Tutorial on OpenCV for Android Setup

EE368/CS232 Digital Image Processing, Winter 2018

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 Studio IDE. You will need to have already installed all the software tools mentioned in that other tutorial before moving onto this tutorial.

Part I: Installing OpenCV for Android

Downloading and Installing Android NDK

The Android NDK enables us to compile and run native C/C++ code on Android. Follow the instructions on the following page (section "Download the NDK and Tools") to download and install the NDK with Android Studio.
https://developer.android.com/ndk/guides/index.html

Downloading and Installing OpenCV SDK for Android

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

1. Download the version 2.4.13.4 of the SDK from this website: https://sourceforge.net/projects/opencvlibrary/files/opencv-android/

2. Unzip the downloaded file to a location without spaces in the path, for example:
   (macOS) `/Users/YourName/Android/OpenCV-android-sdk`
   (Linux) `/home/YourName/Android/OpenCV-android-sdk`
   (Windows) `C:\\Android\\OpenCV-android-sdk`
   In the rest of the tutorial, we will refer to this location as `$OPENCV_PATH/OpenCV-android-sdk`.

Part II: Running OpenCV Sample Applications

Running OpenCV Samples

OpenCV provides many sample projects that can be used as starting points for larger projects. Since these projects were generated with Eclipse, there is a bit of work to make them run under Android Studio. We will show you how to import the Face Detection project, which allow you to see real-time face detections like in Figure 1 displayed on the mobile device.

1. Click File > New > Import Project.
2. Choose the sample that you want in the OpenCV SDK. For example the Face Detection sample can be found at: `$OPENCV_PATH/OpenCV-android-sdk/samples/face-detection`. Click OK.
3. Keep the default settings and click Next, then Finish.
4. At this point the project will probably fail to build since it tries to build with an outdated version of the SDK platform and of Gradle. In order to change that follow the next steps. If the project builds properly, go directly to step 10.
5. Click the "Project" button in the top left corner of the Android Studio window to open the side panel. Make sure that the drop-down menu at the top of this panel is set to Android.
6. Open the current module (e.g. face-detection). Among the subfolders there should be one corresponding to the project (e.g. openCVSamplefacedetection) and another one corresponding to the OpenCV library (e.g. openCVLibrary24134). Open the build.gradle file located in each of these two subfolders.
7. Change the value of the number next to compileSdkVersion so that it refers to the SDK platform version that you installed. If you are not sure which version you have, open the SDK Manager by clicking the icon circled below in the toolbar (make sure the toolbar is shown by clicking View > Toolbar). Open the SDK Platforms tab and note down the number in the API Level column for one of the platforms installed.

8. Depending on the error message, in some cases it may be needed to change the value of the buildToolsVersion as well. You can also find it in the SDK Manager under the SDK

Tools tab. You will need to click the "Show Package Details" checkbox so that it shows the version number for "Android SDK Build - Tools".

9. Open the File > Project Structure window. Click on "Project" and enter your supported Gradle version (the error message should tell you what that version is, normally 4.1 if your Gradle version is up-to-date).
10. Sync the project by clicking the "Sync Project with Gradle Files" icon in the toolbar.

11. If the project does not have a native component (i.e. there is a "jni" folder in the original sample project), you can stop here and run it like a normal project. If it does, follow the next steps to link the native C++ library.
12. If it is not open yet, open the left side panel (cf. step 5).
13. Right click on the current project module (e.g. openCVSamplefacedetection) and choose Link C++ Project with Gradle.
14. Set the "Build System" as ndk-build and set the "Project Path" to the location of the Android.mk file in the current project. For the Face Detection sample, it should be located in `openCVSamplefacedetection/src/main/jni/Android.mk`. Click OK.
15. Open the Android.mk file under "External build files" in the left side panel.
16. Now that we are running this project from a different location than the original one, the path to the OpenCV.mk file needs to be updated. Find the line "`include ../../sdk/native/jni/OpenCV.mk`" and update the path. If you installed at the location we recommended, the path should be:
    `$OPENCV_PATH/OpenCV-android-sdk/sdk/native/jni/OpenCV.mk`.
17. Open the build.gradle script for the current module. It should now be found under "Gradle scripts" in the left panel. Make sure you are opening the correct script (for the Face Detection sample, the module name is openCVSamplefacedetection).
18. Locate the ndk block that should start around line 12 and add the line:
    `abiFilters "armeabi-v7a"`
19. Sync the project by clicking "Sync Project with Gradle Files" in the toolbar. The project should now build without problem. You can run it on your Android device. (Note if you are using an older version of the OpenCV SDK: on the first launch, you will most likely be prompted to install the OpenCV Manager app onto your device, which you need to do).

