ReadSense 人脸SDK 开发文档

基本类型定义

1. 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[] 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.....）
* 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);
```

2. 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
```

使用指南

1. 创建android studio工程

   如何创建工程请访问[如何创建Android Studio工程](#) ,工程创建完毕后添加权限，注意Android设备6.0以上设备需要动态申请相机等权限，详情参看 [Android设备如何动态申请权限](#) ， 也可以参照Demo中使用 [RxPermissions](#) 来管理权限申请。

```
//需要申请的权限
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STOR
AGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" /
```

```
>
//若是使用lisence的付费用户请在首次使用时打开网络验证，申请internet权限
<uses-permission android:name="android.permission.INTERNET" />
```

新建activity并开启相机，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.h
eight) * ImageFormat.getBitsPerPixel(ImageFormat.NV21)) / 8]);

public void onPreviewFrame(byte[] data, Camera camera) {
    camera.addCallbackBuffer(data);

    /* 此处为检测器发起检测处
     .
     .
     .
     */
}
```

)

2. 初始化检测器，准备阶段

   a. 详细阅读本文章

   b. 复制相关文件到指定目录

   - 复制libreadface.jar到工程libs目录。
   - 复制libreadface.so到工程jniLibs，对应平台放置对应文件夹。

   c. 初始化检测器

```
YMFaceTrack faceTrack = new YMFaceTrack();

//检测距离设置，属性设置需要在init之前调用，参数包括DISTANCE_TYPE_NEARST(
最近距离1米内)，DISTANCE_TYPE_NEAR(近距离1米-2米)，DISTANCE_TYPE_FAR(远
距离2-4米)，DISTANCE_TYPE_FARTHESTER(最远距离4-6米)，距离为检测距离，人脸
识别精度可用情况下为5米内
faceTrack.setDistanceType(int distanceType)

//此处为默认初始化（Activity方向为landscape，摄像头为1->CAMERA_FACING_FR
ONT）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(FAC
E_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。
faceTrack.initTrack(this,YMFaceTrack.FACE_0,YMFaceTrack.RESIZE_WIDTH_640);

//普通有效期初始化方法，人脸识别数据库保存位置自定义，人脸数据库faces.db默认保存在应用目录的cache目录下
int result = faceTrack.initTrack(this,YMFaceTrack.FACE_0,YMFaceTrack.RESIZE_WIDTH_640,"/sdcard/mydir");
//license激活版本初始化
int result = faceTrack.initTrack(this, YMFaceTrack.FACE_0,
    YMFaceTrack.RESIZE_WIDTH_640, SenseConfig.appid, SenseConfig.appsecret);
if(result==0)//初始化成功
else //初始化失败
```

3. 发起人脸检测

