

# Android端opensdk人体骨骼关键点计数集成说明文档

---

一、Android studio配置工程

二、SDK调用步骤，功能实现

## 2.1 SDK初始化

接口描述：

初始化接口：

参数说明：

返回值：

具体代码示例如下：

## 2.2 人体骨骼关键点计数API使用

### 2.2.1 创建算法实例

算法实例化接口：

参数说明：

返回值：

具体代码示例如下：

### 2.2.2 单个能力license鉴权

接口描述：

license证书验签接口：

获取证书路径(具体实现可以参考demo)：

参数说明：

返回值：

具体代码示例如下：

### 2.2.3 传入对应检测能力的model模型文件及dat文件

接口描述：

传入文件接口：

参数说明：

返回值：

具体代码示例如下：

#### 2.2.4-1 目标关键点检测及计数返回（重要）

接口描述：

接口示例：

参数说明：

返回值：

调用样例代码如下：

#### 2.2.4-2 目标关键点检测及计数返回（重要）

接口描述：

接口示例：

参数说明：

返回值：

调用样例代码如下：

#### 2.2.5 算法销毁

接口描述：

接口示例：

参数说明：

返回值：

具体代码示例如下：

#### 2.2.6 负反馈说明

负反馈数据获取：

返回数据说明：

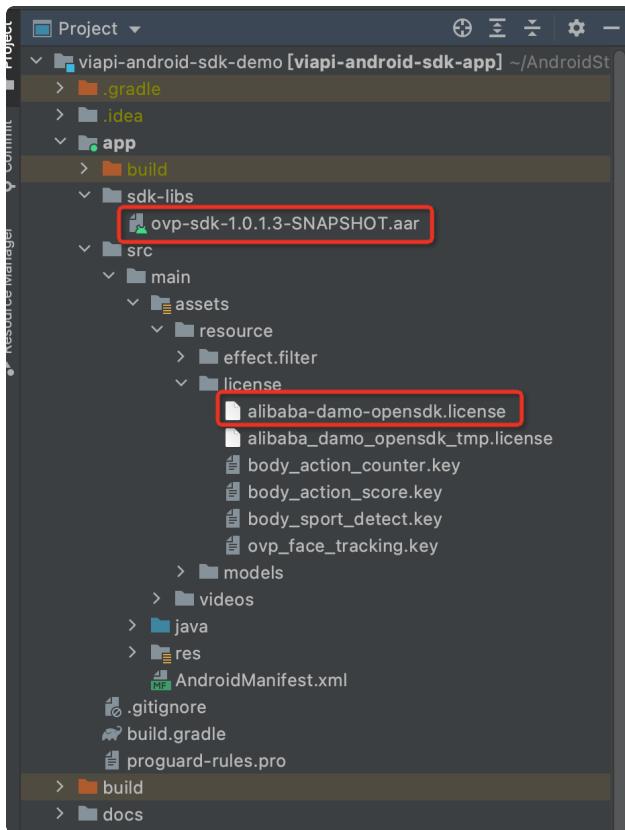
### 三、支持的系统和硬件版本

### 四、注意事项

附 离线鉴权错误码定义

## 一、Android studio配置工程

1、获取相关资源压缩包（由官网线上购买申请或阿里云相关人员提供下载链接）后，解压压缩包，可看到如下资源文件，demo示例工程、支持相关能力的aar及支持相关能力的license文件。如下图：



注意：alibaba-damo-opensdk.license为正式证书（官网下载获取的都是正式证书），\_tmp结尾的为临时证书一般线下提供，**临时证书不能改名，正式license可以改名字，但是不能与tmp license重名**。两个证书只需要调用其中一个进行鉴权就可以。

## 二、SDK调用步骤，功能实现

### 2.1 SDK初始化

#### 接口描述：

算法API使用前先调用SDK初始化接口，初始化之后，各功能才可以正常使用，否则会引起鉴权等异常，初始化建议放在app进程启动时Application onCreate中进行。

#### 初始化接口：

```
1 VIAPICreateApi.getInstance().getVIAPISdkCore().init(Context context, boolean isDebug);
```

## 参数说明：

Context context 应用上下文。

boolean isDebug SDK调试开关。

## 返回值：

int类型，返回0为初始化成功，其它返回为初始化失败。

## 具体代码示例如下：

```
1  private void initSDK() {  
2      int status = VIAPICreateApi.getInstance().getVIAPISdkCore().init(this,  
3          false);  
4      if (status != 0) {  
5          Toast.makeText(this, VIAPIStatusCode.getErrorMsg(status),  
6              LENGTH_LONG).show();  
7      } else {  
8          Toast.makeText(this, "初始化成功!", LENGTH_LONG).show();  
9      }  
10 }
```

