Android OpenCV Configuration Instructions

Credits:

This work is NOT mine originally, I have compiled this document from the following sources:

Stack Overflow post by Kiran, jlhonora, ssimm, and Wodin, see here:
http://stackoverflow.com/questions/27406303/opencv-in-android-studio

The OpenCV Organization, see the following:
http://opencv.org/platforms/android/opencv4android-samples.html
https://github.com/opencv/opencv/tree/master/samples/android

Instructions:

Open Android Studio, close a project if you have one open currently, then choose “Start a new Android Studio project”. Name the project “OpenCVTest” or similar, choose company domain, for example “yournamehere.com”, choose you preferred Project location, choose Next.

Check Phone and Tablet, set Minimum SDK to API 19: Android 4.4 (KitKat), and uncheck the other boxes (for Wear, TV, etc.), choose Next.

Choose Empty Activity, choose Next.

You can leave the “Activity Name” as “MainActivity” and the “Layout Name” as “activity_main”, and leave “Generate Layout File” checked, then choose Finish.

Once Android Studio builds the project, open "build.gradle (Module:app)" and change "targetSdkVersion" to 22, this will force install time permissions rather than using the API 23 or later runtime permissions, which require substantial additional code that we would like to avoid at the moment for simplicity. Android Studio will highlight the 22 and show the warning "Not targeting the latest version of Android; compatibility modes apply. Consider testing and updating this version." This warning is OK, leave targetSdkVersion set to 22.

Next go to activity_main.xml and change the default TextView text from “Hello World!” to anything else, for example, “My Hello World!” or “Hello World 123”, then run the app on your device to verify it will build and everything is working so far.

Download the latest version of OpenCV for Android, ex. “OpenCV-3.1.0-android-sdk.zip”. The first directory in the .zip will contain only one directory, “OpenCV-android-sdk”. Copy this to the root of your file system, ex. “C:\ OpenCV-android-sdk”, then put the version number in-between OpenCV and android, ex. “OpenCV-3.1.0-android-sdk” so you can use different versions at a later time without the different versions overwriting each other.

Back in Android Studio, choose File -> New -> Import Module . . . Set Source directory to your OpenCV for android sdk, then sdk, then java, ex. “C:\OpenCV-3.1.0-android-sdk\sdk\java”. Android Studio should automatically fill in the “Module name:”, ex. openCVLibrary310. Choose Next.

On the next screen you will get 3 check boxes about replacing .jars with dependencies, replacing library sources with dependencies, and Gradle-style module names, leave these all checked and choose Finish.

At this point you will get Gradle errors, if you see an option “Install missing platform(s) and sync project” do NOT choose this, we will fix the Gradle build errors manually in the next steps. If you choose the “Install missing platform(s) and sync project” option, Gradle will do some things to the project that will make it not work later, so do NOT choose this option. If you accidentally chose this option, exit Android Studio, delete the project, and start again from the beginning of these instructions.

Go to File -> Project Structure, chose the “app” module on the left, choose the Dependencies tab, then the plus sign towards the top right, then “Module Dependency”, the applicable OpenCV dependency should be listed, ex “openCVLibrary310”, choose OK, then OK.

You will still get build version errors, we will fix these in the next steps.

In the Android Studio file browser on the left, open Gradle Scripts, then “build.gradle (Module: app)”. Copy paste the compileSdkVersion, buildToolsVersion, minSdkVersion, and targetSdkVersion into Notepad (or anything else you prefer to copy/paste text into), you should have something that looks like the following copy/pasted:

compileSdkVersion 24
buildToolsVersion "24.0.1"
minSdkVersion 19
targetSdkVersion 22

Next we will need to make changes to some files that are not visible by default in Android Studio. To make all files visible in Android Studio, at the top left change the file browser view from “Android” to “Project” to show all files.

Go to OpenCVTest/openCVLibrary310/build.gradle, you will find values for compileSdkVersion, buildToolsVersion, minSdkVersion, and targetSdkVersion, change these to match the values you copied/pasted from your build.gradle file a few steps previously, in our example that would be the 24, “24.0.1”, 19, and 24.

You should see an option appear at the top, “Sync Now”, choose this option to make Gradle re-synch with the updated version numbers. To be on the safe side, at this point, choose Build -> Clean Project, then Build -> Rebuild Project, the project should build without errors.