```
//单脸tracking
YMFace face = faceTrack.track(bytes, iw, ih);
//多脸tracking
List<YMFace> faces = faceTrack.trackMulti(bytes, iw, ih);
```

4. 绘制人脸框以及关键点（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。

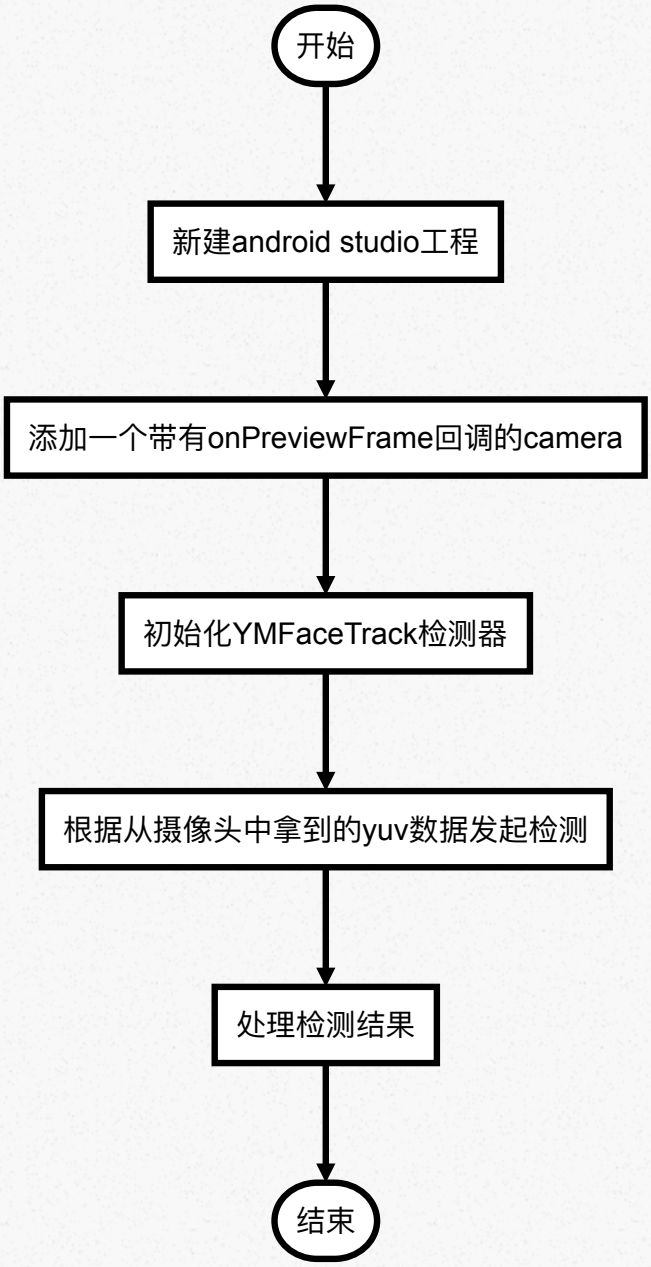
```
//单脸
YMFace face = ...;
//多脸
for(int i = 0;i<faces.size();i++){
    YMFace face = faces.get(i);
    .
    .
    .
}

float rect[] = face.getRect();
float landmarks[] = face.getLandmarks();
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] );
//关键点的设置
float x = previewW - landmarks[i * 2] ;
float y = landmarks[i * 2 + 1] ;
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] );
//关键点的设置
float x = landmarks[i * 2];
float y = landmarks[i * 2 + 1];
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]);
//关键点的设置
float x = previewW - landmarks[i * 2];
float y = landmarks[i * 2 + 1];
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]);
//关键点的设置
float x = landmarks[i * 2];
float y = landmarks[i * 2 + 1];
绘制框和点
canvas.drawRect(faceRect, paint);
canvas.drawPoint(x, y, paint);
```

结束人脸检测，调用释放检测器

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

到此处人脸tracking演示完毕，流程图如下：



功能详解

5. 属性识别功能

   Readsense的人脸SDK可以输出丰富的属性，其中包括：

   - 表情
   - 头部姿态
   - 年龄
   - 性别
   - 颜值
   - 微表情

   上一节描述过人脸关键点以及人脸框，之后不在说明。

a. 表情识别

表情识别输出数据长度为7，对应（喜悦, 悲伤, "预留位", "预留位", 惊讶, 愤怒, 正常），输出数组中值比较最大的那个味当前输出的表情。（此接口目前不再推荐使用，请已使用的客户不要再使用）主要代码如下：

```
String showString;//最终得到的表情
//定义表情序列
String emotions[] = {"喜悦", "悲伤", "", "", "惊讶", "愤怒", "正常"};

//获取表情数组
float[] emos = faceTrack.getEmotion(faceIndex);

//根据输出数组得到最大的那个下标
public static int getMaxFromArr(float arr[]) {
    int position = 0;
    float max = 0;
    for (int j = 0; j < arr.length; j++) {
        if (max <= arr[j]) {
            max = arr[j];
            position = j;
        }
    }
    return position;
}

//最终得到的表情
showString = emotions[getMaxFromArr(result)];
```

b. 年龄性别识别

年龄接口输出为一个int类型的值，即为当前人脸年龄；性别接口输出为0（Female）或者1（Male），主要代码如下：

```
int age = faceTrack.getAge(faceIndex);
int gender = faceTrack.getGender(faceIndex);
```

此处性别接口另有一输出新别可信度接口，目前认为大于90才可信，也可以根据实际场景调整

