人脸SDK接口说明文档

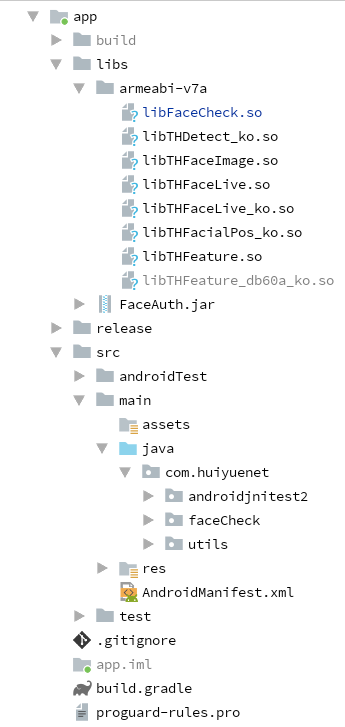
# 人脸SDK接口说明文档

## 人脸应用程序说明

### 人脸应用目录结构

本应用程序基于AndroidStudio开发，核心目录结构以及相关说明如下：

**目录结构：**



人脸示例程序源码目录，主要是演示人脸SDK各个接口（如：特征提取、活体检测）的使用方式、双目Camera调用流程、联网授权流程等核心部分

人脸示例程序的manifest文件，主要用于声明人脸示例程序所需的必要权限，如：联网权限、SD卡访问权限等

人脸核心算法库以及SDK文件，在集成人脸SDK到自己的应用中时，要确保所有的so库都被放置在正确的目录中，FaceAuth.jar被正常依赖

## 人脸SDK使用说明

### 人脸SDK接口介绍

#### FaceCheck类

##### Init()、InitFaceEngine()

**接口声明：**

private static native int **Init**(String read, String write, Context context)

private static native int **InitFaceEngine**(String read, String write);

**接口说明：**

**说明：**

这两个接口主要用于初始化、加载人脸算法，不建议单独调用，而是直接调用封装好的接口FaceCheck.init(context)来初始化人脸算法，只需要在程序启动时调用一次；

**注意：**

1. 调用这两个接口的时候需要传入两个路径，对于read目录 必须使用同示例程序中同样的路径，也就是：

readDir = m*Context*.getApplicationInfo().**nativeLibraryDir**;

对于write目录，可以是其他目录，但是要确保当前应用可以正常访问；

**2. 对于一些性能较差的设备，执行init()方法耗时较长，建议将该方法的调用放在子线程中进行；**

##### Release()、FaceEngineRelease()

**接口声明：**

private static native int **Release**()

private static native int **FaceEngineRelease**();

**接口说明：**

**说明：**

这两个接口主要用于释放算法相关资源，只需要在程序退出时调用一次；

**返回值：**

初始化成功返回>0的数，否则返回<0的错误码

##### CheckFace()

**接口声明：**

public static native int **CheckFace**(

short nChannelID, byte[] pImage, int bpp,

int nWidth, int nHeight, THFI\_FacePos[] pfps,

int nMaxFaceNums, int nSampleSize);

**接口说明：**

**说明：**

人脸检测接口，主要根据输入的图像给对应的人脸信息结构体赋值。

**参数：**

* nChannelID：（输入参数）检测通道，传入0即可
* pImage：（输入参数）BGR图像数据，**注：图像格式不能传错，必须是BGR格式**
* bpp：（输入参数）图像位数，24位、8位
* nWidth：（输入参数）图像宽度
* nHeight：（输入参数）图像高度
* pfps：（输出参数）人脸信息结构体数组，数组大小为检测到的人脸数量，结构体信息参考[THFI\_FacePos类](#_THFI_FacePos类)
* nMaxFaceNums：（输入参数）最大支持人脸检测数
* nSampleSize：（输入参数）人脸检测缩放比例，传入常规值360即可

**返回值：**

人脸检测成功，返回检测到的人脸数量，检测失败 返回<0的数

##### GetFeaturesSize()

**接口声明：**

public static native int **GetFeaturesSize**();

