


A Beginner’s Guide to Setting up OpenCV Android Library on Android Studio

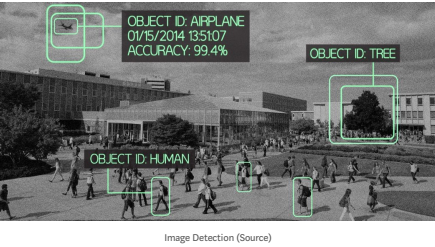


Elvis Chidera

Follow

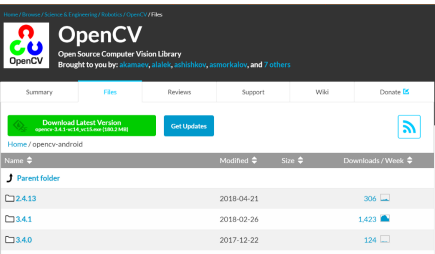
Aug 13, 2018 · 6 min read

I recently started a project that involved working with OpenCV on Android. Most of the guides on setting up the library on Android were outdated or not complete. So, after getting multiple requests from team mates on how to set this up, I decided to just write a dead simple guide on this.

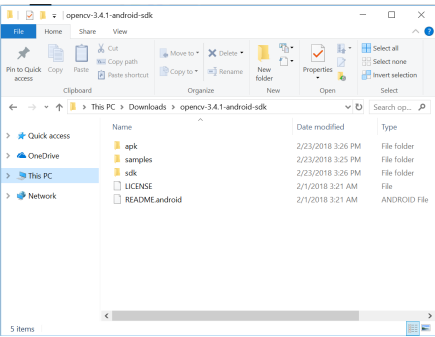


Step 1: Download OpenCV Android Library

Go to the OpenCV Android Sourceforge page and download the latest OpenCV Android library. As at the time of writing this post, the latest available version was **3.4.1**.



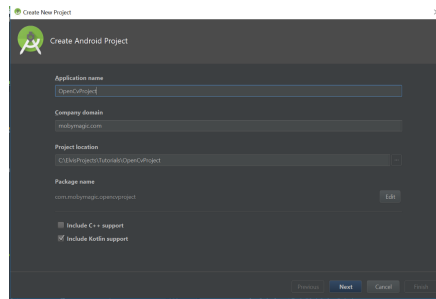
When the download completes, you should extract the contents of the zip file into a folder.



Step 2: Setup project

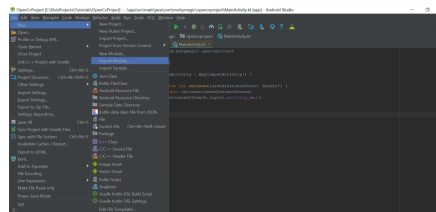
Create a new Android project using Android Studio only if you have not created one already for your computer vision project.

Note: Skip this step if you already have an Android project you want to use the OpenCV library in.

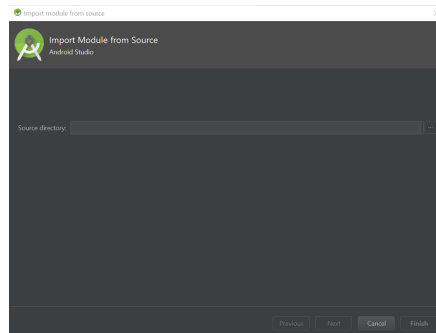


Step 3: Import OpenCV Module

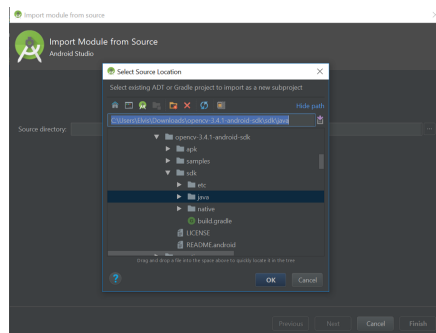
After successfully creating an Android project, it is time to import the OpenCV module into your Android project. Click on **File -> New -> Import Module...**



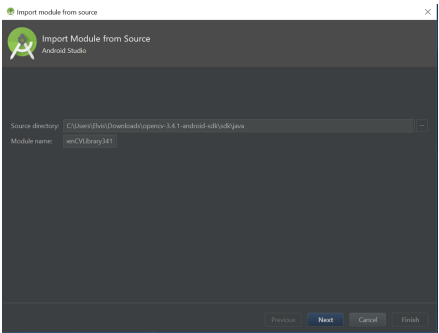
It should bring up a popup like the image below where you can select the path to the module you want to import.



