Tutorial on using Android for Image Processing Projects

EE368 Digital Image Processing, Spring 2010

Linux Version

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-based mobile device (e.g., the DROID phone that you have received for the class assignments). 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 device users. There is a growing community of Android developers and a growing market for Android-based devices (e.g., Motorola DROID, HTC DROID ERIS, Google Nexus One). Android also comes with a vast 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-based 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-based mobile device.

Part I: Creating the Software Development Environment

We will use the Google Android SDK, the Java Runtime, and the Eclipse IDE to design, implement, and debug Android-compatible programs in this class.[1] Make sure your Linux environment can display graphics (e.g., "`ssh -X`" if you are connecting to a server).

Downloading and Installing Java Runtime

1. Download Java JDK 6.0 from this location:
   `http://ee368.stanford.edu/Android/Installation-Files/jdk-6u19-linux-i586.bin`
2. Run the downloaded installer file. Installation instructions for your particular Linux system can be found here:
   `http://java.sun.com/javase/6/webnotes/install/index.html#linux`
3. Write down the path to where the JDK is installed to, for example:
   `/home/username/EE368-Android/jdk1.6.0_19`

Downloading and Installing Eclipse

1. Download "Eclipse IDE for Java Developers (Galileo)" from this location:

[1] Parts of this tutorial borrow explanations from the official Android developers' website (developer.android.com).

`http://ee368.stanford.edu/Android/Installation-Files/eclipse-java-galileo-SR2-linux-gtk.tar.gz`

2. Unzip (e.g., "`tar –xvf`") the downloaded file to a convenient location on your hard disk, for example:
   `/home/username/EE368-Android/eclipse`
3. Change to the directory where you unzipped Eclipse. This directory should contain an "eclipse" executable. Link Eclipse to the installed Java JDK using the following commands:
   ```
   chmod  a+x  eclipse
   ./eclipse  –vm  /home/username/EE368-Android/jdk1.6.0_19/jre/bin  &
   ```
   (Note: Replace "`/home/username/EE368-Android/jdk1.6.0_19`" with the location where you installed the Java JDK, and verify there is a "`jre/bin`" sub-directory present.)
4. When asked to choose a default workspace, pick a folder that is easy to remember and access, for example:
   `/home/username/EE368-Android/eclipse/workspace`
5. Verify that Eclipse starts properly and an IDE window like in Figure 1 is shown.
6. Each time Eclipse is started in the future, it should use the full command in Step 3. You can create a convenient alias in your Linux shell's start-up file (e.g., `.bashrc`, `.cshrc`, etc. in your home directory) by adding a line like the following:
   ```
   alias  myeclipse='/home/username/EE368-Android/eclipse/eclipse
         –vm  /home/username/EE368-Android/jdk1.6.0_19/jre/bin'
   ```

   Be sure to "`source .bashrc`", "`source .cshrc`", etc. for the settings change to take immediate effect.

