人脸识别 Android SDK
开发指南 v2.4.2

上海阅面网络科技有限公司

目录

前言

■ SDK 简介

本 SDK 开发指南指导您如何安装和配置开发环境，如何通过调用 SDK 提供的接口函数(API)进行二次开发与系统集成。用户按照要求调用 SDK 提供的 API 即可实现使用人脸检测/跟踪、活体识别、人脸识别等服务的目的。

SDK 目录结构如下，

face_sdk_readsense

├── doc //SDK 开发指南

├── Face-Demo // Android 示例工程,帮助用户了解如何使用 SDK

├── jniLibs //so 库

├── libs //类库

■ 读者对象

本文档（本指南）主要适用于以下工程师：

✧ 软件工程师

✧ 硬件工程师

✧ 技术支持工程师

■ 修订记录

版本号	变更说明	日期	备注
V2.0	文档结构重新整理 大幅更新文档内容	2018/7/29	
V2.1	人证比对阈值修改为 75	2018/8/2	
V2.2	getFaceQuality 人脸质量阈值设定为 6	2018/8/8	
V2.3	新增红外活体检测接口和根据特征注册人脸接口	2018/8/24	
V2.4	增加双目活体检测流程	2018/9/5	
V2.4.1	分离人脸和人证特征比对	2018/11/20	
V2.4.2	更新激活返回码含义	2018/11/21	

开发前准备

- 获取 SDK

  1)请联系阅面科技商务处获取，邮箱：business@readsense.cn

  2)测试版或正式版 SDK，申请前都需提供应用的包名

- 激活 SDK

- 若您获得是测试版 SDK，无需激活。
- 若您获得是正式版 SDK，则在初始化时（参考“初始化检测器”章节）需提供 appid 和 appsecret 激活 SDK。

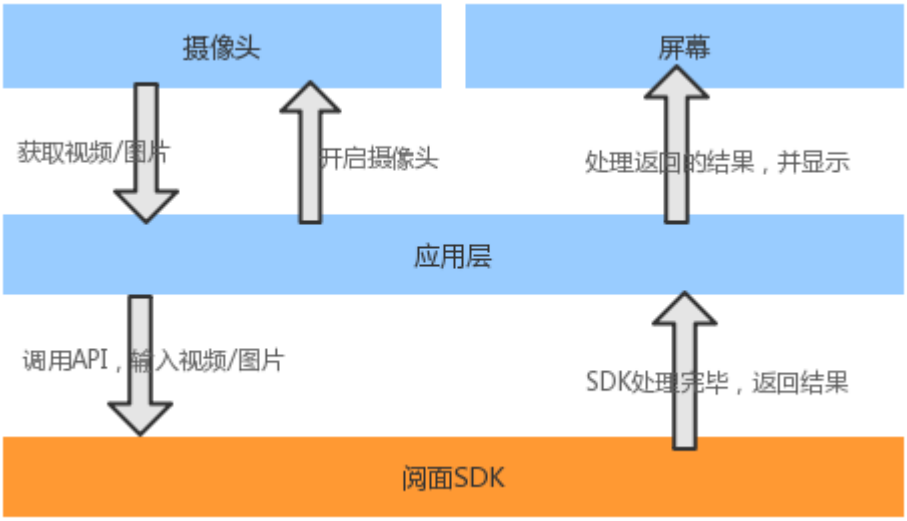
  注： appid 和 appsecret 在提供给您的正式版 SDK 的 demo 中的 SenseConfig 文件中获取 (请保管好自己的密钥，避免不必要的损失)。首次使用 SDK 正式版需要联网认证一下，之后不依赖网络。

