

Android端opensdk人体骨骼关键点集成说明文档

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: 针对所有能力生效

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

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 设置检测间隔

接口描述:

设置检测间隔接口示例:

参数说明:

返回值:

具体代码示例如下:

2.2.5 设置最大检测目标数

接口描述:

设置最大检测目标数:

参数说明:

返回值:

具体代码示例如下:

2.2.6 目标关键点检测 (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:

接口描述:

接口示例:

参数说明:

返回值:

调用样例代码如下:

2.2.7 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:

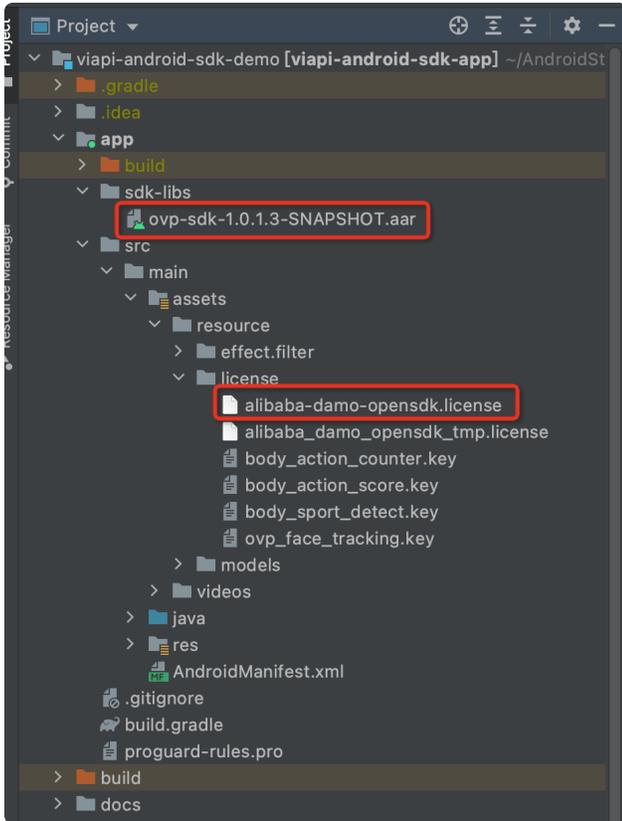
三、支持的系统和硬件版本

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：

```
Plain Text | 复制代码  
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：

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

2.2 人体骨骼关键点API使用

2.2.1 创建算法实例

OVBODYTRACK：在需要用到运动骨骼点检测算法的地方，创建OVBODYTRACK骨骼检测实例，OVBODYTRACK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：

```
Plain Text | 复制代码
1 //骨骼点检测
2 OVBodyTrack mBodyTrack = new OVBodyTrack();
```

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 OVBodyTrack mBodyTrack;
2 mBodyTrack = new OVBodyTrack();
```

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 mBodyTrack.nativeCheckLicense(String licensePath);
```

获取证书路径: 针对所有能力生效

```
Plain Text | 复制代码
1 String licensePath = VIAPICreateApi.getInstance().getVIAPISdkCore().getLicenseFilePath();
```

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（参考Demo）

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：

```
Plain Text | 复制代码  
1 int errorCode = mBodyTrack.nativeCheckLicense(licensePath);
```

2.2.3 传入对应检测能力的model模型文件

接口描述：

传入算法识别所需的model。

传入文件接口：

```
Plain Text | 复制代码  
1 mBodyTrack.nativeBodyTrackCreateHandle(bodySportModelsPath);
```

参数说明：

String modelsPath 传入算法的模型文件路径，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 = mBodyTrack.nativeBodyTrackCreateHandle(modelsPath);
```

2.2.4 设置检测间隔

接口描述:

设置算法检测的间隔

设置检测间隔接口示例:

```
1 mBodyTrack.bodyTrackSetDetectInterval(int intervalCode);
```

参数说明:

int intervalCode 有0、1两个int值供设置，0为每帧都检测，1为按算法策略跳帧检测。

返回值:

int类型，返回0为算法初始化成功，其它返回为算法初始化失败。

具体代码示例如下:

```
1 int status = mBodyTrack.bodyTrackSetDetectInterval(0);
```

2.2.5 设置最大检测目标数

接口描述:

设置算法可检测的最大目标数（目前只支持单人检测）

设置最大检测目标数:

```
1 mBodyTrack.bodyTrackSetDetectMaxObjects(int detectMaxNum);
```

参数说明:

int detectMaxNum 根据设置的数量, 决定算法的检测目标的个数。

返回值:

int类型, 返回0为算法初始化成功, 其它返回为算法初始化失败。

具体代码示例如下:

```
1 int status = mBodyTrack.bodyTrackSetDetectMaxObjects(1);
```

2.2.6 目标关键点检测 (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RGBA数据, 获得运动骨骼关键点返回数据。

接口示例:

```
1 mBodyTrack.startBodyTrack(ByteBuffer outBuffer, int formatType, int DataTyp  
e, int width,  
2 int height, int step, int angle, OVBodyTrackInfo bodyTrac  
kInfo);
```

参数说明:

- outBuffer: 传入算法的ByteBuffer数据。
- formatType: 传入outBuffer的数据格式, 如buffer传入的是RGBA数据, 则传入OVImageFormat.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)。

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 = mBodyTrack.startBodyTrack(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:

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

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, 获得运动骨骼关键点返回数据。

接口示例:

```
1  mBodyTrack.startBodyTrackNv21(byte[] yuv420sp, int width, int height, int
row_stride, int cameraFace, int rotate, OVPBodyTrackInfo trackInfo);
```

参数说明:

- yuv420sp: 传入算法的相机回调数据。
- width: 传入数据的宽。
- height: 传入数据的高。
- row_stride: 算法的步长, 例nv21单通道通道, 步长为width。
- rotate: 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)。

返回值:

int类型, 返回0为算法处理成功, 其它返回为算法处理失败。

调用样例代码如下:

```
int status = mBodyTrack.startBodyTrackNv21(outBuffer,
OSImageFormat.OVP_IMG_FMT_RGBA.formatType, OSDDataType.OVP_UINT8.intType,
size.getWidth(),
size.getHeight(), size.getWidth() * 4, 0, bodyTrackInfo, result);
```

2.2.7 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:

```

▼ Plain Text | 复制代码
1 mBodyTrack.nativeBodyTrackDestroy();
```

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 = mBodyTrack.nativeBodyTrackDestroy();
```

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。