## 2.2人体骨骼关键点计数API使用

### 2.2.1 创建算法实例

OVSportPoseDetect：在需要用到运动骨骼点检测算法的地方，创建OVSportPoseDetect骨骼检测实例，OVSportPoseDetect是视频实时人像骨骼点检测API接口对象，通过此对象可以完成camera实时视频流运动骨骼点检测能力的使用。

## 算法实例化接口：

```
1 //骨骼点检测  
2 OVSportPoseDetect mSportPoseDetect = new OVSportPoseDetect();
```

## 参数说明：

无

返回值:

无

具体代码示例如下:

Plain Text | 复制代码

```
1 private final OVSportPoseDetect mSportPoseDetect;  
2 mSportPoseDetect = new OVSportPoseDetect();
```

## 2.2.2 单个能力license鉴权

接口描述:

license证书验签接口，验签通过后才能成功调用算法。

license证书验签接口:

Plain Text | 复制代码

```
1 mSportPoseDetect.nativeCheckLicense(String licensePath);
```

获取证书路径(具体实现可以参考demo):

Plain Text | 复制代码

```
1 String licensePath = AssetsProvider.getReleaseLicenseFilePath();
```

参数说明:

String licensePath 传入全局证书license路径或自定义的单个能力license路径。

注：如果所有能力使用同一个全局证书默认传入全局证书路径即可，如果接入方有针对此能力的单独证书，则需要传入单独的证书文件的绝对路径

返回值:

int类型，返回0为验签成功，其它返回为验签失败。

具体代码示例如下：

```
1 int errorCode = mSportPoseDetect.nativeCheckLicense(licensePath);
```

### 2.2.3 传入对应检测能力的model模型文件及dat文件

接口描述：

传入算法识别所需的model及dat文件。

传入文件接口：

```
1 mSportPoseDetect.nativePoseCreateHandle(bodySportModelsPath, bodySportDatasP  
ath);
```

参数说明：

String bodySportModelsPath 传入算法的模型文件路径，需具体到文件名，传绝对路径。

String bodySportDatasPath 传入算法的dat文件路径，需具体到文件名，传绝对路径。

返回值：

int类型，返回0为验签成功，其它返回为验签失败。

具体代码示例如下：

```
1 int status = mSportPoseDetect.nativePoseCreateHandle(bodySportModelsPath, b  
odySportDatasPath);
```

### 2.2.4-1 目标关键点检测及计数返回（重要）

## 接口描述:

该方法为处理目标关键点检测及计数的接口，传入Buffer数据，获得运动骨骼关键点及计数结果返回数据。

## 接口示例:

```
Plain Text | 复制代码
1 mSportPoseDetect.nativeStartPoseCount(ByteBuffer outBuffer, int formatType,
2                                         int dataType, int width,
3                                         int height, int step, int angle, BodyTrackInfo bodyTrackInfo,
4                                         OVCountResult result);
```

## 参数说明:

- outBuffer: 传入算法的ByteBuffer数据。
- formatType: 传入outBuffer的数据格式，如buffer数据，则传入 OSImageFormat.OV\_IMG\_FMT\_RGBA.formatType。
- dataType: 传入outBuffer的数据类型。
- width: 传入数据的宽。
- height: 传入数据的高。
- step: 算法的步长，例RGBA为4通道，步长\*4。
- angle: 传给算法的角度，该角度为帧图转正需要的角度，如图是正向传入的则传值为0。
- bodyTrackInfo: 骨骼点数据返回。trackObject.getDetection().getRect()可获取检测范围矩形框的4点坐标。trackObject.getLandmarks().getPoints()  
可获取算法返回的所有14个骨骼检测点。当骨骼点score得分大于0，两点之间即可连线。连线规则为(4, 3) (3, 2) (2, 1) (1, 0) (1, 5) (5, 6) (6, 7) (0, 1) (1, 14)  
(14, 8) (8, 9) (9, 10) (14, 11) (11, 12) (12, 13).
- result: 计数结果返回。需创建计数实例OVCountResult mCountResult = new OVCountResult(), 作为参数传进去，返回结果mCountResult.getTotalNum()，即为计数结果。

## 返回值:

int类型，返回0为图像分割算法处理成功，其它返回为图像分割算法处理失败。

## 调用样例代码如下:

```
int status = mSportPoseDetect.nativeStartPoseCount(outBuffer,
OSImageFormat.OVP_IMG_FMT_RGBA.formatType, OSDDataType.OVP_UINT8.intType,
size.getWidth(),
size.getHeight(), size.getWidth() * 4, 0, bodyTrackInfo, result);
```

注意：算法内部没有对内存进行处理，输出buffer需提前申请内存空间，初始化格式为：

```
Plain Text | 复制代码
```

```
1 outBuffer = ByteBuffer.allocateDirect(textureWidth * textureHeight * 4);
```

