: Image processing functions (blurring, histograms, registration, tracking, detection);
- **highgui**: High-level Graphical User Interface;
- **calib3d**: Camera calibration and 3D Reconstruction;
- **features2d**: Features detection and description;
- **objdetect**: Object detection;
Main Packages

- **core**: Basic functionalities and data structures;
- **imgproc**: Image processing functions (blurring, histograms, registration, tracking, detection);
- **highgui**: High-level Graphical User Interface;
- **calib3d**: Camera calibration and 3D Reconstruction;
- **features2d**: Features detection and description;
- **objdetect**: Object detection;
- **ml**: Machine Learning and Pattern Recognition tools (e.g. k-means, SVM, knn)
**Main Packages**

- **core**: Basic functionalities and data structures;
- **imgproc**: Image processing functions (blurring, histograms, registration, tracking, detection);
- **highgui**: High-level Graphical User Interface;
- **calib3d**: Camera calibration and 3D Reconstruction;
- **features2d**: Features detection and description;
- **objdetect**: Object detection;
- **ml**: Machine Learning and Pattern Recognition tools (e.g. k-means, SVM, knn)

Data structures and functions belong to the namespace `cv`; therefore, to access this functionality from your code, use the `cv::` specifier or the `using namespace cv;` directive.
Let CVHOME be the folder where you have installed OpenCV.

```
CVHOME
  └── bin
  └── doc
  └── include
      └── opencv ......................... old versions’ C/C++ header files
      └── opencv2 ....................... new versions’ C/C++ header files
  └── lib
      └── opencv_{package_name}_{version}.d.lib .......... Debug libs
      └── opencv_{package_name}_{version}.lib .......... Release libs
```

Figure: OpenCV directory tree
Example

If you have installed OpenCV 2.4.2, the core functions and data structures are declared in `include/opencv2/core/core.hpp` and can be found in the library `opencv_core_242d.lib` (Debug) and `opencv_core_242.lib` (Release).
The Good old method

1. Create a new, blank, C++ Project;
2. Open Project Properties;
3. Click on VC++ Directories;
   a. Add CVHOME/include directory to Additional Include Directories;
4. Click on Linker;
   a. Click on General, then add CVHOME/lib directory to Additional Library Directories;
   b. Click on Input, then add the lib filenames you need to Additional dependencies.
Local method

1. Create a new, blank, C++ Project;
2. Open the Property Manager;
3. Right click on Debug, then click on Add New Property Sheet;
4. Apply the **Good old method** and save the property sheet;
5. Do steps 2 and 3 for Release;
6. Next time you want to create a new OpenCV-based project, just click on Add Existing Property Sheet.
Global method

- **Visual Studio 2008**
  1. Go to Tools → Options → Projects and Solutions;
  2. Apply the **Good old method**.

- **Visual Studio 2010**
  1. Apply the local method to the Global Property Sheet (it should be named Microsoft.Cpp.Win32.user).

**Remember that** OpenCV libraries for Debug and Release **differs only in the final d!**
**Be careful ;-)**
Geometric primitives - Points and Rectangles

**Point**  Template class that represents a 2-column vector containing the coordinates of a point in a plane;

Implemented classes:

- **Point**, **Point2i**: `integer` coordinates;
- **Point2d**: `double` coordinates;
- **Point2f**: `float` coordinates.
Geometric primitives - Points and Rectangles

Point Template class that represents a 2-column vector containing the coordinates of a point in a plane;

Implemented classes:

- Point, Point2i: integer coordinates;
- Point2d: double coordinates;
- Point2f: float coordinates.

Point3 Template class that represents a 3-column vector containing the coordinates of a point in the space;

- Point3, Point3i: integer coordinates;
- Point3d: double coordinates;
- Point3f: float coordinates.
Geometric primitives - Points and Rectangles

**Point** Template class that represents a 2-column vector containing the coordinates of a point in a plane;

Implemented classes:

- **Point, Point2i**: integer coordinates;
- **Point2d**: double coordinates;
- **Point2f**: float coordinates.

**Point3** Template class that represents a 3-column vector containing the coordinates of a point in the space;

- **Point3, Point3i**: integer coordinates;
- **Point3d**: double coordinates;
- **Point3f**: float coordinates.

**Rect** Template class that represents a rectangle, defined by the upper-left corner coordinates, width and height.
Geometric primitives - Points and Rectangles

Point  Template class that represents a 2-column vector containing the coordinates of a point in a plane;

Implemented classes:
- Point, Point2i: integer coordinates;
- Point2d: double coordinates;
- Point2f: float coordinates.

