

密级状态：绝密() 秘密() 内部() 公开(☒)

智能视频分析（ROCKIVA）SDK 开发指南

（技术部，图形计算平台中心）

文件状态： [] 正在修改 [<input checked="" type="checkbox"/>] 正式发布	当前版本：	V1.9.0
	作 者：	HPC&AI Team
	完成日期：	2023-08-01
	审 核：	YHC
	完成日期：	2023-08-02

瑞芯微电子股份有限公司

Rockchips Semiconductor Co., Ltd

(版本所有,翻版必究)

更新记录

版本	修改人	修改日期	修改说明	核定人
V1.0	YHC	2021-12-30	初始版本	XW/ZHT
V1.1	YHC	2022-01-13	1. 人脸模块支持获取关联人形结果 2. 人脸模块结果支持获取人脸质量信息	ZHT
V1.2	XDC	2022-04-06	1. 修改部分 API、结构体的名字和参数	YHC
V1.3	XDC	2022-06-06	1. 修改人脸抓拍 API、结构体的名字和参数	YHC
V1.4.0	XDC	2022-07-18	1. 增加车辆车牌识别模块 2. 增加非机动车检测模块 3. 修改部分 API、结构体的名字和参数	YHC
V1.4.2	XDC	2022-09-06	1. 人脸模块支持口罩提前上报和配置人脸抓拍图像不缩放模式 2. 物体检测模块梯控部分增加电瓶车报警标志位 3. 通用模块检测类别增加电瓶车、自行车和车牌，删除物体检测模块的非机动车枚举类型	YHC
V1.4.2	XDC	2023-08-01	4. 更新 API 参考定义 5. 增加客流统计模块 6. 各模块增加示例代码 7. 增加 IPC Demo 使用说明	YHC

目 录

1	概述.....	5
2	授权说明.....	6
2.1	设备上授权	6
2.2	PC 上授权.....	6
3	DEMO 使用说明.....	8
3.1	图像 DEMO 使用说明	8
3.2	IPC DEMO 使用说明	9
4	SDK 开发说明.....	9
4.1	通用模块	10
4.1.1	功能描述.....	10
4.1.2	API 参考.....	13
4.2	检测模块	35
4.2.1	功能描述.....	35
4.2.2	API 参考.....	37
4.3	周界模块	40
4.3.1	功能描述.....	40
4.3.2	API 参考.....	45
4.4	人脸模块	53
4.4.1	功能描述.....	53
4.4.2	API 参考.....	59
4.5	车辆车牌模块.....	85
4.5.1	功能描述.....	85
4.5.2	API 参考.....	88

4.6	客流统计模块.....	98
4.6.1	功能描述.....	98
4.6.2	API 参考.....	100
4.7	物体检测模块.....	107
4.7.1	功能描述.....	107
4.7.2	API 参考.....	110

1 概述

ROCKIVA SDK 是一套智能视频分析（IVA: Intelligence Video Analysis）算法 SDK，能够嵌入 NVR、IPC 等摄像头视频流的产品应用中，提供 AI 智能算法支持。SDK 的主要功能包括目标检测、周界功能、人脸抓拍识别功能、车辆车牌检测识别功能和非机动车检测功能：

- 目标检测功能：检测画面中的人形、人脸、机动车、非机动车和宠物猫狗，返回目标位置；
- 周界功能：支持区域入侵、越界检测、区域进入、区域离开等周界功能；
- 人脸抓拍识别功能：根据抓拍规则对画面中的人脸进行跟踪优选抓拍，可对抓拍人脸做属性分析和特征提取比对搜索；
- 车辆车牌识别检测功能：检测画面中的机动车及其车牌，并返回车牌号码、车辆属性和车牌属性；
- 客流统计功能：支持越界客流和区域人数统计；
- 非机动车检测功能：检测画面中的非机动车位置，支持平视角（马路、走廊等）和俯视角（电梯等）两类场景；

2 授权说明

算法 SDK 需要授权后才能使用，用户首先需要联系业务窗口申请授权账号，然后使用授权工具获取 License 密钥后通过初始化接口传入 License 密钥字符串后即可。

授权工具支持 PC Windows/Linux 或设备上运行获取授权。

2.1 设备上授权

需要确保设备能够联网，然后将对应平台的 rkauth_tool_bin 程序，部署到设备中。

以 RV1109 平台为例，使用 adb 推到设备：

```
adb push Linux/armhf/rkauth_tool_bin /userdata/
```

然后在设备上执行以下命令授权

```
./rkauth_tool_bin --user="xxxxxx" --passwd="xxxxxx" --output="key.lic"
```

执行成功后 License 密钥将保存到"key.lic"文件，对相同设备的重复授权不会减少授权次数，如果 key.lic 丢失后可以重新执行获取。

2.2 PC 上授权

如果设备无法联网，可以在 PC 上授权。首先需要从设备获取设备信息，然后使用 PC 的授权工具根据设备信息来生成 License 密钥。

首先将对应平台的 rkdevice_info 执行程序部署到设备中

以 RV1109 为例，通过 adb 推到设备

```
adb push Linux/armhf/rkdevice_info /userdata/
```

在设备上执行 rkdevice_info 命令，执行成功将在当前目录生成 device.inf 文件，将该文件拷贝回 PC。

确保 PC 能够联网，在 PC 执行授权命令

```
./rkauth_tool_bin --user="xxxxxx" --passwd="xxxxxx" --output="key.lic" --device_info=/path/to/device.inf
```

执行成功后 License 密钥将保存到"key.lic"文件中。

3 Demo 使用说明

3.1 图像 Demo 使用说明

SDK 提供命令行 Demo，通过读取图像来能够简单上手编译运行测试 SDK 的功能。图像 Demo 位于 SDK 包的 demo 目录下，用户只要执行相应平台的编译脚本即可。

以 RV1109 平台为例

```
./build-linux-rv1109.sh
```

编译完成后将在 install 目录下生成 Demo 可执行文件推到设备

```
adb push install/rockiva_rv1109_linux /userdata/
```

接下来还需要将模型文件推到设备中，模型目录可以在 Demo 的初始化参数中指定，Demo 中默认为 “/data/rockiva_data”。

```
adb push ../models/rockiva_data_rv1109 /data/  
adb shell mv /userdata/rockiva_data_rv1109 /data/rockiva_data
```

接着还需要准备测试图像，Demo 通过读取文件夹下连续的图像文件来逐张运行，对于视频流功能（检测、周界、人脸、车辆等）的 Demo，可用 demo/rockiva_demo/tools/video_to_images.py 脚本对视频每三帧（默认）提取图像并按照顺序 0000.jpg、0001.jpg、0002.jpg ... 放入当前目录 image 文件夹中。

```
cd /userdata/rockiva_rv1109_linux /rockiva_demo  
./rockiva_demo -s /userdata/xxxx/ -n 100 -det
```