## 2.2.4-2 目标关键点检测及计数返回（重要）

接口描述：

该方法为处理目标关键点检测及计数的接口，传入camera的原始nv21数据，获得运动骨骼关键点及计数结果返回数据。

接口示例：

```
Plain Text | 复制代码
```

```
1 mSportPoseDetect.nativeStartPoseCountNv21(byte[] yuv420sp, int width,
2                                     int height, int step,int cameraFace , int angle,BodyTrac
kInfo bodyTrackInfo, OVCountResult result);
```

参数说明：

- yuv420sp：传入算法的byte[]数组。
- width：传入数据的宽。
- height：传入数据的高。
- step：算法的步长，例相机返回数据为单通道，步长\*1。
- cameraFace：摄像头前后置1为前置，0为后置。
- angle：传给算法的角度，该角度为帧图转正需要的角度，如图是正向传入的则传值为0。
- bodyTrackInfo：骨骼点数据返回。trackObject.getDetection().getRect()可获取检测范围矩形框的4点坐标。trackObject.getLandmarks().getPoints()

可获取算法返回的所有14个骨骼检测点。当骨骼点score得分大于0，两点之间即可连线。连线规则

为(4, 3) (3, 2) (2, 1) (1, 0) (1, 5) (5, 6) (6, 7) (0, 1) (1, 14)

(14, 8) (8, 9) (9, 10) (14, 11) (11, 12) (12, 13).

- result: 计数结果返回。需创建计数实例OVCountResult mCountResult = new OVCountResult(), 作为参数传进去,返回结果mCountResult.getTotalNum(),即为计数结果。

#### 返回值:

int类型, 返回0为图像分割算法处理成功, 其它返回为图像分割算法处理失败。

#### 调用样例代码如下:

```
int status = mSportPoseDetect.nativeStartPoseCount(yuv420sp, size.getWidth(),
size.getHeight(), size.getWidth(), 1, 0, bodyTrackInfo, result);
```

注意: 算法内部没有对内存进行处理, 输出buffer需提前申请内存空间, 初始化格式为:

```
1 outBuffer = ByteBuffer.allocateDirect(textureWidth * textureHeight * 4);
```

## 2.2.5 算法销毁

#### 接口描述:

在不需要用到算法的地方进行算法销毁。

#### 接口示例:

```
1 mSportPoseDetect.nativePoseDestroyHandle();
```

#### 参数说明:

无

#### 返回值:

int类型, 返回0为删除算法相关成功, 其它返回为删除算法相关处理失败。

具体代码示例如下：

```
Plain Text | 复制代码  
1 int status = mSportPoseDetect.nativePoseDestroyHandle();
```

## 2.2.6 负反馈说明

负反馈数据获取：

```
int[] countResult = mCountResult.getErrorDetails();
```

返回数据说明：

error\_details[20]数据说明

error\_details[0]: 用来指示检测到的关键点质量是否能够支撑计数，

0: OK

1: NG, 关键点缺失严重无法进行计数分析

error\_details[1]: 动作幅度是否不够

0: 正常,

1: 动作幅度不足

error\_details[2]: 指示动作的第几个状态有问题

-1: 一切正常

对照feedback json, 令m=error\_details[2], 若m>-1, 则feedback\_info[m]出现问题。

error\_details[n] (n={4,5,6,...,19}) : 问题详情

若m = error\_details[2]中feedback\_info[m]出现问题, 则具体出问题的信息参考

feedback\_info[m].details[n-4]

error\_details[3]: 暂时忽略

注意：臀桥、跳绳、卷腹、腰绕环只需判断error\_details[0]、error\_details[1]即可

## 三、支持的系统和硬件版本

1、硬件要求：要求设备上有相机模块,陀螺仪模块

2、CPU架构：armeabi-v7a、arm64-v8a

3、系统：最低支持 Android 4.0 (API Level 14) 需要开发者通过minSdkVersion来保证支持系统的检测

## 四、注意事项

- 1、 viapi-android-sdk的 minSdkVersion为 14。
- 2、 demo工程Android Studio 3.4 及以上， Open GLES 2.0 及以上。
- 3、证书分为临时证书以\_tmp结尾和正式证书。临时证书一般作为调试用， 正式证书一般作为发布用。  
根据证书的不同代码需要做相应调整， 全局搜索mIsTmpLicense字段， 如果是\_tmp证书， 需把该字段改为true， 正式证书的话把该字段改为false。

## 附 离线鉴权错误码定义

- 2011 license没有初始化直接调用API接口。
- 2012 当前的license与调用app不是绑定关系， license用在其他app中使用。
- 2013 license无效。
- 2014 license授权时间过期。
- 2015 此license中不包含调用的算法能力（未购买此能力）。
- 2016 bundle id获取失败。
- 2017 临时license时间校验失败。