Department of Electronic Engineering

City University of Hong Kong

EE5415 Mobile Applications Design and Development

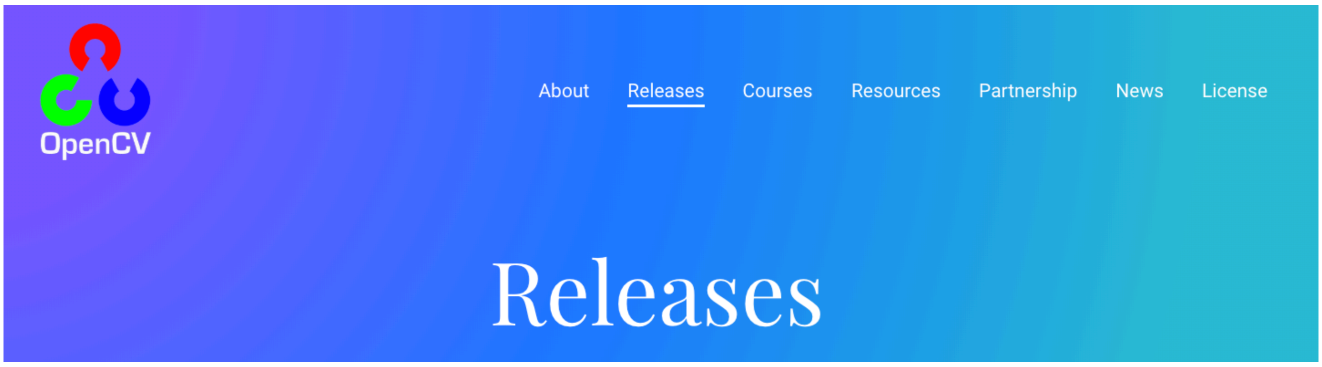
Lab 05: OpenCV Android Library

I. Introduction

OpenCV is a popular open source computer vision and machine learning library and it is aimed at implementing Image Processing and Computer Vision functionalities in real-time applications. Although it is required to import the OpenCV Android Library into Android Studio Project, it is quick and easy to get started with OpenCV on Android. This lab exercise shows students how to run OpenCV on a standard Android Studio AVD. You do not need any special hardware or software, just Android Studio. You do not even need an Android phone as the AVD will work fine.

II. Setup and Integration: OpenCV for Android

Step 1: Download the OpenCV SDK for Android (https://opencv.org/releases/).



It is recommended to download **OpenCV version 3.4.9** for Android, which is evaluated on Android Studio version 3.5.3.

- https://sourceforge.net/projects/opencvlibrary/files/3.4.9/opencv-3.4.9-android-sdk.zip/download

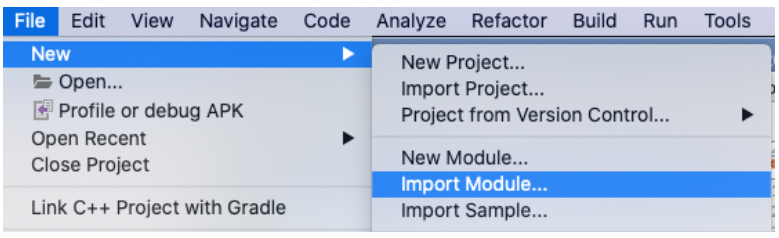
Download the zip file and decompress it in your computer. Make note of the destination folder: **OpenCV-android-sdk**. For example,

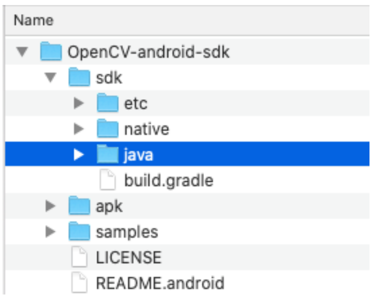
- `/Users/username/Library/Android/OpenCV-anddroid-sdk` for MacOS machine
- `C:\OpenCV-anddroid-sdk` for Windows Machine

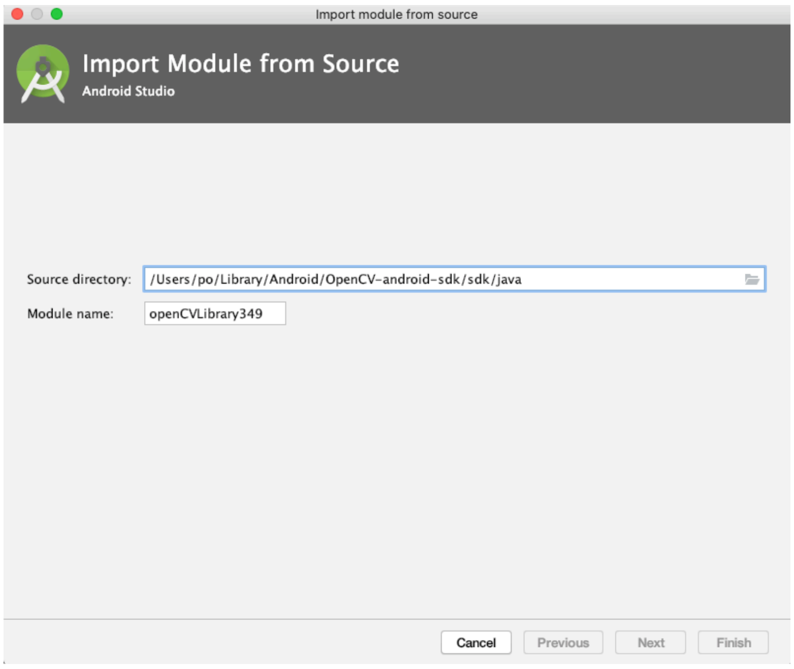
Step 2: In Android Studio, create a new project named "OpenCV349" with following project setting:

- Choose your project : **Empty Activity**
- Application Name : **OpenCV349**
- Package name : **com.xample.opencv349**
- Project location : use the default setting
- Language : **Java**
- Minimum API level: **API 19: Android 4.4 (KitKat)**
- Click **Finish**
- This configuration will create an Android Activity with following files
    - Activity Name : **MainActivity**
    - Layout Name : **activity_main**
    - Title : **MainActivity**

Step 3: Import the OpenCV Module. From the top menu, go to **File => New => Import Module**. Select the Source Directory path to the ***OpenCV-android-sdk/*****sdk/java** folder under your decompression OpenCV-android-sdk folder. Here are some screenshots that will give you more clarity.





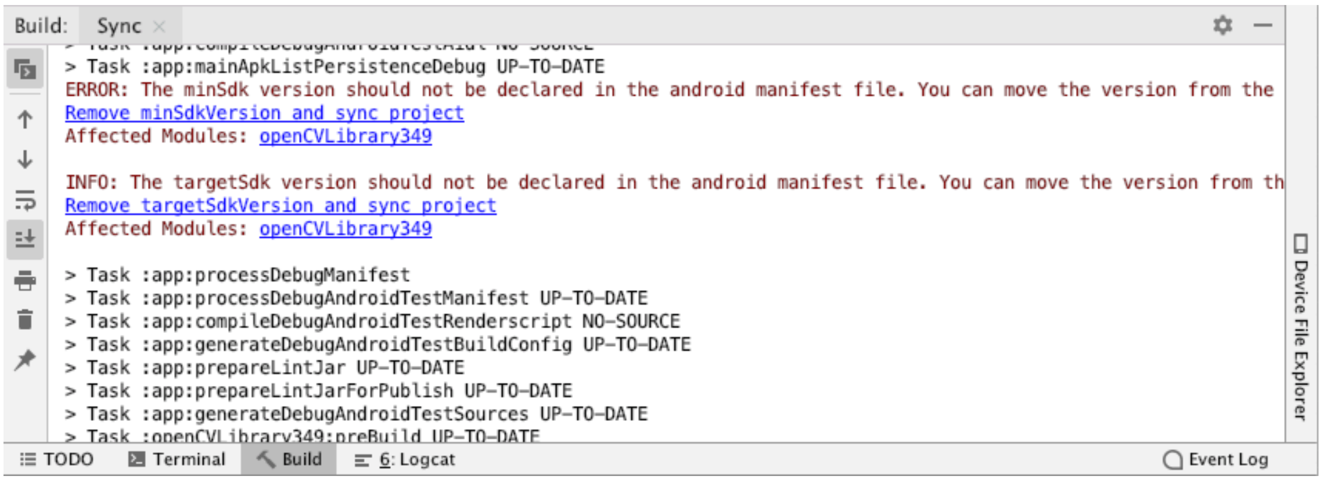


Source directory: /Users/po/Library/Android/OpenCV-android-sdk/sdk/java

Module name: openCVLibrary349

Click Next and Finish to complete the import of the OpenCV Library.

