**赵鹏程\_背景分割\_1月21日**

目录

[0 前言 1](#_Toc62204556)

[1 依赖软件列表介绍 1](#_Toc62204557)

[2 整合思路 1](#_Toc62204558)

[3 SDK打包 7](#_Toc62204559)

[3.1 思路 7](#_Toc62204560)

[3.2 通用步骤 7](#_Toc62204561)

[3.2.1 准备工作 7](#_Toc62204562)

[3.2.2 构建aar包 8](#_Toc62204563)

[3.2.3 aar包的引用 9](#_Toc62204564)

[3.2.4 使用aar包的代码 10](#_Toc62204565)

[4 接口与参数说明 11](#_Toc62204566)

# 0 前言

本文档针对**移动端。**

# 1 依赖软件列表介绍

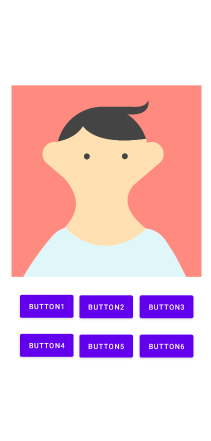
采用IDE版本号为4.1.1，具体依赖包见依赖文档（大家填写的在线文档）。

# 2 整合思路

简单的MVC模式：

在整个流程当中，基本的数据就是从摄像读取的视频流。

View层：在view层中用户进行操作，选出相应的模块，调用摄像头（前/后），基本的原型界面如下所示：



介绍预览

这一部分可以放置项目图标

各个部分模块文字说明等

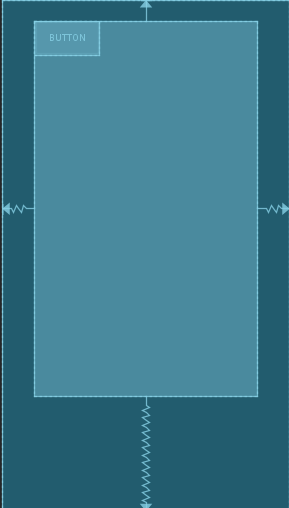
各个功能入口

每个button对应一个模块功能，点击模块跳转到对应的界面。layout布局的xml代码如下，每个人都在这个的入口界面的代码上进行添加，选择button作为你的功能入口，书写**你自己的activity**：

点击button跳转到新的activity界面，使用intent交互，这个就不再描述。（id不用改变，写好之后提供aar包和你写的对应activity的代码，单独模块自己测试完毕之后发送过来）

