Tutorial on Basic Android Setup
EE368/CS232 Digital Image Processing, Spring 2015

Linux Version for SCIEN Lab

Introduction
In this tutorial, we will learn how to set up the Android software development environment and how to implement image processing operations on an Android mobile device. Android is an open-source platform developed by Google and the Open Handset Alliance on which interesting and powerful new applications can be quickly developed and distributed to many mobile devices. There is a large, growing community of Android developers and a vast selection of Android devices, which includes smartphones, tablets, and TV setup boxes. Android also comes with an extension library of useful functions, including functions for user interfaces, image(bitmap) manipulation, and camera control that we will frequently use in EE368/CS232. We look forward to seeing your novel image processing algorithms and applications running on Android devices as the quarter progresses.

The tutorial is split into two parts. In the first part, we will explain how to download and install the Android software tools onto your computer. Then, in the second part, we will explain how to develop image processing programs that can run on an Android mobile device.

Estimated time to complete this tutorial: 30 minutes

Part I: Creating the Software Development Environment

Running Eclipse
We have already installed the Java JDK, Android SDK, and Eclipse on the SCIEN lab computers, so you can start developing immediately following a few simple steps.

1. Add some important folders to your PATH in your Linux shell’s .cshrc start-up file:
   ```
   setenv ANDROID /afs/ir/class/ee368/Android
   setenv SDK $ANDROID/android-sdk-linux
   setenv PATH $SDK/platform-tools:$SDK/tools:$ANDROID/eclipse:$PATH
   ```
   Be sure to “source .cshrc” for the changes to take immediate effect. You can then type “printenv” to confirm that the PATH is properly set.

2. Check Eclipse points to the right path:
   ```
   which eclipse → should print /afs/ir/class/ee368/Android/
   eclipse/eclipse
   ```

3. To start Eclipse, run:
   ```
   eclipse &
   ```
4. When asked to choose a default workspace, pick a folder that is easy to remember and access, for example:
/afs/ir/users/j/d/jdoe/workspace

5. If asked, choose an existing SDK location as /afs/ir/class/ee368/Android/android-sdk-linux. If not asked, go to Window > Preferences > Android and make sure the SDK Location is properly set to /afs/ir/class/ee368/Android/android-sdk-linux

Linking Your Phone to Your Computer
1. Connect your phone to the computer via USB. Turn on your phone.

2. Go to the home screen.

3. Select Settings > Applications > Development and then enable USB debugging.

Part II: Developing Image Processing Programs for Android
Now that the Google Android SDK, the Java Runtime, and the Eclipse IDE are all set up, we are ready to start writing image processing programs that can run an Android-compatible mobile device.

Hello World Example
First, we will build a simple Android program in Eclipse. This simple example will also help you to become familiar with how to create an Android project, how to (auto) compile source code, and how to run the generated executable on the mobile device. Please follow the instructions on this page to develop the “My First App” program. (The tutorial was recently replaced by the corresponding Android Studio tutorial, so we give you a link to an outdated snapshot of that page)
http://goo.gl/KSp30e

In the external “My First App” tutorial, they only run the “My First App” program in an emulator. Additionally, we will now also run the program on the actual Android phone. Make sure your phone is properly linked to your computer.

1. In Eclipse, select the project MyFirstApp in the Project Explorer. Then, select Project > Properties > Android and select the highest Android version if it is not already selected. Thanks to backwards compatibility and support libraries, it does not matter if your device does not support this particular version.

2. Known issue: If you chose as “minimum required SDK” an API < 14, the additional support library “appcompat_v7” will be created. A known bug in Eclipse may show an error on that library, preventing you to build your project. To solve the problem, select the “appcompat_v7” library in the Project Explorer and press F5 (or File > Refresh) to
refresh it. You can now clean it and rebuild it by choosing Project > Clean and then Project > Build project. The error should be gone.

3. Select the project MyFirstApp again, then Run > Run Configurations > Android Application > My First App > Target. Choose “Always prompt to pick device”.

4. Select Run, and in the Device Chooser dialog, select your Android phone. The “My First App” program will be sent to and automatically started on your phone, and you should see the screen similar to Figure 1 on your phone.

![My First App](image)

Figure 1. “My First App” program running on an Android phone.

EE368 Viewfinder Example
Now, having grasped the fundamentals of building and running an Android application, we will create a more complicated project involving the onboard camera and real-time image processing.

1. Create a new Android project with the following parameters.
   Application name: Viewfinder EE368
   Project name: ViewfinderEE368
   Package name: com.example.viewfinderee368
   Minimum Required SDK: API 8: Android 2.2 (Froyo)
   Target SDK: API 8: Android 2.2 (Froyo)
   Compile With: API 22: Android 5.1
   Theme: None
   Create Activity: Blank Activity

2. Copy the text in the following document into AndroidManifest.xml. This defines the main activities and permissions for this program.
   [http://ee368.stanford.edu/Android/ViewfinderEE368/AndroidManifest.xml.txt](http://ee368.stanford.edu/Android/ViewfinderEE368/AndroidManifest.xml.txt)
3. Copy the text in the following document into src : com.example.viewfinderee368 : ViewfinderEE368.java. This defines the classes in this program. 
http://ee368.stanford.edu/Android/ViewfinderEE368/MainActivity.java

4. Check to make sure everything is copied correctly into the project. If there are compilation errors, a red X will appear in the Package Explorer. If you have followed all the previous steps correctly and still get errors, it may be necessary to clean your workspace: Project > Clean > Clean all projects and to rebuild it.

5. Select Run and in the Device Chooser dialog, select your phone. You should see something like Figure 2 on your phone. Point the camera at different objects around you to see how the mean, standard deviation, and histogram of each color channel changes dynamically. You are augmenting the viewfinder in real time!

![Figure 2. “Viewfinder EE368” program running on an Android phone.](image)

6. You may notice that there are many deprecation warnings in the code. This is due to the fact that we compiled our project using Android 5.1, in which most Camera functions are deprecated. One workaround is to suppress the warnings manually. Another possibility is to build the project using another version of the SDK (you can get it on the SDK manager), but this may not work for older APIs – typically API ≤ 13 – since it may not be possible to build the compatibility library with the latest version of the SDK tools.

Real-time Phone Debugging in Eclipse
It is actually possible to view real-time messages from the phone in Eclipse, which can be very helpful for debugging and code development.
1. Select Window > Open Perspective > DDMS.

2. A new tab entitled “DDMS” should appear next to the default “Java” tab. Click on the “DDMS” tab.

3. Select Window > Show View > LogCat. The LogCat view shows a sequential list of real-time messages from the phone. In particular, error messages in red can be very useful when trying to debug a problem.

**Taking a Screenshot of the Phone**

At some point, it may be useful to take a screenshot of the phone, e.g., to use as a figure in your project report.

1. Go to the DDMS perspective in Eclipse.

2. Select your Android device, as shown in Figure 3.

3. Click the camera icon (circled in red in Figure 3), and a panel like Figure 4 should pop up.

4. Finally, when you have the desired screen shown, click Save or Copy to extract the screenshot.

![Figure 3. Dalvik Debug Monitor panel.](image-url)
Figure 4. Device screen capture panel.