**接口说明：**

**说明：**

获取人脸特征长度。

**返回值：**

单个人脸特征长度，当前版本单个特征长度为2560

##### GetFeatures()

**接口声明：**

public static native int **GetFeatures**(

byte[] pImage, int nWidth, int nHeight,

int nChannel, THFI\_FacePos pfps, byte[] pFeature);

**接口说明：**

**说明：**

从图像中提取人脸特征。

**参数：**

* pImage：（输入参数）BGR图像数据，**注：图像格式不能传错，必须是BGR格式**
* nWidth：（输入参数）图像宽度
* nHeight：（输入参数）图像高度
* nChannelID：（输入参数）图像色彩通道，必须为3
* pfps：（输入参数）人脸信息结构体数组，该结构体由[人脸检测函数](#_CheckFace())得到，结构体信息参考[THFI\_FacePos类](#_THFI_FacePos类)
* pFeature：（输出参数）当前图像中提取到的特征数据

**返回值：**

特征提取成功 返回1，否则返回对应的错误码

##### FaceCompare()

**接口声明：**

public static native float **FaceCompare**(byte[] pFeature1, byte[] pFeature2);

**接口说明：**

**说明：**

比对两个人脸特征。

**参数：**

* pFeature1：第一个人脸特征
* pFeature2：第二个人脸特征

**返回值：**

两个特征的比对分数，当前版本比对通过阈值建议设置为0.65f，大于该值则表示比对通过

##### allocateFeatureMemory()

**接口声明：**

public static native int **allocateFeatureMemory**(int capacity);

**接口说明：**

**说明：**

（底层管理特征接口）分配一块空间用于保存已注册特征，已注册特征可以通过[addFeature()](#_addFeature())添加。

这部分空间在native层分配，所以访问起来效率更快一些，通过该方法分配的空间，可以使用addFeature()向这块空间添加特征，

当需要比对的时候，调用ver1N()，传入一个需要比对的特征即可实现1:N

**参数：**

* capacity：最大特征数量，目前允许的最大容量为20,0000个特征

**返回值：**

成功返回0，否则返回<0的数

##### addFeature()

**接口声明：**

public static native int **addFeature**(int index, byte[] feature);

**接口说明：**

**说明：**

（底层管理特征接口）向通过[allocateFeatureMemory()](#_allocateFeatureMemory())分配的空间中添加一个特征。

**参数：**

* index：添加特征的位置，如果被添加的位置已经有特征，则新特征会替换掉已有特征
* feature：被添加的特征

**返回值：**

成功返回0，否则返回<0的数

##### clearFeature()

**接口声明：**

public static native void **clearFeature**();

**接口说明：**

**说明：**

（底层管理特征接口）清空已保存的特征，**注：只是将特征清空，并不会释放保存特征的空间**。

**返回值：**

成功返回0，否则返回<0的数

##### ver1N()

**接口声明：**

public static native int **ver1N**(int matchNum, byte[] feature, int[] matchIndex, float[] matchScore);

**接口说明：**

**说明：**

（底层管理特征接口）特征1:N接口，通过输入单个特征，与已经保存的特征进行比对。

**参数：**

* matchNum：与已保存的特征中的前多少个特征进行比对，通常是已保存特征个数
* feature：需要比对的单个特征
* matchIndex：比对成功的index
* matchScore：比对成功的分数

**返回值：**

成功返回0，否则返回<0的数

##### ver1NN()

**接口声明：**

public static native int **ver1NN**(

int featureCount, byte[] featureList, byte[] featureSingle, int[] matchIndex, float[] matchScore);

**接口说明：**

**说明：**

特征1:N接口，直接输入单个特征和多个特征进行1:N功能。

注：在有些版本的SDK中，并没有将该接口添加到FaceCheck.java中，需要客户手动将以上声明添加到 FaceCheck.java 这个文件中；

**参数：**

* featureCount：与featureList中的前多少个特征进行比对，通常是featureList中保存的特征个数
* featureList：需要比对的N个特征列表
* featureSingle：需要比对的单个特征
* matchIndex：比对成功的index
* matchScore：比对成功的分数