Demo 的 -s 参数为图像的目录，-n 参数为跑的帧数，后面可以跟上需要跑的模块（如 det、face、plate 等）。

执行后会在当前目录生成 out 文件夹，可以将其拷贝到 PC 上查看运行结果。

3.2 IPC Demo 使用说明

SDK 提供基于 RV1106 IPC SDK 的 IPC Demo。使用前需要修改 demo/rockiva_demo/CMakeLists.txt 中的 IPC_LINUX_SDK_PATH 为本地对应的 RV1106 IPC Linux SDK 的路径，并且该 SDK 有完整编译过一次。

执行 Demo 的编译脚本，编译完成将生成可执行程序 rockiva_ipc_demo，操作如下：

```
./build-linux-rv1106.sh
```

编译完成后将在 install 目录下生成的 Demo 推到设备

```
adb push install/rockiva_rv1106_linux /userdata/
```

接下来还需要将模型文件推到设备中，设备上的模型目录可以在 Demo 的初始化参数中指定，Demo 中默认的路径为 “/data/rockiva_data”。

```
adb push ../models/rockiva_data_rv1106 /data/  
adb shell mv /userdata/rockiva_data_rv1106 /data/rockiva_data
```

运行程序

```
adb shell  
cd /userdata/rockiva_rv1106_linux/rockiva_demo  
./rockiva_ipc_demo --det
```

其中--det 表示使能检测模块，还可以设置--ba（周界）、--face（人脸）、--plate（车牌）等。

程序运行成功没有报错的情况下，PC 端可以通过打开 rtsp://<ip>/live/0 查看设备的 rtsp 推流画面和算法显示的结果。

4 SDK 开发说明

ROCKIVA SDK 首先基于目标检测跟踪算法从图像中获取人形、人脸、机动车、非机动车等目标信息，然后根据目标检测跟踪的结果再进行智能分析。当前支持的智能分析功能有周界、人脸、车辆（机动车和非机动车）和车牌等。智能分析每个功能可以独立开启或关闭，也能

够同时使用。

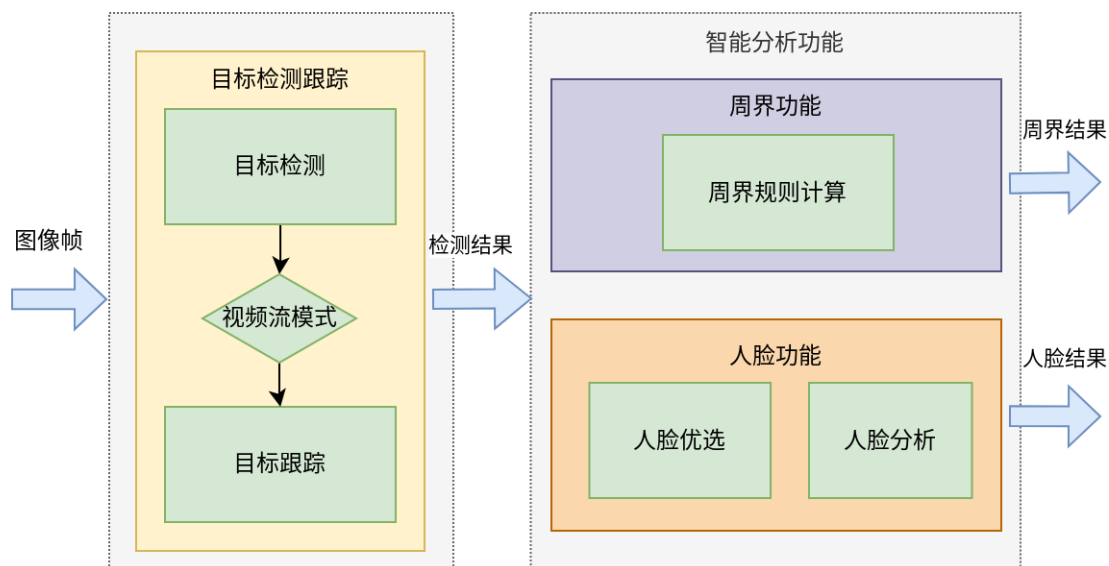


图 4-1 RovkIva SDK 整体流程

4.1 通用模块

4.1.1 功能描述

ROCKIVA SDK 的接口为异步模式，使用的基本操作就是配置初始化，然后按一定的帧率推帧，在回调函数中获取结果并释放图像帧内存，最后如果不需要使用就释放实例。

SDK 的基本调用流程如图 4-1-1 所示。

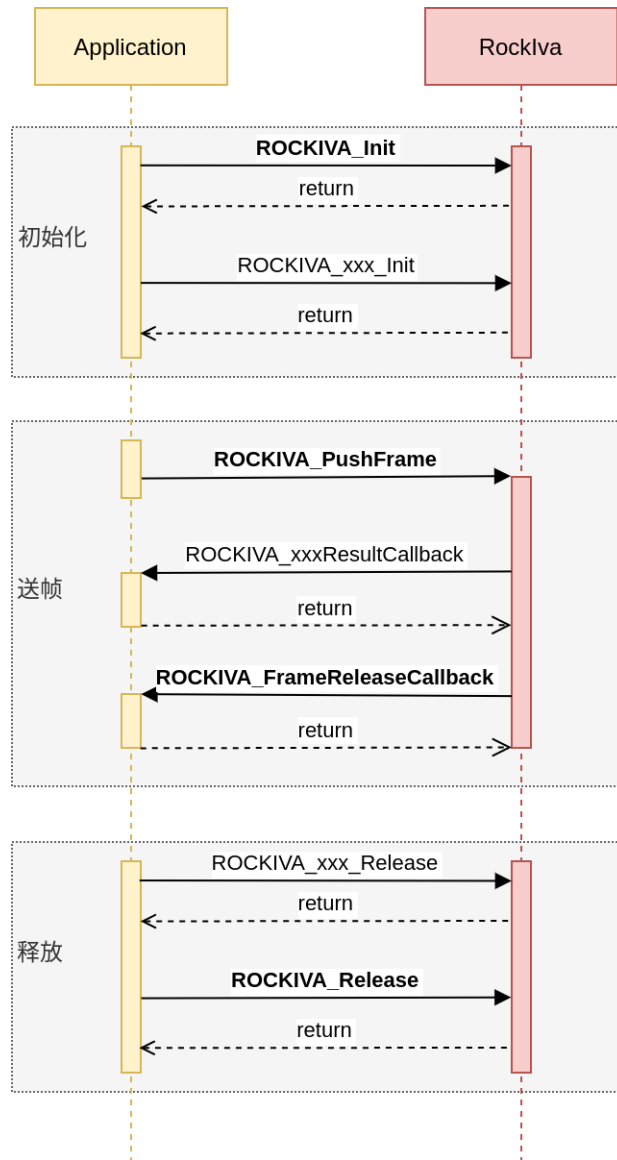


图 4-1-1 SDK 基本调用流程

配置初始化后，SDK 内部会创建线程异步处理每一帧，应用每次推送的图像帧都会放到一个缓存队列中送给处理线程。

回调函数 `ROCKIVA_FrameReleaseCallback` 用来通知应用可以释放的图像帧，SDK 内部会判断如果不需要再使用该图像帧的时候通知外面应用可以进行释放。

