FaceSDK_Android_ API文档_8.1.0.1

1、实体类说明

1.1 BDFaceInstance 实体类

人脸能力实例类：用来创建人脸能力实例对象，对象会在具体的人脸能力初始化的时候使用到 1. 所有的人脸能力都是绑定在具体的人脸能力实例类上的，如果不显式定义的话，人脸库会自创建并使用默认的人脸能力实例对象 2. 在单线程模式下，用户不需要关心此类的使用，直接使用默认的方法即可；在多线程模式下，用户需要使用此类显式创建人脸能力实例类，并根据实际需求将对应的人脸能力绑定到对应的实例上进行人脸能力的使用

1.2 FaceInfo[ ] 实体类

```
public class FaceInfo {

   /**
     * ---------detect---------**
     * 人脸索引值，标记连续视频帧追踪中人脸ID
     */
    public int faceID;

    /**
     * 人脸中心点x坐标
     */
    public float centerX;

    /**
     * 人脸中心点y坐标
     */
    public float centerY;

    /**
     * 人脸宽度
     */
    public float width;

    /**
     * 人脸高度
     */
    public float height;

    /**
     * 人脸角度
     */
    public float angle;

    /**
     * 人脸置信度
     */
    public float score;

    /**
     * 人脸72个关键点数据（鼻子，眼镜，嘴巴，眉毛）
     */
    public float[] landmarks;

    /** -------- config head pose ture------**/

    /**
     * 人脸左右偏转角
     */
```

```
    public float yaw;

    /**
     * 人脸平行平面内的头部旋转角
     */
    public float roll;

    /**
     * 人脸上下偏转角
     */
    public float pitch;

    /** ---------config quality blur illum occluton ture---------**/

    /**
     * 人脸模糊度信息
     */
    public float bluriness;

    /**
     * 人脸光照信息
     */
    public int illum;

    /**
     * 人脸遮挡信息
     */
    public BDFaceOcclusion occlusion;

    /**---------config Attribute ture-------**/

    /**
     * 人脸年龄
     */
    public int age;

    /**
     * 人脸种族（黄，白，黑，印度）
     */
    public BDFaceSDKCommon.BDFaceRace race;

    /**
     * 人脸佩戴眼镜状态（无眼镜，有眼镜，墨镜）
     */
    public BDFaceSDKCommon.BDFaceGlasses glasses;

    /**
     * 人脸性别（男女）状态
     */
    public BDFaceSDKCommon.BDFaceGender gender;

    /**---------config isMouthClose ture-------**/

    /**
     * 嘴巴闭合置信度
     */
    public float mouthclose;

    /**---------config isEyeClose ture-------**/

    /**
     * 左眼闭合的置信度
     */
    public float leftEyeclose;
    /**
     * 右眼闭合的置信度
     */
```

```
    public float rightEyeclose;
     /**
     * 最优人脸的置信度
     */
    public float bestImageScore;

}
```

1.3 BDFaceSDKCommon 枚举类

```
public class BDFaceSDKCommon {

     public enum BDFaceImageType {
        BDFACE_IMAGE_TYPE_RGB,
        BDFACE_IMAGE_TYPE_BGR,
        BDFACE_IMAGE_TYPE_RGBA,
        BDFACE_IMAGE_TYPE_BGRA,
        BDFACE_IMAGE_TYPE_GRAY,
        BDFACE_IMAGE_TYPE_DEPTH,
        BDFACE_IMAGE_TYPE_YUV_NV21, //  YYYYVUVU
        BDFACE_IMAGE_TYPE_YUV_NV12, // YYYYUVUV
        BDFACE_IMAGE_TYPE_YUV_YV12, // YYYYVVUU
    }

    // 检测类型
    public enum DetectType {
        DETECT_VIS,  // 可见光图像
        DETECT_NIR   // 红外图像
    }

    //  人脸关键点类型枚举
    public enum AlignType {
        BDFACE_ALIGN_TYPE_RGB_ACCURATE,   // 精确RGB对齐
        BDFACE_ALIGN_TYPE_RGB_FAST,       // 快速RGB对齐
        BDFACE_ALIGN_TYPE_NIR_ACCURATE,   // 精确NIR对齐
    }

    /**
     * 图片检测类型，目前支持红外，深度图，可见光
     */
    public enum LiveType {
        BDFACE_SILENT_LIVE_TYPE_RGB,   // RGB活体
        BDFACE_SILENT_LIVE_TYPE_NIR,   // NIR活体
        BDFACE_SILENT_LIVE_TYPE_DEPTH, // Depth深度活体
    }

    /**
     * 特征提取图片类型，证件照，可见光，红外，RGBD
     */
    public enum FeatureType {
        BDFACE_FEATURE_TYPE_LIVE_PHOTO,     // 生活照特征提取
        BDFACE_FEATURE_TYPE_ID_PHOTO,       // 证件照特征提取
        BDFACE_FEATURE_TYPE_NIR,            // 近红外特征提取
        BDFACE_FEATURE_TYPE_RGBD,           // RGBD特征提取
    }

    // 质量检测类型
    public enum FaceQualityType {
        BLUR,         // 模糊
        OCCLUSION,    // 遮挡
        ILLUMINATION  // 光照
    }

    // 人脸属性种族
```

```
    public enum BDFaceRace {
        BDFACE_RACE_YELLOW,       // 黄种人
        BDFACE_RACE_WHITE,        // 白种人
        BDFACE_RACE_BLACK,        // 黑种人
        BDFACE_RACE_INDIAN,       // 印度人
    }

    // 戴眼镜状态
    public enum BDFaceGlasses {
        BDFACE_NO_GLASSES,   // 无眼镜
        BDFACE_GLASSES,      // 有眼镜
        BDFACE_SUN_GLASSES,  // 墨镜
    }

    // 性别
    public enum BDFaceGender {
        BDFACE_GENDER_FEMALE, // 女性
        BDFACE_GENDER_MALE,   // 男性
    }

    // 凝视方向
    public enum BDFaceGazeDirection {
        BDFACE_GACE_DIRECTION_UP,           // 向上看
        BDFACE_GACE_DIRECTION_DOWN,         // 向下看
        BDFACE_GACE_DIRECTION_RIGHT,        // 向右看
        BDFACE_GACE_DIRECTION_LEFT,         // 向左看
        BDFACE_GACE_DIRECTION_FRONT,        // 向前看
        BDFACE_GACE_DIRECTION_EYE_CLOSE,    // 闭眼
    }

    public enum BDFaceActionLiveType {
        BDFace_ACTION_LIVE_BLINK,           // 眨眨眼
        BDFACE_ACTION_LIVE_OPEN_MOUTH,      // 张张嘴
        BDFACE_ACTION_LIVE_NOD,             // 点点头
        BDFACE_ACTION_LIVE_SHAKE_HEAD,      // 摇摇头
        BDFACE_ACTION_LIVE_LOOK_UP,         // 抬头
        BDFACE_ACTION_LIVE_TURN_LEFT,       // 向左转
        BDFACE_ACTION_LIVE_TURN_RIGHT,      // 向右转
    }

    /**
     * log种类枚举
     */
    public enum BDFaceLogInfo {
        BDFACE_LOG_VALUE_MESSAGE,   // 打印输出值日志
        BDFACE_LOG_ERROR_MESSAGE,   // 打印输出错误日志
        BDFACE_LOG_ALL_MESSAGE,     // 打印所有日志
    }

    public enum BDFaceCoreRunMode {
        BDFACE_LITE_POWER_HIGH,      //大核运行模式
        BDFACE_LITE_POWER_LOW,       //  小核运行模式
        BDFACE_LITE_POWER_FULL,      // 大小核混用模式
        BDFACE_LITE_POWER_NO_BIND,   // 不绑核运行模式 （推荐）
        BDFACE_LITE_POWER_RAND_HIGH, // 轮流绑定大核模式
        BDFACE_LITE_POWER_RAND_LOW   // 轮流绑定小核模式
    }
}
```

