**技术文档**

一、基于 Android 的 OpenCV 开发环境搭建

1. OpenCV SDK 准备

采用直接加载openCV官方提供的Android SDK来进行导入：

先到官网http://opencv.org/releases.html ，下载Android 包，如：opencv-3.2.0-android-sdk.zip

1. 进入后选择 Android SDK 点击下载即可

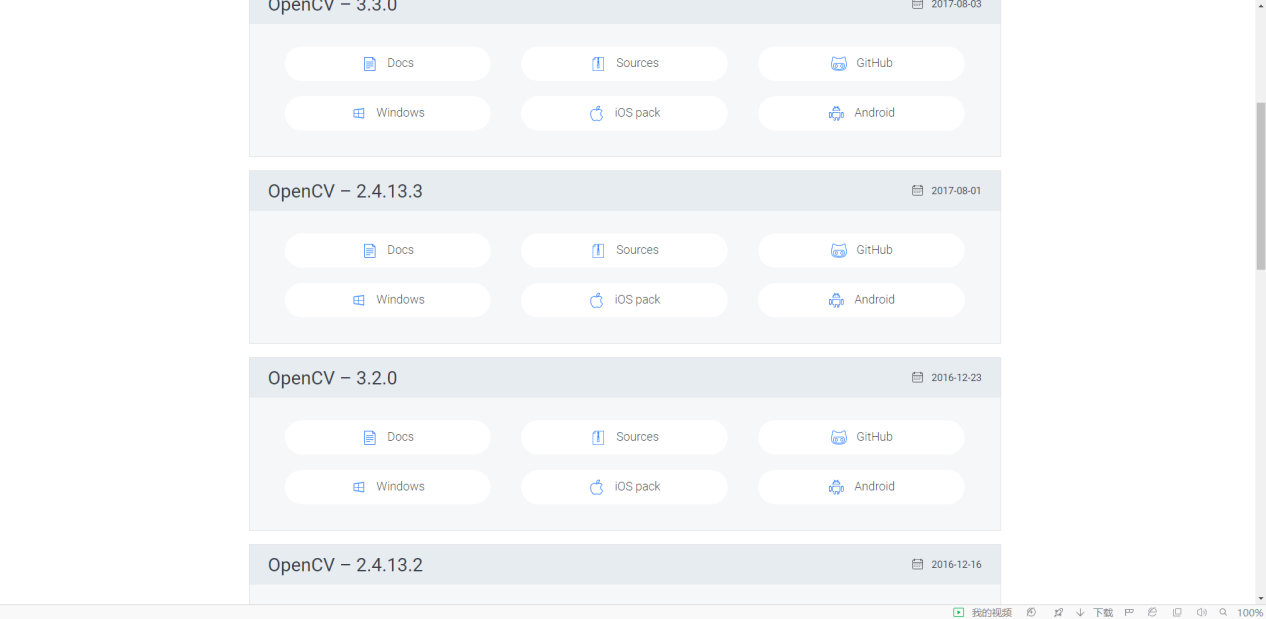


图1.1.1 Opencv下载

1. 下载成功后解压得到我们要的OpenCV 资源



图1.1.2 解压下载资源

1. 将 OpenCV 导入到项目中去
2. 新建 Android 项目

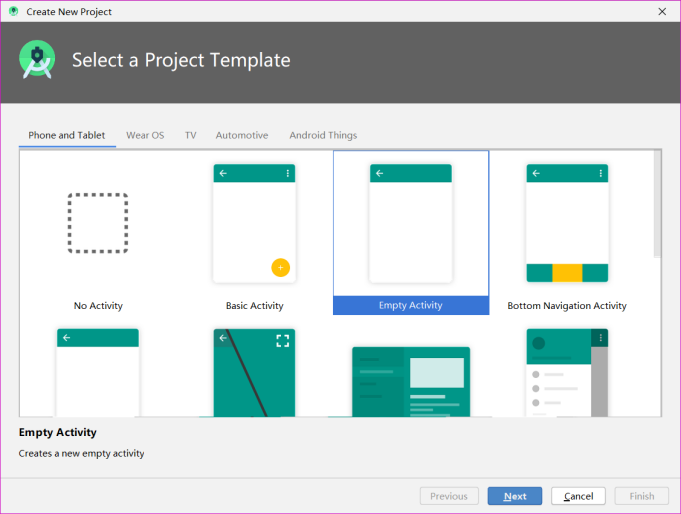