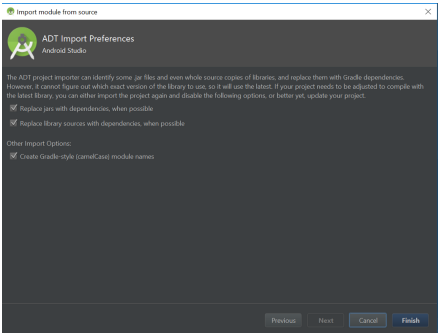
Browse to the folder where you extracted the OpenCV Android library zip file contents. Select the `java` folder inside of the `sdk` folder .



After selecting the correct path and clicking **OK**, you should get a screen like the image below.

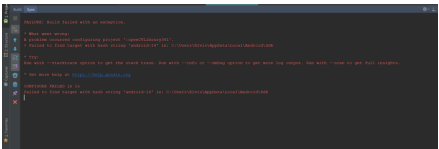


Click on **Next** to go to the next screen. On the next screen (the image below) you should leave the default options checked and click on **Finish** to complete the module import.

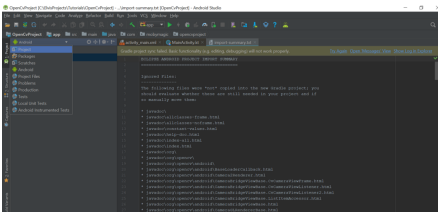


Step 4: Fixing Gradle Sync Errors

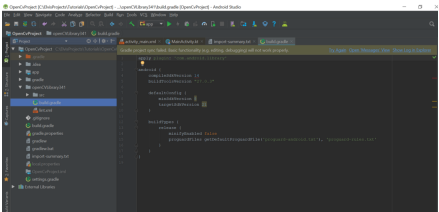
You should get a Gradle build error after you finish importing the OpenCV library. This happens because the library is using an old Android SDK that you probably don't have installed yet.



To quickly fix this error, switch from the Android pane to the Project pane on the left side of Android Studio.

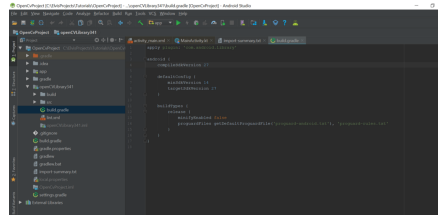


Browse to OpenCV library module and open its `build.gradle` file.



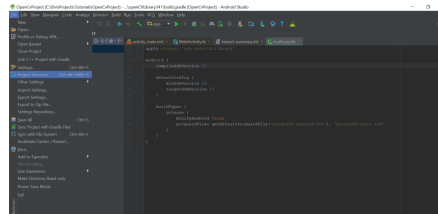
To fix the error, you just have to change the `compileSdkVersion` and `targetSdkVersion` to the latest Android SDK version or the one you have installed on your PC. After changing the version you should click on the **sync** button so that Gradle can sync the project.

Quick tip: `buildToolsVersion` can be ignored



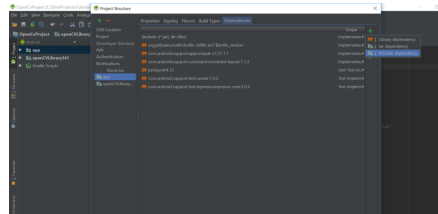
Step 5: Add the OpenCV Dependency

To work with the OpenCV Android library, you have to add it to your `app` module as a dependency. To easily do this on Android Studio, click on **File -> Project Structure**.

