Android+OpenCV

(20181216)

1. 安装教程网上很多，我一开始参考的是以下这篇博文：

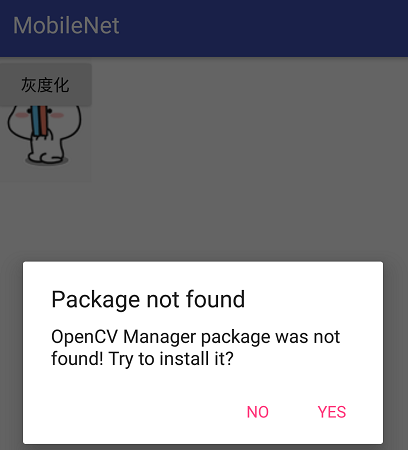
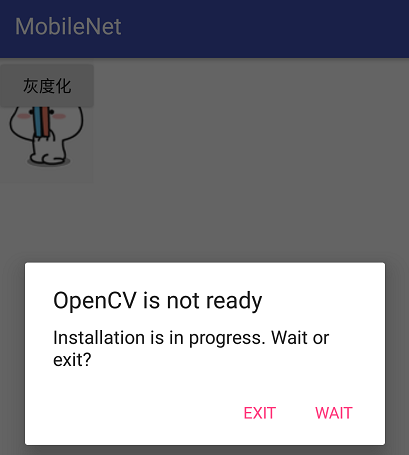
<https://blog.csdn.net/gao_chun/article/details/49359535>

但是这里有两个问题：

1. 他的示例代码中没有给全，实际上要能够运行的话需要在前面添加许多import库，比如博客里面的代码需要添加：（可以根据底下的错误提示来选择）

package com.tengine.openailab.mobilenet;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.graphics.BitmapFactory;  
import android.view.View;  
import android.graphics.Bitmap;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.util.Log;  
  
import butterknife.BindView;  
  
import org.opencv.android.BaseLoaderCallback;  
import org.opencv.android.LoaderCallbackInterface;  
import org.opencv.android.OpenCVLoader;  
import org.opencv.android.Utils;  
import org.opencv.core.Mat;  
import org.opencv.imgproc.Imgproc;

1. 第二个没有解决的问题是OpenCV manager安装的问题，上面博客里面提到了这一点，但是没有说如何解决，在实际运行当中如果没有安装这个会在手机上运行的时候出现下面的错误提示：

而且这个manager已经显示下架了，想安装也装不上（oppo手机如此），于是需要另寻他解，有许多博客说不用安装manager也可以实现，但是最后各种方法试下来，下面这种方法是我最终成功的：

<https://www.cnblogs.com/tail/p/4618790.html>

意思是OpenCV在AS3.0版本里所给出的示例程序就说可以免OpenCV安装，它在初始代码里面的onResume函数当中是这样写的：

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);

}

}

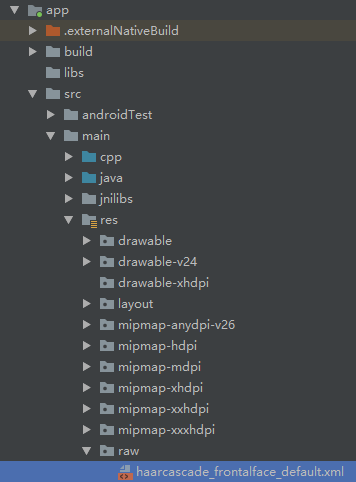
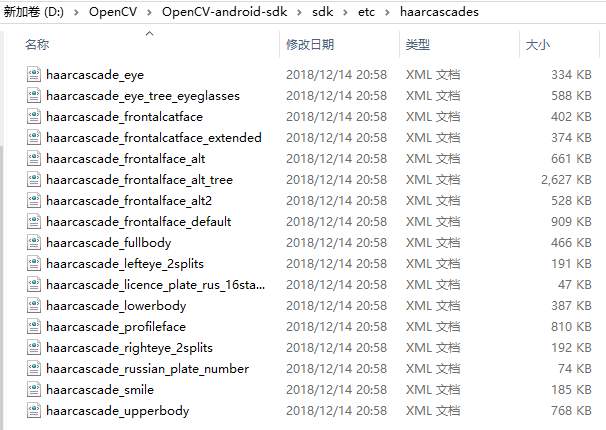
直接替换亲测有效。