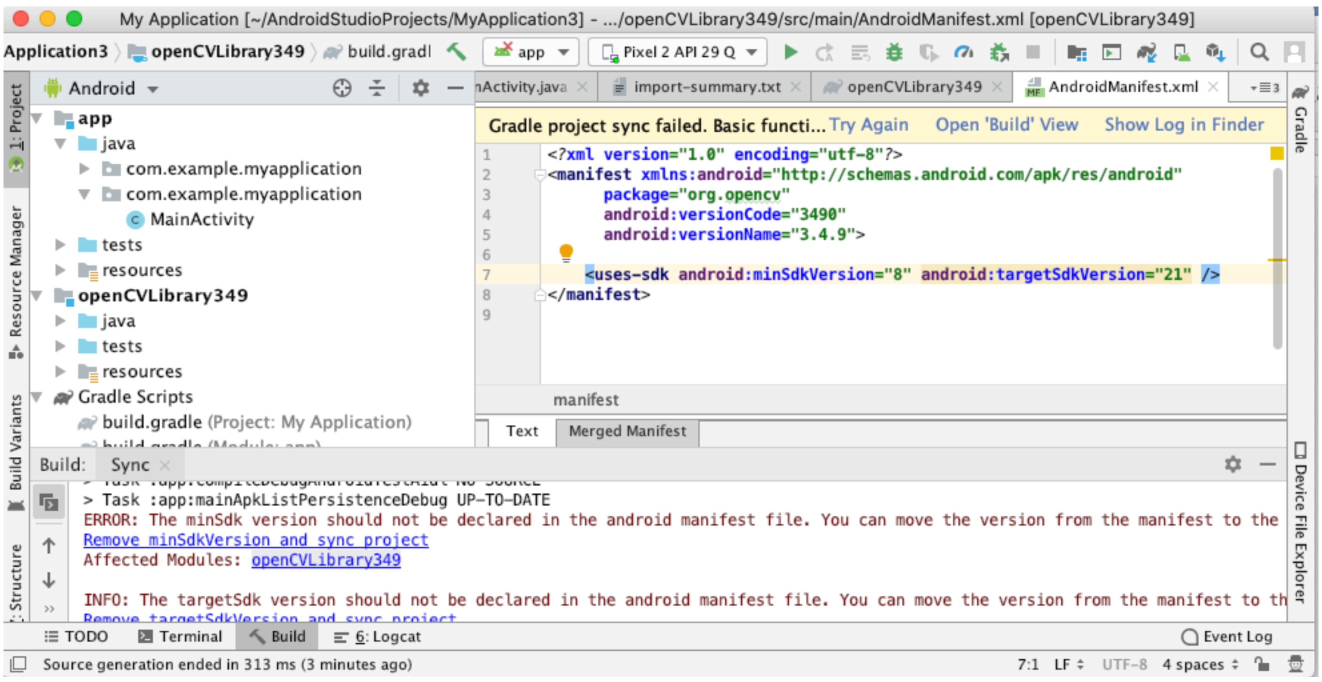
Step 4: **Handle Gradle project sync error**. Gradle script of the project app and of openCV does not match. Thus, there will be a sync error as shown below:



```
> Task :app:mainApkListPersistenceDebug UP-TO-DATE
ERROR: The minSdk version should not be declared in the android manifest file. You can move the version from the
Remove minSdkVersion and sync project
Affected Modules: openCVLibrary349

INFO: The targetSdk version should not be declared in the android manifest file. You can move the version from th
Remove targetSdkVersion and sync project
Affected Modules: openCVLibrary349

> Task :app:processDebugManifest
> Task :app:processDebugAndroidTestManifest UP-TO-DATE
> Task :app:compileDebugAndroidTestRenderscript NO-SOURCE
> Task :app:generateDebugAndroidTestBuildConfig UP-TO-DATE
> Task :app:prepareLintJar UP-TO-DATE
> Task :app:prepareLintJarForPublish UP-TO-DATE
> Task :app:generateDebugAndroidTestSources UP-TO-DATE
> Task :openCVLibrary349:preBuild UP-TO-DATE
```

To fix the first error of "**The targetSdk version should not be declared in the android manifest file".** Click on the link of "openCVLibrary349 as shown above or select ***Android*** from the drop-down at the top of the ***Project*** window and then **Remove minSdkVersion and targetSdkVersion. if it's in AndroidManifest of openCVLibrary.**

```
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="21" />
```



My Application [~/AndroidStudioProjects/MyApplication3] - .../openCVLibrary349/src/main/AndroidManifest.xml [openCVLibrary349]

Gradle project sync failed. Basic functi... Try Again Open 'Build' View Show Log in Finder

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.opencv"
    android:versionCode="3490"
    android:versionName="3.4.9">

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="21" />
</manifest>
```

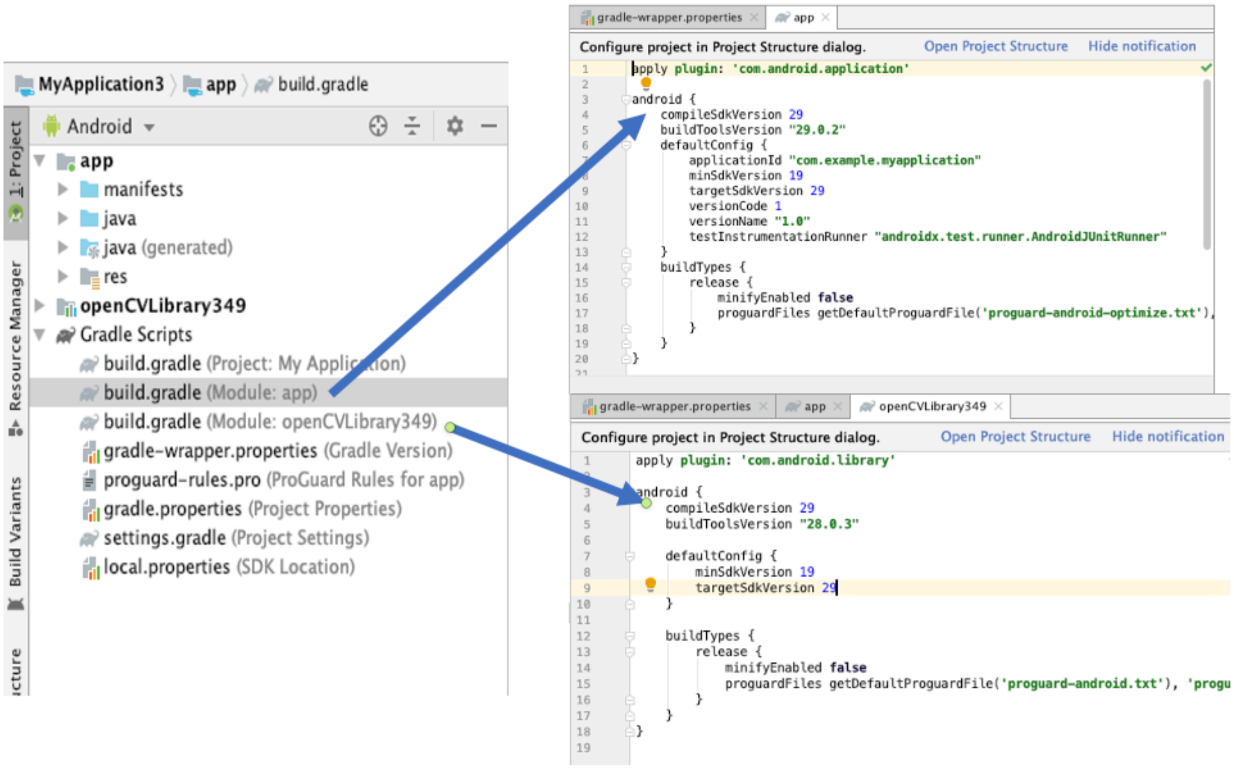
```
> Task :app:mainApkListPersistenceDebug UP-TO-DATE
ERROR: The minSdk version should not be declared in the android manifest file. You can move the version from the manifest to the
Remove minSdkVersion and sync project
Affected Modules: openCVLibrary349

