Tutorial on OpenCV for Android Setup

EE368/CS232 Digital Image Processing, Spring 2014

Windows Version
For API 11 (Android 3.0) or Higher

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. First, we will explain how to download and install the OpenCV library onto your computer. Second, we will explain how to build applications that directly call OpenCV functions on the viewfinder frames of the Android device, as shown in Figure 1.

Figure 1. Android application drawing face detections 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: 1.5 hours

Part I: Installing OpenCV for Android [1]

Downloading and Installing Android NDK

The Android NDK enables us to compile and run native C/C++ code on Android.

1. Download the latest version of the NDK from this website:
   `http://developer.android.com/tools/sdk/ndk/index.html`

2. Unzip the downloaded file to a location without spaces in the path, for example:
   `C:\\android-ndk-r9`

3. Define NDKROOT as a new environment variable pointing to the unzipped location from the last step. Please see the following page if you need help editing environment variables on Windows:
   `http://www.computerhope.com/issues/ch000549.htm`

Updating Tools in Android SDK

Enabling some additional tools in the Android SDK will enable easier development of native applications later on.

1. In Eclipse, click Help > Install New Software. Then, click Add.

2. In the Add Repository dialog, enter "ADT Plugin" for the Name and the following URL for the Location, and click OK:
   `https://dl-ssl.google.com/android/eclipse/`

3. In the Available Software dialog, select all listed components. Click Next. Accept all license agreements. Click Finish.

4. Restart Eclipse after all the tools have been downloaded and installed.

Downloading and Installing OpenCV SDK

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

1. Download version 2.4.4 (or higher) of the SDK from this website:
   `http://sourceforge.net/projects/opencvlibrary/files/opencv-android/`

2. Unzip the downloaded file to a location without spaces in the path, for example:
   `C:\\OpenCV4Android\\OpenCV-2.4.4-android-sdk`

3. In Eclipse, select File > Switch Workspace > Other. Specify a location associated with OpenCV projects, for example:
   `C:\\OpenCV4Android`

[1] Parts of this tutorial borrow explanations from the OpenCV website (http://opencv.org/platforms/android.html).

4. Right-click in the Package Explorer panel and choose Import > General > Existing Projects into Workspace. Select "Specify root directory" and input the location where the SDK contents were unzipped, for example:
   `C:\\OpenCV4Android\\OpenCV-2.4.4-android-sdk`

5. After seeing the OpenCV library and sample projects appear in the "Projects" box, click Finish. It may take a minute for all the projects to be loaded and initialized. Thereafter, you have the projects imported into the workspace, as shown in Figure 2.

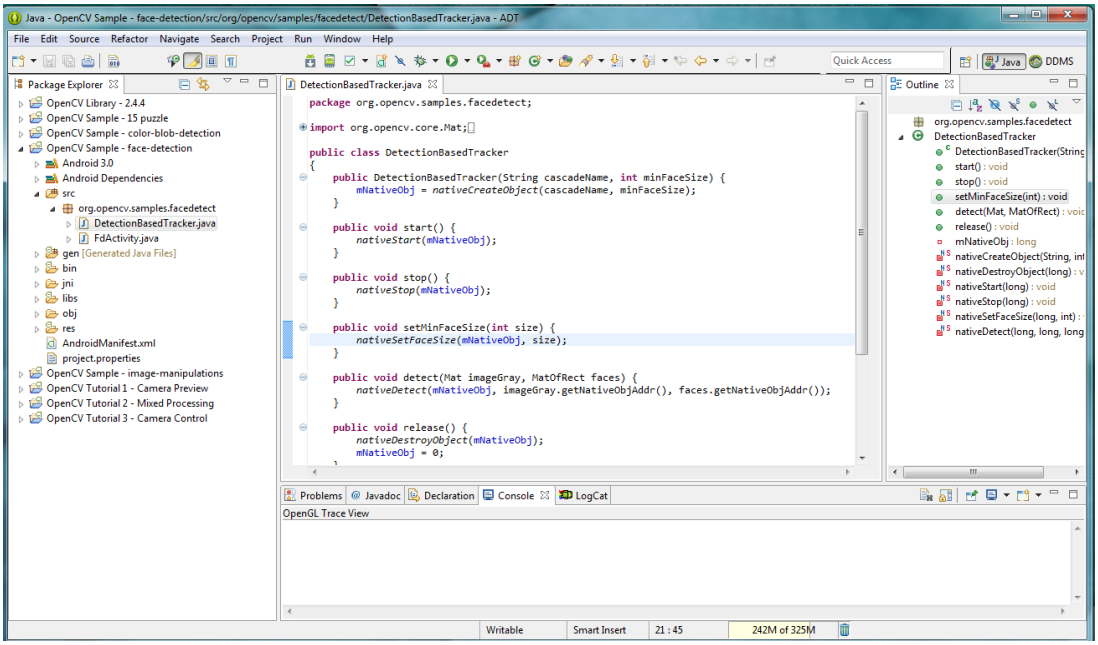

Figure 2. OpenCV library and sample projects loaded into Eclipse workspace.

