Linux arm 人脸识别离线 sdk

用户接入文档

文档名称	人脸识别 linux arm 离线 sdk 用户接入文档
所属平台	Linux C++
文档提交日期	2023.06.12

百度在线网络技术（北京）有限公司

修改记录

No	版本号	修改内容简介	修改日期	修改人
1	V5.0	模型能力等升级到 5.0	2021-03-11	
2、	V7.0	模型能力升级到 7.0，修改设备指纹的小部分 bug 等	2021-08-30	
3、	V7.1	添加人脸库、业务层封装等	2021-12-28	
4	V7.2	1、 修改人脸注册不支持中文的问题 2、 人脸检测框中的 angle 参数删除，检测人脸姿态角可通过 head_pose 实现 3、 其他多项文档优化	2022-12-20	
5	V7.3	1、 更新 rgb 和 depth 等活体模型 2、 优化口罩检测模型 3、 支持多实例多线程 4、 支持在线激活 5、 新增三分类表情识别能力 6、 其他各项文档优化	2023-06-12	

目 录

1 设计背景

- **场景特点：**

-- 网络：对于无网、局域网等情况，无法连接公网，API 方式无法运作。如政府单位、金融保险、教育机构等，其中内网情况最为常见，私有化部署是项目开展的前提条件。

-- 安全：即使可以连接外网，因为人脸数据的敏感性，许多客户不希望将人脸数据传入百度服务器，如大学学生照片、部分企业员工数据等，API 形式也往往不被接受。

-- 速度：由于各地网络线路、机房部署、图片采集方式等诸多原因，API 形式往往耗时较高，容易存在部分请求耗时过长的情况，容易影响业务正常运转。

-- 稳定：API 形式容易受网络抖动、机房故障、线上连带 bug 等影响，存在一定的不稳定因素，可用性保障，往往成为在线调用最容易出现问题的地方。

- **客户特点：**

-- 1：N-小型人脸库检索：多为通道通行、固定区域人群验证等需求，如写字楼闸机门禁、企业考勤打卡等，人脸库范围较小，且不易经常变动。

-- 1：1-自有数据源对比：将当前采集的人脸，与其他数据源中的人脸进行对比，如身份证芯片照、教务系统图片、档案图片等，进行快速的 1：1 对比验证。

- **核心需求：**

-- 基础的人脸采集：包含人脸检测、跟踪、捕获、质量校验等基础功能，获取符合识别条件的人脸。为之前的客户端 SDK 的标准功能，离线版本 SDK 保留以上所有能力。

-- 本地特征抽取：所有在 SDK 中运行的人脸图片，都可以完成本地特征抽取，以便进行对比或识别操作。

-- 1：1 对比：支持两张图片的相似度对比，可直接传入图片，也可调用本地某个人脸特征；

-- 1：N 搜索：支持一定库大小的人脸查找，在指定的人脸集合中查找最相似的人脸，并返回相似度分值；

2 名词解释

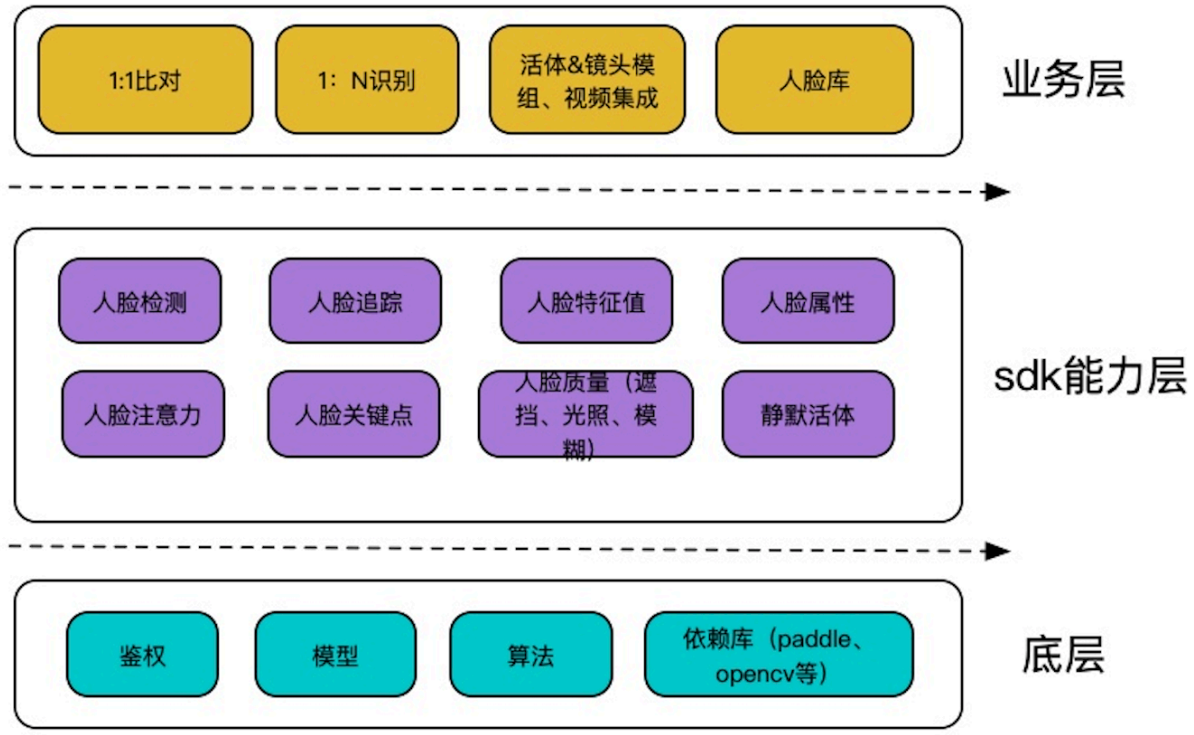
名词	定义
sdk	linux arm C++离线人脸识别 sdk
g++	linux c++编译工具，如 rk3288、3399 等系统一般自带
license	人脸识别激活所需要的激活文件,文档介绍了三种激活方式
key	人脸激活所需的序列号，可从百度 AI 官网申请（ai.baidu.com）
feature	人脸特征值，用来表示人脸特征的 128 个 float 浮点值
landmark	人脸关键点（72 个关键点）
face_token	对应人脸图片的唯一编码，若一个人上传了 2 张不同图片，则可能有 2 个不同的 face_token,它和图片一一对应

3 sdk 简介

本 sdk 适应于 linux arm 平台下的人脸识别系统,为支持 c++语言开发的 sdk,开发者可在 linux arm 平台下面进行开发（支持 armv7hf、armv8 平台）。sdk 采用导出动态库 dll 的方式提供接口，sdk 附带一个示例工程 face_offline_sdk，提供了 sdk 的各种能力及调用示例。

3.1 功能架构

sdk 具有人脸检测、追踪、特征值、静默活体、人脸库、镜头模组集成等诸多功能。架构如下图所示：



3.2 版本及兼容性

本 sdk 支持 armv7hf、armv8 两种 liunx arm 平台。各平台推荐代表如：

armv7hf 架构的如 rk3288 系列。

armv8 架构的如 rk3399 系列。

推荐使用 linux arm 平台上直接编译。

3.3 直接编译和运行

sdk 支持在开发板上直接编译运行。

首先要安装 gcc，g++，可参考文档

https://blog.csdn.net/weixin_31445167/article/details/116887403

利用命令如：sudo apt update 进行更新

sudo apt install build-essential 进行安装。

另外：若开发板还没有 cmake，也需要安装，具体可自行 baidu 一下安装方法。

sdk 支持 ssh 网络登录开发板直接命令行编译。可参考 sdk 中的编译 txt 文件进行编译运行。

在 sdk 工程目录：

```
mkdir build                  #新建 build 目录
cd build                     #进入 build 目录
cmake -DARCH_ABI=armv8 ..    #根据平台分别用命令 armv7hf 或 armv8
make -j4                     #或 make 编译、根据开发板是否支持多核
cd ..                        #回退到 sdk 工程目录
./run.sh armv8               #不同平台请分别修改为如 armv7hf 或 armv8。
```