INFO: The targetSdk version should not be declared in the android manifest file. You can move the version from the manifest to th
```

Source generation ended in 313 ms (3 minutes ago)

The second error is a little bit confusing. There are several .gradle files, which one should you be choosing? Select ***Project*** from the drop-down at the top of the ***Project*** window. **You need to match the following between the *build.gradle* under the app folder and the *build.gradle* of the openCVLibrary folder:**

· ***compileSdkVersion***
· ***buildToolsVersion***
· ***minSdkVersion***
· ***targetSdkVersion***

Alter the build.gradle of the OpenCV Library and modify it to the values of these parameters in the build.gradle of the app. **Do NOT tamper with the *build.gradle* of the app.**
Then press sync.

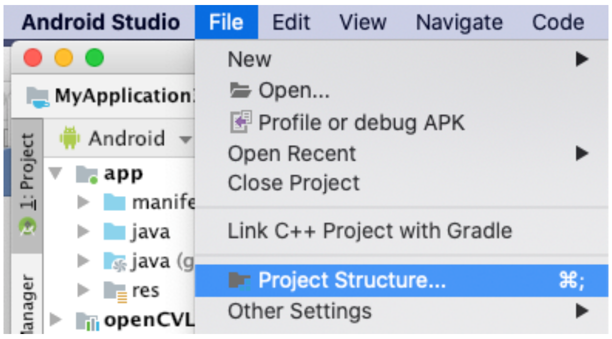

MyApplication3 app build.gradle
Android
app
manifests
java
java (generated)
res
openCVLibrary349
Gradle Scripts
build.gradle (Project: My Application)
build.gradle (Module: app)
build.gradle (Module: openCVLibrary349)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
1: Project
Resource Manager
Build Variants
gradle-wrapper.properties
app
Configure project in Project Structure dialog.
Open Project Structure
Hide notification

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.example.myapplication"
        minSdkVersion 19
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
        }
    }
}
```

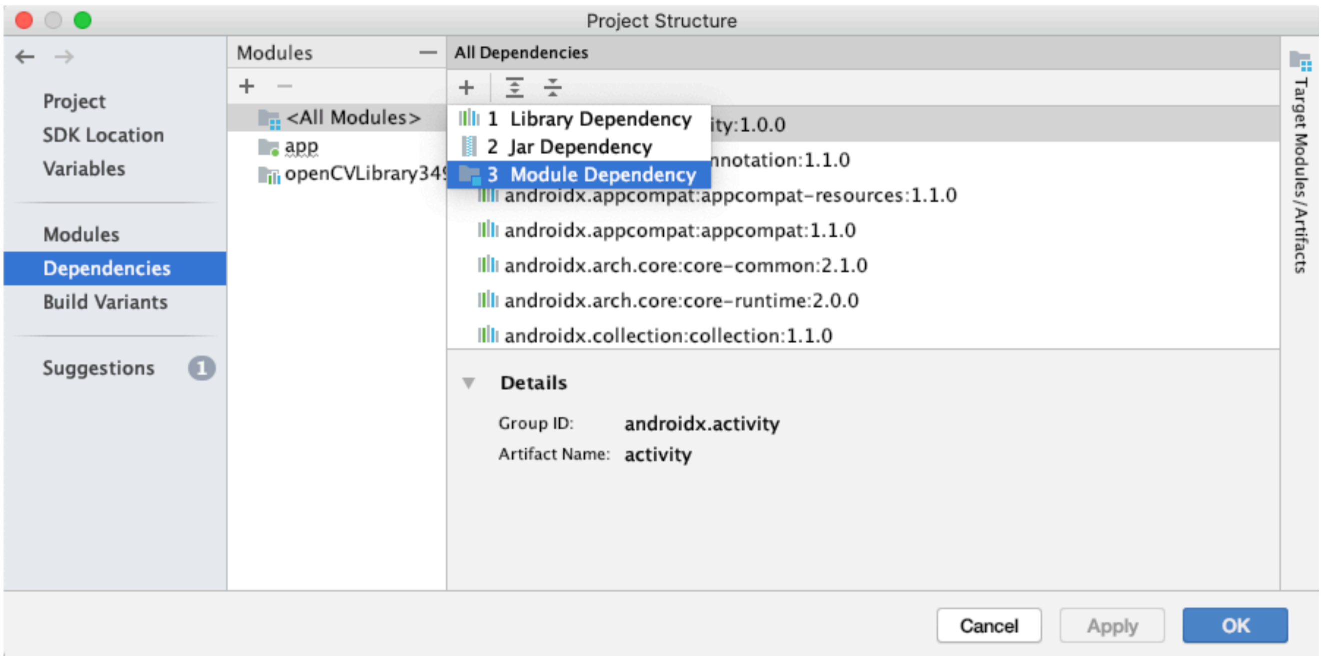
gradle-wrapper.properties
app
openCVLibrary349
Configure project in Project Structure dialog.
Open Project Structure
Hide notification

```
apply plugin: 'com.android.library'

android {
    compileSdkVersion 29
    buildToolsVersion "28.0.3"

    defaultConfig {
        minSdkVersion 19
        targetSdkVersion 29
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'progu
        }
    }
}
```

Step 5: Adding the OpenCV Module Dependencies. R Go to ***File -> Project Structure…*** as shown below:



Select "Dependencies" =><All Modules> => Press "+" sign and select the option call "3 Module Dependency". You will see the ***OpenCVLibrary module***, click on OK after choosing it.

