



Tutorial on OpenCV for Android Setup

EE368/CS232 Digital Image Processing, Spring 2015



Macintosh Version
For Motorola Droid phones

Introduction

In this tutorial, we will learn how to install OpenCV for Android on your computer and how to build Android applications using OpenCV functions. The tutorial is split into two parts. In the first part, we will explain how to download and install the OpenCV library onto your computer. Then, in the second part, we will explain how to build an application that draws different types of feature keypoints directly in the video viewfinder of the Android device, as shown in Figure 1.



Figure 1. Android application drawing MSER feature keypoints on a viewfinder frame.

Please note that this tutorial assumes that you have successfully completed the first Android tutorial for EE368/CS232, “Tutorial on Basic Android Setup”, which explains how to install the Android SDK and the Eclipse IDE. You will need to have already installed all the software tools mentioned in that other tutorial before moving onto this tutorial.

Estimated time to complete this tutorial: 2 hours

Part I: Installing OpenCV for Android

Downloading and Installing XCode

If you do not already have XCode on your computer, please download and install the latest version of XCode from the Mac OSX app store:

<http://developer.apple.com/xcode/>

1. Check if "make" is installed by typing in a command-line terminal:

```
which make
```

If installed, the installation path should be printed.

2. If "make" is not installed, try:
 - a. Start XCode.
 - b. Go to XCode/Preferences.
 - c. Click the "Downloads" tab.
 - d. Click "Components".
 - e. Click "Install" on the command line tools line.

Downloading and Installing CMake

We need to download and install the version 2.8.10.* (or higher) of CMake. Earlier versions are not recommended for the subsequent process of building OpenCV.

1. Download the "cmake-2.8.10.*-Darwin64-universal.dmg" for 64-bit OS or "cmake-2.8.10.*-Darwin-universal.dmg" for 32-bit OS from this page:
<http://www.cmake.org/cmake/resources/software.html>
2. Double-click the dmg file. Then, double-click the extracted pkg file.
3. Continue through the installation wizard. When asked to create symbolic links, enter "/usr/bin" for the "Install Folder" and select "Install Command Line Links".
4. Verify that CMake version.

```
cmake --version
```

The output should be: cmake version 2.8.10.*

Downloading and Installing Android NDK

The Android NDK enables us to compile and run native C/C++ code on Android. We will use a version of the NDK that supports STL and exceptions.

1. Download "android-ndk-r4-darwin-x86-crystax-4.tar.bz2" from this website:

<http://www.crystax.net/android/ndk-r4.php>

2. Unzip the downloaded file to your Android directory, for example:

```
cd /Users/yourname/Android # replace yourname with your user name
tar -xvf android-ndk-r4-darwin-x86-crystax-4.tar.bz2
```

Downloading and Installing PCRE and SWIG

1. Check if “swig” is installed by typing in a command-line terminal:

```
which swig
```

If installed, the installation path should be printed.
2. If “swig” is not installed, download PCRE from:

```
ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.32.zip
```
3. Unzip `pcre-8.32.zip` to `/Developer/pcre-8.32`.
4. Install PCRE:

```
cd /Developer/pcre-8.32
./configure
sudo make
sudo make install
```
5. Download SWIG from:

```
http://prdownloads.sourceforge.net/swig/swig-2.0.9.tar.gz
```
6. Extract `swig-2.0.9.tar.gz` to `/Developer/swig-2.0.9`.
7. Install SWIG:

```
cd /Developer/swig-2.0.9
./configure
sudo make
sudo make install
```
8. Check if SWIG is installed properly by:

```
which swig
```

It should return the installation path.

Downloading and Installing OpenCV

Now, we are ready to download and install the OpenCV library.

1. Download the OpenCV package from this location:

```
http://ee368.stanford.edu/Android/OpenCV/opencv.tar.gz
```
2. Unzip the package in an appropriate location, e.g., your Android folder:

```
cd /Users/yourname/Android # replace yourname with your user name
tar -xvf opencv.tar.gz
```
3. Make sure important programs are in the PATH. If “.bash_profile” does not yet exist in your home folder (e.g., `/Users/username`), create the file. Add the following lines to “.bash_profile”:

```
export NDK=/Users/yourname/Android/android-ndk-r4-crystax
export SDK=/Users/yourname/Android/android-sdk-macosx
```

```
export OPCV=/Users/yourname/Android/opencv
export PATH=$NDK:$SDK/tools:$SDK/platform-tools:$PATH
```

Be sure to change the locations above to match the folders where you placed Android NDK, Android SDK, and OpenCV. Remember to “source .bash_profile” for the changes to take effect. You can check that the desired environment variables are set correctly using “printenv”.