调用代码参考：

```
RockIvaRetCode ret;
RockIvaHandle handle;

// 释放回调函数
void FrameReleaseCallback(const RockIvaReleaseFrames* releaseFrames, void* userdata)
{
    // 处理释放
```

```

    }

    // 模块回调函数，如检测模块
    void DetResultCallback(const RockIvaDetectResult* result, const RockIvaExecuteStatus status, void*
userdata)
    {
        // 处理结果
    }

    // 初始化 IVA 实例
    RockIvaInitParam commonParams;
    memset(&commonParams, 0, sizeof(RockIvaInitParam));
    commonParams.detModel = ROCKIVA_DET_MODEL_CLS7; // 设置检测模型
    ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);
    // 设置回调函数
    ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);

    // 初始化需要的 IVA 模块，如检测模块
    ret = ROCKIVA_DETECT_Init(handle, &detParams, DetResultCallback);

    // 处理图像帧
    while(running) {
        RockIvaImage image;
        memset(&image, 0, sizeof(RockIvaImage));
        // 取帧
        ...
        // 设置帧数据
        image->info.width = image_width;
        image->info.height = image_height;
        image->info.format = image_pixel_format;
        image->dataAddr = image_buffer_virtual_address;
        image->dataFd = image_buffer_fd;
        image->frameId = frame_index;
        ...
        // 送帧
        ROCKIVA_PushFrame(handle, image, NULL);
    }

    // 等待所有帧都处理完成
    ROCKIVA_WaitFinish(handle, -1, 5000);
    // 销毁模块
    ROCKIVA_DETECT_Release(handle)
    // 销毁 IVA 实例
    ROCKIVA_Release(handle);

```

4.1.2 API 参考

4.1.2.1 接口

接口	描述
ROCKIVA_Init	算法 SDK 初始化配置
ROCKIVA_Release	算法 SDK 销毁释放
ROCKIVA_WaitFinish	等待算法帧处理完成
ROCKIVA_SetFrameReleaseCallback	设置帧释放回调函数
ROCKIVA_FrameReleaseCallback	帧释放回调函数
ROCKIVA_PushFrame	输入图像帧
ROCKIVA_PushFrameWithRoi	输入图像帧的特定有效区域
ROCKIVA_GetVersion	获取 SDK 版本

4.1.2.1.1 ROCKIVA_Init

【功能】

算法 SDK 初始化，对传入 handle 进行初始化

【声明】

```
RockIvaRetCode ROCKIVA_Init(RockIvaHandle* handle, RockIvaWorkMode mode,  
                             const RockIvaInitParam* param, void *userdata);
```

【参数】

参数名称	输入/输出	描述
handle	输入	要初始化的实例句柄
mode	输入	工作模式，参见 RockIvaWorkMode 定义
param	输入	初始化参数，参见 RockIvaInitParam 定义
userdata	输入	用户自定义数据

【返回值】

RockIvaRetCode

4.1.2.1.2 ROCKIVA_Release

【功能】

算法 SDK 销毁释放

【声明】

```
RockIvaRetCode ROCKIVA_Release(RockIvaHandle handle);
```

【参数】

参数名称	输入/输出	描述
handle	输入	要释放的实例句柄

【返回值】

RockIvaRetCode

4.1.2.1.3 ROCKIVA_WaitFinish

【功能】

等待算法帧处理完成

【声明】

```
RockIvaRetCode ROCKIVA_WaitFinish(RockIvaHandle handle, long frameId, int timeoutMS);
```

【参数】

参数名称	输入/输出	描述
handle	输入	实例句柄
frameId	输入	要等待的帧 ID，如果小于 0 表示等待所有帧处理完
timeoutMS	输入	超时时间（毫秒），如果小于等于 0 表示不超时等待

【返回值】

RockIvaRetCode

4.1.2.1.4 ROCKIVA_SetFrameReleaseCallback

【功能】

设置帧释放回调函数

【声明】

```
RockIvaRetCode ROCKIVA_SetFrameReleaseCallback(RockIvaHandle handle,  
ROCKIVA_FrameReleaseCallback callback);
```

【参数】

参数名称	输入/输出	描述
handle	输入	实例句柄
callback	输入	帧释放回调函数

【返回值】

RockIvaRetCode

【说明】

建议将帧对应的信息存放到 RockIvaImage 中的 extData 指针，然后释放时候可以取出进行释放

4.1.2.1.5 ROCKIVA_FrameReleaseCallback

【功能】

帧释放回调函数

【声明】

```
typedef void (*ROCKIVA_FrameReleaseCallback)(const RockIvaReleaseFrames* releaseFrames, void *userdata);
```

【参数】

参数名称	输入/输出	描述
releaseFrames	输入	可释放的帧列表，参见 RockIvaReleaseFrames 定义
userdata	输入	用户自定义数据

【返回值】

RockIvaRetCode

4.1.2.1.6 ROCKIVA_PushFrame

【功能】

输入图像帧

【声明】

```
RockIvaRetCode ROCKIVA_PushFrame(RockIvaHandle handle, const RockIvaImage* inputImg, const RockIvaFrameExtraInfo* extraInfo);
```

【参数】

参数名称	输入/输出	描述
handle	输入	实例句柄
inputImg	输入	输入图像帧, 参见 RockIvaImage 定义
extraInfo	输入	额外信息(不需要可设 NULL), 参见 RockIvaFrameExtraInfo 定义

【返回值】

RockIvaRetCode

4.1.2.1.7 ROCKIVA_PushFrame2

【功能】

输入图像帧（多摄）

【声明】

```
RockIvaRetCode ROCKIVA_PushFrame2(RockIvaHandle handle, const RockIvaMultiImage* inputImg,
const RockIvaFrameExtraInfo* extraInfo);
```

【参数】

参数名称	输入/输出	描述
handle	输入	实例句柄
inputImg	输入	输入图像帧, 参见 RockIvaMultiImage 定义
extraInfo	输入	额外信息(不需要可设 NULL), 参见 RockIvaFrameExtraInfo 定义

【返回值】

RockIvaRetCode

4.1.2.1.8 ROCKIVA_SetParam

【功能】

设置参数

【声明】

```
RockIvaRetCode ROCKIVA_SetParam(RockIvaParams* params, const char* key, const char* value);
```

【参数】

参数名称	输入/输出	描述
params	输入	参数, 参见 RockIvaParams 定义
key	输入	关键字
value	输入	值

【返回值】

RockIvaRetCode

4.1.2.1.9 ROCKIVA_GetVersion

【功能】

获取 SDK 版本号

【声明】

```
RockIvaRetCode ROCKIVA_GetVersion(const uint32_t maxLen, char* version);
```

【参数】

参数名称	输入/输出	描述
maxLen	输入	存放版本号 buffer 大小
version	输入	版本号 buffer 地址

【返回值】

RockIvaRetCode

4.1.2.1.10 ROCKIVA_Yuv2Jpeg

【功能】

回调函数, 用于将图像编码为 JPEG 图像

【声明】

```
int (*ROCKIVA_Yuv2Jpeg)(RockIvaImage *inputImg, RockIvaImage *outputImg, void *userdata);
```