1. Opencv实现人脸检测

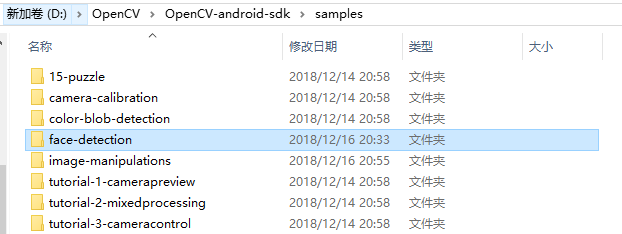
这部分讲述一下使用opencv自带的人脸检测模型来检测人脸实现方式，其实很简单，但是中间过程有许多问题，现在也有很多问题还没有解决。该部分实例参考

TryOpenCV\_SinglePictureagain目录下面的工程。

1. 准备运动：这里就是之前1中所说的将opencv当中的sdk装载到AS工程下面，然后要将opencv当中自带的人脸检测模型拷贝到app下面的raw目录：app/src/main/res/raw，这个自带的模型在opencv安装包当中的目录：

1. 然后我们需要修改三个地方的代码，这些代码都可以参考opencv安装包里面的samples例程，



第一个是activity.java文件：需要修改这几个地方：

【1】package的名字修改成自己的：

package com.tengine.openailab.mobilenet;

【2】activity的名字修改成自己的：

public class MainActivity extends Activity implements CvCameraViewListener2

【3】因为这部分代码既可以使用JAVA\_DETECTOR，也可以使用NATIVE\_DETECTOR，但是NATIVE\_DETECTOR在这里还不能用，因为sample代码是有另外一个.java文件来描述这个NATIVE\_DETECTOR检测器的，于是我们需要将原来代码里面的与NATIVE\_DETECTOR相关的都注释掉：

// System.loadLibrary("detection\_based\_tracker");

//mNativeDetector = new DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);

//mNativeDetector.setMinFaceSize(mAbsoluteFaceSize);

//if (mNativeDetector != null)  
 //mNativeDetector.detect(mGray, faces);

if (type == *NATIVE\_DETECTOR*) {  
 Log.*i*(*TAG*, "Detection Based Tracker enabled");  
 //mNativeDetector.start();  
} else {  
 Log.*i*(*TAG*, "Cascade detector enabled");  
 //mNativeDetector.stop();  
}

1. 第二个文件是布局文件activity\_main.xml，这里我是直接复制过来的，就建立了一个JavaCameraView用作摄像头采集界面：（不晓得这个linearlayout对界面有没有影响？）

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" >  
  
 <org.opencv.android.JavaCameraView  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 android:id="@+id/fd\_activity\_surface\_view" />  
</LinearLayout>

1. 第三个文件是权限文件AndroidManifest.xml，主要需要开通几个权限：

【1】摄像头权限（但是其实在这里开通之后在手机上还是需要在允许调用摄像头操作）

<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" />

【2】设置手机横屏，不设置的话手机是倒的屏幕

<activity android:name=".MainActivity"  
 android:screenOrientation="landscape"> <!--landscape:设置手机横屏-->

1. 这样以后下载到手机上就可以进行人脸框图了：
2. 自己的一点问题：

这里自己遇到了一点问题，那就是原来的设想是不直接使用摄像头摄影来实现人脸检测的功能，而是在之前在灰度化的代码的基础上，能够对单张照片进行识别处理，从而在手机app界面上显示出来。于是就仔细看看AS当中灰度化的工程代码，加上自己的一般化理解来进行代码组合~，下面就我的理解来分析一下这部分代码，但是这里自己实现的时候是app闪退，所以我也不懂为啥，只能在将来多看看安卓的程序才能做得更多吧：

首先看activity.java文件：

【1】前面的package和import就不说了

【2】onCreate主要是界面的创建

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 getWindow().addFlags(WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*);  
 setContentView(R.layout.*activity\_main*);  
  
 img = (ImageView)findViewById(R.id.*img*);  
 btn = (Button)findViewById(R.id.*btn*);  
 btn.setOnClickListener(new ProcessClickListener());  
}

【3】onResume就按照之前说的不需要opencv manager的代码拷贝过来就行了

【4】BaseLoaderCallback

//OpenCV库加载并初始化成功后的回调函数  
 private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {  
  
 @Override  
 public void onManagerConnected(int status) {  
 // *TODO Auto-generated method stub* switch (status){  
// case BaseLoaderCallback.SUCCESS:  
// Log.i(TAG, "成功加载");  
// break;