|  |
| --- |
| <?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=".DisplayMessageActivity">  <ImageView  android:id="@+id/imageView"  android:layout\_width="367dp"  android:layout\_height="662dp"  android:layout\_marginTop="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  tools:srcCompat="@tools:sample/avatars" />  <Button  android:id="@+id/button1"  android:layout\_width="104dp"  android:layout\_height="56dp"  android:text="Button1"  app:layout\_constraintBottom\_toTopOf="@+id/imageView"  app:layout\_constraintEnd\_toStartOf="@+id/imageView"  app:layout\_constraintHorizontal\_bias="0.745"  app:layout\_constraintStart\_toEndOf="@+id/imageView"  app:layout\_constraintTop\_toBottomOf="@+id/imageView"  app:layout\_constraintVertical\_bias="0.165" />  <Button  android:id="@+id/button4"  android:layout\_width="104dp"  android:layout\_height="56dp"  android:text="Button4"  app:layout\_constraintBottom\_toTopOf="@+id/imageView"  app:layout\_constraintEnd\_toStartOf="@+id/imageView"  app:layout\_constraintHorizontal\_bias="0.745"  app:layout\_constraintStart\_toEndOf="@+id/imageView"  app:layout\_constraintTop\_toBottomOf="@+id/imageView"  app:layout\_constraintVertical\_bias="0.061" />  <Button  android:id="@+id/button2"  android:layout\_width="104dp"  android:layout\_height="56dp"  android:text="Button2"  app:layout\_constraintBottom\_toTopOf="@+id/imageView"  app:layout\_constraintEnd\_toStartOf="@+id/imageView"  app:layout\_constraintHorizontal\_bias="0.501"  app:layout\_constraintStart\_toEndOf="@+id/imageView"  app:layout\_constraintTop\_toBottomOf="@+id/imageView"  app:layout\_constraintVertical\_bias="0.164" />  <Button  android:id="@+id/button5"  android:layout\_width="104dp"  android:layout\_height="56dp"  android:text="Button5"  app:layout\_constraintBottom\_toTopOf="@+id/imageView"  app:layout\_constraintEnd\_toStartOf="@+id/imageView"  app:layout\_constraintHorizontal\_bias="0.501"  app:layout\_constraintStart\_toEndOf="@+id/imageView"  app:layout\_constraintTop\_toBottomOf="@+id/imageView"  app:layout\_constraintVertical\_bias="0.059" />  <Button  android:id="@+id/button3"  android:layout\_width="104dp"  android:layout\_height="56dp"  android:text="Button3"  app:layout\_constraintBottom\_toTopOf="@+id/imageView"  app:layout\_constraintEnd\_toStartOf="@+id/imageView"  app:layout\_constraintHorizontal\_bias="0.254"  app:layout\_constraintStart\_toEndOf="@+id/imageView"  app:layout\_constraintTop\_toBottomOf="@+id/imageView"  app:layout\_constraintVertical\_bias="0.164" />  <Button  android:id="@+id/button6"  android:layout\_width="104dp"  android:layout\_height="56dp"  android:text="Button6"  app:layout\_constraintBottom\_toTopOf="@+id/imageView"  app:layout\_constraintEnd\_toStartOf="@+id/imageView"  app:layout\_constraintHorizontal\_bias="0.254"  app:layout\_constraintStart\_toEndOf="@+id/imageView"  app:layout\_constraintTop\_toBottomOf="@+id/imageView"  app:layout\_constraintVertical\_bias="0.059" />  </androidx.constraintlayout.widget.ConstraintLayout> |

**书写Activity的要求：**



使用原生控件，AS里面有的按钮、滑动条等。

设置Margin为16，下方放置功能按钮，左上角放置前后摄像头按钮（如果有的话）。

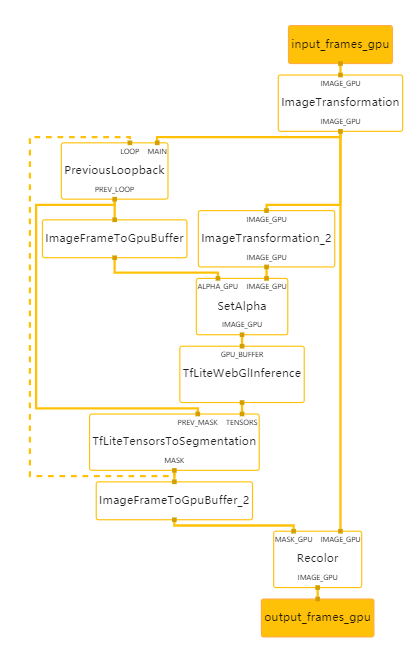
xml代码如下：

|  |
| --- |
| <?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">  <FrameLayout  android:layout\_width="318dp"  android:layout\_height="536dp"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="0.494"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.158">  <Button  android:id="@+id/button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Button" />  </FrameLayout>  </androidx.constraintlayout.widget.ConstraintLayout> |

**Controller层：**

这一部分是封闭的，就是你的aar包，按照下面aar的步骤封装好，在view层测试。

如果使用的是mediapipe，那么mediapipe的controller可以简要描述如下（根据graph图）：



顶部输入视频流，进行rgb处理，染色，通过tflite进行分割，最后输出相应的视频流。

**Model层：**

包括程序需要的数据信息，我们在这一部分所需要的就是从摄像头读取的数据。

（例如下面我在aar构建步骤的例子中，我将代码用aar构建完成之后，然后切换到不同的activity为参考）

总结一下流程如下：

1. 首先统一大家使用各个依赖dependencies版本，这一步决定了大家后面整合到一起的时候是否会有冲突，版本不兼容的问题等。
2. 按照步骤生成自己的sdk包。
3. 然后每个人自己创建项目**测试自己**的sdk是否能使用，并且生成相应的使用文档。
4. 最后进行整合和搭建，书写最后的文档，合并包。