【参数】

参数名称	输入/输出	描述
inputImg	输入	要编码的图像
outputImg	输出	编码后图像(注意,outputImg.dataAddr 需要外部自己申请和释放内存, 当内部对该图像内存不再使用将会在抓拍回调函数 ROCKIVA_FrameReleaseCallback 上报)
userdata	输入	初始化设置的自定义指针

【返回值】

RockIvaRetCode

4.1.2.2枚举

枚举	描述
RockIvaLogLevel	日志打印级别
RockIvaRetCode	函数返回错误码
RockIvaExecuteStatus	执行回调结果状态码
RockIvaImageFormat	图像像素格式
RockIvaImageTransform	输入图像的旋转模式
RockIvaMemType	内存类型
RockIvaObjectType	目标类型
RockIvaObjectState	目标跟踪状态
RockIvaDetModel	前级目标检测算法模型
RockIvaWorkMode	工作模式
RockIvaCameraType	摄像头类型

4.1.2.2.1 RockIvaLogLevel

【功能】

日志打印级别

【声明】

```
typedef enum {  
    ROCKIVA_LOG_ERROR = 0,  
    ROCKIVA_LOG_WARN = 1,  
    ROCKIVA_LOG_DEBUG = 2,  
    ROCKIVA_LOG_INFO = 3,  
    ROCKIVA_LOG_TRACE = 4  
} RockIvaLogLevel;
```

【成员】

成员	描述
ROCKIVA_LOG_ERROR	错误级别（默认）
ROCKIVA_LOG_WARN	警告级别
ROCKIVA_LOG_DEBUG	调试级别
ROCKIVA_LOG_INFO	信息级别
ROCKIVA_LOG_TRACE	跟踪调用级别

【说明】

日志等级数值越大，打印的日志越多。

4.1.2.2.2 RockIvaRetCode

【功能】

函数调用返回错误码

【声明】

```
typedef enum {
    ROCKIVA_RET_SUCCESS = 0,
    ROCKIVA_RET_FAIL = -1,
    ROCKIVA_RET_NULL_PTR = -2,
    ROCKIVA_RET_INVALID_HANDLE = -3,
    ROCKIVA_RET_LICENSE_ERROR = -4,
    ROCKIVA_RET_UNSUPPORTED = -5,
    ROCKIVA_RET_STREAM_SWITCH = -6,
    ROCKIVA_RET_BUFFER_FULL = -7,
} RockIvaRetCode;
```

【成员】

成员	描述
ROCKIVA_RET_SUCCESS	成功
ROCKIVA_RET_FAIL	失败
ROCKIVA_RET_NULL_PTR	空指针
ROCKIVA_RET_INVALID_HANDLE	无效句柄
ROCKIVA_RET_LICENSE_ERROR	License 错误
ROCKIVA_RET_UNSUPPORTED	不支持
ROCKIVA_RET_STREAM_SWITCH	码流切换
ROCKIVA_RET_BUFFER_FULL	缓存区域满

4.1.2.2.3 RockIvaExecuteStatus

【功能】

执行回调结果状态码

【声明】

```
typedef enum {  
    ROCKIVA_SUCCESS = 0,  
    ROCKIVA_UNKNOWN = 1,  
    ROCKIVA_NULL_PTR = 2,  
    ROCKIVA_ALLOC_FAILED = 3,  
    ROCKIVA_INVALID_INPUT = 4,  
    ROCKIVA_EXECUTE_FAILED = 6,  
    ROCKIVA_NOT_CONFIGURED = 7,  
    ROCKIVA_NO_CAPACITY = 8,  
    ROCKIVA_BUFFER_FULL = 9,  
    ROCKIVA_LICENSE_ERROR = 10,  
    ROCKIVA_JPEG_DECODE_ERROR = 11,  
    ROCKIVA_DECODER_EXIT = 12,  
} RockIvaExecuteStatus;
```

【成员】

成员	描述
ROCKIVA_SUCCESS	运行结果正常
ROCKIVA_UNKNOWN	错误类型未知
ROCKIVA_NULL_PTR	操作空指针
ROCKIVA_ALLOC_FAILED	申请空间失败
ROCKIVA_INVALID_INPUT	无效的输入
ROCKIVA_EXECUTE_FAILED	内部执行错误
ROCKIVA_NOT_CONFIGURED	未配置的类型
ROCKIVA_NO_CAPACITY	业务已建满
ROCKIVA_BUFFER_FULL	缓存区域满
ROCKIVA_LICENSE_ERROR	license 异常
ROCKIVA_JPEG_DECODE_ERROR	解码出错
ROCKIVA_DECODER_EXIT	解码器内部退出

4.1.2.2.4 RockIvalmageFormat

【功能】

图像像素格式类型

【声明】

```
typedef enum {  
    ROCKIVA_IMAGE_FORMAT_GRAY8 = 0,  
    ROCKIVA_IMAGE_FORMAT_RGB888,  
    ROCKIVA_IMAGE_FORMAT_BGR888,  
    ROCKIVA_IMAGE_FORMAT_RGBA8888,  
    ROCKIVA_IMAGE_FORMAT_BGRA8888,  
}
```

```
ROCKIVA_IMAGE_FORMAT_YUV420P_YU12,  
ROCKIVA_IMAGE_FORMAT_YUV420P_YV12,  
ROCKIVA_IMAGE_FORMAT_YUV420SP_NV12,  
ROCKIVA_IMAGE_FORMAT_YUV420SP_NV21,  
ROCKIVA_IMAGE_FORMAT_JPEG,  
} RockIvaImageFormat;
```

【成员】

成员	描述
ROCKIVA_IMAGE_FORMAT_GRAY8	单通道灰度图
ROCKIVA_IMAGE_FORMAT_RGB888	三通道彩色图像
ROCKIVA_IMAGE_FORMAT_BGR888	三通道彩色图像
ROCKIVA_IMAGE_FORMAT_RGBA8888	四通道彩色图像
ROCKIVA_IMAGE_FORMAT_BGRA8888	四通道彩色图像
ROCKIVA_IMAGE_FORMAT_YUV420P_YU12	YUV420P_YU12
ROCKIVA_IMAGE_FORMAT_YUV420P_YV12	YUV420P_YV12
ROCKIVA_IMAGE_FORMAT_YUV420SP_NV12	YUV420SP_NV12
ROCKIVA_IMAGE_FORMAT_YUV420SP_NV21	YUV420SP_NV21
ROCKIVA_IMAGE_FORMAT_JPEG	JPEG 图像（输入图像不支持该格式）

【注意事项】

无

4.1.2.2.5 RockIvalImageTransform

【功能】

输入图像的旋转模式

【声明】

```
typedef enum {  
    ROCKIVA_IMAGE_TRANSFORM_NONE = 0x00,  
    ROCKIVA_IMAGE_TRANSFORM_FLIP_H = 0x01,  
    ROCKIVA_IMAGE_TRANSFORM_FLIP_V = 0x02,  
    ROCKIVA_IMAGE_TRANSFORM_ROTATE_90 = 0x04,  
    ROCKIVA_IMAGE_TRANSFORM_ROTATE_180 = 0x03,  
    ROCKIVA_IMAGE_TRANSFORM_ROTATE_270 = 0x07,  
} RockIvaImageFormat;
```