Point3  Template class that represents a 3-column vector containing the coordinates of a point in the space;

- Point3, Point3i: integer coordinates;
- Point3d: double coordinates;
- Point3f: float coordinates.

Rect  Template class that represents a rectangle, defined by the upper-left corner coordinates, width and height.
Geometric primitives - Example

```
Point p1(10, 10);
Point2i p2(11, 11);
Point2d p3(1.0, 1.0);
Point2f p4(2.0f, 2.0f);

Point3i p5(10, 10, 10);
Point3d p6(1.0, 1.0, 1.0);
Point3f p7(1.0f, 1.0f, 1.0f);

Rect<int> r1(1, 1, 100, 100);
Rect<int> r2(p1, p2);
Rect<double> r3(p3, p3*10.0);
```

Listing 1: Ex01.cpp
Mat is the **fundamental** class for doing everything in OpenCV. Mat can represent:

- a matrix;
Mat is the **fundamental** class for doing everything in OpenCV. Mat can represent:

- a matrix;
- a filter;
**Class Mat**

**Mat** is the **fundamental** class for doing everything in OpenCV. **Mat** can represent:

- a matrix;
- a filter;
- an image;
Class Mat

Mat is the **fundamental** class for doing everything in OpenCV. Mat can represent:

- a matrix;
- a filter;
- an image;
- a set of vectors (e.g. descriptors);
Class Mat

Mat is the fundamental class for doing everything in OpenCV. Mat can represent:

- a matrix;
- a filter;
- an image;
- a set of vectors (e.g. descriptors);
- etc.
Class Mat

Mat is the **fundamental** class for doing everything in OpenCV. Mat can represent:

- a matrix;
- a filter;
- an image;
- a set of vectors (e.g. descriptors);
- etc.

A Mat object should always be released before exiting the program using the Mat::release() function!
Mat as an image

```cpp
void loadImage() {
    Mat img = imread("lena.bmp");
    imshow("Window", img);
    waitKey(0);
    img.release();
}
```

Listing 2: Ex02-Img.cpp
# Mat as a matrix

```cpp
... 
#include <opencv2/imgproc/imgproc.hpp> 
... 
void rotationMatrix() { 
    Mat rot; 
    rot = getRotationMatrix2D(Point2f(0.0f, 0.0f), 30.0, 1.0); 

    for(int i=0; i < rot.rows; i++) { 
        for(int j=0; j < rot.cols; j++) 
            cout << rot.at<double>(i, j) << " \t" ;
        cout << endl ;
    }
}
```

Listing 3: Ex02-Mat.cpp
Mat as a filter

#include <opencv2/imgproc/imgproc.hpp>

void createFilter() {

    Mat flt;
    flt = getGaussianKernel(5, 2.0);

    for(int i=0; i < flt.rows; i++) {
        for(int j=0; j < flt.cols; j++)
            cout << flt.at<double>(i, j) << "\t";
        cout << endl;
    }
}

Listing 4: Ex02-Flt.cpp
Mat as a set

```cpp
void createObservationsSet() {
    Mat data(10, 5, CV_32S);
    for(int i=0; i<10; i++) {
        for(int j=0; j<5; j++)
            data.at<int>(i, j) = rand() * 256;
    }

    for(int i=0; i < data.rows; i++) {
        for(int j=0; j < data.cols; j++)
            cout << data.at<int>(i, j) << "\t";
        cout << endl;
    }
}
```

Listing 5: Ex02-Set.cpp
Notes on previous slides

As you probably noticed, we used two functions:

1. `getRotationMatrix2D`;
2. `getGaussianKernel`.

Both belong to the `imgproc` subset, so we included the correspondent header file (see source code for details).
Loading, Showing and Writing an image

- All graphical user interface (GUI) utilities are defined in `highgui` library;
- Include directive: `opencv2\highgui\highgui.hpp`;
- `imread(filename)`: reads the image from the filename passed as a parameter. Returns a `Mat`;
- `imshow(windowname, image)`: show the image in a new window titled `windowname`;
- `imwrite(filename, image, params)`: write the image in a new file named `filename` using the format specific `params`. The format is derived from the extension given in the filename.
Loading, Showing and Writing an image

- All graphical user interface (GUI) utilities are defined in `highgui` library;
- Include directive: `opencv2\highgui\highgui.hpp`;
- `imread(filename)`: reads the image from the filename passed as a parameter. Returns a `Mat`;
- `imshow(windowname, image)`: show the `image` in a new window titled `windowname`;
- `imwrite(filename, image, params)`: write the `image` in a new file named `filename` using the format specific `params`. The format is derived from the extension given in the filename.
Graphic User Interface