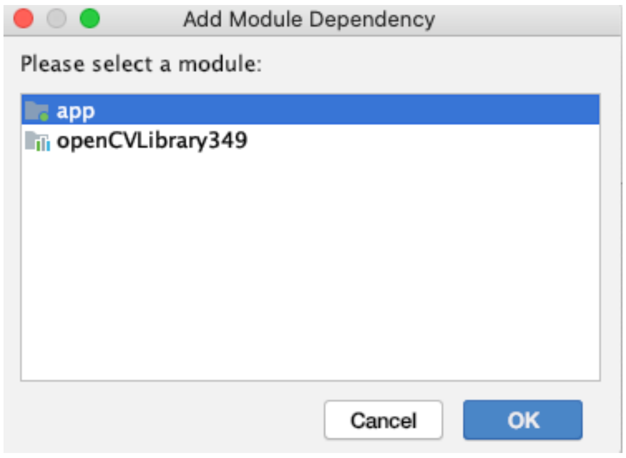


<All Modules>
app
openCVLibrary349

ity:1.0.0
nnotation:1.1.0
androidx.appcompat:appcompat-resources:1.1.0
androidx.appcompat:appcompat:1.1.0
androidx.arch.core:core-common:2.1.0
androidx.arch.core:core-runtime:2.0.0
androidx.collection:collection:1.1.0

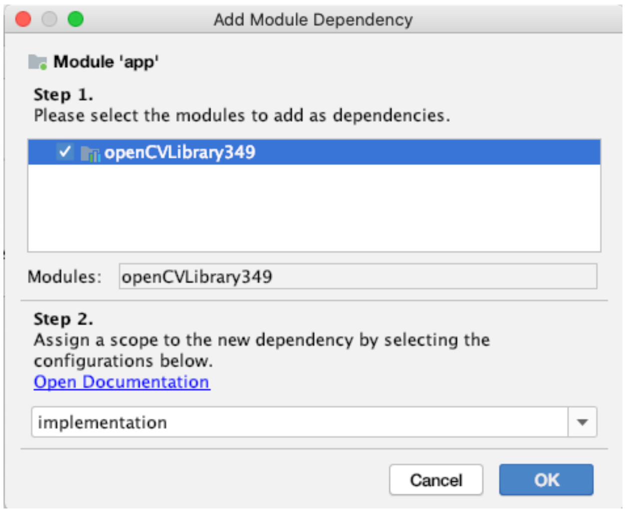
Details

Group ID:	androidx.activity
Artifact Name:	activity



Please select a module:

app
openCVLibrary349



Module 'app'

Step 1.
Please select the modules to add as dependencies.

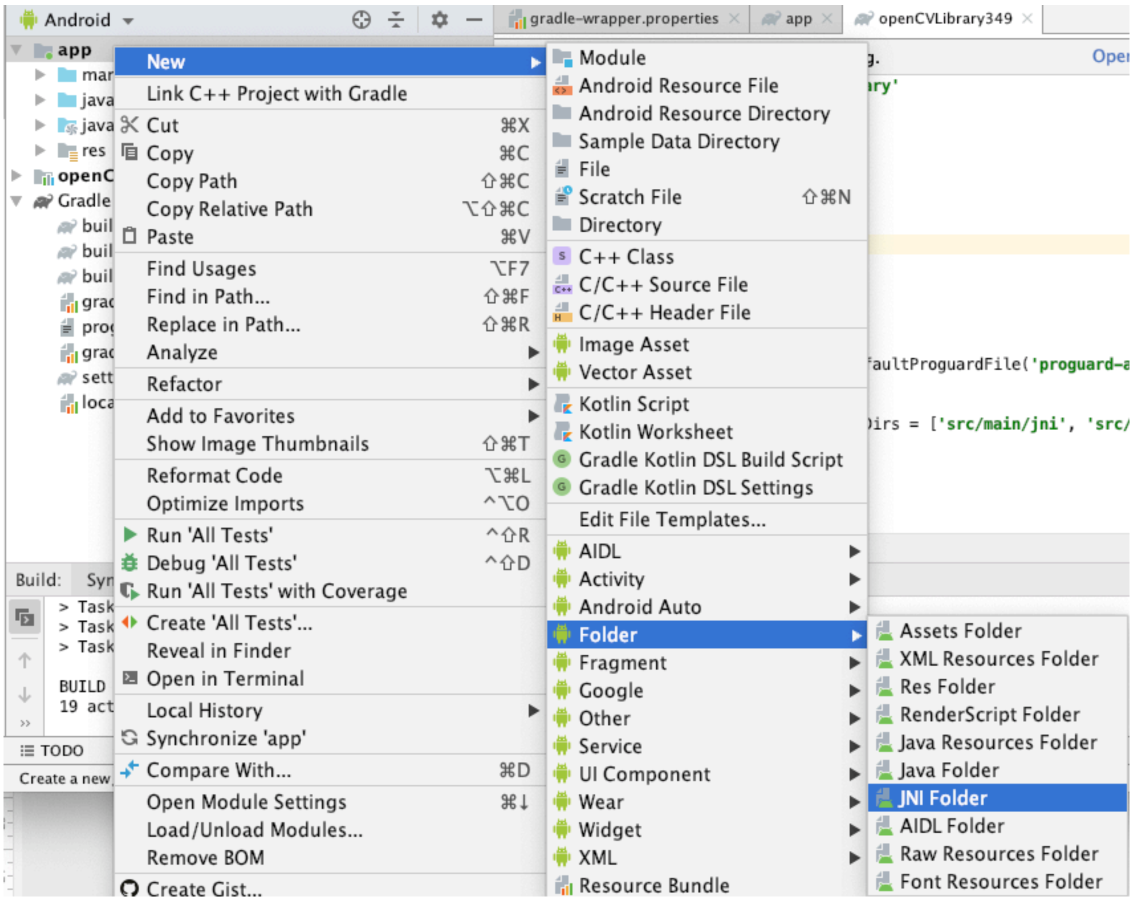
✓ openCVLibrary349

Modules: openCVLibrary349

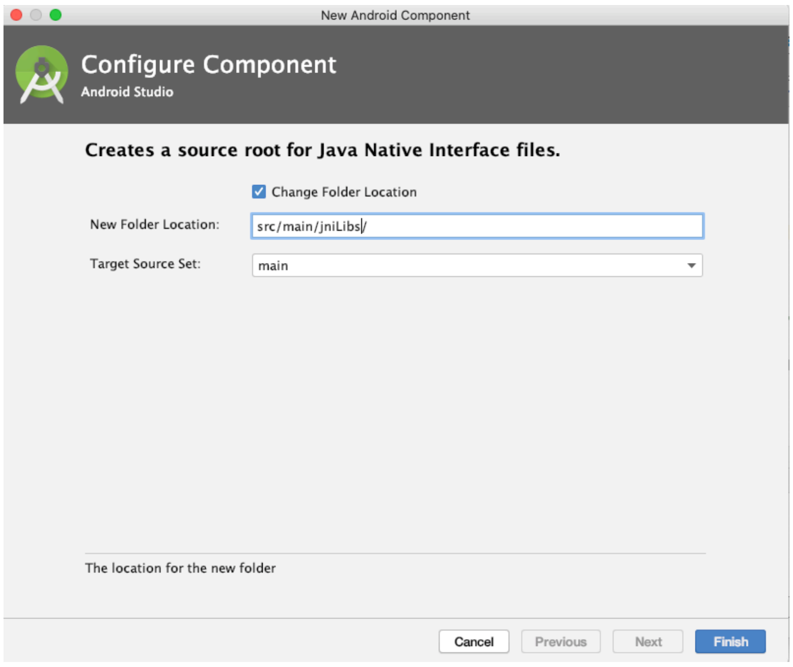
Step 2.
Assign a scope to the new dependency by selecting the configurations below.

implementation

Step 6: Create a **jniLibs** folder under the **app/src/main** folder by right click the **App** => **New** => **Folder** => **JNI folder**:



Select the "Change folder Location" and then rename the folder path as "**src/main/jniLibs/**". Click Finish to create the folder.


New Android Component

Configure Component

Android Studio

Creates a source root for Java Native Interface files.