Let us know if you have trouble during this process. The face detection project can also be found on the EE368 git repository as an Android Studio project, compatible with the latest version of Android Studio. The only step necessary to make it run is to ensure that the path to the OpenCV.mk file is correct in the Android.mk file (step 15 above). If that path is incorrect, Android Studio will notify you and allow you to access this file by clicking "Open File". A sync is needed after modification of this file.

You are strongly encouraged to try other OpenCV samples.

Modifying an OpenCV Sample

In this part, we will modify an existing OpenCV sample. Import the `tutorial-2-mixedprocessing` sample using the same process as above. We will add some code to see a

locally adaptive binarization of some text document like in Figure 2 displayed on the mobile device.

1. Open Tutorial2Activity.java, which is the main Java source file.

2. At the top of the file, add:
```
private static final int VIEW_MODE_THRESH = 3;
private MenuItem mItemPreviewThresh;
```

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

4. 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
);
break;
```

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

The full project can be found on the git repository under `Tutorial2/AdaptiveBinarization`.

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.

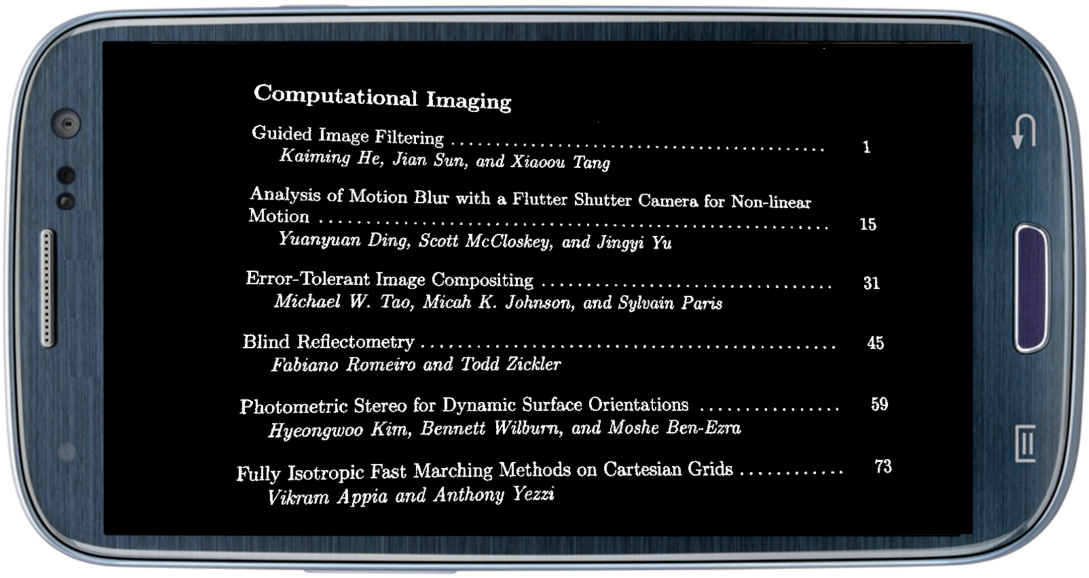

Figure 2. Android application showing adaptive binarization of a viewfinder frame.