```
int gender_confidence = faceTrack.getGenderConfidence();
```

用于校准性别识别是否可靠，一般认为gender_confidence值大于90，当前输出的gender是一个可靠的值。

c. 颜值、微表情以及表情检测

这两个接口是针对个别客户设计，大部分客户不需要关注这两个接口。如无要求，这两个接口是不对外进行开放的，请不要擅自调用。展示如下：

```
//对应人脸颜值<0--100>
int score = faceTrack.getFaceBeautyScore(faceIndex);
//微表情动作，数组长度为6，对应index值为1代表有此动作，为0代表无此动作，对应（"预留位", "OPENMOUTH", "EYECLOSE", "FROWN", "预留位", "POUT"）
int[] facialActions = faceTrack.getFacialActions(faceIndex);
//表情,score为1代表当前为的正向情绪，否则为正常
int score = faceTrack.getHappyScore(faceIndex);
```

d. 识别是否张嘴

```
boolean isMouthOpen = faceTrack.isMouthOpen(faceIndex);
```

a. 其他属性

人脸框rect、关键点landmarks、头部姿态headposes以及多人脸追踪时的trackingId会在追踪过程中直接输出，不用二次调用接口来获取。其他属性如无特殊说明，请忽略。

6. 人脸识别功能

人脸识别作为本SDK核心功能，流程较为复杂，请依照帮助文档以及辅助demo建立自己的人脸识别模块。
人脸识别模块演示demo比较复杂，详情参照SDK中附带Demo

a. 接口描述

- faceTrack.setRecognitionConfidence(75);（必须在初始化接口initTrack之后调用）此接口为设置人脸识别可信度，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，如有更改会特地告知。
- int personId = faceTrack.identifyPerson(faceIndex);对此张人脸发起识别，如personId为大于0说明已经认识此人，personId为唯一标识此人；如personId=-1 error:未识别到人脸;如personId=-111 不认识此人，可以对此人进行注册。
- int personId = faceTrack.addPerson(faceIndex);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。

- int result = faceTrack.updatePerson(personId, faceIndex);对personId添加当前Face List中第faceIndex个人脸，可通过getFaceCountByPersonId获得人脸Id为personId的人脸中共添加了几张人脸，成功result等于0，失败小于0。
- int result = faceTrack.deletePerson(personId);从人脸库中删除此人脸，成功result等于0，失败小于0。
- int result = faceTrack.resetAlbum();清空人脸库，成功result等于0，失败小于0。
- int result = faceTrack.getAlbumSize();获取当前人脸库中共有多少个人。
- int result = faceTrack.getFaceCountByPersonId(personId);获取该personId对应的人脸数目。
- List ids = faceTrack.getEnrolledPersonIds();获取获取当前人脸库中所有的人脸的Id。

b. 注册人脸到人脸库

注册人脸时，请在摄像头预览中只保持一张人脸，并保证图像尽可能清晰，如果在注册时图像模糊，或者多人，可能造成注册失败或者识别出错。

i. 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。

ii. 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.getAbsolutePa
th() ,scale ,scale);
List<YMFace> ymFaces = faceTrack.detectMultiBitmap(bitmap);
```

以上两种方式均支持，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);
```

c. 识别人脸

人脸识别是只需要调用此接口即可；

```
int personId = faceTrack.identifyPerson(i);
```

```
personId > 0 //已经认识
personId < 0 //不认识
```

7. 活体检测（此功能为附加功能，普通sdk未开启，请勿调用）
   活体检测时，请在摄像头预览中只保持一张人脸，并保证图像尽可能清晰。

```
List<YMFace> faces = faceTrack.trackMulti(bytes, iw, ih);
int[] ints = faceTrack.livenessDetect(0);

if (ints[0] == 1) {
    show_str = "活体通过，开始人脸识别";
} else {
    show_str = "活体不通过";
}
```

8. 人脸抠图（此功能为附加功能，普通sdk未开启，请勿调用）
   要求：1. 调用多人脸追踪接口。2. 输入为nv21

   人脸追踪过程中，开启人脸抠图的sdk会在多人脸追踪时持有当前追踪序列，调用抠图接口可抠出此序列最佳人脸照片，相关接口描述如下：

