Introduction to OpenCV 2.x

Jesús Nuevo Chiquero
jnuevo@depeca.uah.es

Department of Electronics, University of Alcalá.

December 9th, 2010

**Slides are support material, not a replacement for the actual lecture*

Universidad de Alcalá

Outline

Universidad de Alcalá

OpenCV 2.1

- Today: an introduction to OpenCV 2.1, and the **new C++ API**.
- OpenCV 2.2 was released 4 days ago (Dec. 5th, 2010).
  - New internal, more modular structure
  - More functionality and new algorithms
  - Nvidia CUDA support (early-beta)
  - Android support
  - Qt, OpenGL support in highgui (integrated tools)

Universidad de Alcalá

OpenCV 2.1: General description

- Multi-platform: GNU/Linux, Mac OS X, MS Windows
- Various APIs: C, **C++**, Python, Matlab/Octave (limited)
- Lots of functionality:
  - **Basic algebra** operations ($\sim$ BLAS).
  - Image/matrix manipulation. Dinamic data structures.
  - **Image processing**: filtering, edge/corner detection, histogram analysis, morphological operations.
  - **Structural analysis**: connected components, contours, distance transform, template correlations, Hough transform, shape fitting,...
  - **Motion analysis and tracking**: optical flow, movement segmentation, tracking,...
  - **Object recognition**: PCA, SVM, ...
  - **Basic GUI**: I/O handling, image loading and saving.

Universidad de Alcalá

OpenCV 2.1: Internal structure & resources

- OpenCV 2.1 has several modules
  - **cxcore:** core libraries. Basic structures, algebra and other math. Drawing functions.
  - **cv:** computer vision. Processing functions.
  - **cvaux:** auxiliar functionality (some experimental).
  - **HighGUI:** image and video I/O. Image displaying, and basic GUI.
  - **ML:** machine learning. Learning and classification algorithms (Neural networks, SVM, Adaboost, etc).
- Resources - check them!
  - Web page: `http://opencv.willowgarage.com`
  - User group: `http://tech.groups.yahoo.com/group/OpenCV/`
  - The CheatSheet is very convenient!

Universidad de Alcalá

Linux: installation

- Available in every big distribution. (K)Ubuntu, Debian: `apt-get install ...`.
- From source: you need **CMake**, and some libraries.

Compilation using CMake

- CMake reads `CMakeLists.txt` and generates the Makefiles.
- See provided files for a template.

Universidad de Alcalá

OpenCV 2.1 C++ API

- We will see the new C++ API. The old C API is still fully supported.
- Internals are common to all APIs (mostly).
- C++ API allows for nice things[1]
  - Closer to *MATLAB syntax*
  - `Smart pointers` $\Rightarrow$ forget about memory management! (mostly)

```
cv::Mat Z0 = Mat::zeros(3,3, CV_32F);
Z0 = Z0 + 4; //add scalar to all elements
cv::Mat B = Z0.diag(0);
cv::Mat C = Z0.row(2);
C = Z0.rowRange(Range(0,2));
cv::Mat im = imread(file);
```

  - Some classes are templates. They can take STL types.
- OpenCV namspace is "cv"

[1] You need to know little about C++ to use this API

Universidad de Alcalá

Basic types I

1. **Point** A 2D point, with integer values.

```
Point point (10,0); //Point p( x,y ) coordinates
Point2f pt_float; pt_float.x = 1.034; //also float values, and 3D points
Point3d pt3(0.1, 0.3, -3e4); //3D, double
//point *4;//check the reference for useful functionality
```

2. **Size** A size

```
Size sz = Size(100,200);
Size2f sz_float; sz_float.width = 10.24;
```

3. **Rect** A rectangle

```
Rect r1(0,0, 100,200), r2 (10, 20, 100, 150);
//some interesting functionality
Rect r = r1 | r2; // r = min rectangle covering r1 and r2
r = r1 & r2; //intersectio of r1 and r2
r = r2 + point; //displace the rectangle
point = r2. tl(); //top-left. .br() = bottom-right point.
```

```
//You can also have Rectangles with float values
Rect_<float> rect2f(Point2f(1.3, 100), Size2f(300.2, 210.234));
```