1.4 BDFaceSDKConfig 配置实体类

```
public class BDFaceSDKConfig {
```

```
    /**
     * 输入图像的缩放系数
     */
    public float scaleRatio = -1;

    /**
     * 需要检测的最大人脸数目
     */
    public int maxDetectNum = 10;

    /**
     * 需要检测的最小人脸大小
     */
    public int minFaceSize = 0;

    /**
     * 人脸置信度阈值（检测分值大于该阈值认为是人脸）
     * RGB
     */
    public float notRGBFaceThreshold = 0.5f;
    /**
     * 人脸置信度阈值（检测分值大于该阈值认为是人脸）
     * NIR
     */
    public float notNIRFaceThreshold = 0.5f;

    /**
     * 未跟踪到人脸前的检测时间间隔
     */
    public float detectInterval = 0;

    /**
     * 已跟踪到人脸后的检测时间间隔
     */
    public float trackInterval = 500;

    /**
     * 质量检测模糊，默认不做质量检测
     */
    public boolean isCheckBlur = false;

    /**
     * 质量检测遮挡，默认不做质量检测
     */
    public boolean isOcclusion = false;

    /**
     * 质量检测光照，默认不做质量检测
     */
    public boolean isIllumination = false;

    /**
     * 姿态角检测，获取yaw(左右偏转角)，roll(人脸平行平面内的头部旋转角)，
     * pitch(上下偏转角),默认不检测
     */
    public boolean isHeadPose = false;
     /**
     * 属性检查，获取年龄，种族，是否戴眼镜等信息，默认不检测
     */
    public boolean isAttribute = false;

    /**
     * 是否扣图，默认不扣图
     */
    private boolean isCropFace = false;

    /**
```

```
     * 是否检测眼睛闭合，默认不检测
     */
    public boolean isEyeClose = false;

    /**
     * 是否检测嘴巴闭合，默认不检测
     */
    public boolean isMouthClose = false;

    /**
     * 是否开启最优人脸检测，默认步开启
     */
    public boolean isBestImage = false;
}
```

1.5 Feature 实体类

```
public class Feature {
    private int id;
    private String faceToken = "";
    private byte[] feature;
    private String userId = "";
    private String groupId = "";
    private long ctime;
    private long updateTime;
    private String imageName = "";
    private String userName = "";
    private String cropImageName = "";
    private boolean isChecked;
    private float score;
}
```

1.6 BDFaceOcclusion 实体类

```
public class BDFaceOcclusion {
   public float leftEye;        // 左眼遮挡置信度
   public float rightEye;       // 右眼遮挡置信度
   public float nose;           // 鼻子遮挡置信度
   public float mouth;          // 嘴巴遮挡置信度
   public float leftCheek;      // 左脸遮挡置信度
   public float rightCheek;     // 右脸遮挡置信度
   public float chin;           // 下巴遮挡置信度
   }
```

