OpenCV Tutorial

(OpenCV 2.1)

Denis Aleshin

Overview

1. Installation (Windows/Linux/Mac)
2. Tutorial 1 - Reading/Writing videos
3. Tutorial 2 - Reading/Writing images and image processing
4. Tutorial 3 - Mouse input and point tracking
5. Tutorial 4 - Haar cascades and face detection

Installation

Linux -

```
sudo apt-get install opencv
```

Mac -

get macports - http://www.macports.org/

```
sudo port install opencv
```

OR

get XCode - http://developer.apple.com/xcode/

and use the OpenCV framework (1.2 instead of 2.1 unless you build your own).

Windows -

Get visual studio - https://www.dreamspark.com/

Download library - http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.2

Making your own projects

To use it, you will likely be writing your code in C++. So you need to link your code to the libraries. Two options:

1 - use an IDE like XCode or Visual Studio

Visual Studio - http://opencv.willowgarage.com/wiki/VisualC%2B%2B
XCode - http://opencv.willowgarage.com/wiki/Mac_OS_X_OpenCV_Port
Other - http://opencv.willowgarage.com/wiki/InstallGuide
(See point 4)

2 - create your own makefile

Makefile example in tutorials
CMake (more advanced) - http://www.cmake.org/

Tutorials

Documentation:

OpenCV 2.1 - http://opencv.willowgarage.com/documentation/python/index.html
OpenCV 2.0 - http://opencv.willowgarage.com/documentation/index.html

These slides will be available online.

Code from tutorials will be available online.

Tutorial 1

Video input is handled by the `VideoCapture` object for files and cameras. link

Functions in the tutorial:

```
VideoCapture capture;    // Declares capture.
capture.open(0);         // Initializes capture from camera.
capture.open(inputName); // Initializes capture from file.
capture.isOpened();
capture >> frame;        // Grabs frame from capture.
```

Tutorial 1

Important function:

```
capture.get(<flag>);
```

Is used to access capture properties, like size, encoding, color depth, etc... Flags used in tutorial:

```
CV_CAP_PROP_FOURCC
CV_CAP_PROP_FPS
CV_CAP_PROP_FRAME_WIDTH
CV_CAP_PROP_FRAME_HEIGHT
```

More Flags

Tutorial 1

Video output is handled by the VideoWriter object. Link

Functions from the tutorial:

```
 VideoWriter videoout; // Declare
videoout.open(const string& filename,
            int fourcc,
            double fps,
            Size frameSize,
            bool isColor=true);
videoout.isOpened();
videoout << frame; // Write frame to file.
```

Will probably use the get() function to get these.

Tutorial 1

First bits of UI. Link

```
cvNamedWindow("video", 1); // Creates a new window.

imshow( "video", frame ); // Shows an image in window.

char c = waitKey(33); // Waits for key input from user for 33 milliseconds.
```

Tutorial 1

Nice thing about OpenCV 2.0 and on - storage is managed for you. Images, video captures and writes, windows cleaned up when program exits.

Let's look at the code!

Tutorial 2

Images are stored in the `Mat` structure. Link

Can be used to store any matrix (histograms, voxels, etc...), and built with extra support for images.
Pixels stored in adjacent memory (so you can do pointer math for really fast code).

```
M.create(int height,int width,int depth);
```

Depth is a CV defined constant. For instance,
  CV_8UC1 is a single channel 8-bit image (greyscale)
  CV_32FC3 is a 3-channel 32 bit image (RGB).
Allocated and deallocated using reference counting (so no need to worry about it!)

See link above for details.

Tutorial 2

Reading and writing images Link

```
Mat img = imread(String filename);

imwrite(String filename, Mat img);
```

Sanity check:

```
img.empty();
```

Image processing.

Huge library! Lots of functions! Link
Usual format:

```
    Filter(Mat src, Mat dst, options...);
```

Examples from code:

```
    cvtColor(out, grey, CV_RGB2GRAY); // Link
    resize(frame, out, Size(0,0), 1/size, 1/size); // Link
    Sobel(grey, grey, CV_8UC1, 1, 1); // Link
```

Tutorial 2

Let's look at the code...

Tutorial 3

Lots of other useful code implemented!

Canny edge detector - *Link*

Find circles and lines - *Link*

Cascade Classifiers - *Link*

Histograms and Back-projection - Link

Meanshift and Camshift - Link

Tutorial 3

A bit more gui... set a callback for a window:

```
    cvSetMouseCallback(const char* windowName,
CvMouseCallback onMouse, void* param=NULL); //Link
```

```
    onMouse(int event, int x, int y, int flags, void* param);
```

And draw circles (or other stuff) on your image Link

```
    circle(image, Point, radius, color);
```

color is a Scalar object, like Scalar(255,0,0);

Tutorial 3

Tracking... several methods Link

```
calcOpticalFlowPyrLK(lastframe, frame, lastpoints, points, status, errors);
```

Let's look at how it does...

Tutorial 4

Haar cascades and code.