- 运行环境

  - 推荐内存：2G
  - 推荐 CPU：1.8GHz，armeabi-v7a, arm64-v8a
  - Android 版本 5.0 及以上

- 调用关系

  用户的应用程序调用阅面 SDK 提供的 API，就可以直接使用人脸检测/追踪、识别等功能。下图完整展示了 SDK 的使用过程。



快速入门

1 添加依赖库

1) 复制 libreadface.jar 到工程 libs 目录。

2) 复制 libreadface.so 到工程 jniLibs，对应平台放置到相应文件夹。

2 初始化检测器

```
YMFaceTrack faceTrack = new YMFaceTrack();
//设置人脸检测距离，此接口必须在 initTrack 前调用才有效
faceTrack.setDistanceType(YMFaceTrack.DISTANCE_TYPE_FAR);

//默认人脸数据库保存在应用目录的 cache 目录下，/data/data/pckname/cache
faceTrack.initTrack(this,YMFaceTrack.FACE_0,YMFaceTrack.RESIZE_WIDTH_640);

//初始化方法，人脸识别数据库保存位置自定义，自定义数据库存储位置位 sd 卡时，应用需要读写权限，请参考本文“获取 Android 权限”章节
faceTrack.initTrack(this,YMFaceTrack.FACE_0,YMFaceTrack.RESIZE_WIDTH_640,"/sdcard/mydir");

//license 激活版本初始化，appid 和 appsecret 在提供的 demo 的 SenseConfig 文件中获取(请保管好自己的密钥，避免不必要的损失)
int result = faceTrack.initTrack(this, YMFaceTrack.FACE_0,YMFaceTrack.RESIZE_WIDTH_640, SenseConfig.appid, SenseConfig.appsecret);
```

初始化(即 result)的返回值：

0 ：成功

-3 ：包名不匹配

-6 ：app_id 不匹配

-5 ：未读取到激活信息

-4 ：激活失败或者网络不正常

-1 ：已过期，当前日期比截止日期要大

-2 ：已过期，当前日期比打包日期要小

-11:异常

-12:句柄初始化失败

- 1001 比对文件格式不正确
- 1002 校验失败
- 2101 激活已满
- 2102 密钥不匹配
- 1128 网络超时，没有网络，
- 1106 DNS 解析出错
- 1107, 1100 服务器异常

注：此处的初始化中，Activity 方向是 landscape；摄像头的方向是 CAMERA_FACING_BACK。

若需配置不同的方向：

竖屏模式，摄像头的前置(FACE_270)和后置(FACE_90)；

横屏模式，摄像头前置(FACE_0)和后置(FACE_0)。

如果设置的不是 FACE_0 则绘制人脸框和关键点时，需对应旋转回来，具体设置请查看 demo 中配置。

3 获取数据

接口支持 NV21 和 bitmap 两种数据格式

- Android Camera 预览回调的 byte[]：仅支持 ImageFormat.NV21

  开启相机，注册 camera 的 PreviewCallback，监听接口，注意使用带缓冲区的

PreviewCallback，如下所示：

```
camera.setPreviewCallbackWithBuffer(this);
camera.addCallbackBuffer(new byte[((previewSize.width * previewSize.height) * ImageFor-
mat.getBitsPerPixel(ImageFormat.NV21)) / 8]);

public void onPreviewFrame(byte[] data, Camera camera) {
                camera.addCallbackBuffer(data);
                /* 此处为检测器发起检测/追踪处
                 .
                 .
                 .
                 */
        }
```

- Android Bitmap：bitmap 宽高不为奇数

功能说明

人脸检测/追踪

1） 接口调用

- 单人脸追踪

```
YMFace face = faceTrack.track(bytes, iw, ih);//yuv 视频流追踪
```

```
YMFace face = faceTrack.track(bitmap);//bitmap 追踪
```

- 多人脸追踪

```
List<YMFace> faces = faceTrack.trackMulti(bytes, iw, ih);//yuv 视频流追
```

```
YMFace face = faceTrack.trackMulti(bitmap);//bitmap 追踪
```

- 人脸检测

```
List<YMFace> faces = faceTrack.trackMulti(bytes, iw, ih);//yuv 视频流检测
```

```
List<YMFace> faces = detectMultiBitmap(bitmap);//bitmap 检测
```