1.7 BDFaceGazeInfo实体类

```
public float leftEyeConf;                // 左眼的置信度
public float rightEyeConf;               // 右眼的置信度
public BDFaceGazeDirection leftEyeGaze;  // 左眼的注意力信息
public BDFaceGazeDirection rightEyeGaze; // 右眼的注意力信息
```

1.8 BDFaceDetectListConf 实体类

```
/**
     * 检测（在track 精度不足时开启）默认不做检测
     */
    public boolean usingDetect = false;
```

```
    /**
     * 检测（在track 精度不足时开启）默认开启对齐
     */
    public boolean usingAlign = true;
    /**
     * 质量检测模糊，默认不做质量检测
     */
    public boolean usingQuality = false;
    /**
     * 姿态角检测，获取yaw(左右偏转角)，roll(人脸平行平面内的头部旋转角)，pitch(上下偏转角),默认不检测
     */
    public boolean usingHeadPose = false;
     /**
     * 属性检查，获取年龄，种族，是否戴眼镜等信息，默认不检测
     */
    public boolean usingAttribute = false;
    /**
     * 是否检测眼睛闭合，默认不检测
     */
    public boolean usingEyeClose = false;
    /**
     * 是否检测嘴巴闭合，默认不检测
     */
    public boolean usingMouthClose = false;
    /**
     * 是否开启最优人脸检测，默认步开启
     */
    public boolean usingBestImage = false;
```

1.9 BDFaceDriverMonitorInfo 驾驶检测结果信息

```
public class BDFaceDriverMonitorInfo {
    float normal = 0;    // 行为正常
    float calling = 0;   // 打电话
    float drinking = 0;  // 喝水
    float eating = 0;    // 吃东西
    float smoking = 0;   // 抽烟
}
```

1.10 BDFaceCropParam 扣图设置参数

```
public class BDFaceCropParam {
    // 额头扩展，大于等于0，0：不进行扩展
    public float foreheadExtend;

    // 下巴扩展，大于等于0，0：不进行扩展
    public float chinExtend;

    // 人脸框与背景比例，大于等于1，1：不进行扩展
    public float enlargeRatio;

    // 输出图像宽，设置为有效值(大于0)则对图像进行缩放，否则输出原图抠图结果
    public int width;

   // 输出图像高，设置为有效值(大于0)则对图像进行缩放，否则输出原图抠图结果
    public int height;
};
```

2.1 FaceAuth鉴权接口

2.1.1 鉴权– 应用鉴权

说明：用户通过读取asset目录下的鉴权文件鉴权

```
void initLicense(final Context context, final String licenseKey, final String licenseFileName, final boolean isRemo
```

参数名	含义
context	当前上下文
licenseKey	鉴权key
licenseFileName	鉴权文件名称
isRemote	是否远程拉取
callback	鉴权结果 void onResponse(int code, String response) code 0 : 成功；code 1 加载失败 response 结果信息

2.1.2 鉴权– 在线鉴权

说明：用户通过申请授权码，在线授权，激活设备

```
void initLicenseOnLine(final Context context, final String licenseKey, final AuthCallback callback)
```

参数名	含义
context	当前上下文
licenseKey	AIPE 鉴权码
callback	鉴权结果 void onResponse(int code, String response) code 0 : 成功；code 1 加载失败 response 结果信息

2.1.3 鉴权-离线授权

说明：用户申请鉴权文件，放在SD 卡下，点击按钮直接鉴权

```
void initLicenseOffLine(final Context context, final Callback callback)
```

参数名	含义
context	当前上下文

参数名	含义
callback	鉴权结果 void onResponse(int code, String response) code 0 : 成功；code 1 加载失败 response 结果信息

2.1.4 鉴权-在线批量授权

说明：用户通过申请在线licenseID，不需要输入任何信息，直接网络请求获取鉴权文件

```
void initLicenseBatchLine(final Context context, final String licenseKey, final Callback callback)
```

参数名	含义
context	当前上下文
licenseKey	鉴权文件Key
callback	鉴权结果 void onResponse(int code, String response) code 0 : 成功；code 1 加载失败 response 结果信息

2.1.5 鉴权-加密芯片授权

说明：用户设备已安装加密芯片，直接调用接口授权即可

```
void initLicenseAuthChip(final Context context, final Callback callback)
```

参数名	含义
context	当前上下文
callback	鉴权结果 void onResponse(int code, String response) code 0 : 成功；code 1 加载失败 response 结果信息

2.1.6 开启底层Log 输出

说明：用于Debug 时候输出LOG 详细信息

```
void setActiveLog(BDFaceLogInfo logInfo)
```

参数名	含义
BDFaceLogInfo	底层log 打印 BDFACE_LOG_VALUE_MESSAGE, // 打印输出值日志 BDFACE_LOG_ERROR_MESSAGE, // 打印输出错误日志 BDFACE_LOG_ALL_MESSAGE, // 打印所有日志

2.1.7 设置核数

说明：根据开发板类型，设置加速对Cpu 核数依赖，调整参数，提高性能

```
setCoreConfigure(BDFaceSDKCommon.BDFaceCoreRunMode runMode, int coreNum)
```