# 3 SDK打包

## 3.1 思路

在Android Studio当中将项目打包成aar文件并且在其他项目中引用，主要目的是能在此基础上进行二次开发。在二次开发的过程当中，不需要开发人员了解mediapipe等框架的知识，根据所用指南，进行普通的app开发。

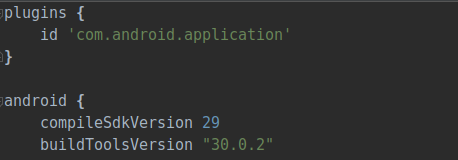
## 3.2 通用步骤

一般的jar包只包含代码文件，但是aar除了包含classes文件，同时包含资源文件（如图片等res文件），并且可以发布到远程或者本地maven库。

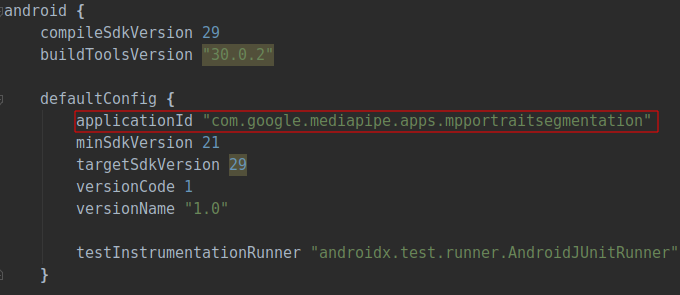
### 3.2.1 准备工作

aar包更像是一个可以调用的库，准备工作简单来说就是将一个完整的项目app“简化”成为一个库。

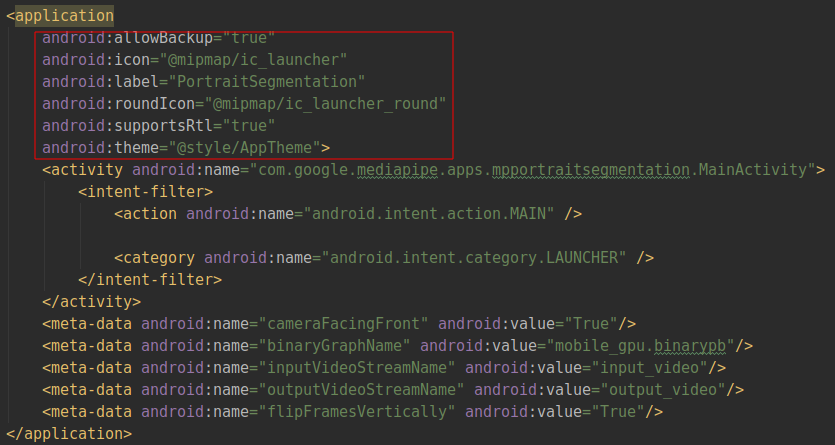
1. build.gradle(app)文件中，首先更改”apply plugin“的标识，将apply plugin: “com.android.application“改为apply plugin:“com.android.library“。