Loading, Showing and Writing an image

- All graphical user interface (GUI) utilities are defined in `highgui` library;
- Include directive: `opencv2\highgui\highgui.hpp`;
- `imread(filename)`: reads the image from the filename passed as a parameter. Returns a `Mat`;
- `imshow(windowname, image)`: show the `image` in a new window titled `windowname`;
- `imwrite(filename, image, params)`: write the `image` in a new file named `filename` using the format specific `params`. The format is derived from the extension given in the filename.
All graphical user interface (GUI) utilities are defined in highgui library;

Include directive: opencv2\highgui\highgui.hpp;

imread(filename): reads the image from the filename passed as a parameter. Returns a Mat;

imshow(windowname, image): show the image in a new window titled windowname;

imwrite(filename, image, params): write the image in a new file named filename using the format specific params. The format is derived from the extension given in the filename.
Loading, Showing and Writing an image

- All graphical user interface (GUI) utilities are defined in highgui library;
- Include directive: opencv2\highgui\highgui.hpp;
- `imread(filename)`: reads the image from the filename passed as a parameter. Returns a Mat;
- `imshow(windowname, image)`: show the image in a new window titled `windowname`;
- `imwrite(filename, image, params)`: write the image in a new file named `filename` using the format specific `params`. The format is derived from the extension given in the filename.
Loading, Showing and Writing an image

Listing 6: Ex03.cpp

```cpp
// Read the image
cv::Mat lena = cv::imread("lena.bmp");

// Show the image
cv::imshow("Lena", lena);
cv::waitKey(0);

// Set params
std::vector<int> jpg_param;
jpg_param.push_back(CV_IMWRITE_JPEG_QUALITY);
jpg_param.push_back(50);

// Write the image
cv::imwrite("lena.jpg", lena, jpg_param);
```

M. Moltisanti – OpenCV
Apply a built-in filter

- Include directive: `opencv2\imgproc\imgproc.hpp`
- Load the image;
- Create the output image;
- Create and apply the filter;
- Show the image;
- Save the image.
Apply a built-in filter

- Include directive: `opencv2\imgproc\imgproc.hpp`
- Load the image;
- Create the output image;
- Create and apply the filter;
- Show the image;
- Save the image.
Apply a built-in filter

- Include directive: `opencv2\imgproc\imgproc.hpp`;
- Load the image;
- Create the output image;
- Create and apply the filter;
- Show the image;
- Save the image.
Apply a built-in filter

- Include directive: `opencv2\imgproc\imgproc.hpp`
- Load the image;
- Create the output image;
- **Create and apply the filter**;
- Show the image;
- Save the image.
Apply a built-in filter

- Include directive: opencv2\imgproc\imgproc.hpp;
- Load the image;
- Create the output image;
- Create and apply the filter;
- Show the image;
- Save the image.
Apply a built-in filter

- Include directive: `opencv2\imgproc\imgproc.hpp`;
- Load the image;
- Create the output image;
- Create and apply the filter;
- Show the image;
- Save the image.
Apply a built-in filter

```cpp
// Read the image
Mat lena = imread("lena.bmp");

// Show the original image
imshow("Lena_original", lena);

// Create the output image
Mat lena_canny = lena.clone();

// Create and apply the filter
Canny(lena, lena_canny, m[0]*0.66, m[0]*1.33);
```

Listing 7: Ex04.cpp
Video from device

- Video handling is included in highgui package;
- Include directive: opencv2\highgui\highgui.hpp;
- Declare a VideoCapture(id) object; id is the camera identification number;
- Enter a loop;
- In each iteration:
  - Grab a frame using the >> operator and store it in a Mat object;
  - Check if the frame is empty;;
  - Show the frame;
Video from device

- Video handling is included in the highgui package;
- Include directive: opencv2\highgui\highgui.hpp;
- Declare a VideoCapture(id) object; id is the camera identification number;
- Enter a loop;
- In each iteration:
  - Grab a frame using the >> operator and store it in a Mat object;
  - Check if the frame is empty;;
  - Show the frame;
Video from device

- Video handling is included in `highgui` package;
- Include directive: `opencv2\highgui\highgui.hpp`;
- **Declare a VideoCapture(id) object; id is the camera identification number**;
- Enter a loop;
- In each iteration:
  - Grab a frame using the `>>` operator and store it in a `Mat` object;
  - Check if the frame is empty;;
  - Show the frame;
Video from device