参数名	含义
runMode	推荐使用0， 1， 3；如果有需要绑核的话，用0和1；如果不需要绑核，系统自动调度的话，用3 BDFACE_LITE_POWER_HIGH=0, 绑定大核运行模式。如果ARM CPU支持big.LITTLE，则优先使用并绑定Big cluster。如果设置的线程数大于大核数量，则会将线程数自动缩放到大核数量。如果系统不存在大核或者在一些手机的低电量情况下会出现绑核失败，如果失败则进入不绑核模式。 BDFACE_LITE_POWER_LOW=1,绑定小核运行模式。如果ARM CPU支持big.LITTLE，则优先使用并绑定Little cluster。如果设置的线程数大于小核数量，则会将线程数自动缩放到小核数量。如果找不到小核，则自动进入不绑核模式。 BDFACE_LITE_POWER_FULL=2,大小核混用模式。线程数可以大于大核数量。当线程数大于核心数量时，则会自动将线程数缩放到核心数量。 BDFACE_LITE_POWER_NO_BIND=3,不绑核运行模式（推荐）。系统根据负载自动调度任务到空闲的CPU核心上。 BDFACE_LITE_POWER_RAND_HIGH=4,轮流绑定大核模式。如果Big cluster有多个核心，则每预测10次后切换绑定到下一个核心。 BDFACE_LITE_POWER_RAND_LOW=5,轮流绑定小核模式。如果Little cluster有多个核心，则每预测10次后切换绑定到下一个核心。

coreNum | 根据如下命令，查看cpu 核数，选择线程数
adb shell
cat /proc/cpuinfo
|

2.1.8 code返回值

说明: 根据code返回值来判断返回错误信息

参数名	含义
code	code 0 成功； code 1 context 为null或者模型路径错误； code == -1 非法的参数；code -2 内存分配失败； code -3 实例对象为空；code == -4 模型内容为空；

参数名	含义
	code -5 不支持的能力类型；code -6 不支持预测类型； code == -7 预测库对象创建失败；code -8 预测库初始化失败； code -9 图像数据为空；code == -10 人脸能力初始化失败； code -11 能力未加载 ；code -12 人脸能力已加载； code == -13 未授权；code -14 人脸能力运行异常 ； code -15 不支持的图像类型；code == -16 图像转换失败；

2.2 FaceDetect 检测接口

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceDetect()
// 有参构造调用（执行自己创建的instance ）
public FaceDetect(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.2.1 检测对齐模型加载

说明：检测模型加载，目前支持可见光模型，近红外检测模型（非必要参数，可以为空），对齐模型

```
void initModel(final Context context, final String visModel,final String nirModel,final String alignModel, final Ca
```

参数名	含义
context	上下文context
visModel	可见光图片检测模型
nirModel	红外图片检测模型
alignModel	对齐模型
callback	模型加载结果 void onResponse(int code, String response) code：请参照此文档 2.1.7 code返回值 response 结果信息

2.2.2 检测对齐模型加载

说明：检测对齐模型加载

```
public void initModel(final Context context,final String detectModel,final String alignModel,final BDFaceSDKCommon.I
```

参数名	含义
context	上下文context
detectModel	检测模型
alignModel	对齐模型
detectType	检测类型 DETECT_VIS 为可见光； DETECT_NIR为近红外
alignType	对齐类型 BDFACE_ALIGN_TYPE_RGB_ACCURATE 为可见光对齐类型; BDFACE_ALIGN_TYPE_NIR_ACCURATE 为 近红外对齐类型
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.2.3 质量检测模型加载

说明：质量检测模型加载，判断人脸遮挡信息，光照信息，模糊信息，模型包含模糊模型，遮挡信息，作用于质量检测接口

```
void initQuality(final Context context, final String blurModel, final String occlurModel, final Callback callback)
```

参数名	含义
context	上下文context
blurModel	模糊检测模型
occlurModel	遮挡检测模型
callback	鉴权结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.2.4 属性模型加载

说明：人脸属性（年龄，性别，戴眼镜等），情绪（喜怒哀乐）模型初始化

```
void initAttrbute(final Context context,final String atttibuteModel,final Callback callback)
```

参数名	含义
context	上下文context
atttibuteModel	属性检测模型
callback	鉴权结果 void onResponse(int code, String response) code 0 : 成功；code 1 加载失败 response 结果信息

2.2.5 眼睛闭合，嘴巴闭合模型加载

说明：人脸眼睛闭合，嘴巴闭合模型初始化

```
void initFaceClose(final Context context,final String eyecloseModel,final String mouthcloseModel,final Callback cal
```

参数名	含义
context	上下文context
eyecloseModel	人脸眼睛闭合模型
mouthcloseModel	人脸嘴巴闭合模型
callback	鉴权结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.2.6 最优人脸模型加载

说明：最优人脸模型初始化

```
void initBestImage(final Context context,final String bestModel,final Callback callback)
```

参数名	含义
context	上下文context
bestModel	最优人脸模型地址
callback	鉴权结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.2.7 配置信息加载

说明：检测最小人脸，是否开启内部质量检测，检测或者追踪时间间隔等配置

```
void loadConfig(BDFaceSDKConfig config)
```

参数名	含义
config	参数配置实体

2.2.8 人脸框检测

说明：人脸框检测，每一帧图片都会检测，返回基本人脸信息和72 关键点，可以绘制人脸框，描绘眼耳鼻嘴关键点，也可作用于后续活体，特征抽取入参。

2.2.8.1 根据图片进行检测

```
FaceInfo[] detect(BDFaceSDKCommon.DetectType type,BDFaceImageInstance imageInstance)
```

参数名	含义
detectType	检测类型
imageInstance	图片数据信息
FaceInfo[]	返回参数(人脸参数信息)