4 sdk 工程结构

5 授权激活

sdk 需要授权激活后才能正常使用，在 sdk 初始化后，若报错误码-13（错误码参考文档最后定义），一般为没有通过授权。通常，sdk 分按设备授权和按应用授权两种方式，大部分采用按设备授权的方式，按应用授权可针对批量大规模客户使用（文档中先只介绍按设备授权，按应用授权可工单或联系百度商务我们提供另外的文档或技术支持）。按设备授权的方式中，sdk 自动激活和激活工具激活需要设备能联网，若设备不能连外网，可采用官网离线激活的方式。

5.1 sdk 自动激活

在 sdk 的目录中有 license 文件夹，里面存放了 2 个文件，license.key 和 license.ini，分别是授权 key 和授权文件，若在百度官网申请了授权系列号 key（16 位），可按 sdk 中的 license 文件夹中 license.key 原格式样子覆盖填写您申请的 key。在设备能联网的情况下，运行 sdk 会自动授权激活并拉取新的授权文件 license.ini 覆盖 sdk 中的旧文件。

5.2 官网离线激活(license_tool)

若设备不能连外网、通过授权还有另外一种方法，即通过运行 sdk 的示例代码 face.cpp，可得到设备指纹 device id，获取到设备硬件指纹信息后，通过百度官网填入指纹信息和申请到的系列号 key，可完成激活并下载获取到 license.ini 文件和 license.key 文件，把这 2 个文件拷贝到 face_offline_sdk 的 license 目录下，重新运行 sdk 亦可通过授权激活。

6 sdk 集成及 demo 示例工程

6.1 sdk 的集成

sdk 集成的 include 需包含 sdk 的接口头文件：baidu_face_api.h、以及 struct_info.h（定义 sdk 的结构体类）两个文件。（若需要使用 face_scene 文件夹里面的原子方法，则需要引入整个 sdk 文件夹的 include 目录）。动态库文件在 lib 文件夹下，根据平台区分 armv7hf 或 armv8 文件夹里面，动态库需要集成时候全部引入。各动态库文件说明如下

so 文件名称	so 文件说明	是否可删除
libbaidu_face_api.so	百度人脸库文件	否
libface_sdk.lib	百度人脸库文件	否
libopencv_world.so	opencv 库文件，用来显示图片，视频等	否、与之一起的有 so.4.1、so.4.1.0
libcurl.so	联网库文件	否、与之一起的有 so.4、so.4.1
libssl.so	openssl 库文件（https 联网）	否、与之一起的有 so.1.1
libcrypto.so	openssl 库文件（https 联网）	否、与之一起的有 so.1.1

6.2 sdk 的使用示例

sdk 工程 face_offline_sdk 包含了 sdk 接口及 demo 示例工程。其中：sdk 接口头文件为前述 include 目录的 baidu_face_api.h，提供了 sdk 的各接口方法。sdk 接口的 lib 库文件为前述 lib 目录。根据平台分别放置在 armv7hf/armv8 目录。其中:baidu_face_api.so 和 face_sdk.so 为人脸 sdk 的 lib 库文件。其他为 curl、opencv 库文件等。

demo 示例工程 face_offline_sdk 展示了如何集成百度人脸识别离线 sdk，并调用 sdk 的方法及使用场景化示例等。

在 face_offline_sdk 中的 face.cpp 的 main()方法中，有使用 sdk 的各个接口方法示例。接入 sdk 及其简单，如下图及解释：

```
// 入口函数
int main()
{
    //api 实例指针
    BaiduFaceApi *api = new BaiduFaceApi();
    //初始化 sdk
    std::cout << "before sdk_init" << std::endl;
    int res = api->sdk_init(nullptr);
    std::cout << "after sdk_init" << std::endl;
    if (res != 0)
    {
        std::cout << "sdk init result is:" << res << std::endl;
        getchar();
        return 0;
    }
    std::time_t time_begin = get_timestamp();
    FaceDemo *demo = new FaceDemo(api);
    demo->face_demo();
    delete demo;
    std::time_t time_end = get_timestamp();
```

```
    std::cout << "time  cost is :" << time_end - time_begin << "ms" << std::endl;
    std::cout << "before delete api" << std::endl;
    // 释放 sdk 实例指针，防止内存泄漏
    delete api;
    getchar();
    std::cout << "end main" << std::endl;
    return 0;
}
```

sdk 使用主要三步：1）初始化实例指针 2）初始化 sdk

第 3 步即可调用示例 demo，实现需要的功能。末尾需要释放 sdk 实例化指针 api。

另外，可通过 is_auth()方法查看是否通过了授权，通常，需要通过授权，才可以使用 sdk 的各种能力。

示例工程中：分别有以下几个文件夹放置文件对应几个常用 sdk 的调用 demo。

解析如下：

文件夹或文件名	文件说明
face.cpp	sdk 总入口，初始化，模型加载，销毁等,见 main 方法
face_demo.cpp	demo 总入口，可打开注释运行各类 demo 示例
face_detect	人脸检测接口及示例 demo
face_track	人脸跟踪接口及示例 demo
face_feature	人脸特征值接口及示例 demo
face_compare	人脸 1:1&1:N(1:N 需要先人脸入库）比对，特征值比较等接口及示例 demo
face_liveness	可见光 RGB、红外 IR 活体检测、RGB&IR 双目摄像头静默活体检测，RGB&DEPTH 双目摄像头静默活体检测等
face_manager	人脸库管理类，包括人脸注册、删除，更新、组管理，人脸

	信息查询等
face_attr	人脸属性(年龄、性别、种族等）接口及示例
face_head_pose	人脸姿态角接口及示例
face_illumination	人脸光照检测接口及示例
face_occlusion	人脸遮挡度检测接口及示例
face_blur	人脸模糊度检测接口及示例
face_crop	人脸扣图接口及示例
face_gaze	双眼注意力检测接口及示例
face_eye_close	眼睛闭合检测接口及示例
face_mouth_close	嘴巴闭合检测接口及示例
face_mouth_mask	是否佩戴口罩检测接口及示例
sdk_info	实现如读取 sdk 版本号、设备指纹等接口示例
driver_monitor	驾驶行为检测示例，如打电话、吃东西等
safety_belt	安全带佩戴检测示例
face_action_live	动作活体检测示例

7 模型能力加载及模型说明

7.1 模型删减说明

sdk 支持按需配置模型和加载能力，若 sdk 有某部分功能不需要使用，可尝试删除一些模型，删除后模型即不会加载也不会占用内存。sdk 中 models 文件夹里的模型及是否可删减说明如下表:

模型文件夹名称	说明	是否可删减	删减说明
detect	人脸检测模型	否	该文件夹不能删减

align	人脸关键点模型	否	该文件夹不能删减
attribute	人脸属性	是	若不使用该功能，该文件夹可删除
best_image	最佳人脸	是	若不使用该功能，该文件夹可删除
blur	人脸质量模糊度检测	是	若不使用该功能，该文件夹可删除
dark_enhance	暗光恢复	是	若不使用该功能，该文件夹可删除
eye_close	眼睛闭合	是	若不使用眼睛闭合及动作活体功能，该文件夹可删除
feature	人脸特征值	是	若不用 nir 近红外的人脸特征值，可删除 feature_nir 开头的文件
gaze	人脸注意力检测	是	若不使用该功能，该文件夹可删除
mouth_close	嘴巴闭合检测	是	若不使用嘴巴闭合及动作活体功能，该文件夹可删除
mouth_mask	口罩佩戴检测	是	若不使用该功能，该文件夹可删除
occlusion	人脸遮挡检测	是	若不使用该功能，该文件夹可删除
silent_live	静默活体检测	是	若只使用 rgb 可见光单目静默活体，则除 liveness_rgb 开头的文件外其他皆可删除
driver_monitor	安全驾驶	是	若不使用驾驶行为检测和安全带佩戴检测，该文件夹可删除