**返回值：**

成功返回0，否则返回<0的数

##### DetectLive2()

**接口声明：**

public static native int **DetectLive2**(

byte[] bgrImageData, byte[] irImageData,

THFI\_FacePos[] facePosRgb, THFI\_FacePos[] facePosIr,

int width, int height, int threshold, float[] score);

**接口说明：**

**说明：**

活体检测接口，根据输入的图像，得出该图像的活体分数。

**参数：**

* bgrImageData：（输入参数）彩色人脸图像对应的BGR数据，**注：图像格式不能传错，必须是BGR格式**
* irImageData：（输入参数）红外人脸图像对应的BGR数据，**注：图像格式不能传错，必须是BGR格式**
* facePosRgb：（输入参数）彩色人脸图像对应的人脸信息结构体，该结构体由[人脸检测函数](#_CheckFace())得到，结构体信息参考[THFI\_FacePos类](#_THFI_FacePos类)
* facePosIr：（输入参数）红外人脸图像对应的人脸信息结构体，该结构体由[人脸检测函数](#_CheckFace())得到，结构体信息参考[THFI\_FacePos类](#_THFI_FacePos类)
* width：（输入参数）图像宽度
* height：（输入参数）图像高度
* threshold：（输入参数）活体检测阈值，传入50即可
* score：（输出参数）彩色图像和红外图像的活体分数，score[0]为彩色人脸图像活体分数，score[1]为红外人脸图像活体分数

**返回值：**

成功返回0，否则返回<0的数

**注意：**

如果要将当前SDK用于单个摄像头设备，调用活体检测接口时，ir相关的参数（irImageData、facePosIr）都传入null即可，同时活体分数数组score只有score[0]有效

##### getPixelsBGR()

**接口声明：**

public static native int **getPixelsBGR**(byte[] dest, byte[] src, int width, int height);

**接口说明：**

**说明：**

将RGB格式的图像数据转换为BGR格式。

**参数：**

* dest：转换后的BGR格式的数据
* src：需要被转换的RGB数据
* width：图像宽度
* height：图像高度

**返回值：**

成功返回0，否则返回<0的数

##### GetSDKVersionInfo()

**接口声明：**

public static native String **GetSDKVersionInfo**();