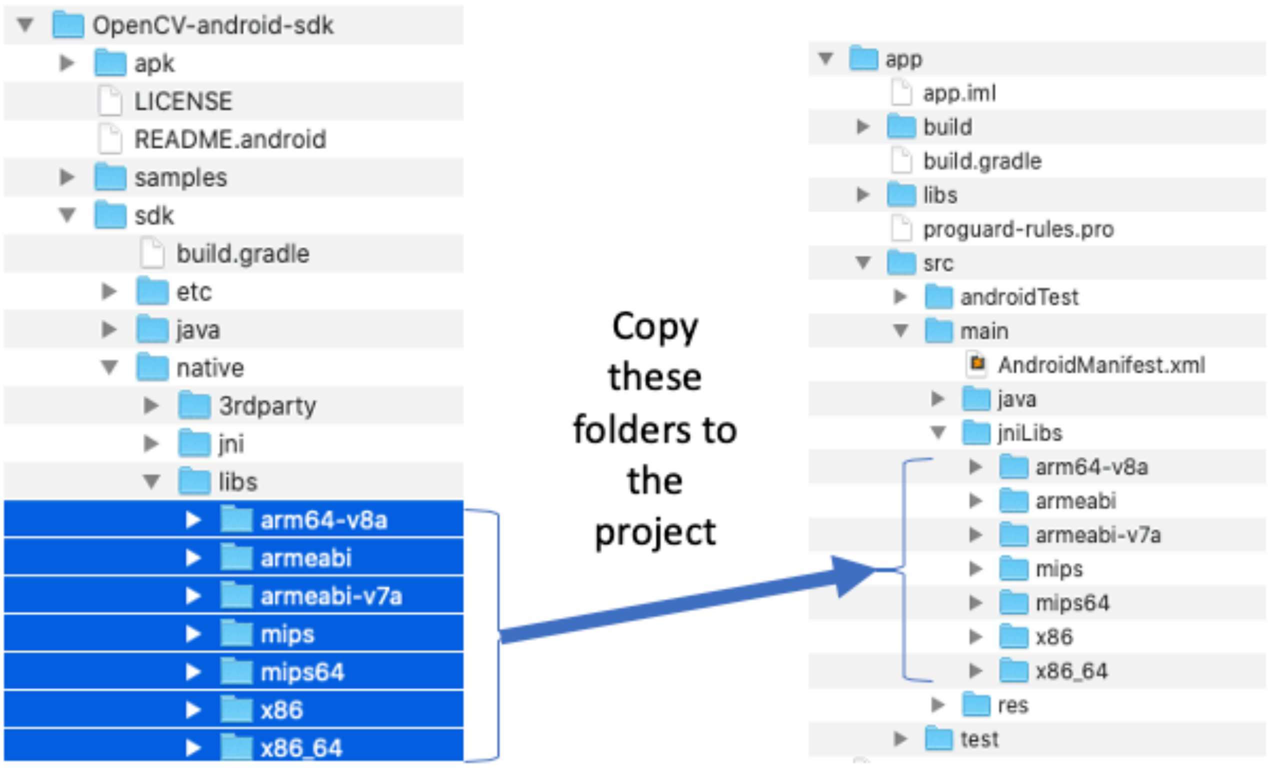
☑ Change Folder Location

New Folder Location: src/main/jniLibs/

Target Source Set: main

The location for the new folder

Step 7: Copy the the *libs* folder present in "OpenCV-android-sdk/sdk/native/libs" to the newly created "app/src/main/jniLib"s folder of the Android project.



Then the basic import of the OpenCV Library is completed and now we make the project to use the library to display the camera video on the MainActivity screen.

Step 8: Add permission in the `AndroidMainfest.xml` to allow the app to use the Camera:

```xml
<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"/>
```

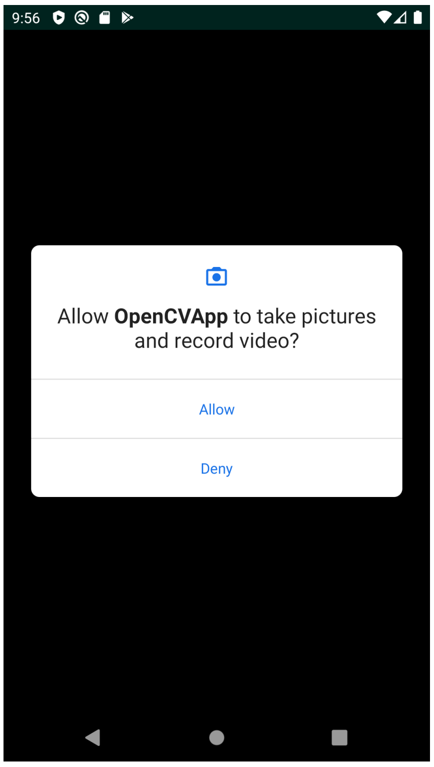
Step 9: Copy the following layout XML content to the `activity_main.xml`, which include an `JavaCamerView` for displaying the video:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <org.opencv.android.JavaCameraView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/javaCameraView" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

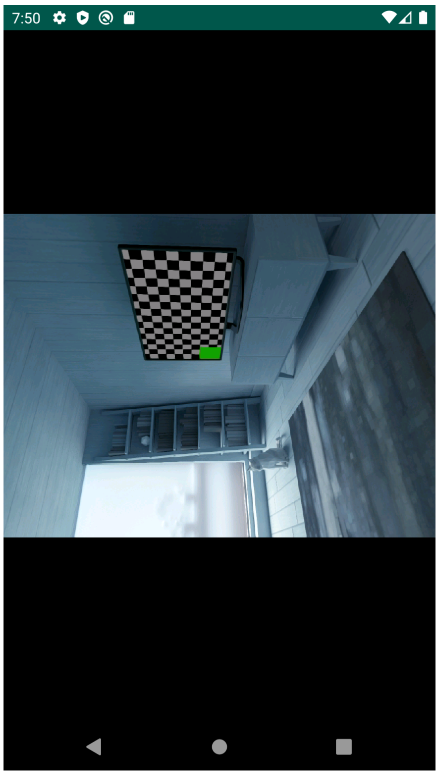
Step 10: Modify the `MainActivty.java` to implement

```java
import java.util.List;
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;
import org.opencv.imgproc.Imgproc;
import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.View.OnTouchListener;
import android.view.SurfaceView;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

public class MainActivity extends Activity implements CvCameraViewListener2 {
    private static final String TAG = "MainActivity";

    private boolean mIsColorSelected = false;
    private CameraBridgeViewBase mOpenCvCameraView;

    // Initialize OpenCV manager.
    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch (status) {
                case LoaderCallbackInterface.SUCCESS: {
                    Log.i(TAG, "OpenCV loaded successfully");
                    mOpenCvCameraView.enableView();
                }
                break;
                default: {
                    super.onManagerConnected(status);
                }
                break;
            }
        }
    };

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

    public void onCameraViewStopped() { }

    public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
        Mat frame = inputFrame.rgba();

        Core.flip(frame, frame, 1);
        return frame;

    }

    // Activity Cycle Call Backs

    // Called when the activity is first created.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "called onCreate");
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        setContentView(R.layout.activity_main);

        if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.CAMERA}, 0);
        }

        mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.javaCameraView);
        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }

    @Override
    public void onPause() {
        super.onPause();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.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_0_0, this, mLoaderCallback);
        } else {
            Log.d(TAG, "OpenCV library found inside package. Using it!");
            mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
    }

    public void onDestroy() {
        super.onDestroy();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }
}
```