7.2 模型路径的定制化

sdk 支持模型文件夹 models 的路径自定义，当 sdk 初始化 api->sdk_init(nullptr)传 null 时候，为 sdk 支持模型文件夹路径在默认路径，即

models 在 sdk 现有位置。同时也支持 models 通过 sdk_init 中传入绝对路径。若把 models 文件夹拷贝到如/home/face 文件夹下面，则可定义：

api->sdk_init(“/home/face”);

此时，授权文件夹 license 也需要随之变为/home/face 文件夹下面。否则会出现授权不通过的问题。

另外，若使用了能力自定义的 config 文件夹，也需要拷贝到/home/face 文件夹下面。否则能力定制化也不会生效而是使用的系统默认。

7.3 能力定制化说明

sdk 支持能力自定义、通过读取配置文件的方式进行能力定制化。sdk 默认能力加载无需定制化，若需要定制化，请把 sdk 根目录里面的 conf 文件夹重命名为 config 文件夹。并且在 sdk 中，把里面的 json 配置按如下说明做修改，可起到定制化能力加载的效果，配置文件简要说明如下（若需定制化修改，请参考示例 json，修改 json 字段的值来达到定制化的目的）

7.3.1 detect.json （人脸检测能力定制配置文件）

配置文件名	detect.json	

说明	人脸检测自定义能力配置文件
	{
	"max_detect_num":5,
	"min_face_size":0,
	"scale_ratio":-1,
	"not_face_thr":0.5
	}	

参数	类型	说明
max_detect_num	int	最大检测的人脸数量，最多支持 50，最小 1,默认 5
min_face_size	int	默认 0，可用来设置检测的最小人脸，比如可设为 10，则表示小于 10*10 的人脸，sdk 检测不到
scale_radio	int	默认-1，表示进行人脸检测时候的图片缩放比率。建议用-1 表示传入原图 sdk 自己缩放（为保证检测效果，该参数建议用默认）

not_face_thr	float	默认 0.5，表示非人脸的阈值，取值范围 0-1

7.3.2 track.json （人脸追踪能力定制配置文件）

配置文件名	track.json	

说明	人脸追踪自定义能力配置文件
	{
	"detect_intv_before_track":0.02,
	"detect_intv_during_track":0.02
	}	

参数	类型	说明
detect_intv_before_track	float	表示人脸追踪前进行人脸检测的时间间隔（单位毫秒）
detect_intv_during_track	float	表示人脸追踪时候进行人脸检测的时间间隔（单位毫秒）

7.3.3 action_live.json （动作活体能力定制配置文件）

配置文件名	action_live.json	

说明	人脸动作活体自定义能力配置文件
	{
	"eye_open_threshold":0.5,
	"eye_close_threshold":0.5,
	"mouth_open_threshold":0.5,
	"mouth_close_threshold":0.5,
	"look_up_threshold":0.5,
	"look_down_threshold":0.5,
	"turn_left_threshold":0.5,
	"turn_right_threshold":0.5,
	"nod_threshold":0.5,
	"shake_threshold":0.5,
	"max_cache_num":1
	}	

参数	类型	说明

eye_open_threshold	float	表示人脸动作活体眼睛睁开的置信度阈值
eye_close_threshold	float	表示人脸动作活体眼睛闭合的置信度阈值
mouth_open_threshold	float	表示人脸动作活体嘴巴张开的置信度阈值
mouth_close_threshold	float	表示人脸动作活体嘴巴闭合的置信度阈值
look_up_threshold	float	表示人脸动作活体抬头的置信度阈值
look_down_threshold	float	表示人脸动作活体低头的置信度阈值
turn_left_threshold	float	表示人脸动作活体向左转头的置信度阈值
turn_right_threshold	float	表示人脸动作活体向右转头的置信度阈值
nod_threshold	float	表示人脸动作活体点头的置信度阈值
shake_threshold	float	表示人脸动作活体摇头的置信度阈值
max_cache_num	float	最大缓存数、推荐为 1 不做修改

7.3.4 crop.json (人脸抠图能力定制配置文件)

配置文件名	crop.json	

说明	人脸抠图自定义能力配置文件
	{
	"is_flat":0,
	"crop_size":200,
	"enlarge_ratio":1
	}	

参数	类型	说明
is_flat	int	默认为 0，表示是否是镜像，该参数目前无效
crop_size	int	表示抠图的大小，如 200，则表示抠出来的是 200*200 的图片
enlarge_ratio	float	默认是 1，若小于 1，比如 0.8，图片有点放大，会稍微模糊，建议自定义可设为 1

8 功能接口

sdk 功能接口的调用可参考各示例 cpp 文件，接口定义在 baidu_face_api.h。sdk 实现的主要功能有人脸实时跟踪检测、人脸特征值提取、动作活体、RGB&IR 静默活体检测、RGB&DEPTH 静默活体检测、人脸注册、人脸更新、组管理、用户管理以及

1:1 人脸对比,1:N 人脸识别、特征值的比对和通过 usb 或笔记本自带摄像头检测视频帧，返回识别出的人脸信息、人脸属性等，另外支持对人脸检测进行能力加载设置，达到根据设置进行识别的目的。

各接口功能及传入参数和返回结果等定义如下：

8.1 人脸检测 detect 接口

方法名	detect			
说明	人脸检测，返回人脸信息			
函数	int detect(std::vector<FaceBox> &out, const cv::Mat* mat, int type = 0)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸结构体数组	是	std::vector<FaceBox>	人脸框信息结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
type	检测类型（传 0 表示 rgb 可见光人脸检测，1 表示 nir）	否	int	0 或 1，不传默认为 0
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.2 人脸跟踪 track 接口

方法名	track			
说明	人脸跟踪，返回人脸信息			
函数	int track(std::vector<TrackFaceInfo>&out, const cv::Mat* mat, int type = 0)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸结构体数组	是	std::vector<TrackFaceInfo>	人脸跟踪结构体信息
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
type	检测类型（传 0 表示	否	int	0 或 1，不传默认为 0

	rgb 可见光人脸检测，1 表示 nir）			
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.3 清除人脸跟踪历史接口

方法名	clear_track_history			
说明	清除人脸跟踪的历史信息			
函数	int clear_track_history(int type = 0)			
请求参数	说明	必须	类型	示例描述
type	检测类型（传 0 表示 rgb 可见光人脸检测，1 表示 nir）	否	int	0 或 1，不传默认为 0
返回信息	函数的返回	是	void	

8.4 人脸关键点接口

方法名	face_landmark			
说明	人脸关键点，返回人脸关键点信息			
函数	int face_landmark(std::vector<Landmarks>&out, const cv::Mat* mat, int type)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸关键点数组	是	std::vector<Landmarks>	人脸关键点信息结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
type	检测类型（传 0 可见光生活照，1、表示近红外）	是	int	
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码

				>0 时候为检测到的人脸数量

8.5 注意力检测接口

方法名	face_gaze			
说明	人脸注意力检测，返回人脸注意力信息			
函数	int face_gaze(std::vector<GazeInfo>& out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸注意力结构体数组	是	std::vector<GazeInfo>	注意力结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.6 人脸属性检测接口

方法名	face_attr			
说明	人脸属性检测，返回人脸属性信息			
函数	int face_attr(std::vector<Attribute>& out, const cv::Mat *mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸属性结构体数组	是	std::vector<Attribute>	人脸属性结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.7 人脸扣图接口

方法名	face_crop

说明	人脸扣图，返回人脸扣图（仅支持单人脸抠图）			
函数	int face_crop(cv::Mat&out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	抠图结果图片帧	是	Opencv mat	
mat	传入的 opencv 视频帧	是	Opencv mat	
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.8 暗光恢复接口

方法名	dark_enhance			
说明	暗光恢复，用于图片比较暗的使之变量利于人类检测（仅支持单人脸抠图）			
函数	int dark_enhance(cv::Mat&out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	暗光恢复结果图片帧	是	Opencv mat	
mat	传入的 opencv 视频帧	是	Opencv mat	
返回信息	函数的返回	是	int	返回：<0 为错误码 =0 表示成功