2) 人脸框绘制

默认绘制在宽高跟 camera previewsize 相同的画布上，根据 android 相机设备前置后置以及应用横屏竖屏，处理不同。

```
        //单脸
        YMFace face = ...;
        //多脸
        for(int i = 0;i<faces.size();i++){
            YMFace face = faces.get(i);
            .
            .
            .
        }

        float rect[] = face.getRect();
        int previewW;//为 camera previewSize 宽度
        int previewH;//为 camera previewSize 高度

        1. 竖屏前置 portrait, cameraId = 1
        //框的设置
        float x1 = previewW - rect[0]  - rect[2] ;
        float y1 = rect[1] ;
        RectF faceRect = new RectF(x1, y1, x1 + rect[2] , y1 + rect[3] );
        2. 竖屏后置 portrait, cameraId = 0
    //框的设置
    float x1 = rect[0] ;
    float y1 = rect[1] ;
    RectF faceRect = new RectF(x1, y1, x1 + rect[2] , y1 + rect[3] );
        3. 横屏前置 landscape, cameraId = 1
    //框的设置
    float x1 = previewW - rect[0] - rect[2];
    float y1 = rect[1];
    RectF faceRect = new RectF(x1, y1, x1 + rect[2], y1 + rect[3]);
        4. 横屏后置 landscape, cameraId = 0
    //框的设置
    float x1 = rect[0];
```

```
float y1 = rect[1];
RectF faceRect = new RectF(x1, y1, x1 + rect[2], y1 + rect[3]);
        5. 绘制框
canvas.drawRect(faceRect, paint);
```

3) 结束人脸检测，调用释放检测器

```
faceTrack.onRelease();
```

活体检测

1) 获取人脸 YMFace，参考“人脸检测/追踪”章节
2) 根据 YMFace 所在 faceIndex 发起检测活体
3) 双目活体检测
   a) 单可见光活体检测
      - 检测追踪可见光视频流；
      - 用 faceTrack.livenessDetect(index)接口来检测活体，返回数组 0 下标位置的指为 1 认为当前帧活体通过；
   b) 单红外活体检测
      - 检测追踪红外视频流；
      - 调用 faceTrack.livenessDetectInfrared(index)接口来检测活体，返回数组 0 下标位置的指为 1 认为当前帧活体通过；
   c) 双目活体检测（可见光+红外）
      - 在需要做活体检测时，同时开启两路摄像头，活体可见光以及红外视频流；
      - 检测追踪可见光视频流；
      - 根据可见光视频流检测追踪到的框位置在红外视频流同样框位置上调用接口 faceTrack.livenessDetectFrame(yuv, widtn, height, rect); 返回值为 1 活体通过。

```
//光活体检测
int[] liveness_ret = faceTrack.livenessDetect(index);
if(liveness_ret[0]==1){}//活体通过
else{}//活体不通过
```

```
//红外活体检测
int[] liveness_ret = faceTrack. livenessDetectInfrared(index);
if(liveness_ret[0]==1){}//活体通过
else{}//活体不通过
```

人脸属性

1) 获取人脸 YMFace，参考“人脸检测/追踪”章节

2) 根据 YMFace 所在 faceIndex 获取属性

```
int ret = faceTrack.getFaceAttribute(faceIndex, faces);
        if(ret==0){
            YMFace face = faces.get(faceIndex);
            int gender = ymFace.getGender();//性别 1 Male,0,FeMale
            int gender_confidence = ymFace.getGenderConfidence();//性别阈值
            int age = ymFace.getAge();//年龄
            int beautyScore = ymFace.getBeautyScore();//颜值

            boolean hasGlass = ymFace.isHasGlass();//是否戴眼镜
            boolean isEyeOpen = ymFace.isEyeClose();//是否闭眼
            boolean isMouthOpen = ymFace.isMouthOpen();//是否张追
            boolean isSmile = ymFace.isSmile();//是否喜悦
            //有可能获取性别可信度不够高，需重新获取一次
            if (gender_confidence >= 90) {
                //选中
            }

        }
```

人脸比对

支持人脸特征比对和人证特征比对

■ 人脸特征比对

两个人脸特征进行比对，confidence 侧为 75 及以上认为是同一个人

```
/**
        *faceIndex          : 人脸 Index
```

```
            *return                 : 返回为特征
            */
            float[] feature = faceTrack.getFaceFeature(int faceIndex);


            /** 输出浮点比对结果
            * feature1      : 特征 1
            * feature2      : 特征 2
            * return   : 返回比对结果
            */
            float confidence = faceTrack.compareFaceFeatureMix(float[] feature1, float[]
feature2);
```

■ 人证特征比对（仅对人证用户提供该接口）

人证比对功能，证件照与实拍图分别提特征进行比对，阈值为 75，比对结果 75 及以上认为是同一个人

```
/**
            *faceIndex          : 人脸 Index
            *return                 : 返回为特征
            */
            float[] feature = faceTrack.getFaceFeatureCard(int faceIndex);

            /** 输出浮点比对结果
            * feature1      : 特征 1
            * feature2      : 特征 2
            * return   : 返回比对结果
            */
            float confidence = faceTrack.compareFaceFeatureCard(float[] feature1, float[]
feature2);
```