图1.2.1 新建项目（1）

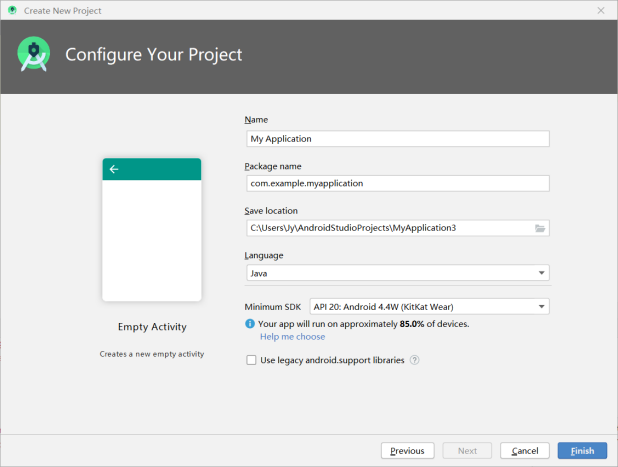


图1.2.2 新建项目（2）

1. 导入 OpenCV SDK
2. 导入刚刚下载解压到 SDK 包：选择：File > New > New Module

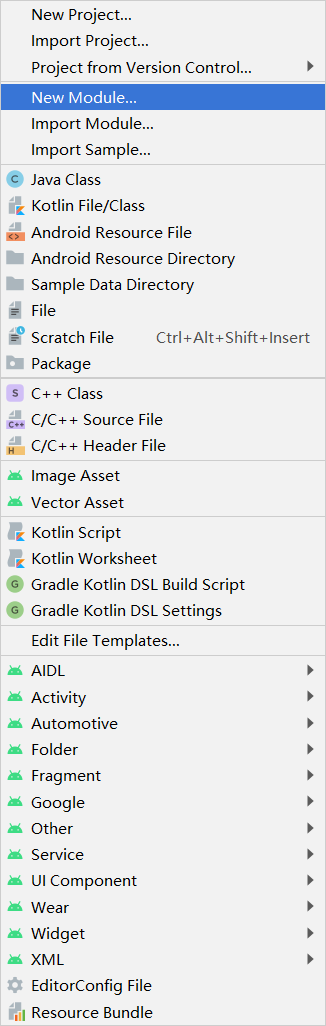


图1.2.3 路径选择

1. 选择Import Eclipse ADT Project

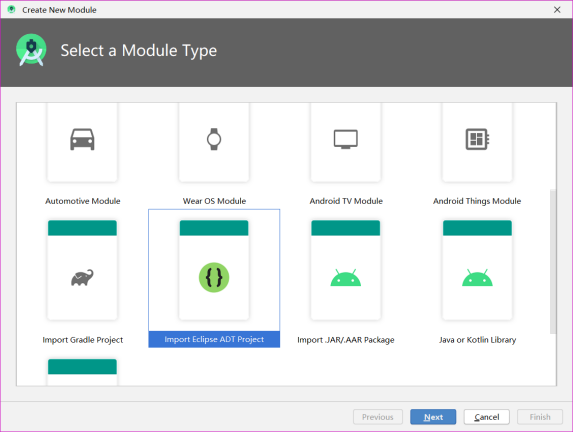


图1.2.4 选择项目

1. 选择刚刚下载解压得到 SDK 文件，点击open 路径为\OpenCV-android-sdk\sdk\java