8.9 眼部状态检测接口

方法名	face_eye_close			
说明	人脸眼部状态检测，返回人脸眼部状态信息			
函数	int face_eye_close(std::vector<EyeClose> &out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸眼部状态结构体数组	是	std::vector<EyeClose>	眼部状态信息结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码

				>0 时候为检测到的人脸数量

8.10 嘴巴闭合检测接口

方法名	face_mouth_close			
说明	人脸嘴巴闭合检测，返回人脸嘴巴闭合信息			
函数	int face_mouth_close(std::vector<MouthClose>& out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸嘴巴闭合结构体数组	是	std::vector<MouthClose>	嘴巴闭合结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.11 口罩佩戴检测接口

方法名	face_mouth_mask			
说明	人脸口罩佩戴检测，返回人脸口罩佩戴信息			
函数	int face_mouth_mask(std::vector<MouthMask>& out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸口罩佩戴结构体数组	是	std::vector<MouthMask>	佩戴口罩结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.12 人脸质量

人脸质量判断可由以下几个因素自由组合综合判断，如姿态角、光照、遮挡、模糊以及最佳人脸。人脸质量判断的推荐阈值：人脸检测置信度>0.63、遮挡度 <0.75、

人脸姿态角 <20 、人脸模糊度 <0.8、最优人脸 >50。

8.12.1 人脸姿态角接口

方法名	face_head_pose			
说明	人脸姿态角检测，返回人脸姿态角信息			
函数	int face_head_pose(std::vector<HeadPose>&out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸姿态角数组	是	std::vector<HeadPose>	人脸姿态角信息结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.12.2 人脸光照检测接口

方法名	face_illumination			
说明	人脸光照检测、（光照分值 0-255、分值越大光照越强）			
函数	int face_illumination(std::vector<Illumination>&out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸光照信息数组	是	std::vector<Illumination>	光照置信度结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.12.3 人脸遮挡检测接口

方法名	face_occlusion
说明	人脸遮挡检测（遮挡分值 0-1，分值越大遮挡度越高）

函数	int face_occlusion(std::vector<Occlusion>& out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸遮挡结构体数组	是	std::vector<Occlusion>	遮挡置信度结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.12.4 人脸模糊度检测接口

方法名	face_blur			
说明	人脸模糊检测(模糊度分值 0-1，分值越大模糊度越高)			
函数	int face_blur(std::vector<Blur> &out, const cv::Mat * mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸模糊结构体数组	是	std::vector<Blur>	模糊度结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.12.5 最优人脸检测接口

方法名	face_best			
说明	最佳人脸检测(最优人脸分值 0-100，分值越高，最佳人脸图片得分越高)			
函数	int face_best(std::vector<Best> &out, const cv::Mat *mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的最佳人脸结构体数组	是	std::vector<Best>	最优人脸结构体
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码

				>0 时候为检测到的人脸数量

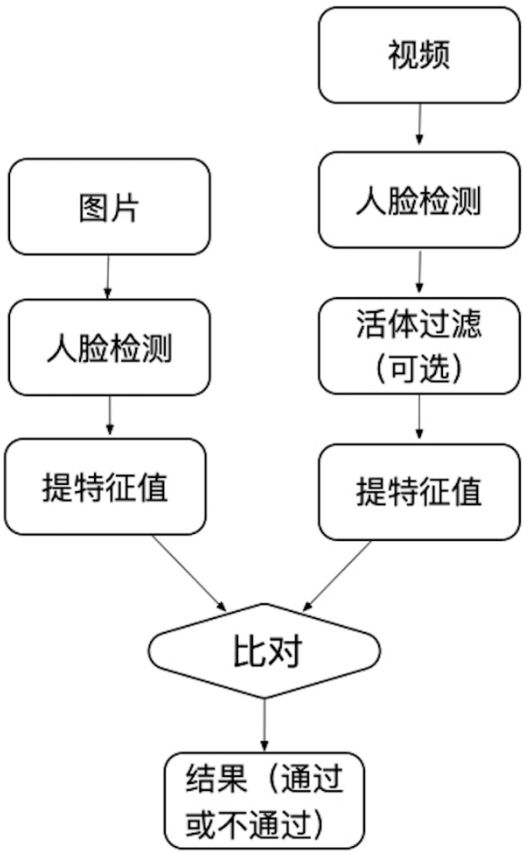
8.12.6 人脸表情接口

方法名	face_emotion			
说明	人脸表情检测，返回人脸表情检测信息			
函数	int face_emotion(std::vector<Emotion>& out, const cv::Mat* mat)			
请求参数	说明	必须	类型	示例描述
out	通过引用返回的人脸表情结构体数组	是	std::vector<Emotion>	表情结构体（三分表情）
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.13 特征值及人脸比对（1:1）

人脸比对的原理实际是特征值比对，通过提取图片中的人脸特征值，根据特征值调用 compare_feature 接口进行比对，推荐比对分值超过 80 分为同一人，可根据实际检测比对情况动态调整。

人脸 1:1 比对流程可如下图，实现如人证比对功能。（人的照片和实时视频比对，可根据使用情况选择是否启用质量检测）



8.13.1 人脸特征值接口

方法名	face_feature			
说明	人脸特征值提取，并返回人脸信息			
函数	int face_feature(std::vector<Feature> &out_fea, std::vector<FaceBox> &out_box, const cv::Mat *mat , int type = 0)			
请求参数	说明	必须	类型	示例描述
out_fea	通过引用返回的人脸特征值结构体数组	是	std::vector<Feature>	人脸特征值结构体信息
out_box	通过引用返回的人脸框结构体数组	是	std::vector<FaceBox>	人脸框结构体信息
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
type	检测类型（传 0 可见光生活照特征值 1、近红外特征值）	否	int	0 或 1, 未传默认为 0

返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.13.2 人脸活体特征值接口

方法名	liveness_feature			
说明	人脸特征值提取，同时有活体校验、并返回活体分值，人脸信息			
函数	int liveness_feature(std::vector<Feature> &out_fea, std::vector<FaceBox> &out_box, float& score, const cv::Mat *mat , int type = 0)			
请求参数	说明	必须	类型	示例描述
out_fea	通过引用返回的人脸特征值结构体数组	是	std::vector<Feature>	人脸特征值结构体信息
out_box	通过引用返回的人脸框结构体数组	是	std::vector<FaceBox>	人脸框结构体信息
mat	传入的 opencv 视频帧	是	Opencv mat	请参考示例
score	返回的活体分值	是	float	
type	检测类型（传 0 可见光特征值，1 表示近红外特征值）	否	int	0 或 1, 未传默认为 0
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.13.3 深度人脸特征值接口

方法名	rgbd_feature			
说明	人脸特征值提取，返回人脸信息			
函数	int rgbd_feature(std::vector<Feature> &out_fea, std::vector<Feature> &out_box, const cv::Mat* rgb_mat, char* depth_mat)			
请求参数	说明	必须	类型	示例描述
out_fea	通过引用返回的人脸特征值结构体数组	是	std::vector<Feature>	人脸特征值结构体信息

out_box	通过引用返回的人脸框结构体数组	是	std::vector<FaceBox>	人脸框结构体信息
rgb_mat	传入可见光的 opencv 视频帧	是	Opencv mat	请参考示例
depth_mat	传入深度的二进制流数据	是	char*	请参考示例
返回信息	函数的返回	是	int	<=0 为未检测到人脸或错误码 >0 时候为检测到的人脸数量

8.13.4 特征值比对接口

方法名	compare_feature			
说明	人脸跟踪，返回人脸信息			
函数	float compare_feature(Feature* f1, Feature *f2, int type)			
请求参数	说明	必须	类型	示例描述
f1	人脸特征值结构体	是	Feature*	人脸特征值结构体信息（仅支持单个结构体的比对）
f2	人脸特征值结构体	是	Feature*	人脸特征值结构体信息（仅支持单个结构体的比对）
type	特征值类型（0,1）	否	int	0：生活照 1：近红外特征值 未传默认为 0
返回信息	函数的返回	是	float	特征值比对分值