【成员】

成员	描述
ROCKIVA_IMAGE_TRANSFORM_NONE	正常
ROCKIVA_IMAGE_TRANSFORM_FLIP_H	水平翻转
ROCKIVA_IMAGE_TRANSFORM_FLIP_V	垂直翻转
ROCKIVA_IMAGE_TRANSFORM_ROTATE_90	顺时针 90 度
ROCKIVA_IMAGE_TRANSFORM_ROTATE_180	顺时针 180 度
ROCKIVA_IMAGE_TRANSFORM_ROTATE_270	顺时针 270 度

4.1.2.2.6 RockIvaObjectType

【功能】

检测的目标类别

【声明】

```
typedef enum {
    ROCKIVA_OBJECT_TYPE_NONE = 0,
    ROCKIVA_OBJECT_TYPE_PERSON = 1,
    ROCKIVA_OBJECT_TYPE_VEHICLE = 2,
    ROCKIVA_OBJECT_TYPE_NON_VEHICLE = 3,
    ROCKIVA_OBJECT_TYPE_FACE = 4,
    ROCKIVA_OBJECT_TYPE_HEAD = 5,
    ROCKIVA_OBJECT_TYPE_PET = 6,
    ROCKIVA_OBJECT_TYPE_MOTORCYCLE = 7,
    ROCKIVA_OBJECT_TYPE_BICYCLE = 8,
    ROCKIVA_OBJECT_TYPE_PLATE = 9,
    ROCKIVA_OBJECT_TYPE_BABY = 10,
    ROCKIVA_OBJECT_TYPE_MAX
} RockIvaObjectType;
```

【成员】

成员	描述
ROCKIVA_OBJECT_TYPE_NONE	未知
ROCKIVA_OBJECT_TYPE_PERSON	人形
ROCKIVA_OBJECT_TYPE_VEHICLE	机动车
ROCKIVA_OBJECT_TYPE_NON_VEHICLE	非机动车
ROCKIVA_OBJECT_TYPE_FACE	人脸
ROCKIVA_OBJECT_TYPE_HEAD	人头
ROCKIVA_OBJECT_TYPE_PET	猫、狗
ROCKIVA_OBJECT_TYPE_MOTORCYCLE	电瓶车
ROCKIVA_OBJECT_TYPE_BICYCLE	自行车
ROCKIVA_OBJECT_TYPE_PLATE	车牌
ROCKIVA_OBJECT_TYPE_BABY	婴幼儿

4.1.2.2.7 RockIvaObjectState

【功能】

目标跟踪状态

【声明】

```
typedef enum {  
    ROCKIVA_OBJECT_STATE_NONE,  
    ROCKIVA_OBJECT_STATE_FIRST,  
    ROCKIVA_OBJECT_STATE_TRACKING,  
    ROCKIVA_OBJECT_STATE_LOST,  
    ROCKIVA_OBJECT_STATE_DISPEAR,  
} RockIvaObjectState;
```

【成员】

成员	描述
ROCKIVA_OBJECT_STATE_NONE	目标状态未知
ROCKIVA_OBJECT_STATE_FIRST	目标第一次出现
ROCKIVA_OBJECT_STATE_TRACKING	目标检测跟踪过程中
ROCKIVA_OBJECT_STATE_LOST	目标检测丢失
ROCKIVA_OBJECT_STATE_DISPEAR	目标消失

4.1.2.2.8 RockIvaDetModel

【功能】

前级目标检测算法模型

【声明】

```
typedef enum {  
    ROCKIVA_DET_MODEL_NONE = 0,  
    ROCKIVA_DET_MODEL_CLS7,  
    ROCKIVA_DET_MODEL_PFCP,  
    ROCKIVA_DET_MODEL_PFP,  
    ROCKIVA_DET_MODEL_PHS,  
    ROCKIVA_DET_MODEL_PHCP,  
    ROCKIVA_DET_MODEL_PERSON,  
    ROCKIVA_DET_MODEL_NONVEHICLE,  
    ROCKIVA_DET_MODEL_FHS,  
    ROCKIVA_DET_MODEL_PHCN,  
    ROCKIVA_DET_MODEL_CLS8,  
} RockIvaDetModel;
```

【成员】

成员	描述
ROCKIVA_DET_MODEL_NONE	未设置，不需要前级目标检测
ROCKIVA_DET_MODEL_CLS7	模型文件：object_detection_ipc_cls7.data 检测类别：人形、人脸、机动车、车牌、非机动车、宠物
ROCKIVA_DET_MODEL_PFCP	模型文件：object_detection_pfcps.data 检测类别：人形、人脸、机动车、宠物
ROCKIVA_DET_MODEL_PFP	模型文件：object_detection_pfps.data 检测类别：人形、人脸、宠物
ROCKIVA_DET_MODEL_PHS	模型文件：object_detection_phs.data 检测类别：人形、头肩
ROCKIVA_DET_MODEL_PHCP	模型文件：object_detection_phcps.data 检测类别：人形、头肩、机动车、宠物
ROCKIVA_DET_MODEL_PERSON	模型文件：object_detection_person.data 检测类别：人形
ROCKIVA_DET_MODEL_NONVEHICLE	模型文件：object_detection_motorcycle.data 检测类别：电动车
ROCKIVA_DET_MODEL_FHS	模型文件：object_detection_fhs.data 检测类别：人脸、头肩
ROCKIVA_DET_MODEL_PHCN	模型文件：object_detection_phcns.data 检测类别：人形、头肩、机动车、非机动车
ROCKIVA_DET_MODEL_CLS8	模型文件：object_detection_v3_cls8.data 检测类别：人形、人脸、机动车、车牌、非机动车、宠物、人头 注：相比 CLS7 速度更快，但内存会略微增加

【说明】

4.1.2.2.9 RockIvaWorkMode

【功能】

工作模式

【声明】

```
typedef enum {  
    ROCKIVA_MODE_VIDEO = 0,  
    ROCKIVA_MODE_PICTURE = 1,  
} RockIvaWorkMode;
```

【成员】

成员	描述
ROCKIVA_MODE_VIDEO	视频流模式(使能跟踪)
ROCKIVA_MODE_PICTURE	图片模式(不使能跟踪)

【说明】

周界功能必须要工作在视频流模式；检测和人脸功能可以使用视频流或图片模式；

4.1.2.2.10 RockIvaMemType

【功能】

内存类型

【声明】

```
typedef enum {  
    ROCKIVA_MEM_TYPE_CPU = 0,  
    ROCKIVA_MEM_TYPE_DMA = 1,  
} RockIvaMemType;
```