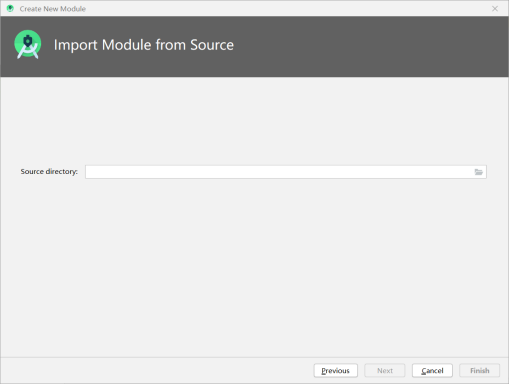


图1.2.5 选择解压文件

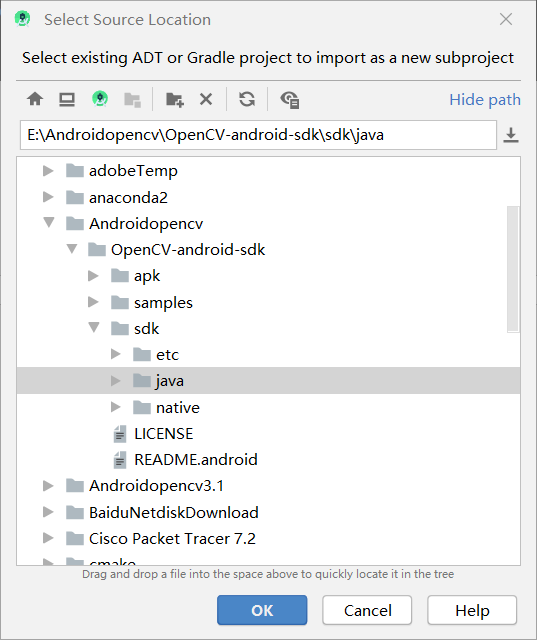


图1.2.6 路径选择

1. 设置 Module 名称，Finish

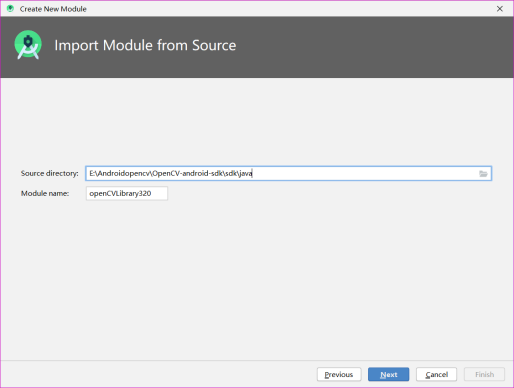


图1.2.7 设置名称（1）

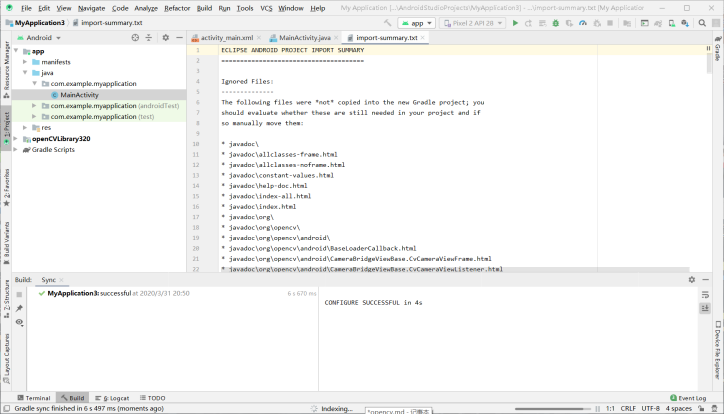


图1.2.8 设置名称（2）

1. 直接在 app 目录下build.gradle 文件里dependencies 大括号下添加 compile project(':openCVLibrary') 版本不同可能要将compile换成implementation