8.13.5 人脸 1：1 比对接口

方法名	match			
说明	人脸比对，返回人脸比对分值			
函数	int match(const cv::Mat& img1, const cv::Mat& img2, int type)			
请求参数	说明	必须	类型	示例描述
img1	传入可见光的 opencv 图	是	cv::Mat	

	片帧			
img2	传入可见光的 opencv 图片帧	是	cv::Mat	
type	比对类型（0,1）	否	int	0：生活照模式、包含 1 寸照 1：近红外特征值 未传默认为 0(1 寸或 2 寸的证据照也推荐当作生活照模式)
返回信息	函数的返回	是	int	人脸比对分值（同特征值比对分值结果应该是 float，这里返回 int 是做了四舍五入）

8.14 动作活体和静默活体

8.14.1 动作活体接口

方法名	face_action_live			
说明	动作活体（眨眨眼、张张嘴等动作校验活体）			
函数	int face_action_live(std::vector<FaceBox>& out, int action_type, int& action_result, const cv::Mat* mat)			
参数	说明	必须	类型	示例描述
out	返回的人脸框结构体数组	是	std::vector<FaceBox>	人脸框结构体
action_type	传入的活体动作	是	int	动作活体类型，如 0 表示眨眨眼，1 表示张嘴等
action_result	动作活体的返回结果	是	int	返回结果为 1 表示存在该动作活体
mat	传入的 opencv 视频帧	是	Mat	视频帧
返回字段描述				
返回字段	int，>=0 返回人脸个数，<0 时候为错误码			

返回示例	

8.14.2 清除动作活体历史接口

方法名	action_live_clear_history			
说明	清除动作活体历史(判断是否有一个动作需要累积多次图片帧的检测，可用来表示如做某动作，执行该接口后，前置在算法中的图片帧检测清零，算法按执行该接口后重新累积图片帧进行计算)			
函数	int action_live_clear_history()			
参数	说明	必须	类型	示例描述
返回字段描述				
返回字段	0 表示成功，其他错误码 int			
返回示例				

8.14.3 rgb 静默活体接口

方法名	rgb_liveness			
说明	rgb 可见光单目静默活体（推荐 rgb+nir 或 rgb+depth 进行双目活体校验）,活体推荐通过分值为 0.8。			
函数	int rgb_liveness(std::vector<TrackFaceInfo>& out, float &score, const cv::Mat* mat)			
参数	说明	必须	类型	示例描述
out	返回的人脸框结构体数组	是	std::vector<TrackFaceInfo>	人脸框结构体（多人则返回最大人脸）
score	返回的动作活体分值	是	float	
mat	传入的 opencv 视频帧	是	Mat	视频帧
返回字段描述				
返回字段	int，>=0 返回人脸个数，<0 时候为错误码			
返回示例				

8.14.4 nir 静默活体接口

方法名	nir_liveness			
说明	Nir 近红外单目静默活体（推荐 rgb+nir 或 rgb+depth 进行双目活体校验）			
函数	int nir_liveness(std::vector<TrackFaceInfo>& out, float &score, const cv::Mat* mat)			
参数	说明	必须	类型	示例描述
out	返回的人脸框结构体数组	是	std::vector<TrackFaceInfo>	人脸框结构体（多人则返回最大人脸）
score	返回的动作活体分值	是	float	
mat	传入的 opencv 视频帧	是	Mat	视频帧
返回字段描述				
返回字段	int，>=0 返回人脸个数，<0 时候为错误码			
返回示例				

8.14.5 rgb+depth 双目静默活体接口

方法名	rgb_depth_liveness			
说明	rgb+depth 双目静默活体			
函数	int rgb_depth_liveness(std::vector<TrackFaceInfo> &out, float &rgbscore, float &depthcore, const cv::Mat * rgb, char* depth)			
参数	说明	必须	类型	示例描述
out	返回的人脸框结构体数组	是	std::vector<TrackFaceInfo>	人脸框结构体（多人则返回最大人脸）
rgbscore	返回的 rgb	是	float	

	动作活体分值			
depthscore	返回的 depth 动作活体分值	是	float	
rgb	传入的 opencv 视频帧	是	Mat	视频帧
depth	传入的二进制深度数据	是	Char *	视频帧
返回字段描述				
返回字段	int，>=0 返回人脸个数，<0 时候为错误码			
返回示例				

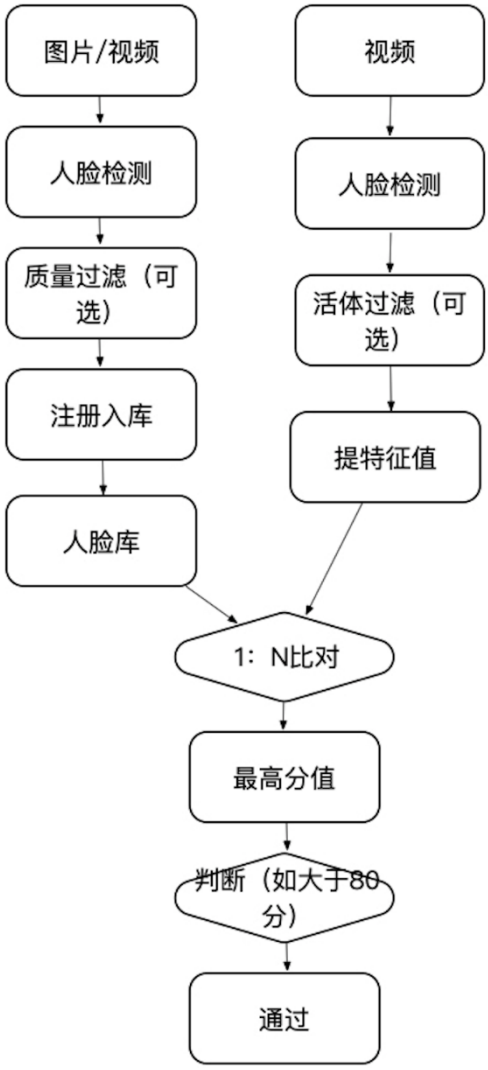
8.15 人脸库管理

sdk 提供支持 5 万以下的人脸库管理，采用的是 sqlite 数据库，sdk 启动后会自动生成 db 文件夹和 face.db 文件（人脸库数据文件）、db 文件夹可手动删除，删除后人脸数据库即被整体删除，sdk 重启后会自动重新创建新库。人脸库数据结构可采用 sqlLiteExpert 等可视化工具查看人脸库表结构。人脸库创建后有三张表,feature 表（用来保存人脸特征值），user 表（用来保存人脸用户信息，如 userid，groupid 以及人脸图片信息，用户信息等）以及 user_group 表（用户组表）。

人脸库可按组(group_id)划分,组就好比一个集团的子公司，人脸注册或查找既可以按整个库查找，也可以按组（子公司）查找，按组查找速度更快（范围小）。用户 id（user_id）是用来标识人脸用户的唯一 id，组 id（group_id）是用来标识组（子公司）的唯一 id。用户信息（user_info）可作为用户 id 的说明，如标识用户名称、住址等信息，也可不填写。人脸的比对或识别、归根结底是人脸特征值的比对。人脸库的保存实际上是保存了对应用户 user_id 的特征值在人脸库上，同时保存了用户 user_id 和 group_id 及其对应关系（一个用户对应一张人脸、一个特征值数据）。除 user_info 字段（用户信息）支持中文外，user_id 和 group_id 仅支持英文、数字和下划线的参数组合。人脸 1：N 识别返回识别的最高分和用户信息，推荐分值超过 80

为识别成功。

人脸库 1：N 识别可如如下图所示流程：



8.15.1 人脸注册接口(通过传入图片帧)

方法名	user_add			
说明	用户注册，该接口支持传入 opencv 图片帧（通常指生活照，1 寸或 2 寸的证件照也可以用这种类型）（图片帧注册会把人脸图片缩略图入库）			
函数	void user_add(std::string& res, const cv::Mat *img, const char* user_id, const char* group_id, const char* user_info="")			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据

img	Opencv 图片帧指针	是	cv::Mat *	人脸图片 opencv 图片帧指针
user_id	用户 id	是	const char*	用户 id（由数字、字母、下划线组成），长度限制 128B
group_id	组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
user_info	用户资料	否	const char*	256 个字符以内
返回字段描述(json)				
errno 及 msg 映射	errno	Msg		
	0	Success		
	<0	失败的原因		
返回示例				