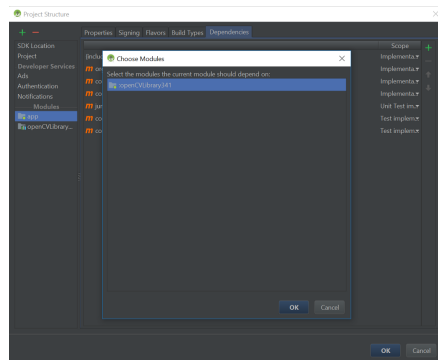


When the project structure dialog opens, click on the `app` module or any other module that you want to use OpenCV library in.

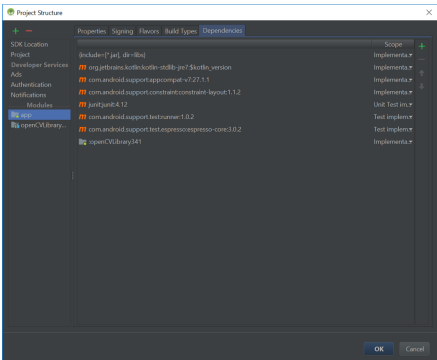
After navigating to the module, click on the **Dependencies** tab. You should see a green plus button on the far right of the dialog, click on it and select **Module dependency**.



When the choose modules dialog opens, select the OpenCV library module and click on **OK**.

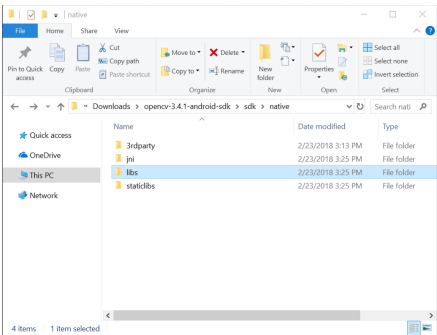


When you return to the dependencies page, confirm that the module was actually added as a dependency then click on the **OK** button to continue.

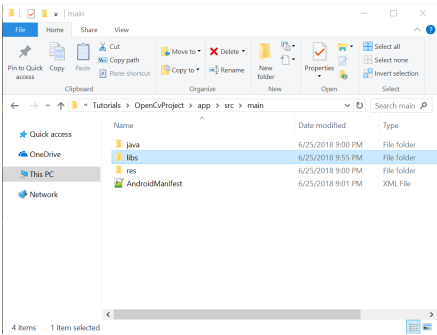


Step 6: Add Native Libraries

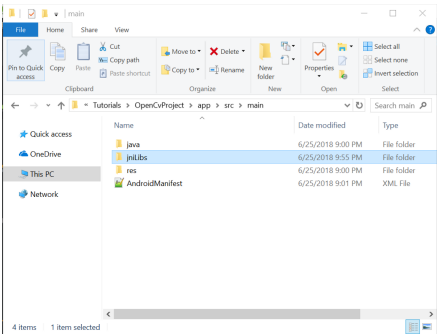
On your file explorer, navigate to the folder where you extracted the content of the OpenCV Android library zip file. Open the `sdk` folder and then the `native` folder (Use the image below as a guide).



Copy the `libs` folder in the `native` folder over to your project `app` module `main` folder (Usually `ProjectName/app/src/main`).



Rename the `libs` folder you just copied into your project to `jniLibs` .



Step 7: Add Required Permissions

To successfully use OpenCV, your app should have the camera permission added to its AndroidManifest.xml file.

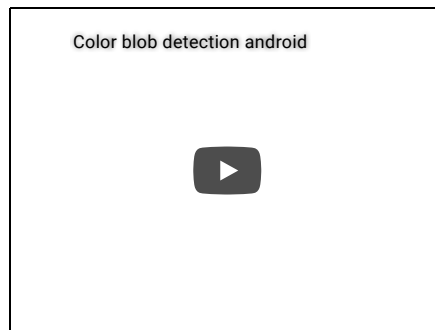
Tip: Don't forget to request for the camera permission at runtime on Android 6 and above.

```
1 <uses-permission android:name="android.permission.CAMERA" />
2
3 <uses-feature android:name="android.hardware.camera" />
4 <uses-feature android:name="android.hardware.camera2" />
5 <uses-feature android:name="android.hardware.camera" />
```

Step 8: Try out Sample

To confirm that you successfully integrated the OpenCV Android library, you can try out one of the samples included in the library zip file.