**接口说明：**

**说明：**

获取当前SDK版本信息。

**返回值：**

返回当前SDK版本信息

#### THFI\_FacePos类

该类包含以下成员：

public RECT **rcFace**; // 人脸矩形框

public POINT **ptLeftEye**; // 左眼中心点坐标

public POINT **ptRightEye**; // 右眼中心点坐标

public POINT **ptMouth**; // 嘴中心坐标

public POINT **ptNose**; // 鼻子中心坐标

public FaceAngle **fAngle**; // 人脸角度

public int **nQuality**; // 人脸图像质量（暂时没用）

public byte[] **pFacialData**; // 人脸核心数据信息

### 人脸SDK注意细节

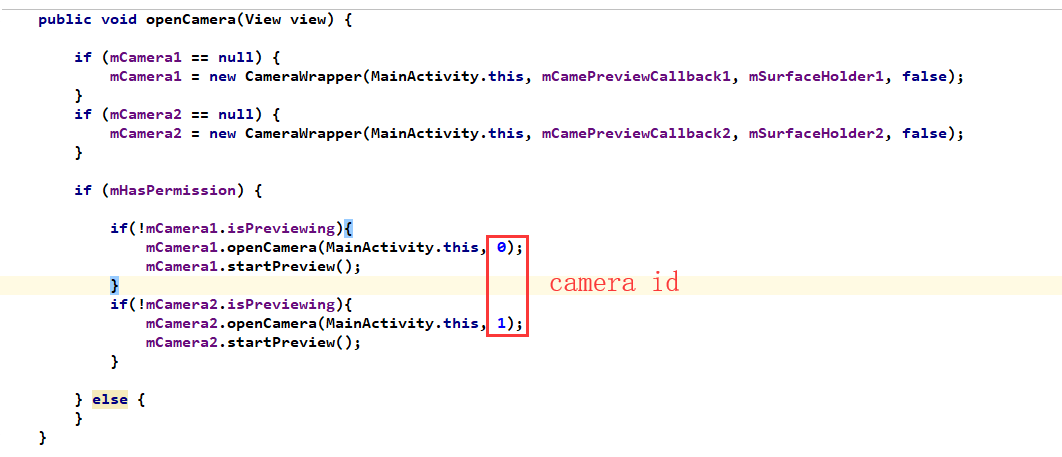
#### 账号填写

示例代码中的授权部分，需要填写自己的账号（MainActivity.java中）：



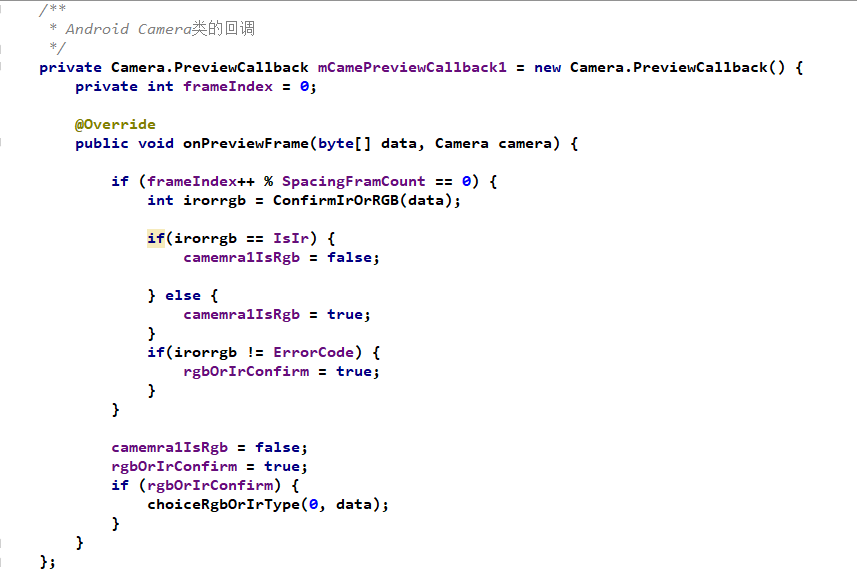