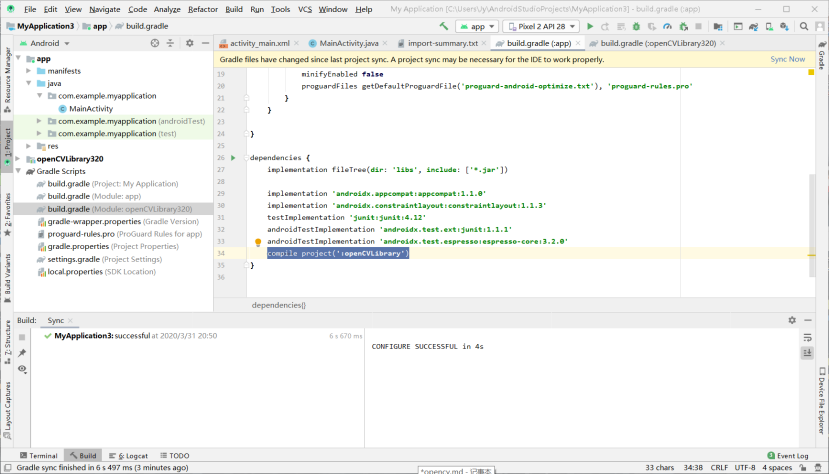


图1.3.1 修改依赖（1）

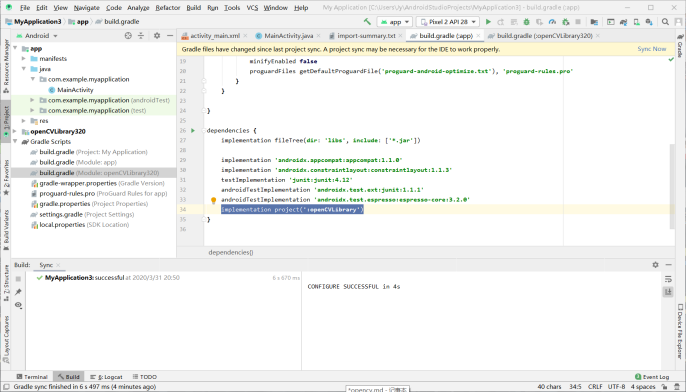


图1.3.2 修改依赖（2）

1. 打开刚导入的模块下 build.gradle 文件，把 compileSdkVersion 、 minSdkVersion 和 targetSdkVersion修改成build.gradle(module:app)的SDK版本

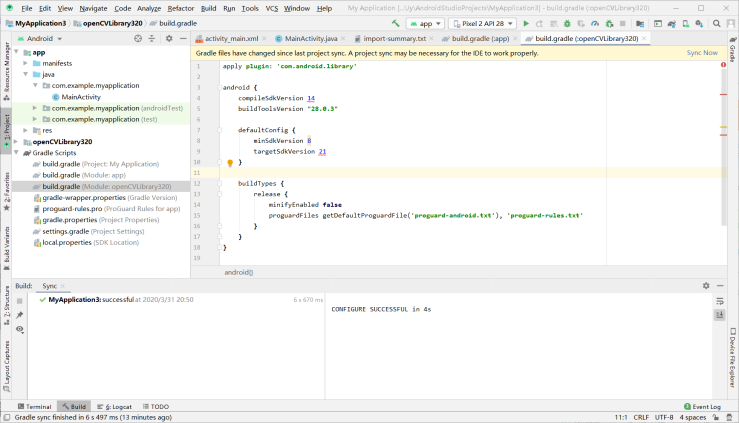


图1.4.1 修改参数（1）

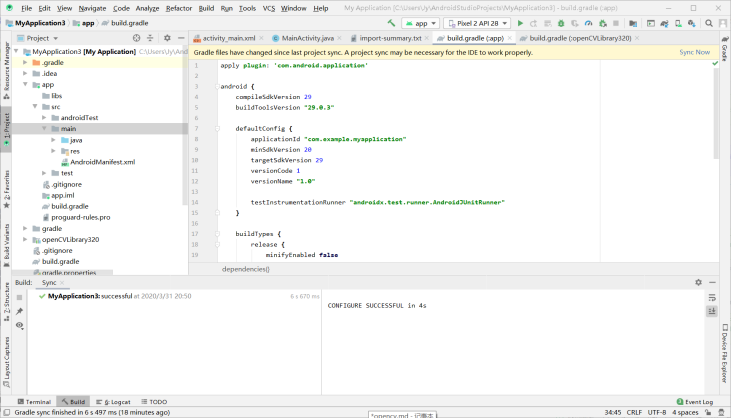


图1.4.2 修改参数（2）

1. 接着在 app/src/main 目录下 创建一个jniLibs 目录，然后把sdk/native/libs 下所有文件拷贝到jniLibs下，编译，运行。