Part II: Running OpenCV Sample Applications

Running OpenCV Samples

1. Install the OpenCV Manager app from the Google Play Market onto your device:
   `https://play.google.com/store/apps/details?id=org.opencv.engine`

2. Connect your device to your computer via USB.

3. In Eclipse, click on any of the sample OpenCV projects. Then, click Run > Run As > Android Application. You can monitor the installation status in the Console.

4. For example, if we run "OpenCV Sample – face-detection" project, we can see real-time face detections like in Figure 1 displayed on the mobile device.

5. Alternatively, we can add some code to the "OpenCV Tutorial 2 – Mixed Processing" project, we can see a locally adaptive binarization of some text document like in Figure 3 displayed on the mobile device.
   a. Open Tutorial2Activity.java, which is the main Java source file.

   b. At the top of the file, add:
```
private static fin tint VIEW_MODE_THRESH = 3;
private MenuItem mItemPreviewThresh;
```

   c. In the method "onCreateOptionsMenu", add:
```
mItemPreviewThresh = menu.add("Thresh.");
```

   d. In the method "onCameraFrame", add:
```
case VIEW_MODE_THRESH:
mRgba = inputFrame.rgba();
int maxValue = 255;
int blockSize = 61;
int meanOffset = 15;
Imgproc.adaptiveThreshold(
   inputFrame.gray(),
   mIntermediateMat,
   maxValue,
   Imgproc.ADAPTIVE_THRESH_MEAN_C,
   Imgproc.THRESH_BINARY_INV,
   blockSize,
   meanOffset
);
Imgproc.cvtColor(
   mIntermediateMat,
   mRgba,
   Imgproc.COLOR_GRAY2RGBA,
   4
);
```

   e. In the method "onOptionsItemSelected", add:
```
else if (item == mItemPreviewThresh) {
   mViewMode = VIEW_MODE_THRESH;
}
```

6. We can look further at the source code of the sample OpenCV projects. All of them have a similar structure: (i) code for initializing the camera, (ii) code for processing and augmenting the viewfinder frames, and (iii) code for regulating the Android activity transitions.

7. We can also examine the "OpenCV Library – 2.4.4" project, which contains the JNI interfaces to the OpenCV library functions. The function interfaces are essentially identical to those listed in the official documentation here: `http://docs.opencv.org/`

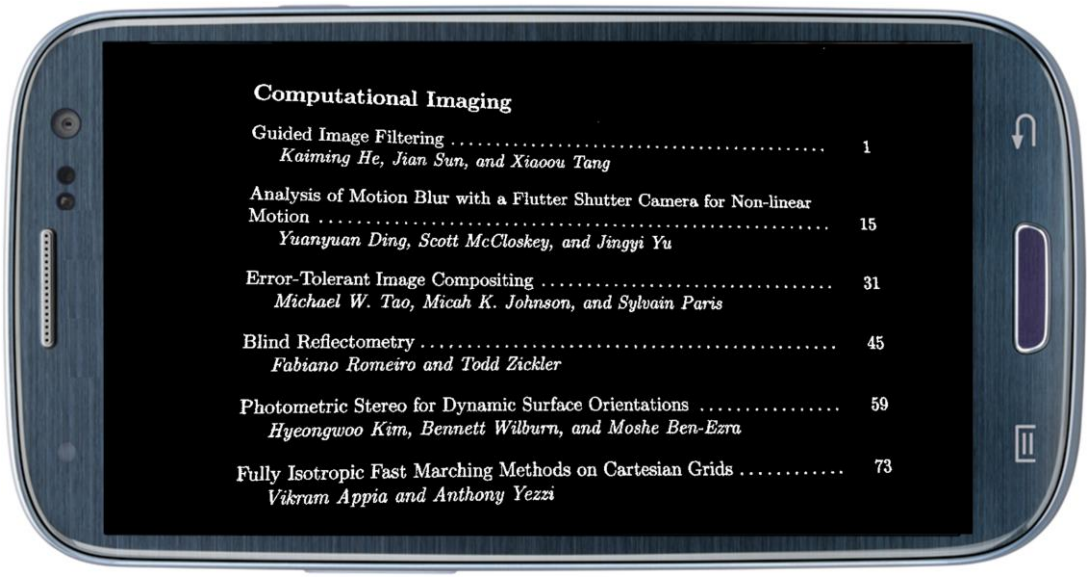

Figure 3. Android application showing adaptive binarization of a viewfinder frame.