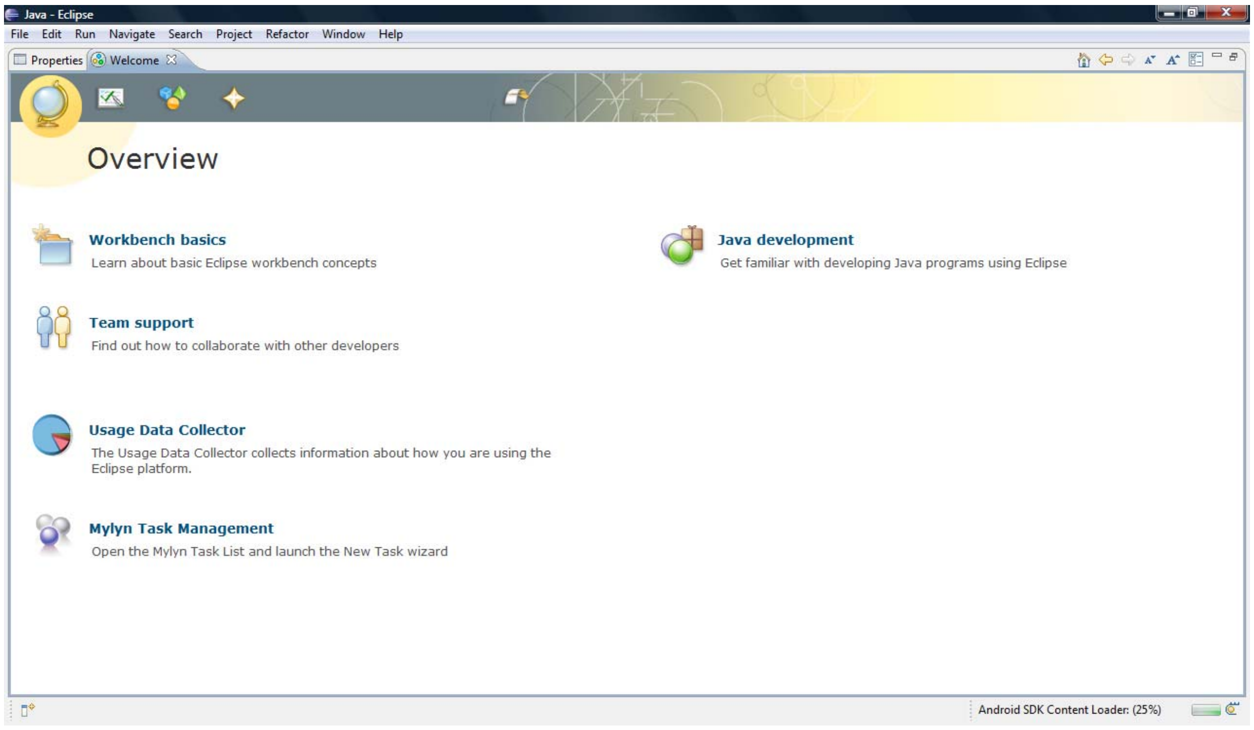

Java - Eclipse
File Edit Run Navigate Search Project Refactor Window Help
Properties
Welcome
Overview
Workbench basics
Learn about basic Eclipse workbench concepts
Java development
Get familiar with developing Java programs using Eclipse
Team support
Find out how to collaborate with other developers
Usage Data Collector
The Usage Data Collector collects information about how you are using the Eclipse platform.
Mylyn Task Management
Open the Mylyn Task List and launch the New Task wizard
Android SDK Content Loader: (25%)

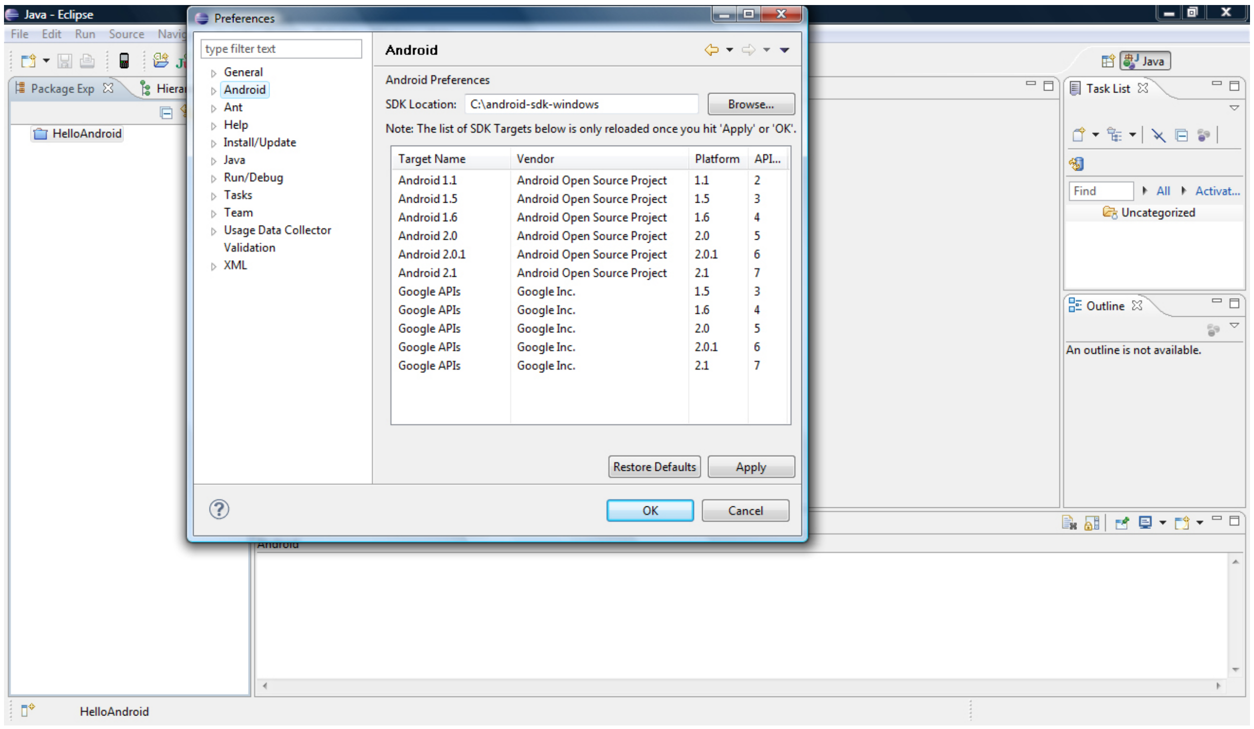
Figure 1. Start-up screen of the Eclipse IDE.