Universidad de Alcalá

Basic types II

4. **Vec** A vector.

```
Vec3f point = Vec3f(10,10,3.2);//Float, 3 components
Mat mat(3,3,CV_32FC3,);//3 channel matrix
Vec3f v3f = mat.at<Vec3f>(y, x);//read color values for pixel (y,x)
mat.at<Vec3f>(y,x) = Vec3f:all(10);//set values for that pixel
```

5. **Scalar** A 4-element vector, double precision.

```
template<typename _Tp> class Scalar_ : public Vec<_Tp, 4>
Scalar v(10);//v[0] == 10, v[1-3] =0
Scalar v(10, 20);//v[0]==10, v[1]==20, v[2-3] =0
Scalar v = Scalar::all(10);//all v[i] =10;
```

6. **Range** A continous range

```
Range range(0, 5);  //Range(init, end). From 'init' to 'end-1'
```

7. **TermCriteria** Termination criteria for iterative operations

```
TermCriteria(int _type, int _maxCount, double _epsilon);
//type can be MAX_ITER, EPS or MAX_ITER+EPS.
//MAX_ITER = maximum iterations
//EPS = algorithm runs until this precision is achieved
//MAX_ITER+EPS = algorithm runs until either criteria is reached
```

Universidad de Alcalá

The Mat class I

This is the basic type in OpenCV. Covers the old CvMat and IplImage.

- Data representation.
  - Do not access member directly: use the methods!*
  - Data is row ordered:
  - Color pixels are interleaved.
- Let's see some important members of the class:
  - Create and initialize

```
// Mat(int _rows, int _cols, int _type);
// Mat(Size _size, int _type); type = CV_8UC3, CV_32FC1, ...
// Mat(Size _size, int _type, const Scalar& _s); fill with values in _s
Mat M(7,7,CV_32FC2,Scalar(1,3));//7x7, float, 2 channels, fill with (1,3)
M.create(Size(15,15), CV_8U);//reallocate (if needed)
//Matlab-like initializers
Mat ident = Mat::eye(3,3, CV_32F);//also Mat::ones(..) and Mat::zeros(..)
int* data = {1,2,3,9,0,-3};
Mat C (2,3,CV_32S, data); //no data copied.
C = C.clone(); //clone the matrix -> now the data is created.
```

Universidad de Alcalá

The Mat class II

- Important things to know:
  - ★ Shallow copy: `Mat A = B;` does **not** copy data.
  - ★ Deep copy: `clone()` and/or `B.copyTo(A);` (for ROIs, etc).
  - ★ Most OpenCV functions can resize matrices if needed
- Lots of convenient functionality (Matrix Expresions):
  - `s` is a `cv::Scalar`, α scalar (double)
  - Addition, scalation, ...: `A±B`, `A±s`, `s±A`, α`A`
  - Per-element multiplication, division...: `A.mul(B)`, `A/B`, α`/A`
  - Matrix multiplication, dot, cross product: `A*B`, `A.dot(B)`, `A.cross(B)`
  - Transposition, inversion: `A.t()`, `A.inv([method])`
  - And a few more.

Universidad de Alcalá

Mat class: element access I

- Rows, columns, ROIs,...

```
Mat A = B.row(int row); //same for B.col()
A = B.rowRange(Range rg);//same for B.colRange()
A = B(Rect r);//use a rectangle to set ROI
```

  - Ranges, ROIs, etc... only create new headers.
  - Where is a ROI in the bigger matrix?

```
Mat A = B(Rect r);
Size s; Point offset;
A.locateROI(s, offset); //'s' and 'offset' will define the rectangle 'rect'
```

- Element access: 3 options
  1. Using at<>()

```
double val = M.at<double>(i,j);//You have to know the type
```

  2. Old C style.

Universidad de Alcalá

Mat class: element access II

```
// compute sum of positive matrix elements
double sum=0;
for(int i = 0; i < M.rows; i++)
{
  const double* Mi = M.ptr<double>(i); //we know it's double data
  for(int j = 0; j < M.cols; j++)
  sum += std::max(Mi[j], 0.);
}
```

3. STL-like iterators

```
// compute sum of positive matrix elements, iterator-based variant
double sum=0;
MatConstIterator_<double> it = M.begin<double>(), it_end = M.end<double>();
for(; it != it_end; ++it)
   sum += std::max(*it, 0.);
```