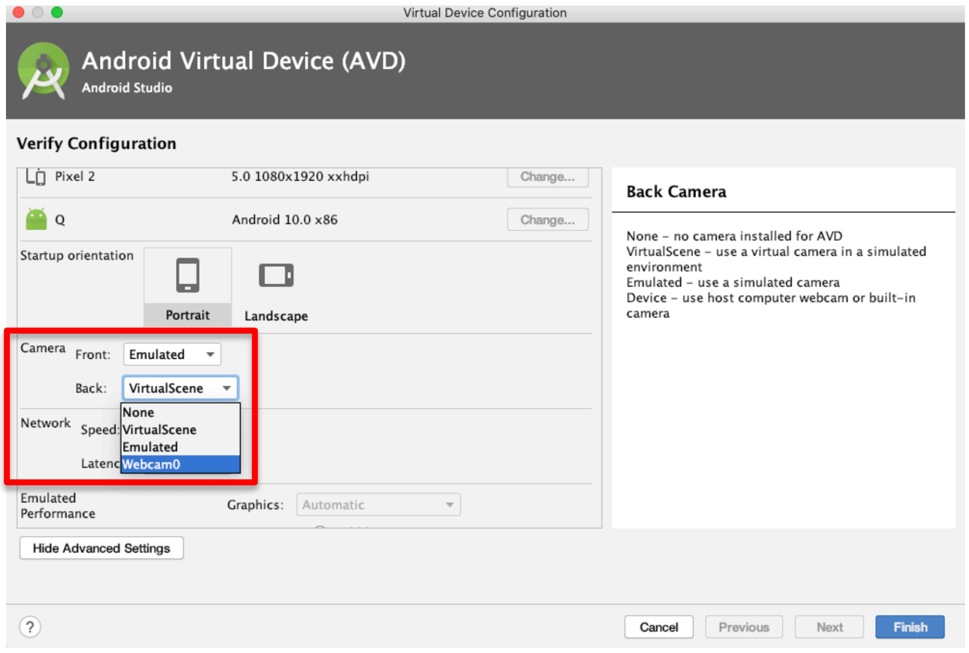
Step 11: Now, you can try to Run the app. However, we need to press "Allow" to let the app obtain the permission to use the Camera resource when the app is run in the first time (The left screen capture as shown below). After that your app will display a virtual scene on AVD screen as shown on the right screen capture . It is because the default settings of AVD's back camera is set to "VirtualScene".


9:56

Allow **OpenCVApp** to take pictures and record video?



If have a webcam connected to your computer, you can open the AVD manager to edit the "Show Advanced Setting" to configure the "Webcam0" as the "Back Camera", then you can have live video displaying on the screen.


Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

Pixel 2 5.0 1080x1920 xxhdpi
Q Android 10.0 x86

Startup orientation: Portrait, Landscape

Camera Front: Emulated
Back: VirtualScene

None
VirtualScene
Emulated
Webcam0

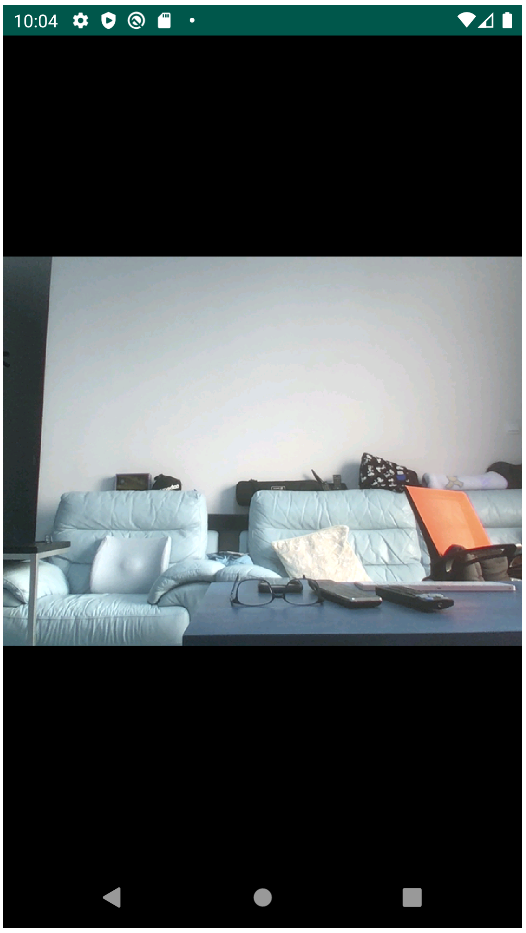
Network Speed: Latency:

Emulated Performance Graphics: Automatic

Back Camera

None – no camera installed for AVD
VirtualScene – use a virtual camera in a simulated environment
Emulated – use a simulated camera
Device – use host computer webcam or built-in camera

After this setting, you can run your OpenCV app again, now your webcam can be used to capture the video and display on the AVD's screen as shown below:

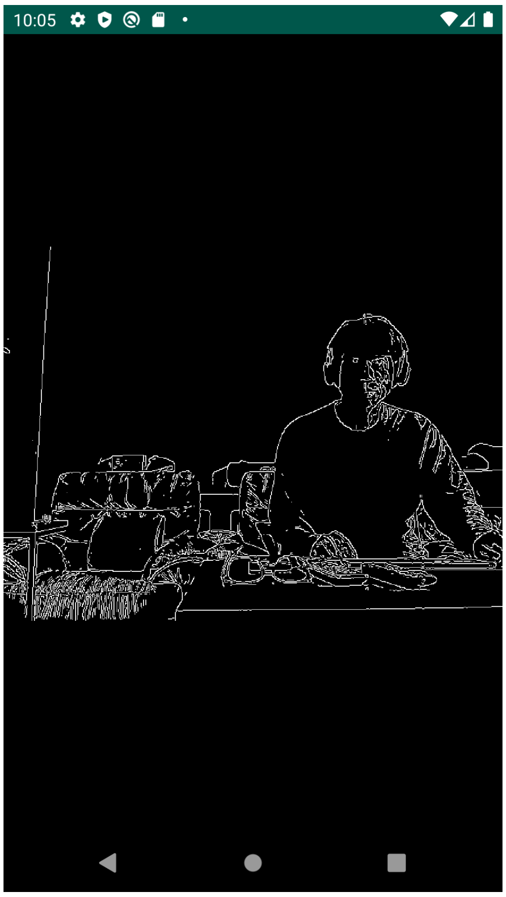


III. OpenCV Image Processing for Android

With the basic setting of the OpenCV Library for your Android project, now you can modify the `onCameraFrame(CvCameraViewFrame inputFrame)`method for perform some image processing operation of the captured video frame. Modify this method with the follow code for performing well-known Canny Edge Detection of the input video frames:

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
    Mat frame = inputFrame.rgba();

    // Perform Canny Edge Detect
    Core.flip(frame, frame, 1);
    Mat img_result = frame.clone();
    Imgproc.Canny(frame, img_result, 80, 90);
    return img_result;

}
```

Run the app again, then you should obtain the results as shown below:



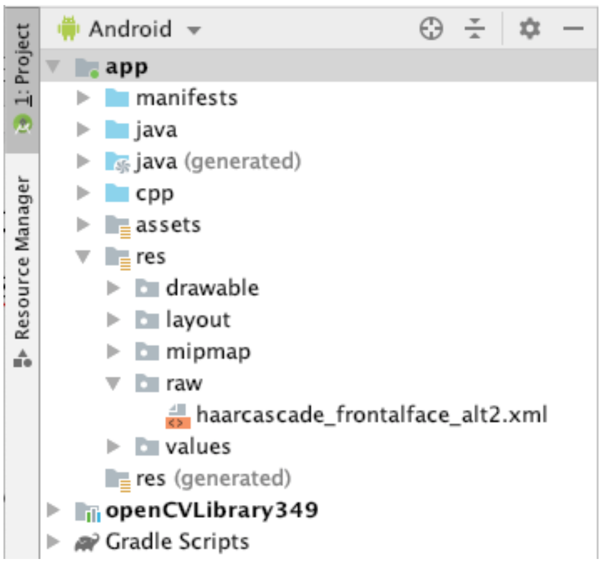
III. Face Detection using OpenCV Android Library

In this section, we will build an Android app to learn how to perform real-time face detection app using Haar Cascases Descriptor. The app demonstrate how to use traditional computer vision technique of Cascade Classifier to detect human faces. The app is designed to pass input video frame through this Haar Cascases classifier and obtain the output bounding box (x, y)-coordinates of each object in the image. Finally the results of each frame are displaying on the screen of MainActivity.

Start a new Android project with name of "**OpenCV Face Detector**" and then repeat the steps in section II for import the OpenCV Library into the project. The Haar Cascases model is available in the OpenCV library under the folder of

"***OpenCV-android-sdk/sdk/etc/haarcacades/*** *haarcasecade_frontalface_alt2.xml*".

Therefore, we can copy and paste this file into the Android project. Before that we need to create a raw folder under the **app/src/main/res** folder by right click the **App => New => Raw Resource Folder**. Then, **app/src/main/res/raw** folder will be created. Then can copy and paste the file of "***OpenCV-android-sdk/sdk/etc/haarcacades/*** *haarcasecade_frontalface_alt2.xml*" to "**app/src/main/res/raw**" folder.



In order to load these files, we need to first declare the File object of `cascFile` and Cascade Classifier object of `faceDetector` at the top of the MainActivity.java as shown below:

```
CascadeClassifier faceDetector;
File cascFile;
```

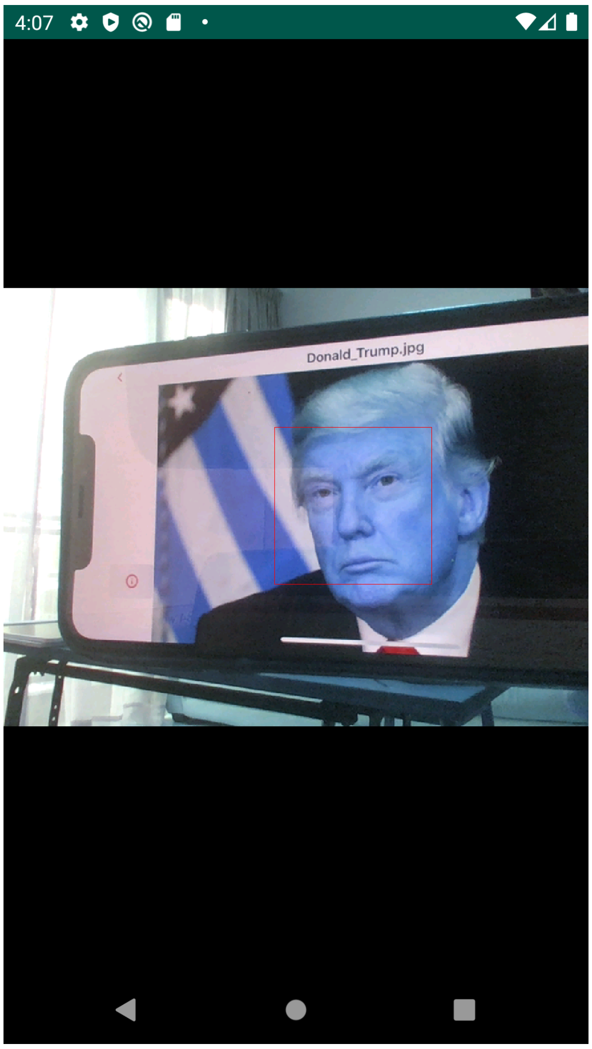
To load the file of the *haarcasecade_frontalface_alt2.xml*, we need to modify the `BaseLoaderCallback` class as shown below to load the HaarCascade face detector.

```
private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
    @Override
    public void onManagerConnected(int status) {
        switch (status) {
            case LoaderCallbackInterface.SUCCESS: {
                Log.i(TAG, "OpenCV loaded successfully");

                // Load the Cascade Classifier
                InputStream is =
getResources().openRawResource(R.raw.haarcascade_frontalface_alt2);
                File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);
                cascFile = new File(cascadeDir, "haarcasecade_frontalface_alt2.xml");

                try {
                    FileOutputStream fos = new FileOutputStream(cascFile);

                    byte[] buffer = new byte[4096];
                    int bytesRead;

                    bytesRead = is.read(buffer);
                    while (bytesRead != -1) {
                        fos.write(buffer, 0, bytesRead);
                        bytesRead = is.read(buffer);
                    }
                    is.close();
                    fos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }

                faceDetector = new CascadeClassifier((cascFile.getAbsolutePath()));

                if (faceDetector.empty()) {
                    faceDetector = null;
                } else {
                    cascadeDir.delete();
                }

                mOpenCvCameraView.enableView();
            }
            break;
            default: {
                super.onManagerConnected(status);
            }
            break;
        }
    }
};
```

In addition, we need to modify the `onCameraFrame()` method as shown below to perform face detection of each input video frame by pass through the frame to the detector.

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
    Mat frame = inputFrame.rgba();

    // Perform Face Detection
    MatOfRect faceDetections = new MatOfRect();
    faceDetector.detectMultiScale(frame, faceDetections);

    for (Rect rect: faceDetections.toArray()) {
        Imgproc.rectangle(frame, new Point(rect.x, rect.y),
                new Point(rect.x + rect.width, rect.y+rect.height),
                new Scalar(255,0,0));
    }

    return frame;

}
```

Run the app again, then you should obtain the results as shown below for detecting human face:

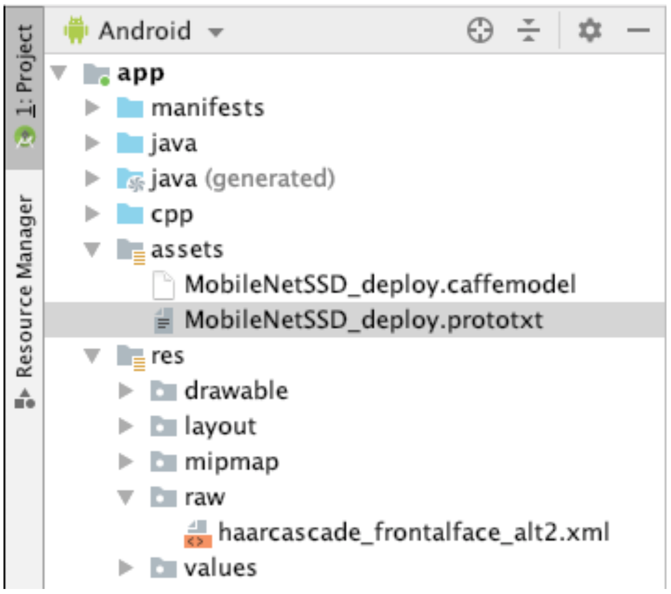


III. Real-time Object Recognition with Deep Learning SSD Net

In this section, we will build an Android app to learn how to perform real-time object detection using deep learning + OpenCV. The object detection is based on Single Shot Detector (SSD) using and MobileNets. The app demonstrate how to use OpenCV's dnn module to load a pre-trained object detection network. This will enable us to pass input video frame through the network and obtain the output bounding box (x, y)-coordinates of each object in the image. Finally the results of applying the MobileNet SSD to each frame are displaying on the screen of MainActivity.