Let's try out the `color-blob-detection` sample. You can see the sample in action below:



Quickly update your app main activity with the code below.

```

1  package com.mobymagic.opencvproject;
2
3  import java.util.List;
4
5  import org.opencv.android.BaseLoaderCallback;
6  import org.opencv.android.CameraBridgeViewBase.CvCame
7  import org.opencv.android.LoaderCallbackInterface;
8  import org.opencv.android.OpenCVLoader;
9  import org.opencv.core.Core;
10 import org.opencv.core.CvType;
11 import org.opencv.core.Mat;
12 import org.opencv.core.MatOfPoint;
13 import org.opencv.core.Rect;
14 import org.opencv.core.Scalar;
15 import org.opencv.core.Size;
16 import org.opencv.android.CameraBridgeViewBase;
17 import org.opencv.android.CameraBridgeViewBase.CvCame
18 import org.opencv.imgproc.Imgproc;
19
20 import android.app.Activity;
21 import android.os.Bundle;
22 import android.util.Log;
23 import android.view.MotionEvent;
24 import android.view.View;
25 import android.view.Window;
26 import android.view.WindowManager;
27 import android.view.View.OnTouchListener;
28 import android.view.SurfaceView;
29
30 public class MainActivity extends Activity implements
31     private static final String TAG = "
32
33     private boolean mIsColorSelected = f
34     private Mat mRgba;
35     private Scalar mBlobColorRgba;
36     private Scalar mBlobColorHsv;
37     private ColorBlobDetector mDetector;
38     private Mat mSpectrum;
39     private Size SPECTRUM_SIZE;
40     private Scalar CONTOUR_COLOR;
41
42     private CameraBridgeViewBase mOpenCvCameraView;
43
44     private BaseLoaderCallback mLoaderCallback = new
45     @Override
46     public void onManagerConnected(int status) {
47         switch (status) {
48             case LoaderCallbackInterface.SUCCESS:
49             {
50                 Log.i(TAG, "OpenCV loaded success
51                 mOpenCvCameraView.enableView();
52                 mOpenCvCameraView.setOnTouchListe
53             } break;
54             default:
55             {
56                 super.onManagerConnected(status);
57             } break;
58         }
59     }
60 };
61
62 public MainActivity() {
63     Log.i(TAG, "Instantiated new " + this.getClas
64 }
65
66 /** Called when the activity is first created. */
67 @Override
68 public void onCreate(Bundle savedInstanceState) {
69     Log.i(TAG, "called onCreate");
70     super.onCreate(savedInstanceState);
71     requestWindowFeature(Window.FEATURE_NO_TITLE)
72     getWindow().addFlags(WindowManager.LayoutParams
73
74     setContentView(R.layout.color_blob_detection_
75
76     mOpenCvCameraView = (CameraBridgeViewBase) fi
77     mOpenCvCameraView.setVisibility(SurfaceView.V
78     mOpenCvCameraView.setCvCameraViewListener(thi
79 }
80
81 @Override
82 public void onPause()
83 {