- ★ This iterators can be used with STL functions, like `std::sort()`

Universidad de Alcalá

The Mat_ class I

- A thin "wrap" around the Mat class. Mat $\leftrightarrow$ Mat_ can be converted freely
  - With care: no data conversion is done
- Type specification is different
- Useful if you do lots of element access.
  - Same internal code, but shorted to write.

```
Mat_<double> M(20,20);//a double matrix 20x20
double k = M(2,18);//no data specification needed
```

- For multichannel, use `Vec`.

```
Mat_<Vec3f> M3f(20,20);//a 20x20 3 channel float matrix
```

Universidad de Alcalá

Thresholding

```
#include <cv.h>
#include <highgui.h>

using namespace std;
using namespace cv;

int main( int argc, char** argv )
{
    Mat src, gray, grayThresh;

    src = imread(argc >= 2 ? argv[1] : "fruits.jpg", 1);

    gray.create(src.size(), CV_8U);//not needed, actually

    namedWindow("src", CV_WINDOW_AUTOSIZE);
    namedWindow("gray", CV_WINDOW_AUTOSIZE);
    namedWindow("grayThreshold", CV_WINDOW_AUTOSIZE);

    cvtColor(src, gray, CV_BGR2GRAY); //color images are BGR!

    threshold(gray, grayThresh, 100, 250, CV_THRESH_BINARY);

    imshow("src", src);    imshow("gray", gray);
    imshow("grayThreshold", grayThresh);

    waitKey(0); //waits for a key: it also handles the GUI events.

    return 0; //no need to free the matrices, they are deleted automatically
  }
```

Universidad de Alcalá

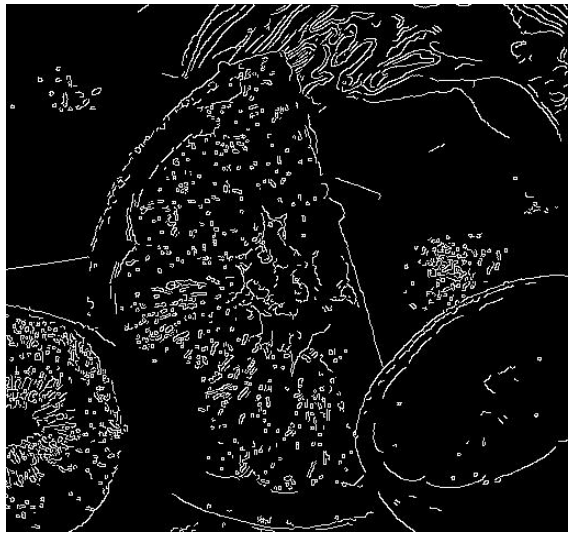
Thresholding

```
#include <cv.h>
#include <highgui.h>

using namespace std;
using namespace cv;

int main( int argc, char** argv )
{
    Mat src, gray, grayThresh;

    src = imread(argc >= 2 ? argv[1] : "fruits.jpg", 1);

    gray.create(src.size(), CV_8U);//not needed, actually

    namedWindow("src", CV_WINDOW_AUTOSIZE);
    namedWindow("gray", CV_WINDOW_AUTOSIZE);
    namedWindow("grayThreshold", CV_WINDOW_AUTOSIZE);

    cvtColor(src, gray, CV_BGR2GRAY); //color images are BGR!

    threshold(gray, grayThresh, 100, 250, CV_THRESH_BINARY);

    imshow("src", src);    imshow("gray", gray);
    imshow("grayThreshold", grayThresh);

    waitKey(0); //waits for a key: it also handles the GUI events.

    return 0; //no need to free the matrices, they are deleted automatically
  }
```

Universidad de Alcalá

Canny border detector

```
int main( int argc, char** argv )
{

    Mat src, dst;

    src = imread(argc >= 2 ? argv[1] : "fruits.jpg", 0);

    //  dst = Mat(src.size(), src.type());

    Canny(src, dst, 100, 150, 3);

    namedWindow("src"); imshow("src", src);
    namedWindow("canny"); imshow("canny", dst);

    waitKey(0);

    return 0;
}
```