【成员】

成员	描述
ROCKIVA_MEM_TYPE_CPU	虚拟地址内存
ROCKIVA_MEM_TYPE_DMA	DMA BUF 内存（RV1106）

【说明】

为图像分配的缓冲区内内存类型，对于 RV1106 平台 RGA 需要使用 DMA BUF 内存类型。

4.1.2.2.11 RockIvaCameraType

【功能】

摄像头类型

【声明】

```
typedef enum {  
    ROCKIVA_CAMERA_TYPE_ONE = 0,  
    ROCKIVA_CAMERA_TYPE_DUAL = 1,  
    ROCKIVA_CAMERA_TYPE_SL = 2,  
} RockIvaCameraType;
```

【成员】

成员	描述
ROCKIVA_CAMERA_TYPE_ONE	单摄
ROCKIVA_CAMERA_TYPE_DUAL	双摄，能够提供 RGB 和 IR 两路图像
ROCKIVA_CAMERA_TYPE_SL	结构光摄像头，能够提供 RGB、IR 和 Depth 三路图像

4.1.2.3 结构体

结构体	描述
RockIvaPoint	点坐标
RockIvaLine	线坐标
RockIvaRectangle	正四边形坐标
RockIvaQuadrangle	四边形坐标
RockIvaArea	多边形区域
RockIvaAreas	多边形区域列表
RockIvaSize	目标大小
RockIvaAngle	角度信息
RockIvaRectExpandRatio	检测框向四周扩展的比例大小配置
RockIvaImageInfo	图像信息
RockIvaImage	图像
RockIvaObjectInfo	单个目标检测结果信息
RockIvaDetectResult	目标检测结果
RockIvaReleaseFrames	需要释放的帧列表
RockIvaInitParam	SDK 全局参数配置

4.1.2.3.1 RockIvaPoint

【功能】

点坐标

【声明】

```
typedef struct {  
    uint16_t x;  
    uint16_t y;
```

```
} RockIvaPoint;
```

【成员】

成员	描述
x	横坐标，万分比表示，0~9999 有效
y	纵坐标，万分比表示，0~9999 有效

4.1.2.3.2 RockIvaLine**【功能】**

线坐标

【声明】

```
typedef struct {  
    RockIvaPoint head;  
    RockIvaPoint tail;  
} RockIvaLine;
```

【成员】

成员	描述
head	头坐标
tail	尾坐标

4.1.2.3.3 RockIvaRectangle**【功能】**

正四边形坐标

【声明】

```
typedef struct {  
    RockIvaPoint topLeft;  
    RockIvaPoint bottomRight;  
} RockIvaRectangle;
```

【成员】

成员	描述
topLeft	左上角坐标
bottomRight	右下角坐标

4.1.2.3.4 RockIvaQuadrangle

【功能】

任意四边形坐标

【声明】

```
typedef struct {
    RockIvaPoint topLeft;
    RockIvaPoint topRight;
    RockIvaPoint bottomLeft;
    RockIvaPoint bottomRight;
} RockIvaQuadrangle;
```

【成员】

成员	描述
topLeft	左上角坐标
topRight	右上角坐标
bottomLeft	左下角坐标
bottomRight	右下角坐标

4.1.2.3.5 RockIvaArea

【功能】

多边形区域坐标

【声明】

```
typedef struct {
    uint32_t pointNum;
    RockIvaPoint points[ROCKIVA_AREA_POINT_NUM_MAX];
} RockIvaArea;
```

【成员】

成员	描述
pointNum	点个数
points	围成区域的所有点坐标

4.1.2.3.6 RockIvaAreas

【功能】

多区域

【声明】

```
typedef struct {  
    uint32_t areaNum;  
    RockIvaArea areas[ROCKIVA_AREA_NUM_MAX];  
} RockIvaAreas;
```

【成员】

成员	描述
areaNum	区域数量
areas	区域数组

4.1.2.3.7 RockIvaSize

【功能】

目标大小向四周

【声明】

```
typedef struct {  
    uint16_t width;  
    uint16_t height;  
} RockIvaSize;
```

【成员】

成员	描述
width	宽度（万分比坐标，值范围 0~9999）
height	高度（万分比坐标，值范围 0~9999）

4.1.2.3.8 RockIvaRectExpandRatio

【功能】

检测框向四周扩展的比例大小配置

【声明】

```
typedef struct {  
    float up;  
    float down;  
    float left;  
    float right;  
} RockIvaRectExpandRatio;
```

【成员】

成员	描述
up	检测框向上按框高扩展的比例大小
down	检测框向下按框高扩展的比例大小
left	检测框向左按框宽扩展的比例大小
right	检测框向右按框宽扩展的比例大小

4.1.2.3.9 RockIvaAngle

【功能】

人脸角度信息

【声明】

```
typedef struct {  
    int16_t pitch;  
    int16_t yaw;  
    int16_t roll;  
} RockIvaAngle;
```

【成员】

成员	描述
pitch	俯仰角,表示绕 x 轴旋转角度
yaw	偏航角,表示绕 y 轴旋转角度
roll	翻滚角,表示绕 z 轴旋转角度

4.1.2.3.10 RockIvalmageInfo

【功能】

图像信息

【声明】

```
typedef struct {
    uint16_t width;
    uint16_t height;
    RockIvaImageFormat format;
    RockIvaImageTransform transformMode;
} RockIvaImageInfo;
```

【成员】

成员	描述
width	图像宽度
height	图像高度
format	图像像素格式
transformMode	旋转模式

4.1.2.3.11 RockIvalmage

【功能】

图像

【声明】

```
typedef struct {
    uint32_t frameId;
    uint32_t channelId;
    RockIvaImageInfo info;
    uint32_t size;
    uint8_t* dataAddr;
    uint8_t* dataPhyAddr;
    int32_t dataFd;
    void* extData;
} RockIvaImage;
```

【成员】

成员	描述
frameId	原始采集帧序号（应用自定义）
channelId	通道号（应用自定义，单通道置为 0）
info	图像信息
size	图像数据大小，图像格式为 JPEG 时需要
dataAddr	输入图像数据虚地址
dataPhyAddr	输入图像数据的物理地址（没有用则置为 NULL）
dataFd	输入图像数据的文件句柄（没有用则置为 0）
extData	用户自定义扩展数据

4.1.2.3.12 RockIvaObjectInfo

【功能】

单个目标检测结果信息

【声明】

```
typedef struct {  
    uint32_t objId;  
    uint32_t frameId;  
    uint32_t score;  
    RockIvaRectangle rect;  
    RockIvaObjectType type;  
} RockIvaObjectInfo;
```

【成员】

成员	描述
objId	目标 ID[0~2 ³²)
frameId	帧 ID
score	目标检测分数 [1-100]
rect	目标区域框 (万分比)
type	目标类别

4.1.2.3.13 RockIvaDetectResult

【功能】

目标检测结果

【声明】

```
typedef struct {  
    uint32_t frameId;  
    RockIvaImage frame;
```



```
uint32_t channelId;  
uint32_t objNum;  
RockIvaObjectInfo objInfo[ROCKIVA_MAX_OBJ_NUM];  
} RockIvaDetectResult;
```