4. Build the OpenCV library.

a. Open the file “CMakeLists.txt” in the folder “\$OPCV/android” and replace line 60:

```
set(NDK_ROOT "$ENV{HOME}/android-ndk-r4-crystax" CACHE STRING "the
crystax ndk directory")
```

with:

```
set(NDK_ROOT "$ENV{NDK}")
```

b. Open the file “sample.local.env.mk” in the folder “\$OPCV/android/android-jni” and replace line 7:

```
ANDROID_NDK_ROOT=$(HOME)/android-ndk-r4-crystax
```

with:

```
ANDROID_NDK_ROOT=$(NDK)
```

c. Execute these commands:

```
cd $OPCV/android
mkdir build
cd build
cmake ..
```

d. Open the file “android-opencv.mk” in the folder “\$OPCV/android/build”. Remove the following expression from line 17:

```
$(OPENCV_ROOT)/modules/index.rst/include
```

This expression causes a compilation error if included.

e. Execute the following command to compile the OpenCV libraries:

```
make
```

This compilation can take a while, maybe 30 minutes or more.

5. Build the Java Native Interface (JNI) to the OpenCV library.

a. Execute the command

```
android list targets -c
```

and keep note of one of the Android versions displayed (e.g. android-19)

b. Open the file “project_create.sh” in the folder “\$OPCV/android/android-jni” and add

```
--target android-19
```

at the end of the last line (after “--path .”), where you should replace the version number with your version.

c. Execute the following commands

```
cd $OPCV/android/android-jni
```

```
make
make
sh ./project_create.sh
cp project.properties default.properties
ant debug
```

Part II: Developing an OpenCV Camera Application

Building the Default Camera Application

First, we will build and run the default camera application included in the OpenCV package. This application will demonstrate how to extract feature keypoints supported by OpenCV from viewfinder frames captured through the phone's camera and then overlay these keypoints in the viewfinder.

1. Execute the following command:

```
cd $OPCV/android/apps/CVCamera
```

2. Edit the file “sample.local.env.mk” in the current folder and replace line 3:

```
ANDROID_NDK_ROOT=$(HOME)/android-ndk-r4-crytax
```

with:

```
ANDROID_NDK_ROOT=$(NDK)
```

3. Like in part I of this tutorial, open the file “project_create.sh” and add

```
--target android-19
```

at the end of the last line (after “--path .”), where you should replace the version number with your version.

4. Make sure your device is connected to your computer. Execute the following commands:

```
make
make
sh ./project_create.sh
cp project.properties default.properties
ant debug
ant debug install
```

Assuming no errors occurred in the above steps, an application called “CVCamera” should have been installed to your Android device. Try opening the application and switching between the different feature keypoints (FAST, STAR, SURF).

More information about the different keypoint types can be found in the OpenCV documentation:

http://docs.opencv.org/modules/imgproc/doc/feature_detection.html

Building Our Improved Camera Application

Next, we will make some improvements to the previous camera application. Specifically, we will address the following issues:

- In the default application, the keypoints drawn in the viewfinder are sometimes a little hard to see against the background.
- In the default application, the keypoints are all drawn at the same scale, giving no indication of the true underlying scale, which is important to know for multi-scale detectors like SURF.
- In the default application, the popular Maximally Stable Extremal Region (MSER) is not included.

All three issues are fixed in our improved camera application:

- We enhance the visibility of the keypoints against a cluttered background by drawing yellow-on-black keypoints.
- We resize each keypoint according to its true underlying scale for multi-scale detectors like SURF.
- We add MSER keypoints as a new option to the three other types (FAST, STAR, SURF). MSER actually runs a lot faster than SURF.

Here are the steps to build the new CVCamera applications.