Universidad de Alcalá

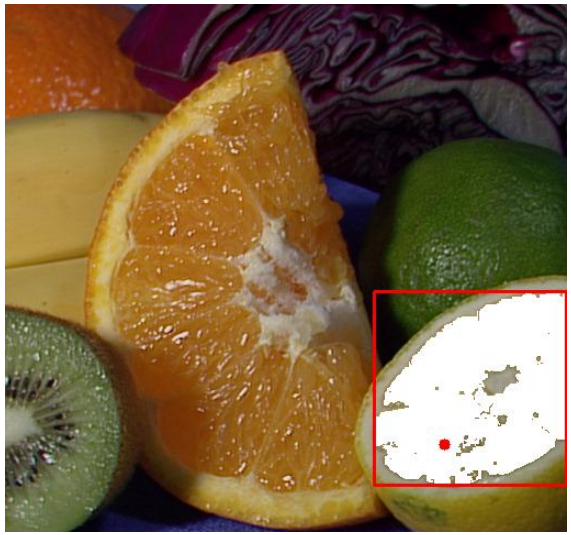
Contour analysis

```
int main( int argc, char** argv )
{
    Mat newImg, grayImg, contourImg;
    vector<vector<Point> > contours;

    namedWindow("SRC");
    namedWindow("Contours");
    namedWindow("Edges");

    newImg = imread(argc >= 2 ? argv[1] : "fruits.jpg", 1);

    imshow("SRC", newImg);

    cvtColor(newImg, grayImg, CV_BGR2GRAY);

    Canny(grayImg, grayImg, 100, 150, 3);
    imshow("Edges", grayImg);

    contourImg = newImg.clone();

    findContours(grayImg, contours, CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);
    drawContours(contourImg, contours, -1, CV_RGB(0, 255, 0));

    imshow("Contours", contourImg);

    waitKey(0);

    return 0;
}
```

Universidad de Alcalá

Flood and fill

```
int main( int argc, char** argv )
{

    Mat newImg, ffImg;
    Rect rect;

    namedWindow("SRC");
    namedWindow("Flood & fill");

    newImg = imread(argc >= 2 ? argv[1] : "fruits.jpg", 1);

    imshow("SRC", newImg);

    ffImg = newImg.clone();

    floodFill(ffImg, Point(400,400), CV_RGB(255,255,255), &rect,
              CV_RGB(7,7,7), CV_RGB(5,5,5));


    circle(ffImg,Point(400,400), 5, CV_RGB(255,0,0), -1);
    rectangle(ffImg, rect, CV_RGB(255,0,0),1);

    imshow("Flood & fill", ffImg);
    waitKey(0);

    return 0;
}
```



Universidad de Alcalá

Loading and saving images

- Reading and writing images is very easy

```
Mat imread(const string& filename, int flags=1);
//flags =0 -> always grayscale
//flags >0 -> always color
//flags <0 -> read image as-is

bool imwrite(const string& filename, const Mat& img,
   const vector<int>& params=vector<int>());
//params set compressions values. defaults are fine.

Mat img = imread("filename.jpg", 1);

imwrite("file.png", myImage);
```

Universidad de Alcalá

Reading and writing sequences (AVIs)

- Requires FFMPEG.
- An example of how to read an AVI file

```
...
VideoCapture cap(video_file); // open the file
if(!cap.isOpened()) // check if we succeeded
 return -1;
Mat edges;
namedWindow("edges",1);
for(;;)
{
  Mat frame;
  cap >> frame; // get a new frame from camera
  cvtColor(frame, edges, CV_BGR2GRAY);
  GaussianBlur(edges, edges, Size(7,7), 1.5, 1.5);
  Canny(edges, edges, 0, 30, 3);
  imshow("edges", edges);
  if(waitKey(30) >= 0) break;
}

cap.release();
...
```

- Use `.get(..)` and `.set(..)` to obtain and set properties of the video stream.

Universidad de Alcalá

Writing sequences (AVIs)

- An example of how to read an AVI file

```
VideoWriter writer;

Mat myImage = Mat(Size(320,240), CV_8UC3);

writer.open("my_sequence.avi", CV_FOURCC('M','P','4','2'), 25, myImage.size());
...
if(writer.isOpened())
    writer << myImage;
...

return 0;
```

Universidad de Alcalá

Some useful tips

- OpenCV error handling uses C++ exceptions.
- Check the API reference (online). Some functions only take matrices of certain types.
- Do **check the data** that you read (images, or other).
- **Document your progress**: it will help if things go wrong.
- **Test** you code as you write: so you don't accumulate errors.
  - This helps a lot when debugging.
- Finally, when programming:
  1. Think first about your algorithm
  2. Think again
  3. **Then code**

Universidad de Alcalá