人脸识别

注：在进行人脸注册、识别前，必须判定当前人脸是否符合角度以及质量的要求

```
float []headposes = face.getHeadpose();
        if ((Math.abs(headposes[0]) > 30
        || Math.abs(headposes[1]) > 30
        || Math.abs(headposes[2]) > 30)) {
            //角度不佳不再注册识别
            next = false;
        }
        int faceQuality = faceTrack.getFaceQuality(anaIndex);

        if (faceQuality < 90) {
            //人脸质量不佳，不再注册识别
            next = false;
        }
```

1） 注册人脸

支持两种注册方式

■ 图片注册

图片注册人脸时，理应保持图像中只有一张人脸，接口中的 faceIndex 应当均为 0。图片注册准备的数据可以有两种。

一种是直接从摄像头中取得数据：

```
List<YMFace> faces = faceTrack.trackMulti(bytes, iw, ih);
```

另一种是从图片（Bitmap）中直接获得数据；

```
Bitmap bitmap = BitmapUtil.decodeScaleImage(image.getAbsolutePath() ,scale ,scale);
List<YMFace> ymFaces = faceTrack.detectMultiBitmap(bitmap);
```

接下来的步骤均一致，首先先判断当前图像是否已经认识

```
int personId = faceTrack.identifyPerson(faceIndex);
personId > 0 //已经认识，不能再添加，可以选择删除之前的重新添加。
```

```
personId < 0 //还不认识，可以添加
```

判断完成后可以添加，然后添加人脸到人脸库

```
int personId = faceTrack.addPerson(faceIndex);
personId > 0 //添加成功，此返回值即为数据库对当前人脸的中唯一标识
personId < 0 //添加失败
```

添加完成第一张人脸后，为了提高识别的速度以及精确度，往往会为这个 personId 添加多张多角度的图像，最多可添加 10 张；

```
int updateResult = faceTrack.updatePerson(personId, faceIndex);
```

■ 视频注册

视频注册时应注意人脸缓慢变换角度，保证有抬头时的图像即可。

开始注册前，先将人脸库中已有的 personId 拿到，用于判断视频注册的结果是否是重复的。可以直接调用 getEnrolledPersonIds，或者使用自己储存的 User 库。

```
List<Integer> ids = faceTrack.getEnrolledPersonIds();
```

开始注册时，在摄像头回调处调用此接口

```
faceTrack.registerFromVideo(bytes, iw, ih);
```

3 秒后调用

```
int personId = faceTrack.registerFromVideoEnd();
```

得到 personId，需要判断注册前的人脸库中是否已经存在此 Id，若存在说明注册前的人脸库中已有此人。若得到的 personId 是小于 0 的，说明注册失败。

2) 识别人脸

人脸识别是只需要调用此接口即可

```
int personId = faceTrack.identifyPerson(faceIndex);
personId > 0 //已经认识
personId < 0 //不认识
```

获取 Android 权限

如果您在自己的工程中集成 SDK，请确保已经在工程 AndroidManifest.xml 文件中添加如下权限：

```
//开启摄像头
<uses-permission android:name="android.permission.CAMERA" />
//如果是 license 激活版本，需要网路
<uses-permission android:name="android.permission.INTERNET" />
//写入文件权限
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
//读取 wifi
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

注意事项

1　请严格按照本文中的 API 调用次序，方可实现功能，减少错误。

2　SDK 的 demo 中，libutils.jar 是非必须的。它用来提供 Android 工具类，不建议直接使用，可提取所需方法到自家工程中。demo 中关于 dou.utils 包中的类缺失，可以直接删除。

3　demo 的配置：旋转角度配置位于 BaseCameraActivity.initCameraMsg； 绘制点框问题位于 TrackUtil.drawAnim。

4　使用 demo 过程中发现部分厂商的摄像头翻转 180 的问题

首先 CameraHelper 的修改摄像头的参数为 parameters.set("rotation", "180"); camera.setDisplayOrientation(180); 初始化检测器的角度设置偏移 180°即可。CameraHelper.java 是示例开启摄像头的工具类，若不满足需求可自定义。

类定义

本 SDK 中有两个重要的开放类，分别为 YMFace、YMFaceTrack

YMFace

功能：用于保存人脸检测、识别后的结果，属性如下：