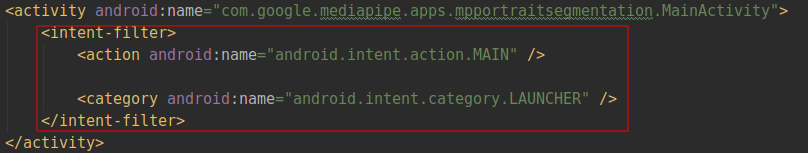
1. 依旧是在这个文件当中，将applicationId去掉。



1. 将AndroidManifest.xml中application中的各项属性删去（name、theme、lable等）

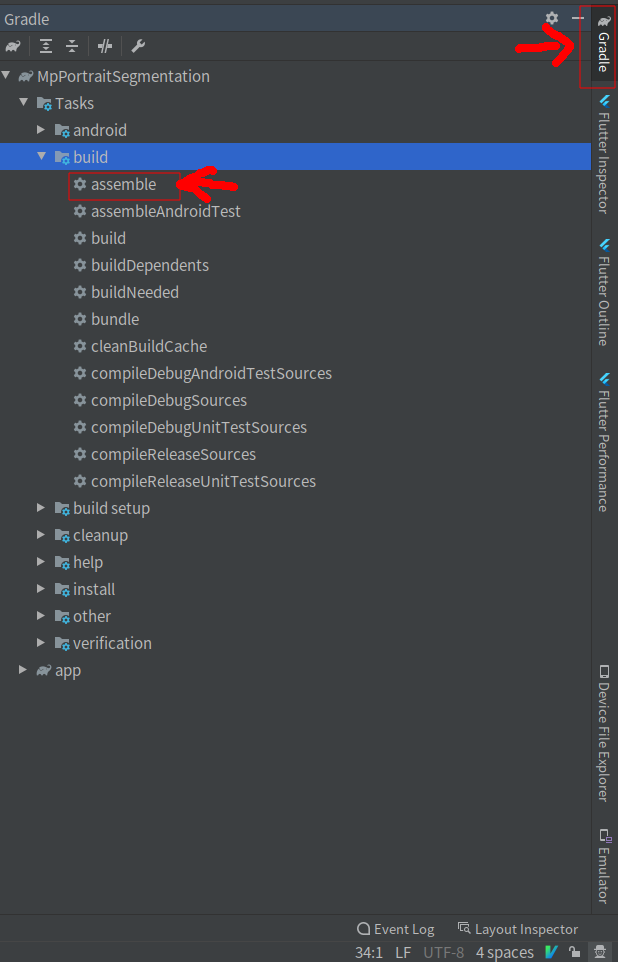


1. 同时也删除main入口



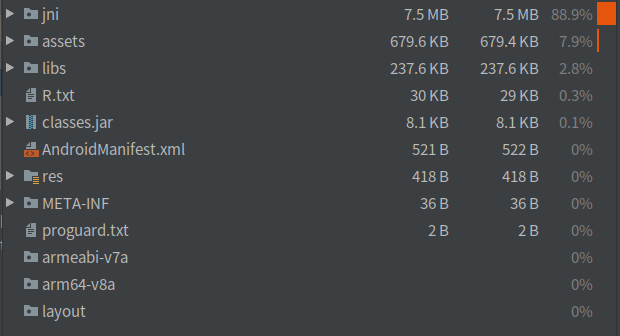
### 3.2.2 构建aar包

1. 点击右侧Gradle，选中assemble进行构建，构建完成之后在app/build/outputs/aar目录下生成相应的aar文件。



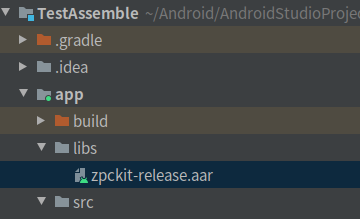
1. 双击aar文件，可以看到aar文件的文件结构（对于mediapipe项目而言，其打包了基本的classes

文件，res资产库，jni等所需opencv外部文件）：



### 3.2.3 aar包的引用

1. 创建一个新的安卓项目，将aar包放在libs（app/libs）文件目录



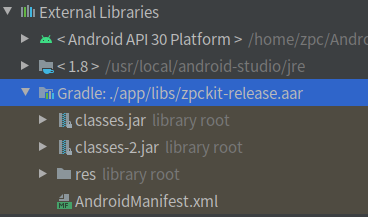
1. 在下build.gradle（app）中添加以下代码：

|  |
| --- |
| repositories {  flatDir {  dirs 'libs'  }  } |

1. 在dependencies中添加（在libs目录下引用所有的jar包和aar包）：

|  |
| --- |
| implementation fileTree(dir: 'libs', include: ['\*.jar', '\*.aar']) |