Downloading and Installing Android SDK

1. Download the Google Android SDK from this location:
   `http://ee368.stanford.edu/Android/Installation-Files/android-sdk_r05-linux_86.tgz`
2. Unzip (e.g., "`tar –xvf`") the file to a convenient location on your hard disk, for example: `/home/username/EE368-Android/android-sdk`
3. Add the location where you installed the SDK to your system PATH by adding a line like the following to your Linux shell's start-up file (e.g., `.bashrc`):
   ```
   export  PATH=/home/username/EE368-Android/android-sdk/tools:${PATH}   OR
   setenv  PATH  /home/username/EE368-Android/android-sdk/tools:$PATH
   ```

   Be sure to "`source .bashrc`", "`source .cshrc`", etc. for the settings change to take immediate effect. You can check the path has been modified using "`printenv`".

4. Install the ADT plugin for Eclipse.
   a. Open Eclipse.
   b. From top menubar, choose Help > Install New Software.
   c. In Available Software dialog, click Add …
   d. In Add Site dialog, enter "Android Plugin" in the "Name" field and enter the following URL in the "Address" field:
      `http://dl-ssl.google.com/android/eclipse/`
   e. Click OK.
   f. In Available Software dialog, you should see "Developer Tools" listed. Select the checkbox next to "Developer Tools" and click Next.
   g. In Install Details dialog, you should see the "Android DDMS" and "Android Development Tools" listed. Click Next, accept all license agreements, and click Finish.
   h. Restart Eclipse (be sure to use the "`myeclipse`" alias established earlier).
5. Link Eclipse to the installed Android SDK.
   a. In Eclipse, select Window > Preferences. A window like that in Figure 2 should pop up.
   b. Select Android from the left panel.
   c. For the "SDK Location", click Browse and find where you installed the Android SDK.
   d. Click Apply and then click OK.
   e. Restart Eclipse (be sure to use the "`myeclipse`" alias established earlier).
6. Download updates for the Android SDK.
   a. In Eclipse, select Window > Android SDK and AVD Manager. A window like that in Figure 3 should pop up.
   b. Select Available Packages from the left panel.
   c. Click the checkboxes for all available packages and click Install Selected.
   d. After installation, select Installed Packages from the left panel and verify that the packages are correctly installed.
7. If you encountered problems in this section, please take a look at the tips on these sites:
   `http://developer.android.com/sdk/index.html`
   `http://developer.android.com/sdk/installing.html`
   `http://developer.android.com/sdk/eclipse-adt.html`