2.2.8.2 根据传入的人脸框信息进行检测

说明：可灵活配置的人脸检测接口，使用输入的faceInfos通过配置bdFaceDetectListConfig可以控制是否进行一下能力的预测：人脸检测，关键点提取，头部姿态角，光照，模糊，属性，情绪，闭眼，闭嘴。输出返回值为预测后的faceInfos，可作用于后续活体，特征抽取入参。

```
FaceInfo[] detect(BDFaceSDKCommon.DetectType type,BDFaceSDKCommon.AlignType alignType, BDFaceImageInstance imageIns
```

参数名	含义
detectType	检测类型
alignType	对齐类型
imageInstance	图片数据信息
faceInfos[]	人脸框数据(可以通过人脸追踪能力获取人脸框)
bdFaceDetectListConfig	功能开关
FaceInfo[]	返回参数(人脸参数信息)

2.2.9 人脸跟踪-多人脸检测 (接口只支持RGB跟踪)

说明：视频人脸跟踪检测，追踪图片中多个人脸信息，通过参数num 配置，接口包含检测和跟踪功能，返回基本人脸信息和72 关键点，可以绘制人脸框，描绘眼耳鼻嘴关键点，也可作用于后续活体，特征抽取入参。 该接口只支持RGB检测和对齐；

```
FaceInfo[] track(DetectType detectType, BDFaceImageInstance imageInstance)
```

参数名	含义
detectType	检测类型 （ DETECT_VIS ）
imageInstance	图片数据信息
FaceInfo[]	返回参数(人脸参数信息)

2.2.10 人脸跟踪-多人脸检测

说明：视频人脸跟踪检测，追踪图片中多个人脸信息，通过参数num 配置，接口包含检测和跟踪功能，返回基本人脸信息和72 关键点，可以绘制人脸框，描绘眼耳鼻嘴关键点，也可作用于后续活体，特征抽取入参。

```
FaceInfo[] track(BDFaceSDKCommon.DetectType detectType, BDFaceSDKCommon.AlignType alignType,BDFaceImageInstance ima
```

参数名	含义
detectType	检测类型
alignType	对齐类型
imageInstance	图片数据信息
FaceInfo[]	返回参数(人脸参数信息)

2.2.11 检查模型卸载

```
void uninitModel()
```

2.3 FaceLive活体接口

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceLive()
// 有参构造调用（执行自己创建的instance ）
public FaceLive(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.3.1 活体模型加载

说明：静默活体检测模型初始化，可见光活体模型，深度活体，近红外活体模型初始化

```
void initModel(final Context context,final String visModel,final String vis2dmaskModel,final String visHandModel,fi
```

参数名	含义
context	上下文context
visModel	可见光图片活体模型
vis2dmaskModel	2d_mask多因子活体模型
visHandModel	屏幕多因子活体模型
visReflectionModel	手部多因子活体模型
nirModel	红外图片活体模型
depthModel	深度图片活体模型
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.3.2 人脸静默活体检测

说明：静默活体分值检测，返回0-1结果，建议超过0.9 为活体

```
public float silentLive(LiveType type, BDFaceImageInstance bdFaceImageInstance, float[] landmarks)
```

参数名	含义
type	LIVEID_VIS 可见光图像静默活体检测 LIVEID_NIR 红外图像静默活体检测
bdFaceImageInstance	图像对象
landmarks	检查后landmark

```
public float silentLive(LiveType type, BDFaceImageInstance bdFaceImageInstance, float[] landmarks, float liveThresh
```

参数名	含义
type	LIVEID_VIS 可见光图像静默活体检测 LIVEID_NIR 红外图像静默活体检测
bdFaceImageInstance	图像对象
landmarks	检查后landmark
liveThreshold	活体阈值，通过该阈值控制多因子阈值，0 ~ 0.6：0.05；0.61 ~ 0.0.8：0.1； 0.81 ~ 1.0：0.5

2.3.3 人脸静默多帧活体检测

说明：静默活体检测，是否为活体， true： 活体， false： 非活体

```
public boolean strategySilentLive(LiveType type, BDFaceImageInstance bdFaceImageInstance,
                                  FaceInfo faceInfo, int strategyCount, float liveThreshold)
```

参数名	含义
type	LIVEID_VIS 可见光图像静默活体检测 LIVEID_NIR 红外图像静默活体检测
bdFaceImageInstance	图像对象
faceInfo	人脸信息
strategyCount	多帧次数
liveThreshold	活体阈值

2.3.4 活体模型卸载

说明：静默活体模型卸载

```
void uninitModel()
```

2.4 FaceFeature特征接口

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceFeature()
// 有参构造调用（执行自己创建的instance ）
public FaceFeature(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.4.1 特征模型加载

说明：离线特征获取模型加载，目前支持可见光模型，近红外检测模型（非必要参数，可以为空），证件照模型；用户根据自己场景，选择相应场景模型

2.4.1.1 旧版本模型加载

```
initModel(final Context context,final String idPhotoModel,final String visModel,final String nirModel,final Callbac
```

参数名	含义
context	上下文context
idPhotoModel	证件照图片模型
visModel	可见光图片模型
nirModel	红外图片模型（非必要参数，可以为空）
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.4.1.1 新版本模型加载

```
initModel(final Context context,final String idPhotoModel,final String visModel,final String nirModel,final String r
```

参数名	含义
context	上下文context
idPhotoModel	证件照图片模型
visModel	可见光图片模型
nirModel	红外图片模型
rgbdModel	RGBD图片模型
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值