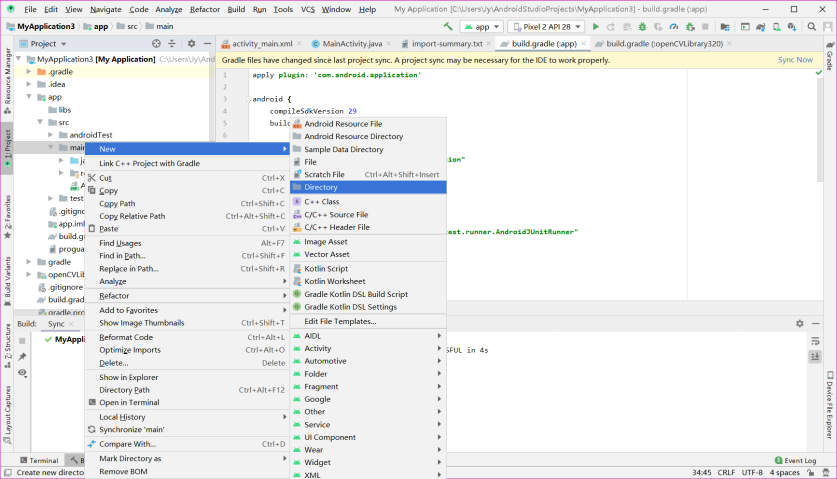


图1.5.1 创建目录

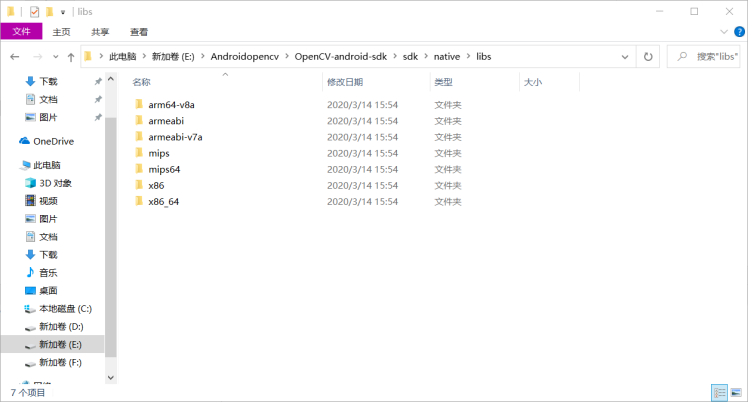


图1.5.2 拷贝文件（1）

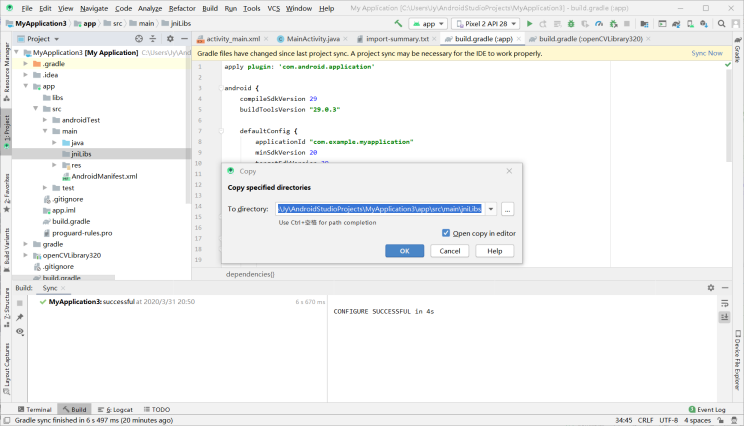


图1.5.3 拷贝文件（2）

二、基本代码

1、界面布局

创建一个JavaCameraView用于显示图像，以及一个按钮切换前置后置。



图2.1.1 界面布局

2、创建一个MainActivity .java

实现打开相机权限请求



图2.2.1 打开相机



图2.2.2 请求权限

3、创建一个detectActivity.java

定义如上变量，用来获取摄像机cameraView，级联分类器classifier，Mat对象grayscaleImage包含了图像的各种基本信息与图像像素数据。 isFrontCamera用来判断是不是前置摄像头。

