Tutorial on Using Android for Image Processing Projects

EE368 Digital Image Processing, Spring 2012

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. 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  SDK  /usr/local/android-sdk-linux
   setenv  PATH  $SDK/platform-tools:$SDK/tools:/usr/local/bin:$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. To start Eclipse, run:
   ```
   eclipse &
   ```

3. 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
   ```

4. Go to Window > Preferences > Android and make sure the SDK Location is set to /usr/local/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

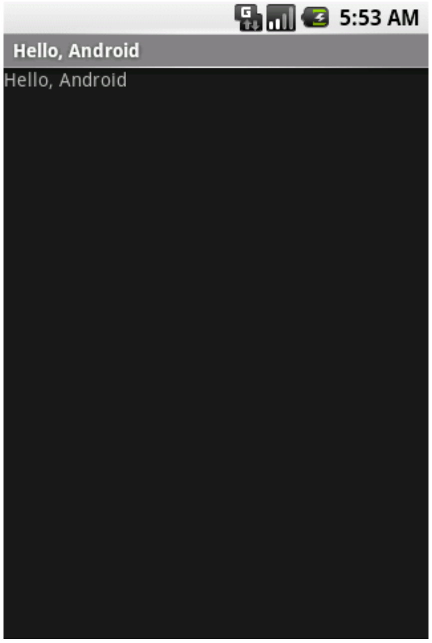
First, we will build a simple Android program in Eclipse. This simple example will also help you to become familiar with how to create an Android project, how to (auto) compile source code, and how to run the generated executable on the mobile device. Please follow the instructions on this page to develop the "Hello World" program:
`http://developer.android.com/resources/tutorials/hello-world.html`

Note: In case there are strange errors when the project is created about "R.java", simply add a comment line (e.g., "// dummy comment") at the top of "gen : com.example.helloandroid : R.java" and save the file. This will make the errors disappear.

In the external "Hello World" tutorial, they only run the "Hello World" 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 Run > Run Configurations > Android Application > HelloWorld > Target. Choose Manual for Deployment Target Selection Mode.

2. Select Run, and in the Device Chooser dialog, select your Android phone. The "Hello World" program will be sent to and automatically started on your phone, and you should see the screen similar to Figure 1 on your phone.



Hello, Android

Figure 1. “Hello World” 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.
   Project name: ViewfinderEE368
   Check the box for Android 2.1
   Application name: Viewfinder EE368
   Package name: com.example.viewfinderee368
   Check the box for Create Activity and enter: ViewfinderEE368
   Min SDK Version: 7

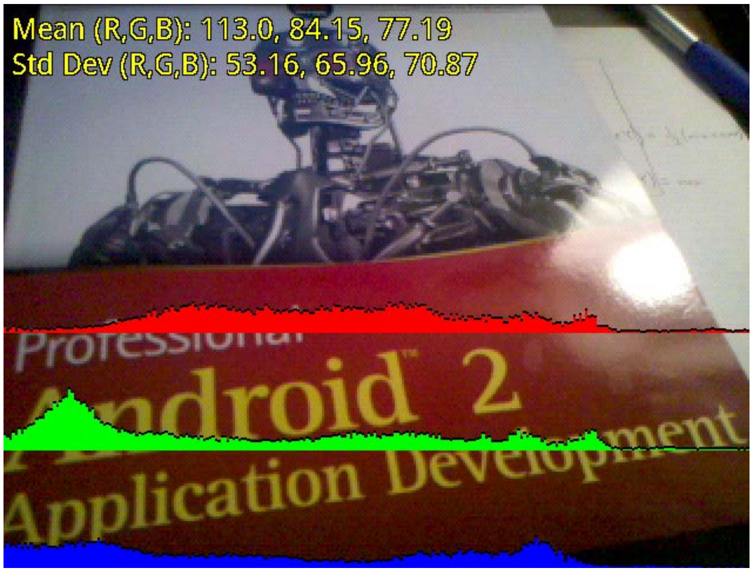
2. Copy the text in the following document into AndroidManifest.xml. This defines the main activities and permissions for this program.
   `http://ee368.stanford.edu/Android/ViewfinderEE368/AndroidManifest.xml`

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/ViewfinderEE368.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.

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!



Mean (R,G,B): 113.0, 84.15, 77.19
Std Dev (R,G,B): 53.16, 65.96, 70.87

Figure 2. "Viewfinder EE368" program running on an Android phone.
Demo video: http://ee368.stanford.edu/Android

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

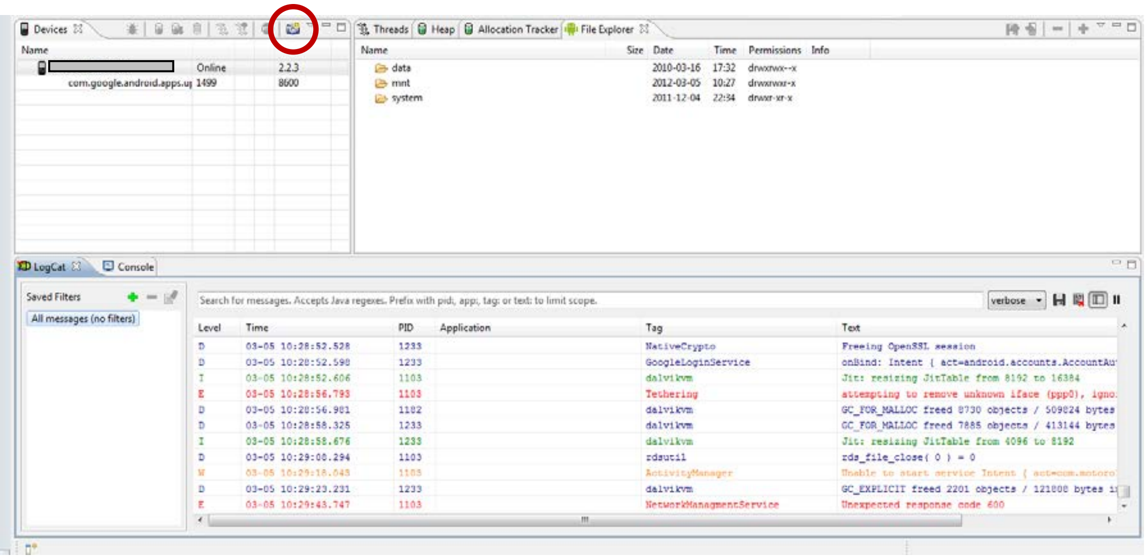
At some point, it may be useful to take a screenshot of the phone, e.g., to use as a figure in your project report.

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 screen shot.



Name		
[redacted]	Online	2.2.3
com.google.android.apps.u	1499	8600

Name	Size	Date	Time	Permissions	Info
data		2010-03-16	17:32	drwxrwx--x	
mnt		2012-03-05	10:27	drwxrwxr-x	
system		2011-12-04	22:34	drwxr-xr-x	

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