8.15.2 人脸注册接口(通过传入特征值)

方法名	user_add			
说明	用户注册，该接口通过传入提取的人脸图片特征值注册（特征值注册无法把人脸缩略图入库）			
函数	void user_add(std::string& res, Feature* f1, const char* user_id, const char* group_id, const char* user_info="")			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
f1	特征值结构体	是	Feature*	人脸特征值结构体
user_id	用户 id	是	const char*	用户 id（由数字、字母、下划线组成），长度限制 128B
group_id	组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
user_info	用户资	否	const char*	256 个字符以内(中文数减半)

	料			
返回字段描述(json)				
errno 及 msg 映射	errno	Msg		
	0	Success		
	<0	失败的原因		
返回示例				

8.15.3 人脸更新接口

方法名	user_update			
说明	人脸更新			
函数	void user_update(std::string& res, const cv::Mat* img, const char* user_id, const char* group_id, const char* user_info="")			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
img	opencv 图片帧指针	是	cv::Mat*	人脸图片 opencv 图片帧指针
user_id	用户 id	是	const char*	用户 id（由数字、字母、下划线组成），长度限制 128B
group_id	组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
user_info	用户信息	否	const char*	用户资料，长度限制 256B
返回字段描述(json)				
errno 及 message 映射	**errno**	Msg		
	0	Success		
	<0	失败的原因		
返回示例				

8.15.4 用户删除接口

方法名	user_delete			
说明	用户删除			
函数	void user_delete(std::string&res, const char* user_id, const char* group_id)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
user_id	用户 id	是	const char*	用户 id（由数字、字母、下划线组成），长度限制 128B
group_id	组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
返回字段描述(json)				
errno 及 msg 映射	errno	Msg		
	0	Success		
	<0	失败的原因		
data 字段	log_id	string	请求日志标识	
返回示例				

8.15.5 创建用户组接口

方法名	group_add			
说明	创建用户组			
函数	void group_add(std::string&res, const char* gourp_id)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
group_id	用户组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
返回字段描述(json)				
errno 及 msg 映射	errno	msg		
	0	Success		
	<0	失败的原因		

data 字段	log_id	string	请求日志标识
返回示例			

8.15.6 用户组删除接口

方法名	group_delete			
说明	用户组删除			
函数	void group_delete(std::string&res, const char* group_id)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
group_id	组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
返回字段描述(json)				
errno 及 msg 映射	Errno	Msg		
	0	Success		
	<0	失败的原因		
data 字段	log_id	string		请求日志标识
返回示例				

8.15.7 用户信息查询接口

方法名	get_user_info			
说明	用户信息查询接口			
函数	void get_user_info(std::string&res, const char* user_id, const char* group_id)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
user_id	用户 id	是	const char*	用户 id（由数字、字母、下划线组成），长度限制 128B
group_id	组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
返回字段描述(json)				
errno 及	errno	Msg		

msg 映射	0	Success	
	<0	失败的原因	
data 字段	log_id	string	请求日志标识
	result	array	识别结果列表
	group_id	string	组 id
	face_token	string	人脸特征的唯一标识
	user_info	string	用户信息
	create_time	string	人脸首次注册时间
返回示例			

8.15.8 用户人脸图片查询接口

方法名	get_user_image			
说明	用户人脸图片查询接口			
函数	int get_user_image(cv::Mat & img, const char* user_id, const char* group_id)			
参数	说明	必须	类型	示例描述
img	通过引用返回的图片结果	是	cv::Mat &	
user_id	用户 id	是	const char*	用户 id（由数字、字母、下划线组成），长度限制 128B
group_id	组 id	是	const char*	用户组 id，标识一组用户（由数字、字母、下划线组成），长度限制 128B
返回字段描述: 返回 int，0 为成功，其他为错误码				

8.15.9 用户组列表查询接口

方法名	get_user_list
说明	用户组列表查询接口

函数	void get_user_list(std::string&res, const char* group_id, int start = 0, int length = 100)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
group_id	用户组 id	是	const char*	用户组 id
start	查询起始序号	否	int	默认值为 0
length	返回数量	否	int	默认值 100，最大值 1000
返回字段描述(json)				
errno 及 msg 映射	Errno	Msg		
	0	Success		
	<0	失败的原因		
data 字段	log_id	string	请求日志表识	
	user_id_list	array	user_id 列表数组	
返回示例				

8.15.10 人脸库人脸数量查询

方法名	db_face_count			
说明	查询数据库人脸数量 (传入组 id 表示查询该组都人脸数量，null 表示查整个库的人脸数量)			
函数	int db_face_count(const char* group_id = nullptr)			
参数	说明	必须	类型	示例描述
group_id	组 id	是	const char*	人脸分组的组 id(传 null 表示查询整个库的数量)
返回示例	int >=0 (返回的数量)			

8.15.11 群组列表查询接口

方法名	get_group_list			
说明	组列表查询			
函数	void get_group_list(std::string& res, int start = 0, int length = 100)			
参数	说明	必须	类型	示例描述

res	返回的结果	是	string	json 数据
start	起始序号	否	Int	默认值为 0，从 0 开始
length	返回数量	否	Int	默认值为 100，最大为 1000
返回字段描述(json)				
errno 及 msg 映射	errno	Msg		
	0	Success		
	<0	失败的原因		
data 字段	log_id	string	请求日志标识	
	group_id_list	array	group_id 列表数组	
返回示例				

8.15.12 人脸识别接口(1:N) (传入 opencv 图片帧)

方法名	identify			
说明	人脸识别（1：N）,可传入人脸组，比对某个组的所有人脸（人脸库可按单位分组，缩小组范围，减少识别比对时间）、返回最高分的用户及信息			
函数	void identify(std::string&res, const cv::Mat *img, const char* group_id_list, const char* user_id, int type = 0)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
img	opencv 图片帧指针	是	cv::Mat *	人脸 opencv 图片帧指针
group_id_list	组列表	是	const char*	组列表
user_id	用户 id	否	const char*	用户 id
type	特征值类型	否	shi	特征值类型（0、是生活照、1、证件照 2、近红外，参考特征值提取），未传默认为 0
返回字段描述				
返回字段	比对分值	float		
返回示例				

8.15.13 人脸识别接口(1:N)(传入特征值)

方法名	identify			
说明	人脸识别（1：N）,可传入人脸组，比对某个组的所有人脸（人脸库可按单位分组，缩小组范围，减少识别比对时间）、返回最高分的用户及信息			
函数	void identify(std::string&res, Feature* f1, const char* group_id_list, const char* user_id, int type = 0)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
f1	起始序号	是	Feature*	特征值结构体
group_id_list	组列表	是	const char*	组列表
user_id	用户 id	否	const char*	用户 id
type	特征值类型	否	shi	特征值类型（0、是生活照、1、证件照 2、近红外，参考特征值提取），未传默认为 0
返回字段描述				
返回字段	比对分值	float		
返回示例				

8.15.14 人脸识别接口(1:N) (传入 opencv 图片帧)

方法名	identify_with_all			
说明	人脸识别（1：N）,和整个人类库比对 、返回最高分的用户及信息			
函数	void identify_with_all(std::string& res, const cv::Mat *img, int type = 0)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
img	opencv 图片帧	是	cv::Mat *	人脸 opencv 图片帧
type	特征值类型	否	int	特征值类型（0、是生活照，包含 1 寸照、1、网格证件照 2、近红

				外，参考特征值提取），未传默认为 0
返回字段描述				
返回字段	比对分值	float		
返回示例				

8.15.15 人脸识别接口(1:N) (传入特征值)