Next in a file browser, go to “C:\OpenCV-3.1.0-android-sdk\sdk\native\” and copy the directory “libs” to “(your project location)\OpenCVTest\openCVLibrary310\src\main”, then rename it from “libs” to “jniLibs”. Next return to the file browser within Android Studio and check to see that Android Studio has detected this change, you may need to do another Build -> Clean Project, Build -> Rebuild Project to make Android Studio pick up on the directory you have added.

In Android Studio, you should now change the file browser view at the top left from “Project” back to “Android”. Keep in mind you may need to change it back to “Project” later on if you need to see all the project files for troubleshooting.

Go to MainActivity, change “extends AppCompatActivity” to “extends Activity”. You will need to use the Alt + Enter keyboard shortcut to add the import for Activity, and you can remove the unused import for AppCompatActivity.

Next in the onCreate method, after the lines the Android Studio project build wizard added, add the following additionally:

```
if (!OpenCVLoader.initDebug()) {
    Log.e(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), not working.");
} else {
    Log.d(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), working.");
}
```

Use the Alt + Enter keyboard shortcut option to include the libraries for OpenCVLoader and Log, then do a Build -> Clean Project, then wait for the project to start on your device (verify you see “Hello World 123” or whatever you changed the text to at the beginning on your device), then back in Android Studio in the Android Monitor Logcat text search box on the right, enter “OpenCVLoader.initDebug()” and you should see “OpenCVLoader.initDebug(), working”. Alternatively, enter your version of OpenCV, ex. “3.1.0”, and you should see something to the effect of “General configuration for OpenCV 3.1.0”. If you choose the “X” in the Logcat search box and look through Logcat you should be able to find the OpenCV load information, which will start with the “General configuration for OpenCV 3.1.0” text.

Now that we’ve got OpenCV installed we will implement one of the OpenCV for Android samples. Google search “GitHub Android OpenCV samples”, the GitHub OpenCV Android samples should appear as the first search result, or you can go directly to:
https://github.com/opencv/opencv/tree/master/samples/android

Choose the “tutorial-1-camerapreview” example, next we will copy/paste in the applicable code to get the example working within Android Studio.

In Android Studio, go to AndroidManifest.xml, comment out everything with <!-- --> to save the original wizard created version, then move it down and copy/paste in the entire AndroidManifest.xml from the OpenCV samples GitHub site. Next make the following changes:
-change the line ***package="org.opencv.samples.tutorial1”*** to your package name, for example ***package="com.chrisdahms.opencvtest"***
-remove the ***android:versionCode=*** and ***android:versionName=*** lines since Gradle overwrites those
-change the line: ***android:icon="@drawable/icon”*** to: ***android:icon="@mipmap/ic_launcher"***

-change the line android:name="Tutorial1Activity" to the name full extension name to your main activity, for example ***android:name="com.chrisdahms.opencvtest.MainActivity"***

-remove the line ***<uses-sdk android:minSdkVersion="8" />*** since Gradle overwrites this anyway

-cut the ***supports-screens***, ***uses-permission***, and ***uses-features*** sections and paste them back in after the beginning of the ***manifest*** section, but before the ***application*** section

-you may still get a warning about the application section not being Google search indexable, don't worry about this warning, it is not really necessary to fix it

After the above changes, for example, my manifest looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.chrisdahms.opencvtest">

    <supports-screens android:resizeable="true"
        android:smallScreens="true"
        android:normalScreens="true"
        android:largeScreens="true"
        android:anyDensity="true" />

    <uses-permission android:name="android.permission.CAMERA"/>

    <uses-feature android:name="android.hardware.camera" android:required="false" />
    <uses-feature android:name="android.hardware.camera.autofocus" android:required="false" />
    <uses-feature android:name="android.hardware.camera.front" android:required="false" />
    <uses-feature android:name="android.hardware.camera.front.autofocus" android:required="false" />

    <application
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:theme="@android:style/Theme.NoTitleBar.Fullscreen">

        <activity android:name=".MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="landscape"
            android:configChanges="keyboardHidden|orientation"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

    </application>

</manifest>
```

Next in Android Studio, go to activity_main.xml, comment out everything with <!-- --> to save what you have so far, then move it down and copy/paste in the entire tutorial1_surface_view.xml from the OpenCV samples GitHub site, additional changes to this file should not be necessary. For example, my activity_main.xml looks like this:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:opencv="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <org.opencv.android.JavaCameraView
        android:id="@+id/javaCameraView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:visibility="gone"
        opencv:show_fps="true"
        opencv:camera_id="any" />

</FrameLayout>
```