Android

Android Preferences

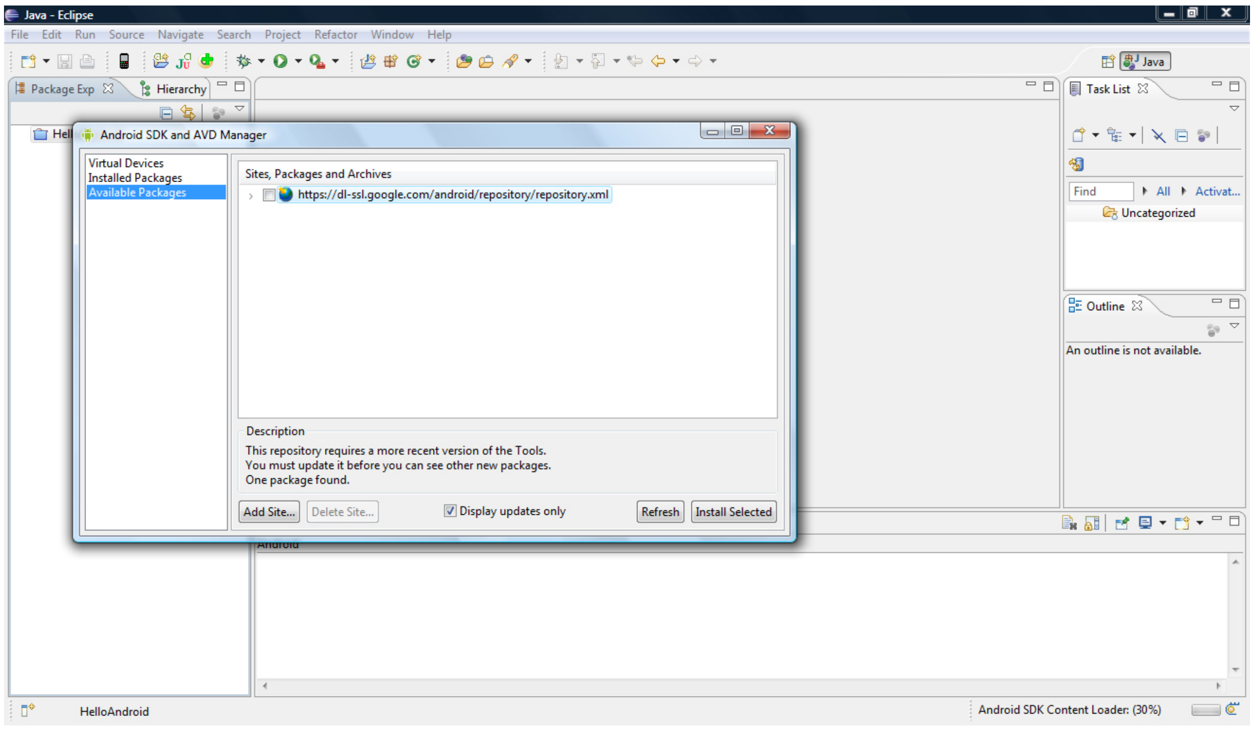
SDK Location: C:\android-sdk-windows

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	API...
Android 1.1	Android Open Source Project	1.1	2
Android 1.5	Android Open Source Project	1.5	3
Android 1.6	Android Open Source Project	1.6	4
Android 2.0	Android Open Source Project	2.0	5
Android 2.0.1	Android Open Source Project	2.0.1	6
Android 2.1	Android Open Source Project	2.1	7
Google APIs	Google Inc.	1.5	3
Google APIs	Google Inc.	1.6	4
Google APIs	Google Inc.	2.0	5
Google APIs	Google Inc.	2.0.1	6
Google APIs	Google Inc.	2.1	7

An outline is not available.

Figure 2. Android preferences panel in Eclipse.



Sites, Packages and Archives

https://dl-ssl.google.com/android/repository/repository.xml

Description

This repository requires a more recent version of the Tools.
You must update it before you can see other new packages.
One package found.

☑ Display updates only

An outline is not available.

Android SDK Content Loader: (30%)

Figure 3. Android update panel in Eclipse.