【成员】

成员	描述
frameId	帧 ID
frame	对应的输入图像帧
channelId	通道 ID
objNum	目标个数
objInfo	各目标检测信息

4.1.2.3.14 RockIvaReleaseFrames**【功能】**

可以释放的帧列表

【声明】

```
typedef struct {  
    uint32_t channelId;  
    uint32_t count;  
    RockIvaImage frameId[ROCKIVA_MAX_OBJ_NUM];  
} RockIvaReleaseFrames;
```

【成员】

成员	描述
channelId	通道 ID
count	数量
frameId	可释放的帧

4.1.2.3.15 RockIvaMediaOps**【功能】**

多媒体操作函数

【声明】

```
typedef struct {  
    int (*ROCKIVA_Yuv2Jpeg)(RockIvaImage *inputImg, RockIvaImage *outputImg, void *userdata);  
} RockIvaMediaOps;
```

【成员】

成员	描述
ROCKIVA_Yuv2Jpeg	将图像编码为 JPEG 图像

4.1.2.3.16 RockIvaInitParam

【功能】

算法全局参数配置

【声明】

```
typedef struct {  
    RockIvaLogLevel logLevel;  
    char logPath[ROCKIVA_PATH_LENGTH];  
    char modelPath[ROCKIVA_PATH_LENGTH];  
    RockIvaMemInfo license;  
    uint32_t coreMask;  
    uint32_t channelId;  
    RockIvaImageInfo imageInfo;  
    RockIvaAreas roiAreas;  
    RockIvaCameraType cameraType;  
    RockIvaDetModel detModel;  
    uint32_t detClasses;  
    RockIvaMemInfo detModelData;  
    uint8_t trackerVersion;  
    RockIvaMediaOps mediaOps;  
} RockIvaInitParam;
```

【成员】

成员	描述
RockIvaLogLevel	日志等级
logPath	日志输出路径
modelPath	存放算法模型路径
license	License 信息
coreMask	指定使用哪个 NPU 核跑(仅 RK3588 平台有效)
channelId	通道号
imageInfo	输入图像信息
roiAreas	有效区域
cameraType	摄像头类型
detModel	前级目标检测模型
detClasses	正常不需要配置，只有一些特殊检测类别需要配置，如婴儿检测： ROCKIVA_OBJECT_TYPE_BITMASK(ROCKIVA_OBJECT_TYPE_BABY)
detModelData	指定检测模型数据（用于快启直接配置模型内存数据直接用于加载）
trackerVersion	指定跟踪算法版本 0:默认 v2; 1:v1; 2:v2
mediaOps	设置媒体操作回调函数

4.2 检测模块

4.2.1 功能描述

检测模型主要获取图像中检测到的目标的框坐标和分数等信息。检测模块对应头文件为 rockiva_det_api.h，可以通过 ROCKIVA_DETECT_Init 配置初始化参数并设置检测结果回调。基本调用流程如图 4-2-1 所示。

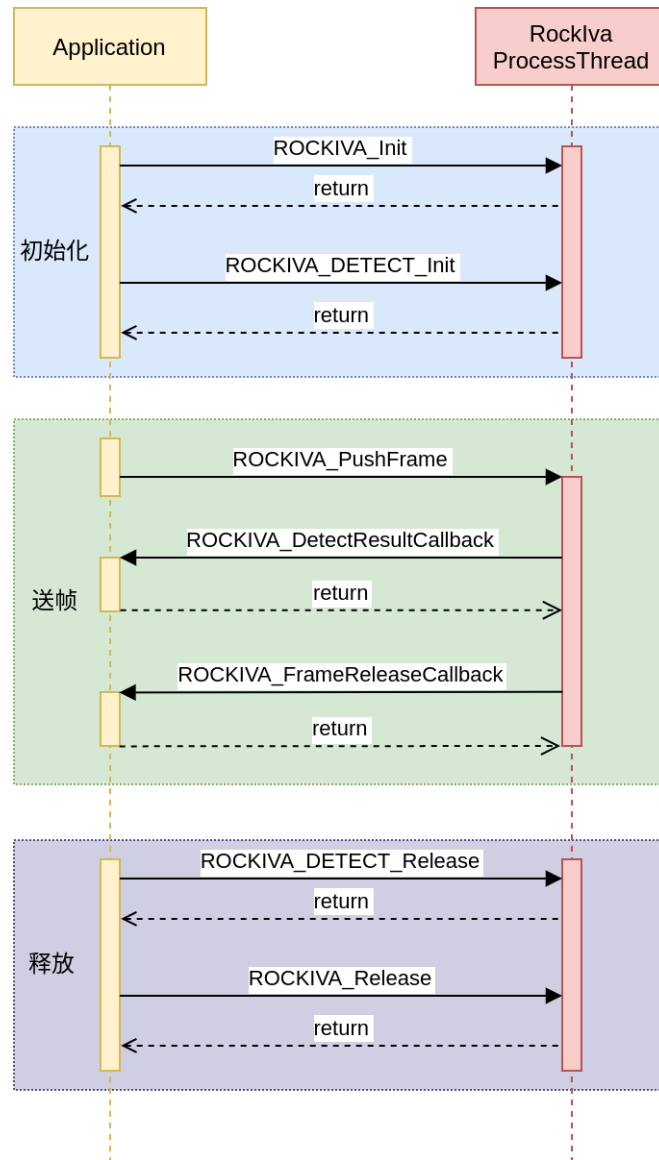


图 4-2-1 检测功能调用流程

配置初始化后，SDK 内部会创建线程异步处理每一帧，应用每次推送的图像帧都会放到一个缓存队列中送给处理线程。需要注意的是结果回调函数 `ROCKIVA_DetectResultCallback` 是运行在该内部处理线程，因此尽量不要做耗时太久的操作，否则会影响处理帧率。

检测模块可以工作在视频流模式或图片模式。视频流模式下会使能目标跟踪，检测回调中的每个检测目标的 `objId` 有效；图片模式下会关闭目标跟踪，检测目标的 `objId` 值无效。

对于用户推送进来的每一帧同样是通过 `ROCKIVA_FrameReleaseCallback` 回调函数返回的结果进行释放。

以下为检测模块的参考代码实例：

```
RockIvaRetCode ret;  
RockIvaHandle handle;
```

```
// 释放回调函数（同通用模块示例代码）
...

// 检测模块回调函数
void DetResultCallback(const RockIvaDetectResult* result, const RockIvaExecuteStatus status, void*
userdata)
{
    // 处理结果
}

// 初始化 IVA 实例
RockIvaInitParam commonParams;
memset(&commonParams, 0, sizeof(RockIvaInitParam));
commonParams.detModel = ROCKIVA_DET_MODEL_CLS7; // 设置检测模型
ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);
// 设置回调函数
ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);

// 初始化检测模块
RockIvaDetTaskParams detParams;
memset(&detParams, 0, sizeof(RockIvaDetTaskParams));
// 设置上报目标类型
detParams->detObjectType |= ROCKIVA_OBJECT_TYPE_BITMASK(ROCKIVA_OBJECT_TYPE_PERSON)
// 设置检测分数阈值，只设置第 0 个可以对所有类别生效
detParams->scores[0] = 30;
ret = ROCKIVA_DETECT_Init(handle, &detParams, DetResultCallback);

// 送帧处理（同通用模块示例代码）
...

// 等待所有帧都处理完成
ROCKIVA_WaitFinish(handle, -1, 5000);
// 销毁检测模块
ROCKIVA_DETECT_Release(handle)
// 销毁 IVA 实例
ROCKIVA_Release(handle);
```