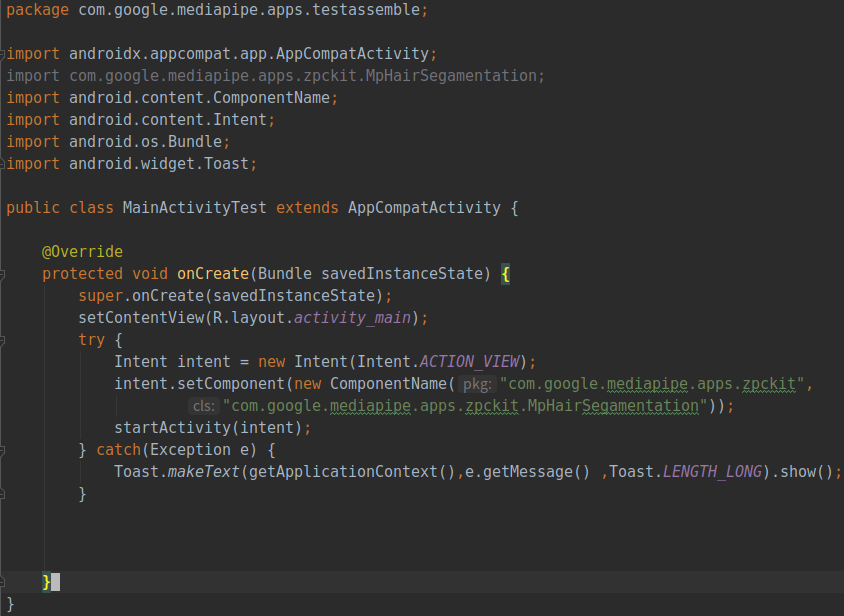
1. 之后运行sync，gradle一下，加载相应的aar文件(可以看到你的aar文件已经被加载进去了)：



### 3.2.4 使用aar包的代码

在aar包中已经包含了一个可以展示的layout，和普通的activity一样，它们之间使用intent来进行通信。

对于已经打包好的文件来说（以我的zpckit-release.aar为例子）：



和一般的intent之间的通信没什么区别，就不再介绍了，主要注意引入的时候路径要写对。 import和package部分的路径，还有就是setComponent里面的路径。

最后sync，运行可以看到正确调用。

# 4 接口与参数说明

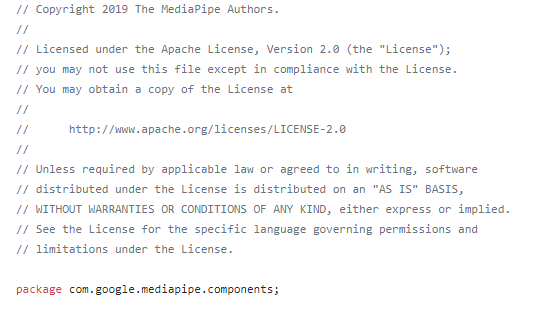
在书写Activity的同时，书写自己相关的代码与接口说明（给出关键函数与参数的接口和使用方式）：

对于交付的代码文件，需要保证你的代码抬头包含需要的license信息，代码的所有人签名，每个类需要有描述，每个函数需要表明对应的description，args，return（描述、参数、返回值）：

例如：（java代码类似）



抬头信息（license）：



所有者与编码格式：



在我的代码中MainActivity.java接口与参数说明如下：

类成员变量：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| BINARY\_GRAPH\_NAME | string | 存储graph图的地址 |
| INPUT\_VIDEO\_STREAM\_NAME | string | 输入视频流的名称 |
| OUTPUT\_VIDEO\_STREAM\_NAME | string | 输出视频流的名称 |
| CAMERA\_FACING | CameraHelper.CameraFacing | 调用前/后摄像头参数 |
| red\_progress | int | 初始rgb数值 |
| blue\_progress | int | 初始rgb数值 |
| green\_progress | int | 初始rgb数值 |
| processor | FrameProcessor | 帧处理器 |

* onCreate函数：

参数：savedInstanceState，Bundle类型。

返回值：空

描述：活动创建函数

函数变量ColorDisk

类型：ColorPickerView

描述：存储色盘对象

使用：调用onColorBack监听器函数，传入a（透明度），r，g，b三个通道色彩。

* onResume函数：

参数：/

返回值：空

描述：摄像头在暂停之后继续执行的代码

* onPause函数：

参数：/

返回值：空

描述：暂停摄像头

* onRequestPermissionsResult函数：

参数：int requestCode, String[] permissions, int[] grantResults

返回值：空

描述：查看摄像头权限是否打开

* setupPreviewDisplayView

参数：/

返回值：空

描述：设置输出视频流的尺寸，显示的大小

* startCamera()

参数：/

返回值：空

描述：开启摄像头

* changeCamera()

参数：View view（绑定button）

返回值：空

描述：切换摄像机（前/后）

* RGBHandler（内部类）
  + onWillFrame

参数：long timestamp（时间戳）

返回值：空

描述：读取rgb进入processor进行处理