方法名	identify_with_all			
说明	人脸识别（1：N）,和整个人类库比对 、返回最高分的用户及信息			
函数	void identify_with_all(std::string& res, Feature *f1, int type = 0)			
参数	说明	必须	类型	示例描述
res	返回的结果	是	string	json 数据
f1	起始序号	是	Feature*	特征值结构体
type	特征值类型	否	int	特征值类型（0、是生活照、1、证件照 2、近红外，参考特征值提取），未传默认为 0
返回字段描述				
返回字段	比对分值	float		
返回示例				

8.16 sdk 系统信息接口

8.16.1 获取 sdk 版本号接口

方法名	sdk_version			
说明	通过引用返回 sdk 版本号			
函数	void sdk_version(std::string& version)			
参数	说明	必须	类型	示例描述
version	返回的 sdk 版本号结果	是	string	如：6.3.3

返回字段描述

8.16.2 获取设备指纹接口

方法名	get_device_id			
说明	通过引用返回 sdk 的设备指纹信息（可用来标识 sdk 的唯一性）			
函数	void get_device_id(std::string& version)			
参数	说明	必须	类型	示例描述
device_id	返回的 sdk 设备指纹结果	是	string	如：0BAF96961A8377AB74797B05F7F2C805
返回字段描述				

8.17 安全驾驶接口

8.17.1 驾驶行为检测接口

方法名	driver_monitor			
说明	驾驶行为检测接口，返回驾驶行为信息			
函数	int driver_monitor(std::vector<DriverMonitor> &out, const cv::Mat *mat)			
参数	说明	必须	类型	示例描述
out	返回的驾驶行为结构体数组信息	是	std::vector<DriverMonitor>	DriverMonitor 结构体见后续结构体定义
mat	传入的 opencv 图片帧	是	cv::Mat *	
返回字段描述	int，>=0 返回人脸个数，<0 时候为错误码			

8.17.2 安全带佩戴检测接口

方法名	safety_belt

说明	安全带佩戴检测接口，返回佩戴置信度分值			
函数	int safety_belt(std::vector<SafetyBelt> &out, const cv::Mat *mat)			
参数	说明	必须	类型	示例描述
out	返回的安全带佩戴结构体数组信息	是	std::vector<SafeBelt>	SafeBelt 结构体见后续结构体定义
mat	传入的 opencv 图片帧	是	cv::Mat *	
返回字段描述	int，>=0 返回人脸个数，<0 时候为错误码			

9 结构体描述

9.1 人脸跟踪信息结构体

结构体名	TrackFaceInfo	

说明	人脸跟踪信息结构体
	struct TrackFaceInfo
	{
	long face_id;
	FaceBox box;
	Landmarks facelandmarks;
	}	

参数	类型	说明
face_id	long	人脸 id
box	FaceBox	人脸框结构体
facelandmarks	Landmarks	人脸关键点结构体

9.2 人脸框信息结构体

结构体名	FaceBox
说明	人脸框信息结构体

	struct FaceBox
	{
	int index;
	float center_x;
	float center_y;
	float width;
	float height;
	float score;
	}	

参数	类型	说明
index	int	人脸索引值
center_x	float	人脸中心点 x 坐标
center_y	float	人脸中心点 y 坐标
width	float	人脸宽度
height	float	人脸高度
score	float	人脸置信度

9.3 人脸关键点信息结构体

结构体名	Landmarks	

说明	人脸关键点信息结构体
	struct Landmarks
	{
	int index;
	int size;
	float data[144];
	float score;
	}	

参数	类型	说明
index	int	人脸关键点索引
size	int	人脸关键点数量（size 的值通常为 144,72 个 x，y 坐标）
data	float[144]	人脸关键点坐标数组（144）（72 个 x，y 坐标）

score	float	人脸关键点置信度

9.4 人脸特征值结构体

结构体名	Feature	

说明	人脸特征值结构体
	struct Feature
	{
	int size;
	float data[128];
	}	

参数	类型	说明
size	int	特征值大小（通常为 128 个 float 浮点值数据）
data	float[]	128 个浮点值数组

9.5 人脸姿态角结构体

结构体名	HeadPose	

说明	人脸姿态角结构体
	struct HeadPose
	{
	float yaw;
	float roll;
	float pitch;
	}	

参数	类型	说明
yaw	float	左右偏转角
roll	float	与人脸平行平面内的头部旋转角
pitch	float	上下偏转角

9.6 人脸属性信息结构体

结构体名	Attribute
说明	人脸属性信息结构体 struct Attribute

	{
	int age;
	Race race;
	AttributeEmotionType emotion;
	Glasses glasses;
	Gender gender;
	}	

参数	类型	说明
age	int	年龄

race	Race	种族
		enum Race
		{
		BDFace_RACE_YELLOW = 0, // 黄种人
		BDFace_RACE_WHITE = 1, //白种人
		BDFace_RACE_BLACK = 2, // 黑种人
		BDFace_RACE_INDIAN = 3, // 印第安人
		}

emotion	AttributeEmotionType	表情
		enum AttributeEmotionType
		{
		BDFACE_ATTRIBUTE_EMOTION_FROWN = 0,//皱眉
		BDFACE_ATTRIBUTE_EMOTION_SMILE = 1, //笑
		BDFACE_ATTRIBUTE_EMOTION_CALM = 2, //平静
		}

glasses	Glasses	戴眼镜状态
		enum Glasses
		{
		BDFACE_NO_GLASSES = 0 , // 无眼镜
		BDFACE_GLASSES = 1, // 有眼镜
		BDFACE_SUN_GLASSES = 2 //墨镜
		}

gender	Gender	性别
		enum Gender
		{
		BDFACE_GENDER_FEMILE = 0, // 女性
		BDFACE_GENDER_MALE = 1, //
		}

9.7 嘴巴闭合结构体

结构体名	MouthClose	

说明	嘴巴闭合置信度结构体
	struct MouthClose
	{
	float score;
	}	

参数	类型	说明
score	float	置信度分值

9.8 口罩佩戴结构体

结构体名	MouthMask	

说明	口罩佩戴置信度结构体
	struct MouthMask
	{
	float score;
	}	

参数	类型	说明
score	float	置信度分值

9.9 最优人脸置信度结构体

结构体名	Best
说明	最优人脸置信度结构体 struct Best {

	float score; }	
参数	类型	说明
score	float	置信度分值

9.10 人脸模糊度置信度结构体

结构体名	Blur	

说明	人脸模糊度置信度结构体
	struct Blur
	{
	float score;
	}	

参数	类型	说明
score	float	置信度分值

9.11 人脸光照置信度结构体

结构体名	Illumination	

说明	光照置信度结构体
	struct Illumination
	{
	int score;
	}	

参数	类型	说明
score	int	置信度分值

9.12 人脸遮挡置信度结构体

结构体名	Illumination
说明	人脸遮挡置信度结构体 struct Illumination { float left_eye; float right_eye;

	float nose; float mouth; float left_cheek; float right_cheek; float chin; }	
参数	类型	说明
left_eye	float	左眼遮挡置信度
right_eye	float	右眼遮挡置信度
nose	float	鼻子遮挡置信度
mouth	float	嘴巴遮挡置信度
left_cheek	float	左脸遮挡置信度
right_cheek	float	右脸遮挡置信度
chin	float	下巴遮挡置信度

9.13 人眼闭合状态结构体

结构体名	EyeClose	

说明	人眼闭合状态结构体
	struct EyeClose
	{
	float left_eye_close_conf;
	float right_eye_close_conf;
	}	

参数	类型	说明
float left_eye_close_conf	float	置信度分值
float right_eye_close_conf	float	置信度分值

9.14 注意力结构体

结构体名	GazeInfo
说明	注意力结构体 struct GazeInfo

	{
	Gaze left_eye;
	Gaze right_eye;
	}
	struct Gaze
	{
	GazeDirection direction; // 凝视方向
	float confidence; // 置信度
	}
	struct GazeDirection
	{
	BDFACE_GAZE_DIRECTION_UP = 0, // 向上看
	BDFACE_GAZE_DIRECTION_DOWN = 1, // 向下看
	BDFACE_GAZE_DIRECTION_RIGHT = 2, // 向右看
	BDFACE_GZE_DIRECTION_LEFT = 3, // 向左看
	BDFACE_GAZE_DIRECTION_FRONT = 4, // 向前看
	BDFACE_GAZE_DIRECTION_EYE_CLOSE = 5, // 闭眼
	}