```
private float[] rect;
private float[] landmarks;
private float[] emotions;
private float[] aus;
private float[] headpose;
private float[] headOrientations;
private int[] facialActions;
private int gender;
private int genderConfidence = -1;
private int age;
private int beautyScore;
private int personId;
private int confidence;
private int trackId;

描述：
* rect：检测到得人脸框坐标，对应（left,top,width,height）
* landmarks：关键点坐标，对应（x1,y1,x2,y2.....）
* emotions：表情定义，长度为 7，对应（"喜悦", "悲伤", "预留位", "预留位", "惊讶", "愤怒", "正常"）
* aus：面部动作，比如，嘴角扬起程度，眉毛抬起程度
* headpose：头部转向，低头抬头，左转右转等头部姿态
        headposes[0]：对应 roll - 左正右负
        headposes[1]：对应 pitch - 下正上负
        headposes[2]：对应 yaw - 左负右正

* headOrientations：用于构建头部坐标系
* facialAction：微表情定义，长度为 6，对应（"预留位"，"OPENMOUTH"，"EYECLOSE"，"FROWN"，"预留位"，"POUT"）
* gender：性别，
* genderConfidence：性别的可信度，目前认为大于 90 才可信，也可以根据实际场景调整
* age：年龄
* beautyScore：人脸颜值 0-100
* personId：人脸 ID（人脸识别唯一标识）
* confidence：人脸置信度，人脸识别时使用
```

* trackid：人脸 tracking 的 id，换张或者离开屏幕再进入，会加 1，单脸人检测此值为无效，只针对多人脸接口

在人脸检测过程中，rect、landmarks、headpose 以及 trackid(多人脸)会被检测接口赋值后输出，其他的 Face 参数需要根据返回的人脸集合，自行调用接口获取各属性的值，然后调用 YMFace 中 set 方法进行各项赋值。下面展示如何给 age 赋值，其他属性类似。

Age 示例：

```
//单人脸设置
YMFace face = ...;
face.setAge(faceTrack.getAge(0));
//多人脸设置
List<YMFace> faces = ...;
int age0 = faceTrack.getAge(faceIndex);
faces.get(faceIndex).setAge(age0);
```

YMFaceTrack

功能：初始化检测器，进行集成检测

参数：