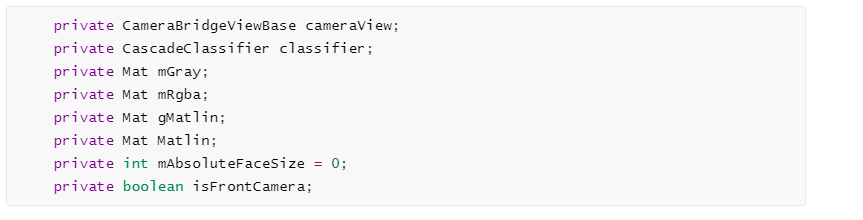


图2.3.1 创建detectActivity.java

4、初始化屏幕

设置屏幕为常亮满屏。

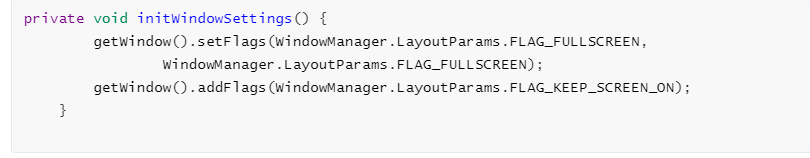


图2.4.1 初始化屏幕

5、相机功能

通过setCvCamera设置相机监听，调用enableView将相机开启，获取button组件，使用实现接口的方式来进行注册监听器。



图2.5.1 注册监听器

在Oncreate中为Button switchCamera的点击事件注册一个监听器，使用实现接口的方式来进行注册监听器。触发时先将cameraView设为不可见，当isFrontCamera=true时，表示当前状态为前置摄像头。所以点击该按钮则要将其变为后置，即



图2.5.2 前后置摄像头选择

此时isFrontCamera设为false。



图2.5.3 设置摄像头

修改CvCameraViewListener2接口自带的方法，在相机启动时创建mat mGray、mRgba ，结束时释放。



图2.5.4 修改方法

6、人脸检测

为了得到更好的人脸检测性能，OpenCV在SDK中提供了多个frontface检测器(人脸模型)，存放在

..\opencv-3.3.0-android-sdk\OpenCV-android-sdk\sdk\etc\目录下，该模型包括了3000个正样本和1500个负样本。



图2.6.1 检测器

我们将其拷贝到AS工程的res/raw目录下，并通过getDir方法保存到/data/data/com.yl.fecedetectdemo/ cascade目录下。

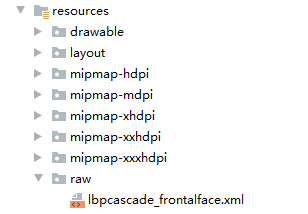


图2.6.2 保存目录

用getResources().openRawResource导入opencv自带的人脸检测训练集lbpcascade\_frontalface.xml。

OpenCV中提供有分类器类CascadeClassifier。

创建一个分类器对象cascadeClassifier，使用File方法将人脸分类器“lbpcascade\_frontalface.xml”载入到cascadeClassifier对象中。



图2.6.3 初始化人脸级联分类器

下面这个方法是在摄像头传帧时被调用到。

因为opencv要在横屏时才能得到较好的结果，那么可以先把竖屏时得到的图像顺时针旋转90度即

Core.rotate(mGray, gMatlin, Core.ROTATE\_90\_CLOCKWISE);

得到和横屏时相同的结果，然后再把得到识别绿框的图像逆时针旋转90度，即

Core.rotate(Matlin, mRgba, Core.ROTATE\_90\_COUNTERCLOCKWISE);