参数	类型	说明
left_eye	Gaze	左眼注意力
right_eye	Gaze	右眼注意力

9.15 静默活体置信度结构体

结构体名	LivenessScore	

说明	活体信度结构体
	struct LivenessScore
	{
	int score;
	}	

参数	类型	说明
score	float	置信度分值

9.16 驾驶行为结构体

结构体名	DriverMonitor	

说明	驾驶行为结构体
	struct DriverMonitor
	{
	float normal;
	float calling;
	float drinking;
	float eating;
	float smoking;
	}	

参数	类型	说明
normal	float	行为正常的置信度分值（置信度范围 0-1）
calling	float	打电话行为的置信度分值（置信度范围 0-1）
drinking	float	喝水的置信度分值（置信度范围 0-1）
eating	float	吃东西的置信度分值（置信度范围 0-1）
smoking	float	抽烟的置信度分值（置信度范围 0-1）

9.17 安全带佩戴结构体

结构体名	SafetyBelt	

说明	安全带佩戴行为结构体
	struct SafetyBelt
	{
	float score;
	}	

参数	类型	说明
score	float	安全带佩戴的置信度分值（置信度范围 0-1）

9.18 人脸表情结构体

结构体名	Emotion

说明	人脸表情结构体
	struct Emotion
	{
	int expression; // 情绪
	float expression_conf; // 对应的置信度
	float expression_conf_list[3]; // 所有情绪的置信度
	}	

参数	类型	说明
expression	int	情绪（共 3 种情绪，分别是 0: 负面 1:正面 2:中 性）
expression_conf	float	情绪对应的置信度
expression_conf_list	float	所有情绪的置信度

10 多端特征值对齐

对于 linux arm sdk 来说，支持和安卓特征值对齐，linux arm sdk7.x 系列，特征值对齐安卓 7.x 系列，但和如安卓 6.x 系列或 5.x 系列，则不对齐，且只支持 rgb 可见光特征值对齐。

在 linux arm sdk 中，可通过提取的特征值 128 个 float 数组，保存成二进制数据，即和安卓中的 512 个 byte 保存成字节流的数据对齐。

sdk 中提供了 float 数组转换为二进制 buffer 的示例代码以及二进制 buffer 数据转换为特征值 float 数组的示例代码。可参考 util 文件夹中的 FeatureAlign 类。

11 多线程运行

sdk 新增支持多实例的多线程使用。可参考 sdk 的示例代码 multi_thread 文件夹，有具体使用例子。若单个实例，sdk 不支持多线程同时调用如人脸检测(detect)和提取特征值(face_feature)。但支持多个实例中多线程调用，如可 new 一个实例中调人脸检测方法(detect)，另外一个实例中调特征值提取方法(face_feature)。同理，若需要用 3 个功能，可分别使用 3 个实例，调用不同的功能接口。

另外，如示例所示，使用多个不同的实例的时候，需要 new 不同的 BaiduFaceApi

实例，同时，需要调多次 sdk_init 方法。销毁时候也一样，注意销毁多个实例，避免内存泄漏。

12 错误码及错误信息

各接口返回结果 error_code 及 msg 信息如下：

错误码	错误内容	错误描述
0	SUCCESS	成功
-1	ILLEGAL_PARAMS	失败或非法参数
-2	MEMORY_ALLOCATION_FAILED	内存分配失败
-3	INSTANCE_IS_EMPTY	实例对象为空
-4	MODEL_IS_EMPTY	模型内容为空
-5	UNSUPPORT_ABILITY_TYPE	不支持的能力类型
-6	UNSUPPORT_INFER_TYPE	不支持的预测库类型
-7	NN_CREATE_FAILED	预测库对象创建失败
-8	NN_INIT_FAILED	预测库对象初始化失败
-9	IMAGE_IS_EMPTY	图像数据为空
-10	ABILITY_INIT_FAILED	人脸能力初始化失败
-11	ABILITY_UNLOAD	人脸能力未加载
-12	ABILITY_ALREADY_LOADED	人脸能力已加载
-13	NOT_AUTHORIZED	未授权
-14	ABILITY_RUN_EXCEPTION	人脸能力运行异常
-15	UNSUPPORT_IMAGE_TYPE	不支持的图像类型
-16	IMAGE_TRANSFORM_FAILED	图像转换失败
-1001	SYSTEM_ERROR	系统错误
-1002	PARARM_ERROR	参数错误
-1003	DB_OP_FAILED	数据库操作失败
-1004	NO_DATA	没有数据
-1005	RECORD_UNEXIST	记录不存在
-1006	RECORD_ALREADY_EXIST	记录已经存在
-1007	FILE_NOT_EXIST	文件不存在

-1008	GET_FEATURE_FAIL	提取特征值失败
-1009	FILE_TOO_BIG	文件太大
-1010	FACE_RESOURCE_NOT_EXIST	人脸资源文件不存在
-1011	FEATURE_LEN_ERROR	特征值长度错误
-1012	DETECT_NO_FACE	未检测到人脸
-1013	CAMERA_ERROR	摄像头错误或不存在
-1014	FACE_INSTANCE_ERROR	人脸引擎初始化错误
-1015	LICENSE_FILE_NOT_EXIST	授权文件不存在
-1016	LICENSE_KEY_EMPTY	授权序列号为空
-1017	LICENSE_KEY_INVALID	授权序列号无效
-1018	LICENSE_KEY_EXPIRE	授权序序列号过期
-1019	LICENSE_ALREADY_USED	授权序列号已被使用
-1020	DEVICE_ID_EMPTY	设备指纹为空
-1021	NETWORK_TIMEOUT	网络超时
-1022	NETWORK_ERROR	网络错误

13 常见问题：

13.1 sdk 推荐使用在开发板如 rk3288 或 rk3399 上直接编译，安装 gcc/g++/cmake。

13.2 工程运行过程中，若不能正常运行功能，可在 build 目录下，生成 face_conf.json 文件，内容为{"log_open":true}，通过这个 json 配置文件，可打开 sdk 的日志模式，运行后会输出及各接口返回的错误码日志等判断问题所在。

13.3 模型文件可定制化：在 main 方法入口，可在 sdk_init 方法中传入模型文件夹的绝对路径，达到模型文件定制化的目的。若不定制化路径，sdk_init 中传入 null 即可，默认模型文件在 sdk 的 models 文件夹里面，无需更改。

13.4 激活后是否可以把激活文件 license.ini 和 license.key 拷贝到其他设备运行？

不能，离线 sdk 和设备绑定，每个设备对应一个 key 和一个 license 文件，换设备无法运行。但对同一台设备，可把 Release 下的 license.ini 和 license.key 拷贝到本电脑的另外 sdk，该设备也等同于激活，可以使用。

13.5 是否支持 debug 模式？只支持 Release 模式，不支持 debug 模式

13.6 sdk 支持 armv7hf、armv8 等平台，请根据对应平台运行。

13.7 人脸库不支持中文参数？用户信息 user_info 支持中文，其他人脸管理参数目前暂只支持英文、数字下划线组合模式，工程所在的路径也建议不用放入中文路径中，可能影响人脸库创建。

13.8 特征值多端是否对齐？在人脸 7.0 系列 sdk 中，安卓和 linux 端生活照模式下的特征值都是对齐的。linux 中请把提取出来的 128 个 float 数据保存成二进制数据（float 数组转二进制保存）即可和安卓的 512 个 byte 二进制数据对齐。linux 中人脸库保存的数据是进行了 base64 编码保存的。

13.9 sdk 是否支持多线程运行？sdk 支持多实例的多线程运行，请参考文档有详细说明。

13.10 海思等其他开发板是否支持？sdk 推荐用在 rk3288 开发板上，其他开发板是 armv7hf 或 armv8 平台，也可支持运行、部分第三库如 opencv 等，若报错，可推荐在对应开发板上编译产出库文件，opencv 推荐使用 4.1 版本。