```
private int orientation;
private int resizeScale;
```

描述：

* orientation：图像需要旋转的角度
* resizeScale：图像需要压缩到此值，此值已失效，请忽略

API 列表

1 setDistanceType(int distanceType)

接口描述：设置检测距离

备注：此方法要求在 initTrack 方法之前调用

	参数名称	Type	备注
输入参数	dis-tanceType	int	YMFaceTrack.DISTANCE_TYPE_NEARST(最近距离 1米内)， YMFaceTrack.DISTANCE_TYPE_NEAR(近距离 1米-2米)， YMFaceTrack.DISTANCE_TYPE_FAR(远距离 2-4米)， YMFaceTrack.DISTANCE_TYPE_FARTHESTER(最远距离 4-6米)， 距离为检测距离，人脸识别精度可用用情况下为 5米内
输出参数			

2 initTrack (Context mContext，int orientation，int resizeScale)

接口描述：初始化检测器

备注：人脸识别数据库之前保存在应用目录的 cache 目录下

	参数名称	Type	备注
输	mContext	Context	上下文对象

入参数	orientation	int	图片旋转的角度 YMFaceTrack.FACE_0 YMFaceTrack.FACE_90 YMFaceTrack.FACE_180 YMFaceTrack.FACE_270
	resizeScale (失效参数)	int	重置图片尺寸 RESIZE_WIDTH_NONE RESIZE_WIDTH_60 RESIZE_WIDTH_80 RESIZE_WIDTH_160 RESIZE_WIDTH_240 RESIZE_WIDTH_320 RESIZE_WIDTH_480 RESIZE_WIDTH_640 RESIZE_WIDTH_1080 RESIZE_WIDTH_1920 RESIZE_WIDTH_2560
输出参数	initResult	int	0：成功 其他：失败

3 initTrack(Context mContext, int orientation, int resizeScale, String db_dir)

接口描述：初始化，可以指定人脸数据库的路径

备注：

	参数名称	Type	备注
输	mContext	Context	上下文对象

入参数	orientation	int	图片旋转的角度 YMFaceTrack.FACE_0 YMFaceTrack.FACE_90 YMFaceTrack.FACE_180 YMFaceTrack.FACE_270
	resizeScale (失效参数)	int	重置图片尺寸 RESIZE_WIDTH_NONE RESIZE_WIDTH_60 RESIZE_WIDTH_80 RESIZE_WIDTH_160 RESIZE_WIDTH_240 RESIZE_WIDTH_320 RESIZE_WIDTH_480 RESIZE_WIDTH_640 RESIZE_WIDTH_1080 RESIZE_WIDTH_1920 RESIZE_WIDTH_2560
	db_dir	String	人脸数据库路径
输出参数	initResult	int	0：成功 其他：失败

4 initTrack (Context mContext，int orientation，int resizeScale，String ap-pid, String appsecret)

接口描述：SDK 正式版初始化

备注：人脸识别数据库之前保存在应用目录的 cache 目录下

	参数名称	Type	备注
输	mContext	Context	上下文对象

入参数	orientation	int	图片旋转的角度 YMFaceTrack.FACE_0 YMFaceTrack.FACE_90 YMFaceTrack.FACE_180 YMFaceTrack.FACE_270
	resizeScale (失效参数)	int	重置图片尺寸 RESIZE_WIDTH_NONE RESIZE_WIDTH_60 RESIZE_WIDTH_80 RESIZE_WIDTH_160 RESIZE_WIDTH_240 RESIZE_WIDTH_320 RESIZE_WIDTH_480 RESIZE_WIDTH_640 RESIZE_WIDTH_1080 RESIZE_WIDTH_1920 RESIZE_WIDTH_2560
	appid	String	正式版 SDK 的 key
	appsecret	String	正式版 SDK 的密钥
输出参数	initResult	int	0：成功 其他：失败

5 initTrack (Context mContext，int orientation，int resizeScale，String appid, String appsecret，String db_dir)

接口描述：sdk 正式版初始化 并指定人脸数据库位置

备注：

	参数名称	Type	备注
输	mContext	Context	上下文对象

入参数	orientation	int	图片旋转的角度 YMFaceTrack.FACE_0 YMFaceTrack.FACE_90 YMFaceTrack.FACE_180 YMFaceTrack.FACE_270
	resizeScale (失效参数)	int	重置图片尺寸 RESIZE_WIDTH_NONE RESIZE_WIDTH_60 RESIZE_WIDTH_80 RESIZE_WIDTH_160 RESIZE_WIDTH_240 RESIZE_WIDTH_320 RESIZE_WIDTH_480 RESIZE_WIDTH_640 RESIZE_WIDTH_1080 RESIZE_WIDTH_1920 RESIZE_WIDTH_2560
	appid	String	
	appsecret	String	
	db_dir	String	人脸数据库路径
输出参数	initResult	int	0：成功 其他：失败

6 setRecognitionConfidence(int confidence)