```
    //初始化之前（调用initTrack之前）需要调用此接口，默认为1，设置抠取人脸大小
,
    faceTrack.setCropScale(float scale);
    //初始化之前（调用initTrack之前）需要调用此接口，默认为10，不建议设置过大，
    faceTrack.setCropMaxCache(10);
    //如果需要保存原图需要调用此接口（调用initTrack之前），默认false不保存原图
    faceTrack.setCropBg(boolean cropBg);
```

```
    //抠取人脸，
    /**
    * path_face     : 保存图片的绝对路径
    * trackId       : 需要抠取人脸序列的trackId
    * orientation   : 需要顺时针旋转的角度
    **/
    faceTrack.nativeCropFaceByTrackId(path_face, trackId, orientation
);
```

```
    //抠取人脸以及保存原图
```

```
    /**
    * path_face      : 保存人脸图片的绝对路径
    * path_image     : 保存原图的路径
    * trackId        : 需要抠取人脸序列的trackId
    * orientation    : 需要顺时针旋转的角度
    **/
    faceTrack.nativeCropFaceByTrackId(path_face, path_image, trackId,
 orientation);
```

使用详情参考demo BaseCameraActivity.java初始化处以及PointActivity.java注释部分使用

9. 人脸比对
   人脸比对功能，提取两张人脸的特征进行比对，输出相似度。

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

    /**
    * feature1  : 特征1
    * feature2  : 特征2
    * return    : 返回比对结果
    */
    int confidence = faceTrack.compareFaceFeature(float[] feature1, f
loat[] feature2);

    /** 输出小数点后数据
    * feature1  : 特征1
    * feature2  : 特征2
    * return    : 返回比对结果
    */
    float confidence = faceTrack.compareFaceFeatureMix(float[] featur
e1, float[] feature2);
```

10. 人证比对
    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37，比对结果超过37认为是同一个人

```
    /**
    *faceIndex       : 人脸Index
    *return          : 返回为特征
```

```
    */
    float[] feature = faceTrack.getFaceFeatureCard(int faceIndex);

    /**
    * feature1  : 特征1
    * feature2  : 特征2
    * return    : 返回比对结果
    */
    int confidence = faceTrack.compareFaceFeature(float[] feature1, f
loat[] feature2);

    /** 输出小数点后数据
    * feature1  : 特征1
    * feature2  : 特征2
    * return    : 返回比对结果
    */
    float confidence = faceTrack.compareFaceFeatureMix(float[] featur
e1, float[] feature2);
```

注意事项

1. 本文档附带的所有demo，均需要将贵公司在阅面公司商务处申请的SDK中的so库添加进去才能正常使用，应用包名需替换成申请SDK时所使用的包名 。
2. SDK中附带demo中libutils.jar是非必须的，，只是提供各种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包种的类缺失，都是些工具类，可以直接删除或参考：https://github.com/littledou/android_utils 。
3. SDK中附带demo配置注意 关于旋转角度配置位于BaseCameraActivity .initCameraMsg关于绘制点框问题位于TrackUtil.drawAnim。
4. SDK附带demo针对部分厂商的摄像头安装翻转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。
5. 收费正式版首次使用需要联网认证一下，之后不依赖网络。
6. 初始化返回码
    a. 0 : 成功
    b. -3 : 包名不匹配
    c. -6 : app_id不匹配
    d. -5 : 未读取到激活信息
    e. -4 : 激活失败或者网络不正常

f. -1 : 已过期
g. -11:异常
h. -12:句柄初始化失败

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_STORAG
E"/>
//读取wifi
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

License 激活

联网激活方案

license 激活主要在FaceTrack初始化处修改

```
    //license激活版本初始化
    int result = faceTrack.initTrack(this, YMFaceTrack.FACE_0,
            YMFaceTrack.RESIZE_WIDTH_640, SenseConfig.appid, SenseConfi
g.appsecret);
```

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**)