#### Camera ID处理

人脸示例应用默认打开CameraID为0和1的camera，如果需要修改，请在下图对应的代码处进行修改

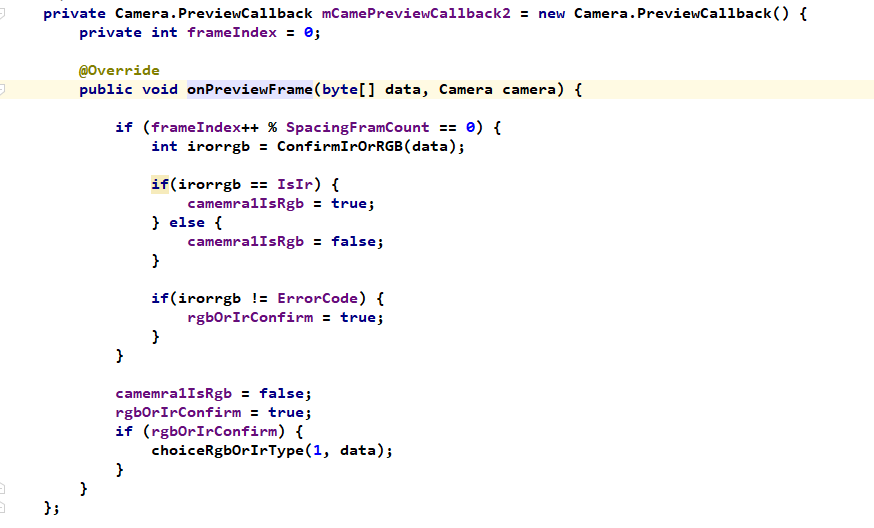


#### RGB/IR 数据处理（重要）

双目摄像头需要对RGB数据和IR数据进行各自对应的处理，对于外接USB双目摄像头来说，Camera的数据回调接口中既有可能是RGB数据，又有可能是IR数据，因此需要加一些判断逻辑，用于判断当前数据是IR还是RGB，设备不同，逻辑也不同，客户需要根据自己的设备进行修改，具体逻辑在每个Camera的回调接口中，如下图所示：



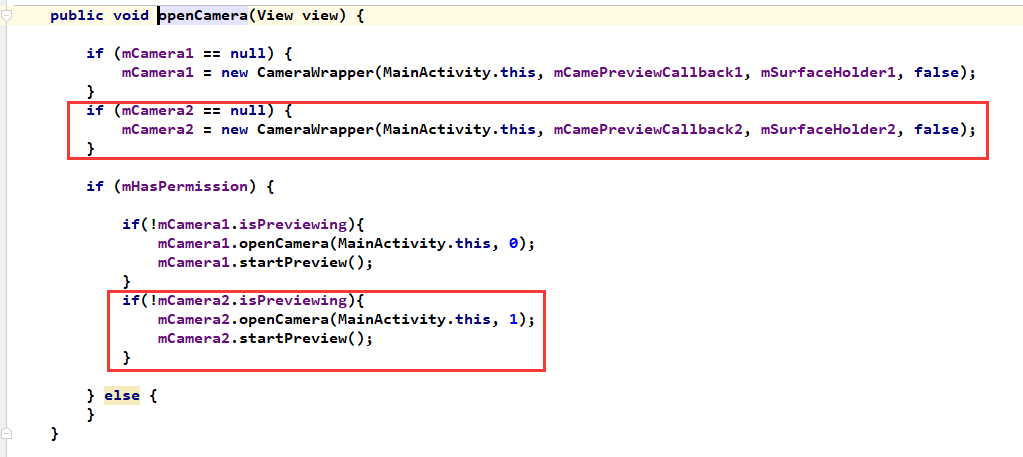
主要靠camera1IsRgb这个变量来控制Camera1是否为RGB数据，通常是应该经过一些针对自己设备的判断（*比如这里的onPreviewFrame中第一个if中的逻辑*）然后对camera1IsRgb这个变量赋上正确的值。由于设备不同，导致判断逻辑不同，这里将camera1IsRgb固定赋值为false，**开发人员在集成时需要改成适合自己设备的逻辑**。