接口描述：设置人脸识别置信度

备注：**此值不推荐修改**。有可能会根据算法以及模型的迭代修改此值，如有更改会及时告知。

	参数名称	Type	备注

输入参数	confidence	int	可信度阈值（默认 75）
输出参数		void	

7 track (byte[] bytes,int iw, int ih)

接口描述： 单个人脸追踪（输入 Camera 预览回调上来的数据）

备注：

	参数名称	Type	备注
输入参数	bytes	Byte[]	Camera 预览回调上来的数据
	iw	int	预览图像宽度
	ih	int	预览图像高度
输出参数	ymFace	YMFace	rect：检测到得人脸框坐标，对应（left,top,width,height） landmarks：关键点坐标，对应（x1,y1,x2,y2.....） headpose：头部转向，低头抬头，左转右转等头部姿态 headposes[0]：对应 roll - 左正右负 headposes[1]：对应 pitch - 下正上负 headposes[2]：对应 yaw - 左负右正 trackid：人脸 tracking 的 id，换张或者离开屏幕再进入，会加 1，单脸人检测此值为无效，只针对多人脸接口

8 track (Bitmap bitmap)

接口描述：单个人脸追踪(输入静态图片)

备注：

	参数名称	Type	备注
输入参数	bitmap	Bitmap	静态图片
输出参数	ymFace	YMFace	人脸数据

9 trackMulti (byte[] bytes,int iw, int ih)

接口描述： 追踪多个人脸（输入 Camera 预览回调上来的数据）

备注：

	参数名称	Type	备注
输入参数	bytes	Byte[]	Camera 预览回调上来的数据
	iw	int	预览图像宽度
	ih	int	预览图像高度
输出参数	ymFaces	List<YMFace>	人脸数据

10 trackMulti (Bitmap bitmap)

接口描述：追踪多个人脸(输入静态图片)

备注：

	参数名称	Type	备注
输入参数	bitmap	Bitmap	输入静态图片
输出参数	ymFaces	List<YMFace>	人脸数据

11 faceDetect (byte[] bytes,int iw, int ih)

接口描述： 检测多个人脸

备注：

	参数名称	Type	备注
输入参数	bytes	Byte[]	Camera 预览回调上来的数据
	iw	int	预览图像宽度
	ih	int	预览图像高度
输出参数	ymFaces	List<YMFace>	人脸数据

12 detectMultiBitmap (Bitmap bitmap)

接口描述： 检测多个人脸

备注：

	参数名称	Type	备注
输入参数	bitmap	Bitmap	静态图片
输出参数	ymFaces	List<YMFace>	人脸数据列表

13 getEnrolledPersonIds ()

接口描述： 获取所有已经注册的人的 id

备注：

	参数名称	Type	备注
输入参数			
输出参数	personIds	List<Integer>	用户 id 列表

14 registerFromVideo (byte[] bytes,int iw, int ih)

接口描述： 开始视频注册人脸

备注： 视频注册时应注意人脸缓慢变换角度，保证有抬头时的图像即可

	参数名称	Type	备注
输入参数	bytes	Byte[]	Camera 预览回调上来的数据
	iw	int	预览图像宽度
	ih	int	预览图像高度
输出参数	success	bool-ean	True：成功 False：失败

15 registerFromVideoEnd ()

接口描述： 结束视频注册人脸，将人脸注册到数据库

备注： registerFromVideo 接口调用 3 秒后调用此接口

	参数名称	Type	备注
输入参数			
输出参数	personId	int	personId>0：成功 其他：失败

16 addPerson (int faceIndex)

接口描述：添加新的人到数据库

备注： 添加此人脸到人脸库，返回 personId 大于 0，则注册首张人脸成功，反之失败。

	参数名称	Type	备注
输入参数	faceIndex	int	检测到的人脸集合下标
输出参数	personId	int	personId > 0 :添加成功，此返回值即为数据库对当前人人脸的中唯一标识 personId < 0 :添加失败

17 addPerson(float[] feature)

接口描述：添加新的人到数据库

备注： 添加此人脸到人脸库，返回 personId 大于 0，则注册首张人脸成功，反之失败

	参数名称	Type	备注
输入参数	feature	float[]	人脸特征
输出参数	personId	int	personId > 0 :添加成功，此返回值即为数据库对当前人人脸的中唯一标识 personId < 0 :添加失败

18 updatePerson (int personId, int faceIndex)

接口描述： 给用户添加人脸

备注：添加完成第一张人脸后，为了提高识别的速度以及精确度，往往会为这个 personId 添加多张多角度的图像，最多可添加 10 张

	参数名称	Type	备注
输入参数	personId	int	用户 id
	faceIndex	int	检测到的人脸集合下标
输出参数	personId	int	personId > 0 :添加成功，此返回值即为数据库对当前人人脸的中唯一一标识 personId < 0 :添加失败

19 updatePerson(int personId, float[] feature);

接口描述： 给用户添加人脸

备注：添加完成第一张人脸后，为了提高识别的速度以及精确度，往往会为这个 personId 添加多张多角度的图像，最多可添加 10 张

	参数名称	Type	备注
输入参数	personId	int	用户 id
	feature	float[]	人脸特征
输出参数	personId	int	personId > 0 :添加成功，此返回值即为数据库对当前人人脸的中唯一一标识 personId < 0 :添加失败