参数名	含义
	response 结果信息

2.4.2人脸特征提取

说明：离线特征提取接口，通过featureType 提取不同图片特征数据，函数返回特征个数，特征存储在feature 参数中

2.4.2.1 无RGBD特征提取

```
float feature(FeatureType featureType, BDFaceImageInstance imageInstance,float[] landmarks, byte[] feature)
```

参数名	含义
featureType	BDFACE_FEATURE_TYPE_LIVE_PHOTO 生活照 BDFACE_FEATURE_TYPE_ID_PHOTO 证件照 BDFACE_FEATURE_TYPE_NIR 红外
imageInstance	图像信息
landmarks	检测后产出的数据
feature	出参：人脸特征 feature 数组，默认初始化512空字节
float	返回128个特征数据

2.4.2.1 有RGBD特征提取

```
float featureRGBD(FeatureType featureType, BDFaceImageInstance imageInstance,BDFaceImageInstance imageInstance_depth
```

参数名	含义
featureType	BDFACE_FEATURE_TYPE_LIVE_PHOTO //生活照 BDFACE_FEATURE_TYPE_ID_PHOTO // 证件照 BDFACE_FEATURE_TYPE_NIR // 红外 BDFACE_FEATURE_TYPE_RGBD // RGBD特征提取
imageInstance	RGB图像信息
imageInstance	Depth图像信息
landmarks	检测后产出的数据
feature	出参：人脸特征 feature 数组，默认初始化512空字节
float	返回 256个特征数据

2.4.3 卸载特征模型

```
void uninitModel()
```

2.5 BDFaceImageInstance 图片接口

2.5.1 图片构造

说明：创建图片对象

```
BDFaceImageInstance(byte[] data, int height, int width,BDFaceSDKCommon.BDFaceImageType imageType, float angle, int
```

参数名	含义
data	图片字节数
height	图片高
width	图片宽
imageType	图片类型
angle	图片旋转角度
isMbyteArrayror	是否镜像

2.5.2 图片构造

说明：创建图片对象

```
BDFaceImageInstance(Bitmap bitmap)
```

参数名	含义
bitmap	图片bitmap

2.5.3 图片信息获取（送检）

说明：获取图片对象

```
BDFaceImageInstance getImage()
```

2.5.4 图片销毁

说明：销毁图片对象

```
```

```
int destory();
```

2.6 FaceGaze注意力检测

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceGaze()
// 有参构造调用（执行自己创建的instance ）
public FaceGaze(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.6.1 注意力模型加载

说明：眼睛状态检测，同时判断出左眼，右眼6种状态，分别为向上，向下，向左，向右，向前，闭合

```
void initModel(final Context context, final String gazeModel, final Callback callback)
```

参数名	含义
context	上下文context
gazeModel	眼睛状态模型
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.6.2 注意力状态获取

说明：通过图片和关键点获取眼睛注意力状态信息

```
BDFaceGazeInfo gaze(BDFaceImageInstance imageInstance, float[] landmarks)
```

参数名	含义
imageInstance	图像信息
landmarks	检测后产出的关键点数据

2.6.3 注意力模型卸载

说明：卸载注意力模型

```
int uninitGazeModel()
```

说明：0表示成功，非0失败

2.7 FaceActionLive动作活体

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceActionLive()
// 有参构造调用（执行自己创建的instance ）
public FaceActionLive(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.7.1 动作活体模型加载

说明：加载眼睛闭合，嘴巴闭合模型加载

```
void initActionLiveModel(final Context context,final String eyecloseModel, final String mouthcloseModel,final Callba
```

参数名	含义
context	上下文context
eyecloseModel	人脸眼睛闭合模型
mouthcloseModel	人脸嘴巴闭合模型
callback	鉴权结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.7.2 动作活体检测

说明： 通过图片和关键点获取眼睛状态信息

```
int  actionLive(BDFaceSDKCommon.BDFaceActionLiveType type,BDFaceImageInstance imageInstance, float[] landmarks, Ato
```

参数名	含义
type	动作活体类型
imageInstance	图像信息

参数名	含义
landmarks	检测后产出的关键点数据
exist	（出参）是否存在这个动作； 0为不存在该动作 ；1为存在该动作

2.7.3 清除动作活体历史数据

说明：再进行新一轮的动作活体检测时调用此功能，将之前缓存的人链图片信息清空

```
int clearHistory()
```

说明：0表示成功，非0失败

2.7.3 动作活体相关模型卸载

说明：调用此功能，会卸载掉绑定实例的眼睛闭合，嘴巴闭合模型能力

```
int uninitActionLiveModel()
```

说明：0表示成功，非0失败

2.8 FaceCrop抠图能力

说明：可以根据人脸框或者人脸关键点进行抠图 构造方法：

```
// 无参构造调用（执行默认创建的instance ）
FaceCrop()
// 有参构造调用（执行自己创建的instance ）
FaceCrop(BDFaceInstance thisBdFaceInstance)
```

2.8.1 initFaceCrop抠图能力加载

```
void initFaceCrop(final Callback callback)
```

参数名	含义
callback	鉴权结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.8.2 uninitFaceCrop抠图能力卸载

```
int uninitFaceCrop()
```

参数名	含义
int	0 success；other 参数异常；

2.8.3 cropFaceByBox使用人脸框进行人脸扣图

```
public static BDFaceImageInstance cropFaceByBox(BDFaceImageInstance imageInstance, FaceInfo faceinfo, float enlarge
```

参数名	含义
imageInstance	图片数据信息
faceinfo	包含人脸框的人脸信息
enlargeRatio	抠图放大倍数
correction	是否进行人脸矫正