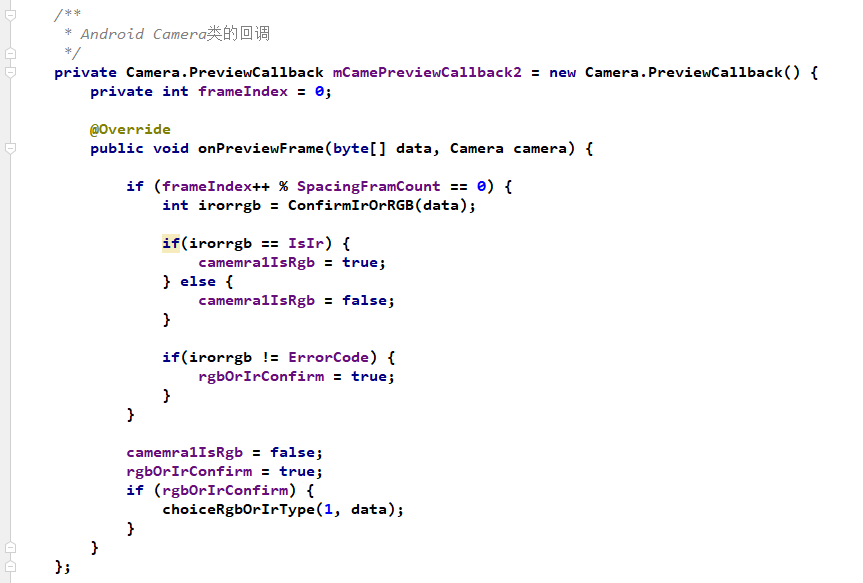


#### 双摄改单摄

如果要将双目示例Demo用在只有一个Camera的设备上，需要做以下修改，否则无法正常使用：

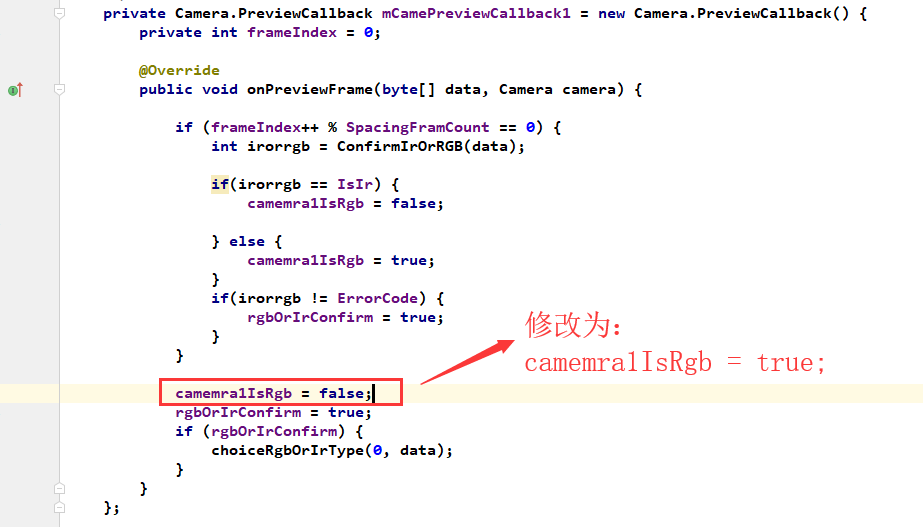
1. 将camera2相关的代码都注释掉，同时需要将camera1的cameraid配置成自己设备对应的ID值，需要注释掉的地方（注：以下代码都位于MainActivity.java中）：







1. 修改RGB/IR数据判断处理：

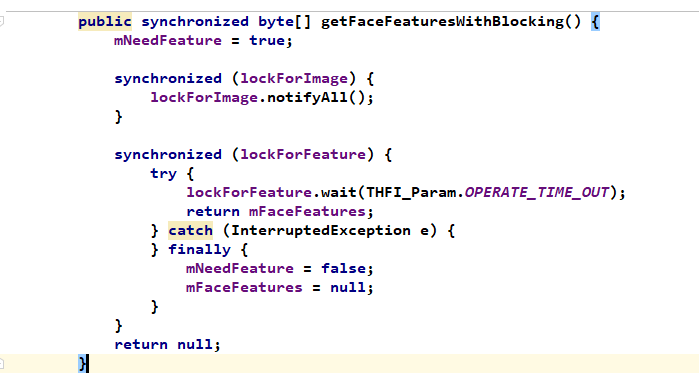


1. 修改人脸检测线程中的逻辑，如下图所示：



#### 超时配置

抽取特征（建模、比对的时候需要用到）时，在LiveCheckThread中抽取，抽取特征有一个超时时间，超过该时间 还未抽取成功（比如：活体分数不够、人脸图像质量较差等原因），则会返回null，超时时间通过 **THFI\_Param.OPERATE\_TIME\_OUT** 来配置



#### 特征管理

示例程序中特征操作是通过SDK提供的接口进行操作，也就是特征数据保存在底层，而不是应用层，如果需要应用层自己保存、管理特征，那么标记有 底层管理特征接口 的那几个接口（allocateFeatureMemory、addFeature、clearFeature、ver1N）都不需要调用。只需要在比对的时候调用 [ver1NN](#_ver1NN()) 即可。

## 结尾