20 identifyPerson(int faceIndex)

接口描述：人脸识别

备注：

	参数名称	Type	备注
输入参数	faceIndex	int	检测到的人脸集合下标
输出参数	personId	int	personId 为唯一标识此人 personId > 0 :已经认识 personId < 0 :不认识

21 livenessDetect (int faceIndex)

接口描述：可见光活体检测

备注：

	参数名称	Type	备注
输入参数	faceIndex	int	检测到的人脸集合下标
输出参数	results	int []	Results[0]==1 > 0 活体检测通过 其他 :不通过

22 livenessDetectInfrared(int faceIndex)

接口描述：红外活体检测接口

备注：

	参数名称	Type	备注
输入参数	faceIndex	int	检测到的人脸集合下标
输出参数	results	int []	Results[0]==1 > 0 活体检测通过 其他 :不通过

23 livenessDetectFrame(byte[] bytes, int iw, int ih, float[] rect)

接口描述：双目活体检测接口

备注：

	参数名称	Type	备注
输入参数	bytes	byte[]	红外 yuv 图像
	Iw	int	红外 yuv 图像宽度
	Ih	int	红外与图像高度
	rect	float[]	可见光检测追踪到的人脸框
输出参数	results	int[]	Results[0]==1 > 0 活体检测通过 其他 :不通过

24 getFaceFeature (int faceIndex)

接口描述：获取人脸识别时的人脸特征值

备注：

	参数名称	Type	备注
输入参数	faceIndex	int	检测到的人脸集合下标
输出参数	feature	float[]	人脸特征值

25 getFaceFeatureCard (int faceIndex)

接口描述：获取人证比对时的人脸特征值

备注：

	参数名称	Type	备注
输入参数	faceIndex	int	检测到的人脸集合下标
输出参数	feature	float[]	人脸特征值

26 compareFaceFeatureMix(float[] feature1, float[] feature2)

接口描述：比对人脸特征值

备注： 人脸比对功能，提取两张人脸的特征进行比对，输出相似度，包括小数点后数据

	参数名称	Type	备注
输入参数	feature1	float[]	人脸特征值 1
	Feature2	float[]	人脸特征值 2
输出参数	confidence	float	相似度

27 deletePerson (int personId)

接口描述： 删除注册的用户

备注：

	参数名称	Type	备注
输入参数	personId	int	人脸的 personId
输出参数	result	int	0：成功 其他：失败

28 getAlbumSize ()

接口描述： 获取当前人脸库中共有多少个人

备注：

	参数名称	Type	备注
输入参数			
输出参数	result	int	返回用户数量

29 getFaceCountByPersonId (int personId)

接口描述： 根据 personId 获取对应人脸张数

备注：

	参数名称	Type	备注
输入参数	personId	int	人脸的 personId
输出参数	result	int	人脸数目

30 getRecognitionConfidence ()

接口描述：当前人脸识别时，获取与人脸数据库中所有人脸比对的最大的相似度

备注：

	参数名称	Type	备注
输入参数			
输出参数	result	int	

31 onRelease ()

接口描述： 释放检测器

备注：

	参数名称	Type	备注
输入参数			无
输出参数			无

32 getVersion ()

接口描述： 获取算法版本号

备注：

	参数名称	Type	备注
输入参数			无
输出参数	result	String	

33 setOrientation (int orientation)

接口描述： 设置检测器对图片旋转的角度

备注：

	参数名称	Type	备注
输入参数	orientation	int	图片旋转的角度 YMFaceTrack.FACE_0 YMFaceTrack.FACE_90 YMFaceTrack.FACE_180 YMFaceTrack.FACE_270
输出参数			

34 getFaceQuality (int faceIndex)

接口描述： 获取人脸的质量

备注：

	参数名称	Type	备注
输入参数	faceIndex	int	检测到的人脸集合下标
输出参数	quality	int	(0-10), 大于等于 6 认为质量比较好

35 cropFace (byte[] bytes, int w, int h, float[] react, String path, int var)

接口描述：抠图

备注：

	参数名称	Type	备注
输入参数	bytes	byte[]	YUV 数据
	w	int	宽
	h	int	高
	react	float[]	人脸框坐标
	path	String	存储路径
	var	int	颜色模式 public static final int CROP_MODE_GRAY = 0; 灰度

			public static final int CROP_MODE_COLOR = 1;彩色
输出参数			

36 getFaceAttribute (int index, List<YMFace> ymFaces)

接口描述： 输出当前人脸的所有属性

备注：

	参数名称	Type	备注
输入参数	index	Int	检测到的人脸集合下标
	ymFaces	List<YMFace>	人脸数据
输出参数	Result	int	0：成功 非 0：失败

37 resetAlbum ()

接口描述： 清空人脸数据库

备注：

	参数名称	Type	备注
输入参数			
输出参数	result	int	0：成功 其他：失败