1. Download and unzip the following zip file to the “\$OPCV/android/apps” folder:
http://ee368.stanford.edu/Android/OpenCV/CVCamera_MSER.zip
2. In this folder, edit the “project_create.sh” script with the relevant android version and execute these commands:

```
sh project_create.sh
make clean
make V=0
ant clean
ant debug
ant debug install
```

When the new CVCamera application runs, you have the options shown in Figure 3. Choosing “MSER” yields the type of keypoints shown in Figure 1.

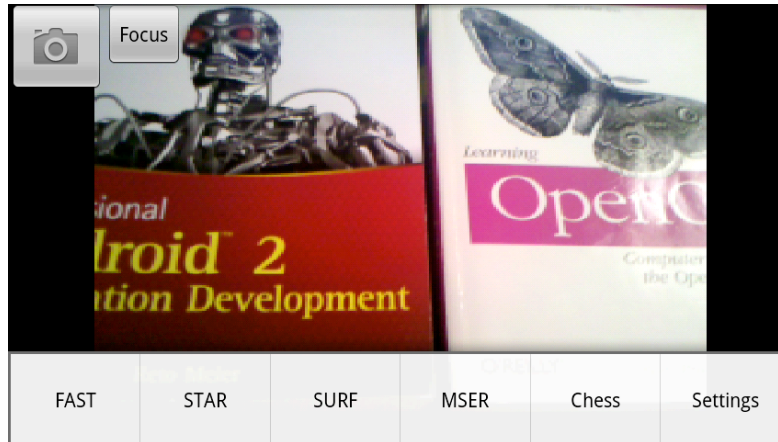


Figure 3. Menu for choosing between different types of feature keypoints.

Using Eclipse to Compile NDK Applications

Using the command line to compile and link the NDK applications can become tedious after a while. Fortunately, there is a way to integrate the compilation and linking process into Eclipse. After we perform the integration, each time a C/C++ source file in the project changes, Eclipse will perform compilation and linking on our behalf.

Here are the required steps for integration.

1. Import “Android JNI” project into Eclipse.
 - a. Open Eclipse. Choose File > New > Project > Android > Android Project from Existing Source.
 - b. Browse for the full path to the “Android OpenCV JNI” project, for example:
/Users/username/Android/opencv/android/android-jni
 - c. Make sure the project name is “OpenCV”. Click Finish.
 - d. Choose Project > Properties. Click on Java Compiler.
 - e. Make sure “Enable project specific settings” is selected. Then, make sure the “Compiler compliance levels” is at least 1.6. Accept any warnings about rebuilding the project.
2. Import “CVCamera_MSER” project into Eclipse.
 - a. Import the “CVCamera_MSER” project into Eclipse following the same type of instructions as in Step 1. Be sure to find the full path to the “CVCamera_MSER” project, for example:
/Users/username/Android/opencv/android/apps/CVCamera_MSER
 - b. Choose Project > Properties. Click on Builders. Click New. Choose Program.

- c. In the Main tab, enter the following information and then click Apply.
 - i. Name: Native Builder
 - ii. Location: /usr/bin/make
 - iii. Working Directory:
/Users/username/Android/opencv/android/apps/CVCamera_MSER
 - iv. Arguments: V=0

- d. In the Refresh tab, execute the following steps and then click Apply.
 - i. Make sure “Refresh resources upon completion” is checked.
 - ii. Select “Specify resources”.
 - iii. Make sure “Recursively include sub-folders” is checked.
 - iv. Click on Specify Resources, make sure (only) “CVCamera_MSER/libs” is checked, and click Finish.

- e. In the Build Options tab, execute the following steps and then click Apply.
 - i. Make sure “After a Clean” is checked.
 - ii. Make sure “During manual builds” is checked.
 - iii. Make sure “During auto builds” is checked.
 - iv. Make sure “During a Clean” is not checked.
 - v. Click on Specify Resources, make sure (only) “CVCamera_MSER/jni” is checked, and click Finish.

- f. Try adding a comment to one of the “.cpp” files in the “jni” folder and save the changes. Eclipse should automatically invoke the compilation and linking process and then take the place of Ant in packaging everything into an “.apk” file.