2.8.4 cropFaceByLandmark使用人脸关键点进行人脸扣图

说明：根据人脸检测结果扣图，扣图结果为矫正之后的人脸信息

```
public BDFaceImageInstance cropFaceByLandmark(BDFaceImageInstance imageInstance,
                                              float[] landmark,
                                              float enlargeRatio,
                                              boolean correction,
                                              AtomicInteger isOutofBoundary)
```

参数名	含义
imageInstance	图片数据信息
landmark	检测后产出数据
enlargeRatio	抠图放大倍数
correction	是否进行人脸矫正
isOutofBoundary	抠图是否：是否超出图像范围（是否有黑边） 0为未超出，1为超出

返回值：扣图图像信息，包含宽，高，图片类型，图片数据

```
BDFaceImageInstance
```

2.8.4 查看人脸是否在边界处

```
public BDFaceIsOutBoundary cropFaceByBoxIsOutofBoundary(
            BDFaceImageInstance imageInstance,
            FaceInfo faceinfo,
            BDFaceCropParam cropParam)
```

参数名	含义
imageInstance	图片数据信息
FaceInfo	检测后产出的人脸信息
cropParam	配置参数

```
public BDFaceIsOutBoundary cropFaceByLandmarkIsOutofBoundary(
            BDFaceImageInstance imageInstance,
            float[] landmark,
            BDFaceCropParam cropParam)
```

参数名	含义
imageInstance	图片数据信息
landmark	关键点信息
cropParam	配置参数

2.8.5 根据人脸box 进行扣图

```
public BDFaceImageInstance cropFaceByBoxParam(
            BDFaceImageInstance imageInstance,
            FaceInfo faceinfo,
            BDFaceCropParam cropParam)
```

参数名	含义
imageInstance	图片数据信息
FaceInfo	检测后产出的人脸信息
cropParam	配置参数

2.8.6 根据人脸关键点进行扣图

```
public BDFaceImageInstance cropFaceByLandmarkParam(
            BDFaceImageInstance imageInstance,
            float[] landmark,
            BDFaceCropParam cropParam)
```

参数名	含义
imageInstance	图片数据信息
landmark	关键点信息
cropParam	配置参数

2.9 FaceDriverMonitor 驾驶行为监测

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
FaceDriverMonitor()
// 有参构造调用（执行自己创建的instance ）
FaceDriverMonitor(BDFaceInstance thisBdFaceInstance)
```

2.9.1 驾驶行为监测能力加载

说明：加载驾驶行为监测能力

```
void initDriverMonitor(final Context context, final String driverMonitorModel, final Callback callback)
```

参数名	含义
context	上下文context
driverMonitorModel	驾驶行为监测模型路径
callback	模型加载结果 void onResponse(int code, String response) code：请参照此文档 2.1.7 code返回值 response 结果信息

2.9.2 驾驶行为监测能力能力卸载

```
void uninitDriverMonitor()
```

2.9.3 驾驶行为监测检测

```
public BDFaceDriverMonitorInfo driverMonitor(BDFaceImageInstance imageInstance,
                                        FaceInfo faceinfo)
```

参数名	含义
imageInstance	图片数据信息
FaceInfo	人脸参数信息（需要包含人脸检测的人脸框数据）
BDFaceDriverMonitorInfo	返回参数：驾驶行为监测结果

2.10 FaceMouthMask口罩检测

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceMouthMask()
// 有参构造调用（执行自己创建的instance ）
public FaceMouthMask(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.10.1 口罩检测模型加载

说明：加载口罩检测模型

```
int initModel(final Context context, final String mouthMaskModel, final Callback callback)
```

参数名	含义
context	上下文context
mouthMaskModel	口罩检测模型
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.10.2 口罩检测结果获取

说明：通过图片和人脸框数据获取口罩检测置信度数据

```
float[] checkMask(BDFaceImageInstance imageInstance, FaceInfo[] faceInfos)
```

参数名	含义
imageInstance	图像信息
faceInfos	人脸框数据
float[]	返回的戴口罩置信度

2.10.3 口罩检测模型卸载

说明：卸载口罩检测模型

```
int uninitModel()
```

说明：0表示成功，非0失败

2.11 ImageIllum 图片光照检测接口

```
int imageIllum(BDFaceImageInstance imageInstance, AtomicInteger illumScore)
```

参数名	含义
imageInstance	图像信息
illumScore	图片光照强度

2.12、FaceSafetyBelt安全带检测

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceSafetyBelt()
// 有参构造调用（执行自己创建的instance ）
public FaceSafetyBelt(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.12.1、模型加载

```
public void initFaceSafetyBelt(final Context context, final String FaceSafetyBeltModel, final Callback callback)
```

参数名	含义
context	上下文context
FaceSafetyBeltModel	安全带检测模型
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.12.2、安全带检测

```
public float faceSafetyBelt(BDFaceImageInstance imageInstance,
                                float[] landmark)
```

参数名	含义
imageInstance	图像信息
landmark	人脸关键点
返回值	0~1之前 0为系安全带， 1为未系安全带

2.12.3、 能力卸载

```
public int uninitFaceSafetyBelt()
```

2.13、FaceDarkEnhance暗光恢复检测

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceDarkEnhance()
// 有参构造调用（执行自己创建的instance ）
public FaceDarkEnhance(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.13.1、模型加载