More detail of the deep learning based object detector can be found in
https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/

Start a new Android project with name of "OpenCV SSDNet" and then repeat the steps in section II for import the OpenCV Library into the project. After that we need to create an **assets** folder under the **app/src/main** folder by right click the **App** => **New** => **Assests Folder**. Then, **app/src/main/assets** folder will be created and we store the SSD model and pre-trained parameters files in this folder. The MobileNet based SSD object detection model can be downloaded from https://github.com/chuanqi305/MobileNet-SSD. Moreover, **copies of these model files can also be downloaded from the course website**. We need a configuration file MobileNetSSD_deploy.prototxt and weights MobileNetSSD_deploy.caffemodel. After downloading the files, we just need to copy and paste them into the **app/src/main/assets/** folder as shown below:


```
Android
app
  manifests
  java
  java (generated)
  cpp
  assets
    MobileNetSSD_deploy.caffemodel
    MobileNetSSD_deploy.prototxt
  res
    drawable
    layout
    mipmap
    raw
      haarcascade_frontalface_alt2.xml
    values
```

In order to load these files, we need to modify the `onCameraViewStarted()` as shown below to load the model and parameters of the SSD object detector.

```
public void onCameraViewStarted(int width, int height) {

        //Load a network
        String proto = getPath("MobileNetSSD_deploy.prototxt", this);
        String weights = getPath("MobileNetSSD_deploy.caffemodel", this);
        net = Dnn.readNetFromCaffe(proto, weights);
        Log.i(TAG, "Network loaded successfully");
}

// Upload file to storage and return a path.
private static String getPath(String file, Context context) {
    AssetManager assetManager = context.getAssets();
    BufferedInputStream inputStream = null;
    try {
        // Read data from assets.
        inputStream = new BufferedInputStream(assetManager.open(file));
        byte[] data = new byte[inputStream.available()];
        inputStream.read(data);
        inputStream.close();
        // Create copy file in storage.
        File outFile = new File(context.getFilesDir(), file);
        FileOutputStream os = new FileOutputStream(outFile);
        os.write(data);
        os.close();
        // Return a path to file which may be read in common way.
        Log.i(TAG, file + " is successfully uploaded");
        return outFile.getAbsolutePath();
    } catch (IOException ex) {
        Log.i(TAG, "Failed to upload a file");
    }
    return "";
}
```

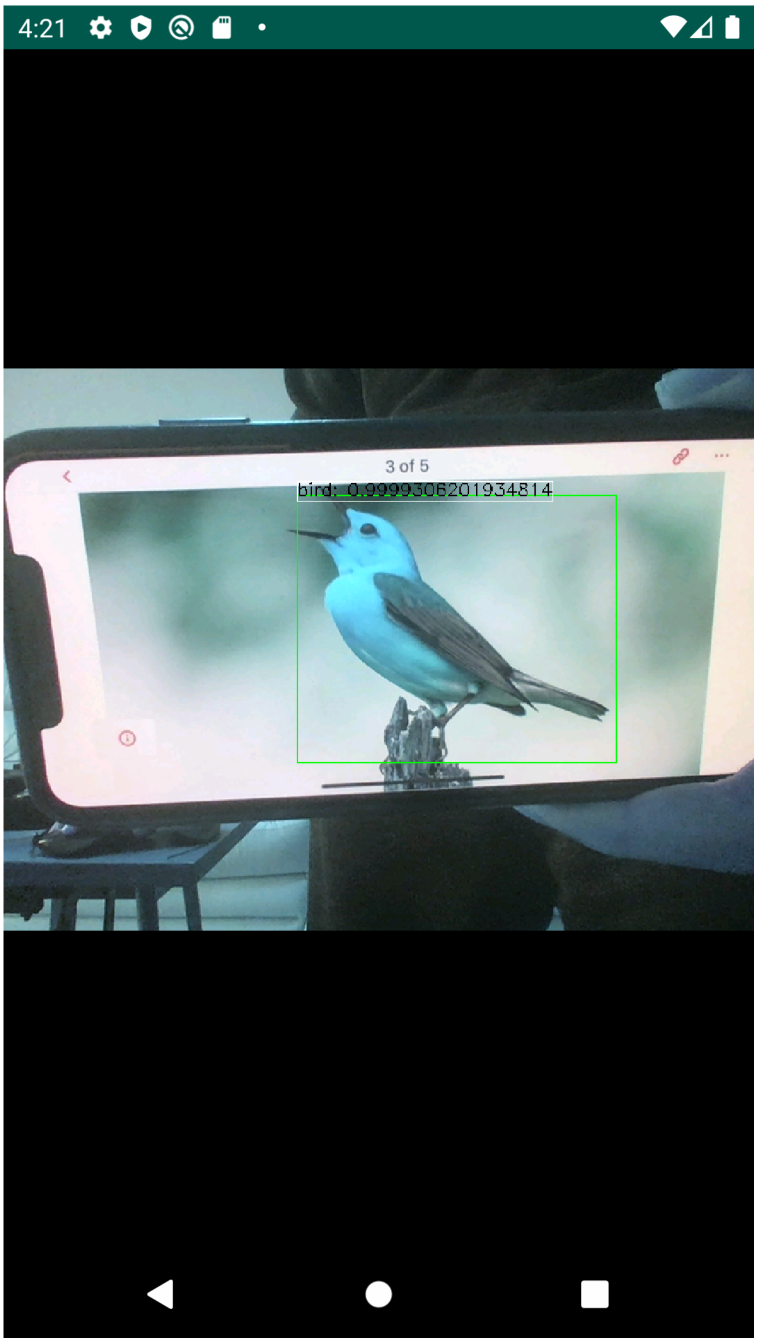
In addition, we need to modify the `onCameraFrame()` method as shown below to perform object recognition of each input video frame by pass through the frame to the SSD network.

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {

        // MobileNet Object Recognition
        final int IN_WIDTH = 300;
        final int IN_HEIGHT = 300;
        final float WH_RATIO = (float)IN_WIDTH / IN_HEIGHT;
        final double IN_SCALE_FACTOR = 0.007843;
        final double MEAN_VAL = 127.5;
        final double THRESHOLD = 0.2;
        // Get a new frame
        Mat frame = inputFrame.rgba();
        Imgproc.cvtColor(frame, frame, Imgproc.COLOR_RGBA2RGB);
        // Forward image through network.
        Mat blob = Dnn.blobFromImage(frame, IN_SCALE_FACTOR,
                new Size(IN_WIDTH, IN_HEIGHT),
                new Scalar(MEAN_VAL, MEAN_VAL, MEAN_VAL), /*swapRB*/false, /*crop*/false);
        net.setInput(blob);
        Mat detections = net.forward();
        int cols = frame.cols();
        int rows = frame.rows();
        detections = detections.reshape(1, (int)detections.total() / 7);
        for (int i = 0; i < detections.rows(); ++i) {
            double confidence = detections.get(i, 2)[0];
            if (confidence > THRESHOLD) {
                int classId = (int)detections.get(i, 1)[0];
                int left   = (int)(detections.get(i, 3)[0] * cols);
                int top    = (int)(detections.get(i, 4)[0] * rows);
                int right  = (int)(detections.get(i, 5)[0] * cols);
                int bottom = (int)(detections.get(i, 6)[0] * rows);
                // Draw rectangle around detected object.
                Imgproc.rectangle(frame, new Point(left, top), new Point(right, bottom),
                        new Scalar(0, 255, 0));
                Log.i(TAG, "calassID"+classId+"confidence:"+confidence);

                String label = classNames[classId] + ": " + confidence;
                int[] baseLine = new int[1];
                Size labelSize = Imgproc.getTextSize(label, Core.FONT_HERSHEY_SIMPLEX, 0.5, 1,
baseLine);
                // Draw background for label.
                Imgproc.rectangle(frame, new Point(left, top - labelSize.height),
                        new Point(left + labelSize.width, top + baseLine[0]),
                        new Scalar(255, 255, 255), 1);
                // Write class name and confidence.
                Imgproc.putText(frame, label, new Point(left, top),
                        Core.FONT_HERSHEY_SIMPLEX, 0.5, new Scalar(0, 0, 0));
            }
        }
        return frame;

    }
```

Finally, we to declare the object **classNames** string array and the neural network **net** object at the top of the MainActivity.java as shown below:

```
private static final String[] classNames = {"background",
        "aeroplane", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair",
        "cow", "diningtable", "dog", "horse",
        "motorbike", "person", "pottedplant",
        "sheep", "sofa", "train", "tvmonitor"};

private Net net;
```

Run the app again, then you should obtain the results as shown below:


4:21
3 of 5
bird: 0.9999306201934814