```

Then create a new class called `ColorBlobDetector` and copy the code below into it.

```

1  package com.mobymagic.opencvproject;
2
3  import java.util.ArrayList;
4  import java.util.Iterator;
5  import java.util.List;
6
7  import org.opencv.core.Core;
8  import org.opencv.core.CvType;
9  import org.opencv.core.Mat;
10 import org.opencv.core.MatOfPoint;
11 import org.opencv.core.Scalar;
12 import org.opencv.imgproc.Imgproc;
13
14 public class ColorBlobDetector {
15     // Lower and Upper bounds for range checking in HSV color space
16     private Scalar mLowerBound = new Scalar(0);
17     private Scalar mUpperBound = new Scalar(0);
18     // Minimum contour area in percent for contours filtering
19     private static double mMinContourArea = 0.1;
20     // Color radius for range checking in HSV color space
21     private Scalar mColorRadius = new Scalar(25,50,50);
22     private Mat mSpectrum = new Mat();
23     private List<MatOfPoint> mContours = new ArrayList<>();
24
25     // Cache
26     Mat mPyrDownMat = new Mat();
27     Mat mHsvMat = new Mat();
28     Mat mMask = new Mat();
29     Mat mDilatedMask = new Mat();
30     Mat mHierarchy = new Mat();
31
32     public void setColorRadius(Scalar radius) {
33         mColorRadius = radius;
34     }
35
36     public void setHsvColor(Scalar hsvColor) {
37         double minH = (hsvColor.val[0] >= mColorRadius.val[0]) ?
38             hsvColor.val[0] - mColorRadius.val[0] :
39             hsvColor.val[0] + mColorRadius.val[0];
40         double maxH = (hsvColor.val[0] <= mColorRadius.val[0]) ?
41             hsvColor.val[0] + mColorRadius.val[0] :
42             hsvColor.val[0] - mColorRadius.val[0];
43
44         mLowerBound.val[0] = minH;
45         mUpperBound.val[0] = maxH;
46
47         mLowerBound.val[1] = hsvColor.val[1] - mColorRadius.val[1];
48         mUpperBound.val[1] = hsvColor.val[1] + mColorRadius.val[1];
49
50         mLowerBound.val[2] = hsvColor.val[2] - mColorRadius.val[2];
51         mUpperBound.val[2] = hsvColor.val[2] + mColorRadius.val[2];
52
53         mLowerBound.val[3] = 0;
54         mUpperBound.val[3] = 255;
55
56         Mat spectrumHsv = new Mat(1, (int)(maxH-minH), CvType.CV_8UC3);
57
58         for (int j = 0; j < maxH-minH; j++) {
59             byte[] tmp = {(byte)(minH+j), (byte)255, (byte)255};
60             spectrumHsv.put(0, j, tmp);
61         }
62
63         Imgproc.cvtColor(spectrumHsv, mSpectrum, Imgproc.COLOR_HSV2RGB);
64     }
65
66     public Mat getSpectrum() {
67         return mSpectrum;
68     }
69
70     public void setMinContourArea(double area) {
71         mMinContourArea = area;
72     }
73 }

```

Finally, update your app main activity layout file with the layout code below.


```

1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
3      xmlns:android="http://schemas.android.com/apk/res/
4      android:layout_width="match_parent"
5      android:layout_height="match_parent" >
6
7      <org.opencv.android.JavaCameraView
8          android:layout_width="match_parent"
9          android:layout_height="match_parent"

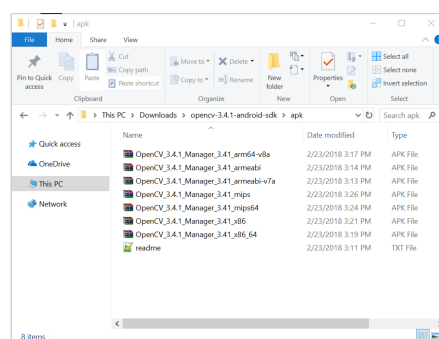
```

Step 9: Using OpenCV Manager

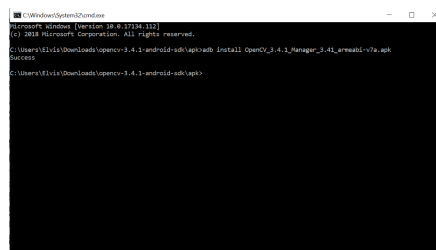
Because you are bundling native libraries in your app APK, you will end up with a very large app APK size.

One way to solve this is to use ABI splits, so you only have the necessary libraries needed for each device in an APK.

The other way around the size issue is to use the OpenCV Manager. In the folder where you extracted the library contents into, there is a folder called `apk` that contains the OpenCV manager for various architectures.



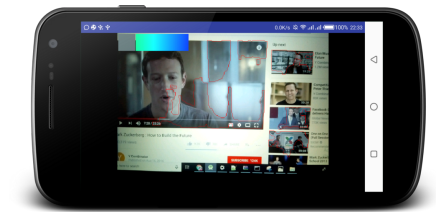
Select the APK with the right architecture of your test device and use ADB via command prompt/line to install it on your device.



Step 10: Test the App

Finally, hit the run button and run the app on your test device.

Note: Don't forget to grant camera permission on Android 6 and above.



That is all.

If you enjoyed this story, please click the 🌟 button and share to help others find it!