Next in Android Studio, go to MainActivity, comment out everything with /* */ to save what you have so far, then move it down and copy/paste in the entire Tutorial1Activity.java from the OpenCV samples GitHub site, then make the following changes to clear the errors:
-change package name as applicable, for example change ***org.opencv.samples.tutorial1;*** to ***com.chrisdahms.opencvtest;***
-change the class name from ***Tutorial1Activity*** to ***MainActivity***
-change the constructor name from ***Tutorial1Activity*** to ***MainActivity***
-change the line ***setContentView(R.layout.tutorial1_surface_view);*** to ***setContentView(R.layout.activity_main);***
-to make the resolution lower so the program runs faster, after the line
mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.tutorial1_activity_java_surface_view);
add the following:

```
// set resolution to 640x480, may seem to slow if we used full resolution
try {
   mOpenCvCameraView.setMaxFrameSize(640, 480);
} catch (Exception exception) {
   Log.e(TAG, "unable to change resolution");
}
```

-for the method ***onCameraFrame***, remove the line return ***inputFrame.rgba();*** and add the following lines:

```
Mat imgOriginal = inputFrame.rgba();
Mat imgCanny = new Mat();
Imgproc.Canny(imgOriginal, imgCanny, 70, 100);
return(imgCanny);
```

then use the Alt + Enter keyboard shortcut to add the library for the Imgproc.Canny line

After the above changes, for example, my ActivityMain looks like this:

```
package com.chrisdahms.opencvtest;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceView;
import android.view.WindowManager;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Mat;
import org.opencv.imgproc.Imgproc;

////////////////////////////////////////////////////////////////////////////////////////////////
public class MainActivity extends Activity implements CameraBridgeViewBase.CvCameraViewListener2 {

    // member variables ////////////////////////////////////////////////////////////////////////
    private static final String TAG = "MainActivity";

    private CameraBridgeViewBase mCameraBridgeViewBase;

    // member variable instantiated via an anonymous inner class //////////////////////////////
    private BaseLoaderCallback mBaseLoaderCallback = new BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            if(status == LoaderCallbackInterface.SUCCESS) {
                Log.i(TAG, "OpenCV loaded successfully");
                mCameraBridgeViewBase.enableView();
            } else {
                super.onManagerConnected(status);
            }
        }
    };

    // constructor /////////////////////////////////////////////////////////////////////////////
    public MainActivity() {
        Log.i(TAG, "Instantiated new " + this.getClass());
    }

    ////////////////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "called onCreate");
        super.onCreate(savedInstanceState);
```

```
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    setContentView(R.layout.activity_main);
    mCameraBridgeViewBase = (CameraBridgeViewBase)findViewById(R.id.javaCameraView);

    // set resolution to 640x480, may seem to slow if we used full resolution
    try {
        mCameraBridgeViewBase.setMaxFrameSize(640, 480);
    } catch (Exception exception) {
        Log.e(TAG, "unable to change resolution");
    }
    mCameraBridgeViewBase.setVisibility(SurfaceView.VISIBLE);
    mCameraBridgeViewBase.setCvCameraViewListener(this);
}

////////////////////////////////////////////////////////////////////////////////////////////////
@Override
public void onPause() {
    super.onPause();
    if (mCameraBridgeViewBase != null)
        mCameraBridgeViewBase.disableView();
}

////////////////////////////////////////////////////////////////////////////////////////////////
@Override
public void onResume() {
    super.onResume();
    if (!OpenCVLoader.initDebug()) {
        Log.d(TAG, "Internal OpenCV library not found. Using OpenCV Manager for initialization");
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this, mBaseLoaderCallback);
    } else {
        Log.d(TAG, "OpenCV library found inside package. Using it!");
        mBaseLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
    }
}

////////////////////////////////////////////////////////////////////////////////////////////////
public void onDestroy() {
    super.onDestroy();
    if (mCameraBridgeViewBase != null)
        mCameraBridgeViewBase.disableView();
}

////////////////////////////////////////////////////////////////////////////////////////////////
public void onCameraViewStarted(int width, int height) {
}

////////////////////////////////////////////////////////////////////////////////////////////////
public void onCameraViewStopped() {
}

////////////////////////////////////////////////////////////////////////////////////////////////
```

```
    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
        Mat imgOriginal = inputFrame.rgba();
        Mat imgCanny = new Mat();
        Imgproc.Canny(imgOriginal, imgCanny, 70, 100);
        return(imgCanny);
    }

}
```

At this point go ahead and run the app, it will show the Canny edge image of whatever the rear camera is point at, and it should run pretty quick at 640x480.