***//以下部分是来自sample的onmanagerconnected部分*** case LoaderCallbackInterface.*SUCCESS*:  
 {  
 Log.*i*(*TAG*, "OpenCV loaded successfully");  
 // Load native library after(!) OpenCV initialization  
 // System.loadLibrary("detection\_based\_tracker");  
  
 try {  
 // load cascade file from application resources  
 InputStream is = getResources().openRawResource(R.raw.*haarcascade\_frontalface\_default*);  
 File cascadeDir = getDir("cascade", Context.*MODE\_PRIVATE*);  
 mCascadeFile = new File(cascadeDir, "haarcascade\_frontalface\_default.xml");  
 FileOutputStream os = new FileOutputStream(mCascadeFile);  
  
 byte[] buffer = new byte[4096];  
 int bytesRead;  
 while ((bytesRead = is.read(buffer)) != -1) {  
 os.write(buffer, 0, bytesRead);  
 }  
 is.close();  
 os.close();  
 ***//定义一个新的分类器***  
 mJavaDetector = new CascadeClassifier(mCascadeFile.getAbsolutePath());  
 btn.setText("运行140");  
 if (mJavaDetector.empty()) {  
 Log.*e*(*TAG*, "Failed to load cascade classifier");  
 mJavaDetector = null;  
 } else  
 Log.*i*(*TAG*, "Loaded cascade classifier from " + mCascadeFile.getAbsolutePath());  
  
 //mNativeDetector = new DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);  
  
 cascadeDir.delete();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 Log.*e*(*TAG*, "Failed to load cascade. Exception thrown: " + e);  
 }  
 btn.setText("运行163");  
 }break;  
 default:  
 super.onManagerConnected(status);  
 Log.*i*(*TAG*, "加载失败");  
 break;  
 }  
 ***//以下部分是原来的灰度化函数修改而来的***  
 Mat rgbMat = new Mat();  
 Mat grayMat = new Mat();  
 Mat facedetectMat = new Mat();  
 srcBitmap = BitmapFactory.*decodeResource*(getResources(), R.drawable.*wechat*);  
 grayBitmap = Bitmap.*createBitmap*(srcBitmap.getWidth(), srcBitmap.getHeight(), Bitmap.Config.*RGB\_565*);  
 Utils.*bitmapToMat*(srcBitmap, rgbMat);//convert original bitmap to Mat, R G B.  
 Imgproc.*cvtColor*(rgbMat, grayMat, Imgproc.*COLOR\_RGB2GRAY*);//rgbMat to gray grayMat  
 Utils.*matToBitmap*(grayMat, grayBitmap); //convert mat to bitmap  
 Log.*i*(*TAG*, "procSrc2Gray sucess...");  
  
 if (mAbsoluteFaceSize == 0) {  
 int height = grayMat.rows();  
 if (Math.*round*(height \* mRelativeFaceSize) > 0) {  
 mAbsoluteFaceSize = Math.*round*(height \* mRelativeFaceSize);  
 }  
 }  
  
 MatOfRect faces = new MatOfRect();  
 btn.setText("运行190");  
 if (mJavaDetector != null)  
 mJavaDetector.detectMultiScale(grayMat, faces, 3, 2, 2, // *TODO: objdetect.CV\_HAAR\_SCALE\_IMAGE* new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());  
 btn.setText("运行194");  
 Rect[] facesArray = faces.toArray();  
 for (int i = 0; i < facesArray.length; i++)  
 Imgproc.*rectangle*(facedetectMat, facesArray[i].tl(), facesArray[i].br(), *FACE\_RECT\_COLOR*, 3);  
  
 Utils.*matToBitmap*(facedetectMat, facedetectBitmap); //convert mat to bitmap  
 btn.setText("运行200");  
  
 }  
  
 };

从上述可以看到，我们最后要得到的是一个Bitmap文件，这个是最后能够显示出来的文件格式。

【5】这部分就是在点击按钮的时候需要做的事情

public class ProcessClickListener implements View.OnClickListener{  
  
 @Override  
 public void onClick(View v) {  
 // *TODO Auto-generated method stub* if(*isFirst*){  
 //procSrc2Gray();  
 *isFirst* = false;  
 btn.setText("运行262");  
 }  
 if(*flag*){  
 //img.setImageBitmap(facedetectBitmap);  
 img.setImageBitmap(grayBitmap);  
 btn.setText("查看原图");  
 *flag* = false;  
 }else{  
 //img.setImageBitmap(srcBitmap);  
 img.setImageBitmap(facedetectBitmap);  
 btn.setText("灰度化");  
 *flag* = true;  
 }  
 }  
}

然而这个代码最后实现的结果是可以编译通过，下载到手机上的时候遇到闪退的问题，目前还没有解决，网上也没有找到可以借鉴的案例。