```
Level  Time                PID   Application  Tag                       Text
D      03-05 10:28:52.528  1233               NativeCrypto              Freeing OpenSSL session
D      03-05 10:28:52.598  1233               GoogleLoginService        onBind: Intent { act=android.accounts.AccountAu
I      03-05 10:28:52.606  1103               dalvikvm                  Jit: resizing JitTable from 8192 to 16384
E      03-05 10:28:56.793  1103               Tethering                 attempting to remove unknown iface (ppp0), igno
D      03-05 10:28:56.981  1182               dalvikvm                  GC_FOR_MALLOC freed 8730 objects / 509824 bytes
D      03-05 10:28:58.325  1233               dalvikvm                  GC_FOR_MALLOC freed 7885 objects / 413144 bytes
I      03-05 10:28:58.676  1233               dalvikvm                  Jit: resizing JitTable from 4096 to 8192
D      03-05 10:29:08.294  1103               rdsutil                   rds_file_close( 0 ) = 0
W      03-05 10:29:18.043  1103               ActivityManager           Unable to start service Intent { act=com.motoro
D      03-05 10:29:23.231  1233               dalvikvm                  GC_EXPLICIT freed 2201 objects / 121808 bytes i
E      03-05 10:29:43.747  1103               NetworkManagmentService   Unexpected response code 600
```

Figure 3. Dalvik Debug Monitor panel.

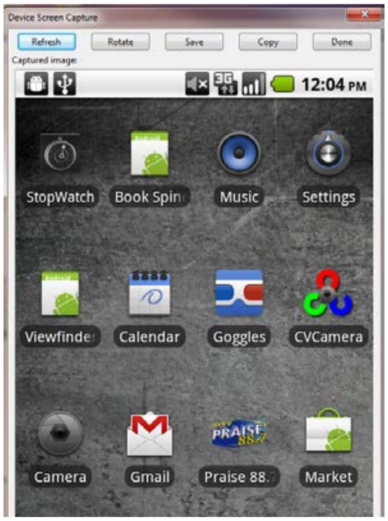


Figure 4. Device screen capture panel.