再输出就能做到竖屏时实现人脸检测了。



图2.6.4 人脸检测完整代码（1）



图2.6.5 人脸检测完整代码（2）

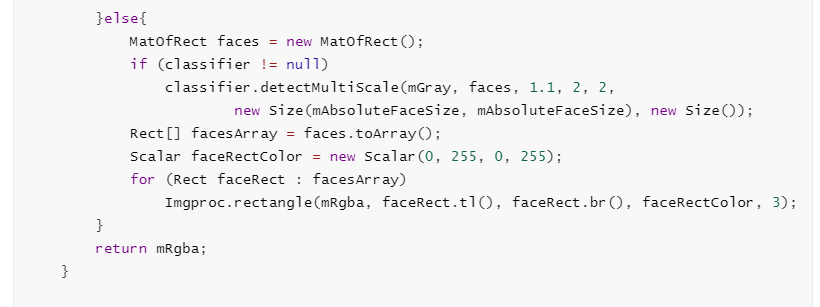


图2.6.6 人脸检测完整代码（3）



图2.6.7 人脸检测完整代码（4）

三、功能优化

OpenCv官方例子中，竖屏时Camera的预览图像总是会左旋90度并且不能全屏。

且它只能在横屏条件下进行人脸检测，同时只能用后置摄像头。

所以本项目中对这些方面进行了优化。

优化点为：增加了前后置切换，实现竖屏模式下能进行较好的人脸检测。



图3.1.1 优化前

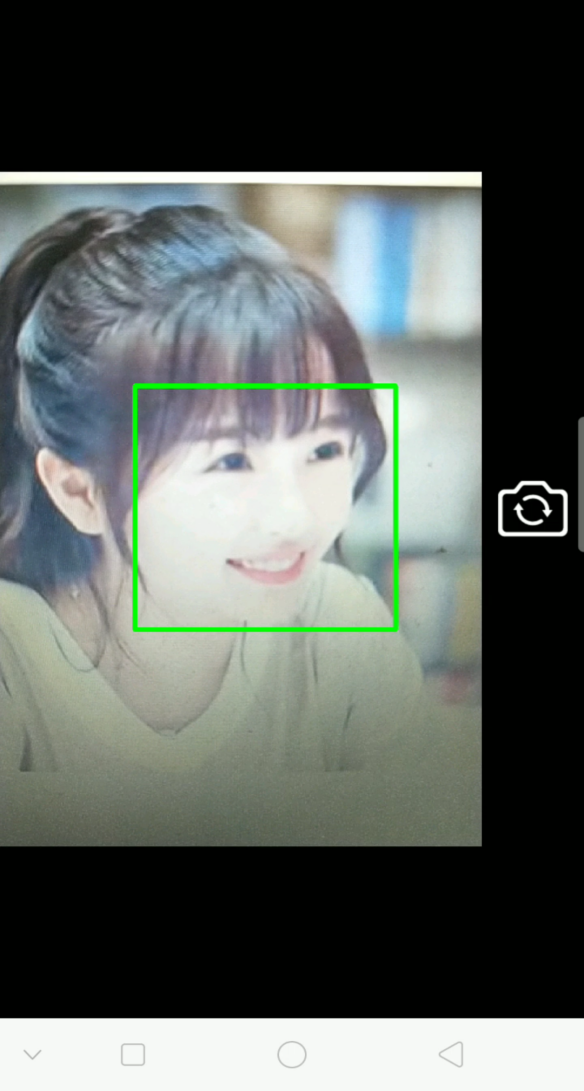


图3.1.2 优化后

竖屏不能人脸识别的问题，其实就是绿框和图片不能同时旋转所导致的。固改为旋转成横屏状态识别好，然后再整体旋转回来。

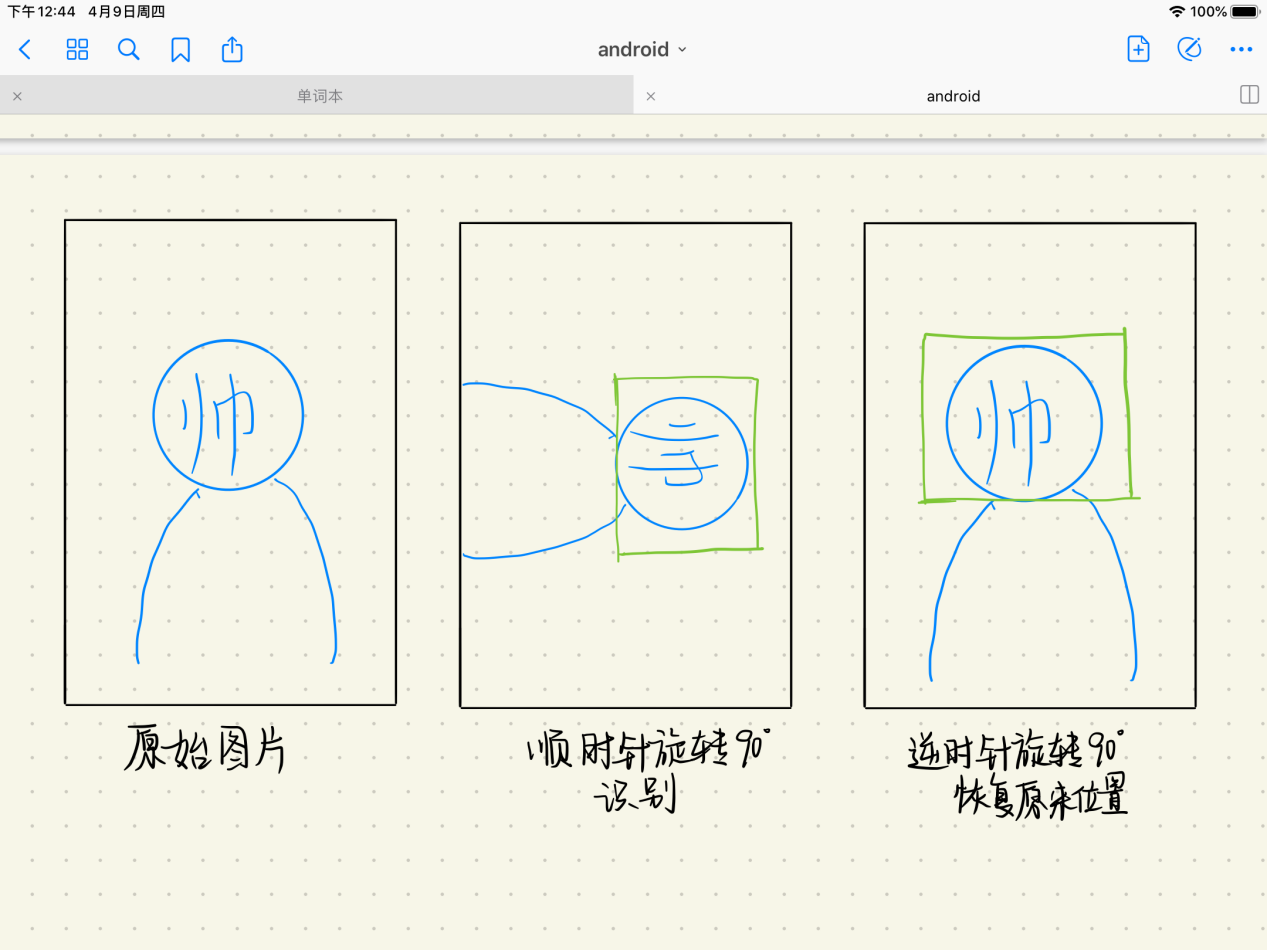


图3.1.3 优化思路

根据 rotation获取屏幕旋转角度，来判断是否竖屏还是横屏

int rotation = openCvCameraView.getDisplay().getRotation();

正常情况下有四种情况，即90、180、270、360，这个项目我们只取了90、180。



图3.1.4 获取旋转角度

接着修改CameraBridgeViewBase.java库文件，该文件位于当前工程\openCVLibrary330\java\org.openv\android文件夹下。

将这个protected void deliverAndDrawFrame(CvCameraViewFrame frame) 函数里

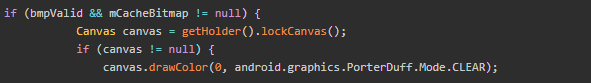


图3.1.5 原函数段

函数段后面把原方法替换成

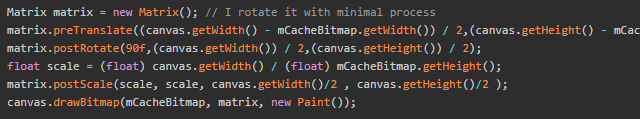


图3.1.6 新增函数段