```
public void initFaceDarkEnhance(final Context context, final String FaceDarkEnhanceModel, final Callback callback)
```

参数名	含义
context	上下文context
FaceDarkEnhanceModel	暗光恢复模型
callback	模型加载结果 void onResponse(int code, String response) code : 请参照此文档 2.1.7 code返回值 response 结果信息

2.13.2、暗光检测

```
public BDFaceImageInstance faceDarkEnhance(BDFaceImageInstance imageInstance)
```

参数名	含义
imageInstance	图像信息
返回值	暗光增强后的图像信息

2.13.3、能力卸载

```
public int uninitFaceDarkEnhance()
```

2.14、FaceSafetyHat安全帽检测

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceSafetyHat()
// 有参构造调用（执行自己创建的instance ）
public FaceSafetyHat(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.14.1、模型加载

```
public void initModel(final Context context, final String safetyHatModel, final Callback callback)
```

参数名	含义
context	上下文context
safetyHatModel	安全帽模型
callback	模型加载结果 void onResponse(int code, String response) code：请参照此文档 2.1.7 code返回值 response 结果信息

2.14.2、安全帽检测

```
public float[] checkHat(BDFaceImageInstance bdFaceImageInstance, FaceInfo[] faceInfos)
```

参数名	含义
imageInstance	图像信息
faceInfos	人脸信息
返回值	安全帽置信度，例如：单个人脸，取值为：float[0]

2.14.3、能力卸载

```
public int uninitModel()
```

2.15、FaceSearch 人脸比对检索检测

构造方法：

```
// 无参构造调用（执行默认创建的instance ）
public FaceSearch()
// 有参构造调用（执行自己创建的instance ）
public FaceSearch(BDFaceInstance thisBdFaceInstance)
```

参数名	含义
thisBdFaceInstance	绑定指定的BDFaceInstance实例，否则使用默认的BDFaceInstance实例

2.15.1 入库回调

```
public interface InputDBListener {
        void onInputDB(int id, int index);
    }
```

2.15.1 参数设置

```
// 是否要入库， true:是；false:否； 默认false
public void setNeedJoinDB(boolean needJoinDB)
// 注册照比对阈值 取值范围：0-1，默认：0.8
public void setRegisterCompareThreshold(float registerCompareThreshold)
// 更新照比对阈值 取值范围：0-1， 默认：0.9
public void setUpdateCompareThreshold(float updateCompareThreshold)
// 入库阈值 取值范围：0-1 默认：0.92
public void setInputDBThreshold(float inputDBThreshold)
// 最大更新数量 默认：10
public void setMaxUpdateSize(int maxUpdateSize)
// 入库间隔时长 默认：24小时
public void setInputDBIntervalTime(long intervalTime)
// 获取缓存库内所有人员特征点
public Map<Integer, byte[]> getFeatureMap()
// 获取缓存库内数目
 public int getSize()
```

2.15.2 1:1 比对

```
public float compare(BDFaceSDKCommon.FeatureType featureType, byte[] feature1, byte[] feature2, boolean isPercent)
```

参数名	含义
featureType	参照FeatureType
feature1	特征1
feature2	特征2
isPercent	控制参数：true返回0~100数值；false 返回0~1
float	比对结果

2.15.3 1:N特征设置

说明： 特征集合预加载接口，继承Feature，必须初始化id 和 feature 字段，用于1：N 内部实现和数据返回。

2.15.3.1 批量添加数据，主要用于程序启动时

```
public int pushPersonFeatureList(List<? extends Feature> features)
```

参数名	含义
features	特征集合

2.15.3.2 添加单个数据，主要用于人脸注册以及其他单个人脸添加

```
public int pushPersonById(int pointID, byte[] feature)
```

参数名	含义
pointID	人员ID
features	人员特征

2.15.3.3 根据人员ID删除单个特征数据， 主要用于人脸库删除操作时使用

```
public int delPersonById(int pointID)
```

参数名	含义
pointID	人员ID

2.15.4 1:N检索

说明：当前feature和预加载Feature 集合比对，返回预加载Feature集合中命中的id，feature 字段和比对分值score；用户可以通过id 在数据库中查找全量信息。

```
public List<? extends Feature> search(BDFaceSDKCommon.FeatureType featureType,
                                      float threshold,
                                      int topNum,
                                      byte[] feature,
                                      boolean isPercent)
```

参数名	含义
featureType	FeatureType.FEATURE_VIS生活照 FeatureType.FEATURE_ID_PHOTO证件照照
threshold	比对阈值
topNum	获取前num 个feature+id映射数组
feature	当前检查人脸特征值
isPercent	控制参数：true返回0~100数值；false 返回0~1

```
public List<? extends Feature> search(BDFaceSDKCommon.FeatureType featureType, float threshold,
                                      byte[] feature,
                                      boolean isPercent)
```

参数名	含义
featureType	FeatureType.FEATURE_VIS生活照 FeatureType.FEATURE_ID_PHOTO证件照照
threshold	比对阈值
feature	当前检查人脸特征值
isPercent	控制参数：true返回0~100数值；false 返回0~1

```
public List<? extends Feature> search(BDFaceSDKCommon.FeatureType featureType, int topNum,
                                      byte[] feature,
                                      boolean isPercent)
```

参数名	含义
featureType	FeatureType.FEATURE_VIS生活照 FeatureType.FEATURE_ID_PHOTO证件照照
topNum	获取前num 个feature+id映射数组
feature	当前检查人脸特征值
isPercent	控制参数：true返回0~100数值；false 返回0~1