- Video handling is included in `highgui` package;
- Include directive: `opencv2\highgui\highgui.hpp`;
- Declare a `VideoCapture(id)` object; `id` is the camera identification number;
- Enter a loop;
- In each iteration:
  - Grab a frame using the `>>` operator and store it in a `Mat` object;
  - Check if the frame is empty;;
  - Show the frame;
Video from device

- Video handling is included in **highgui** package;
- Include directive: `opencv2\highgui\highgui.hpp`;
- Declare a `VideoCapture(id)` object; *id* is the camera identification number;
- Enter a loop;
- In each iteration:
  - Grab a frame using the `>>` operator and store it in a **Mat** object;
  - Check if the frame is empty;;
  - Show the frame;
Video from device

- Video handling is included in `highgui` package;
- Include directive: `opencv2\highgui\highgui.hpp`;
- Declare a `VideoCapture(id)` object; `id` is the camera identification number;
- Enter a loop;
- In each iteration:
  - Grab a frame using the `>>` operator and store it in a `Mat` object;
  - Check if the frame is empty;;
  - Show the frame;
Video from device

- Video handling is included in *highgui* package;
- Include directive: `opencv2\highgui\highgui.hpp`;
- Declare a `VideoCapture(id)` object; *id* is the camera identification number;
- Enter a loop;
- In each iteration:
  - Grab a frame using the `>>` operator and store it in a `Mat` object;
  - Check if the frame is empty;;
  - *Show the frame;*
Video from device

```cpp
VideoCapture vc(0);
char res = '0';
int fps = vc.get(cv::CAP_PROP_FPS);
Mat frame;
while(1) {
    vc >> frame;
    if(frame.empty())
        break;
    imshow("Video", frame);
    res = (char)(waitKey(0));
    if(res == 'q')
        break;
}
vc.release();
```

Listing 8: Ex05-In.cpp
The difference between using a video from file and a video from a device is only in the constructor. Instead of VideoCapture(id) function, we will use the overloaded version VideoCapture(filename), where filename is the path to a video file.

Example

VideoCapture vc("video.mp4")
Save a video

- Declare a `VideoWriter` object and set the properties for the new video using the constructor;
- Put the frames in the `VideoWriter` object using the `<<` operator;
- Release the `VideoWriter` object.
Save a video

- Declare a `VideoWriter` object and set the properties for the new video using the constructor;
- Put the frames in the `VideoWriter` object using the `<<` operator;
- Release the `VideoWriter` object.
Save a video

- Declare a `VideoWriter` object and set the properties for the new video using the constructor;
- Put the frames in the `VideoWriter` object using the `<<` operator;
- Release the `VideoWriter` object.
Save a video

```cpp
Size s(vc.get(CV_CAP_PROP_FRAME_WIDTH), vc.get(CV_CAP_PROP_FRAME_HEIGHT));
VideoWriter vw("videoOut.avi", vc.get(CV_CAP_PROP_FOURCC), vc.get(CV_CAP_PROP_FPS), s, true);

while(1) {

    ...
    flip(frame, newFrame, 1);
    vw << newFrame;
    imshow("It's something!", newFrame);
    res = (char) waitKey(1000/fps);
    ...
}
```

Listing 9: Ex05-Out.cpp
Available methods

- Statistical Models;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
- SVM;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
- SVM;
- Decision Trees;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
- SVM;
- Decision Trees;
- Boosting;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
- SVM;
- Decision Trees;
- Boosting;
- Gradient Boosted Trees;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
- SVM;
- Decision Trees;
- Boosting;
- Gradient Boosted Trees;
- Random Trees;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
- SVM;
- Decision Trees;
- Boosting;
- Gradient Boosted Trees;
- Random Trees;
- EM algorithm;
Available methods

- Statistical Models;
- Normal Bayes Classifier;
- K-NN;
- SVM;
- Decision Trees;
- Boosting;
- Gradient Boosted Trees;
- Random Trees;
- EM algorithm;
- Neural Networks.
Available methods

Include Directive

Include path: `<opencv2\ml\ml.hpp>

Documentation is fundamental using ml functions!
Tips & Tricks

Tip #1
Documentation is your biggest friend while developing!

Tip #2
Community is your second biggest friend!
Google to find out how your colleagues all around the world solved their problems!

Tip #3
Don’t try to copy!
In conclusion...

If you have doubts don’t hesitate to contact me at:
- moltisanti@dmi.unict.it;
- IPLab (room 146 - ground floor);
- Room 31 - first floor in the classrooms area.

Slides available at
- Course page
- Personal page

Code available here - Personal page
Question time!