Linking Your Phone to Your Computer

1. Turn on your phone.
2. Go to the home screen.
3. Select Applications > Development and then enable USB debugging.
4. After you have downloaded updates for the Android SDK in Eclipse above, the USB driver should have been included. Install the USB driver on your computer, following the tips on the following page:
   `http://developer.android.com/guide/developing/device.html#setting-up`

   (Note: On SCIEN computers, the device file in `/etc/...` is already configured.)

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 on your computer, 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-based 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-based phone. The “Hello World” program will be sent to and automatically started on your phone, and you should see the screen similar to Figure 4 on your phone.

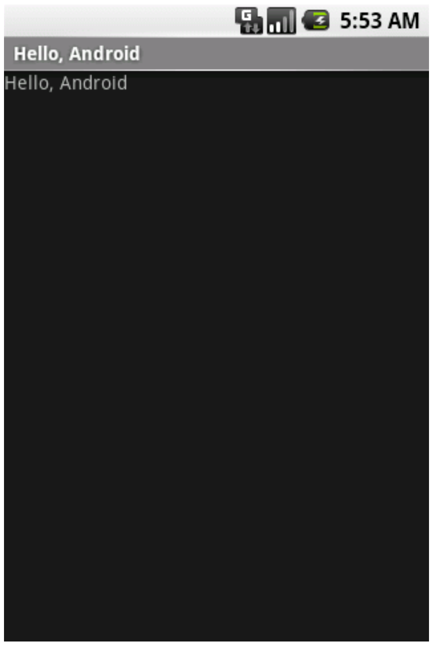


Figure 4. “Hello World” program running on Android-based phone.

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 2.0.1
   Application name: Viewfinder EE368
   Package name: com.example.viewfinderee368
   Check the box for Create Activity and enter: ViewfinderEE368
   Min SDK Version: 6

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 5 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!

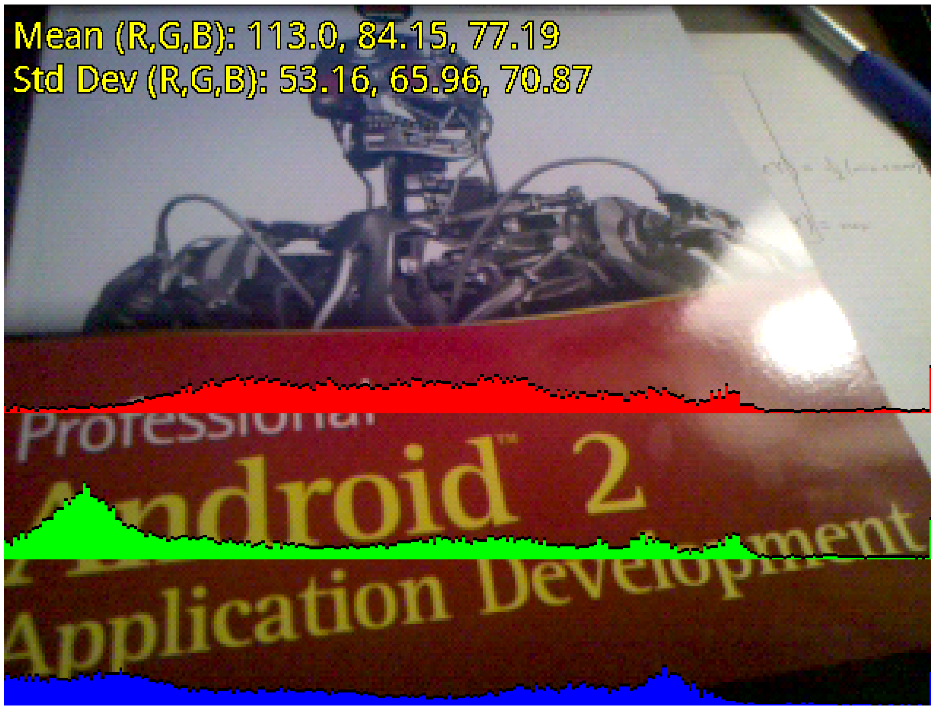

Mean (R,G,B): 113.0, 84.15, 77.19
Std Dev (R,G,B): 53.16, 65.96, 70.87

Figure 5. “Viewfinder EE368” program running on Android-based phone.

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.