4.2.2 API 参考

4.2.2.1 函数接口

接口	描述
ROCKIVA_DetectResultCallback	结果回调函数
ROCKIVA_DETECT_Init	初始化

ROCKIVA_DETECT_Release	释放
ROCKIVA_DETECT_Reset	运行时重新配置(重新配置会导致内部的一些记录清空复位，但是模型不会重新初始化)

4.2.2.1.1 ROCKIVA_DetectResultCallback

【功能】

结果回调函数

【声明】

```
typedef void (*ROCKIVA_DetectResultCallback)(
    const RockIvaDetectResult* result,
    const RockIvaExecuteStatus status,
    void* userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输入	结果
status	输入	状态码
userdata	输入	用户自定义数据

【返回值】

RockIvaRetCode

4.2.2.1.2 ROCKIVA_DETECT_Init

【功能】

初始化检测模块

【声明】

```
RockIvaRetCode ROCKIVA_DETECT_Init(RockIvaHandle handle,
    const RockIvaDetTaskParams* params,
    const ROCKIVA_DetectResultCallback resultCallback);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	实例句柄
params	输入	初始化参数
resultCallback	输入	结果回调函数

【返回值】

RockIvaRetCode

4.2.2.1.3 ROCKIVA_DETECT_Release**【功能】**

释放

【声明】

```
RockIvaRetCode ROCKIVA_DETECT_Release(RockIvaHandle handle);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	实例句柄

【返回值】

RockIvaRetCode

4.2.2.1.4 ROCKIVA_DETECT_Reset**【功能】**

运行时重新配置(重新配置会导致内部的一些记录清空复位，但是模型不会重新初始化)

【声明】

```
RockIvaRetCode ROCKIVA_DETECT_Reset(RockIvaHandle handle ,  
                                      RockIvaDetTaskParams* params);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要重置的 handle，运行时重新配置(重新配置会导致内部的一些记录清空复位，但是模型不会重新初始化)
params	输入	初始化参数

【返回值】

RockIvaRetCode

4.2.2.2 结构体

结构体	描述
RockIvaDetTaskParams	目标检测业务初始化参数配置

4.2.2.2.1 RockIvaDetTaskParams

【功能】

目标检测业务初始化参数配置

【声明】

```
typedef struct {  
    uint32_t detObjectType;  
    RockIvaAreas roiAreas;  
    uint8_t scores[ROCKIVA_OBJECT_TYPE_MAX];  
    uint8_t min_det_count;  
} RockIvaDetTaskParams;
```

【成员】

成员	描述
detObjectType	配置要返回的检测目标
roiAreas	配置有效检测区域（仅影响检测结果）
scores	各类别过滤分数阈值，0 为内部自动
min_det_count	检测目标稳定检测帧数大于该数值后再上报，过滤一些小概率误检

4.3 周界模块

4.3.1 功能描述

周界功能的流程基本和检测类似，通过 ROCKIVA_BA_Init 初始化周界相关配置以及配置周

界结果回调。

周界功能模块代码调用流程如图 4-3-1 所示。

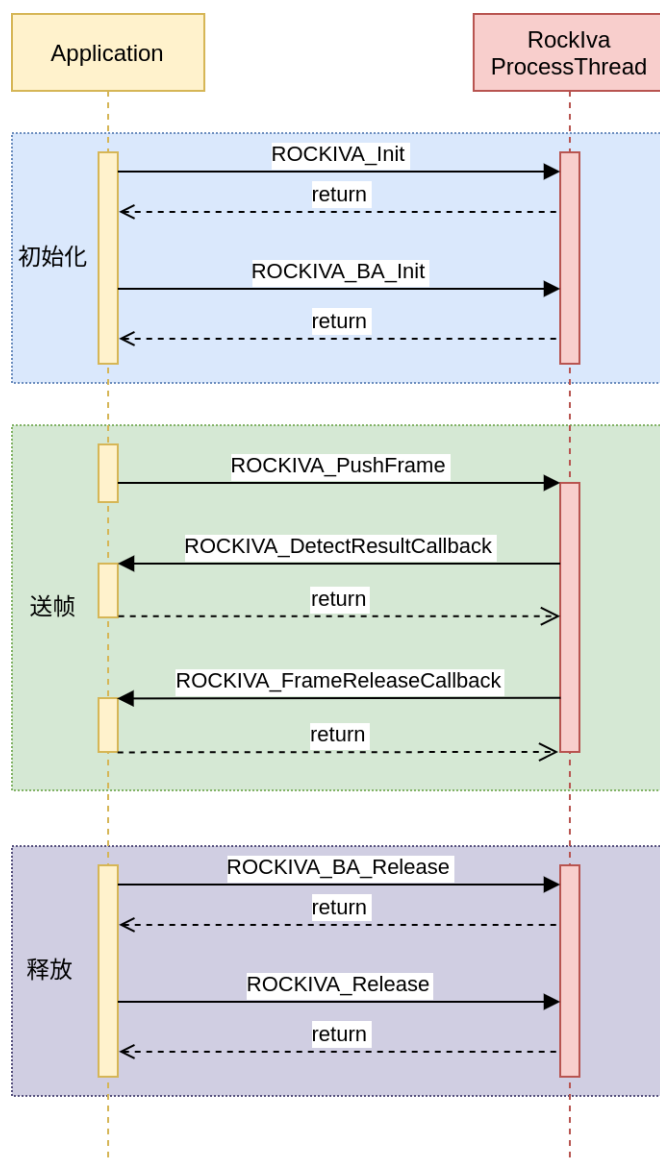


图 4-3-1 周界功能调用流程

注意周界的工作模式必须要是视频流模式（`ROCKIVA_MODE_VIDEO`），因为周界的规则判断需要根据目标前后帧的结果进行计算。

周界功能有四种类型：

- 区域入侵：目标进入设定区域并停留超过设定时间触发
- 区域进入：目标从设定区域外进入区域内触发
- 区域离开：目标从设定区域内离开区域外触发
- 越界检测：目标从设定的线段越过（符合设定方向）触发

可配置规则主要有：

规则配置	说明
区域	设定规则区域：有顺序的最多 6 个点构成的多边形
界线	设定规则界线：两个点构成的线段，可设置方向
目标类型过滤	设定触发的目标类型：人形、车辆、非机动车
目标大小过滤	设定目标大小：最小和最大宽高

区域入侵结果展示如图 4-3-2 和图 4-3-3 所示。目标进入禁区后会上报事件。

越界结果展示如图 4-3-4 所示。目标由禁止的方向越过边界线时会上报事件。



图 4-3-2 区域入侵进入禁区前



图 4-3-3 区域入侵进入禁区后

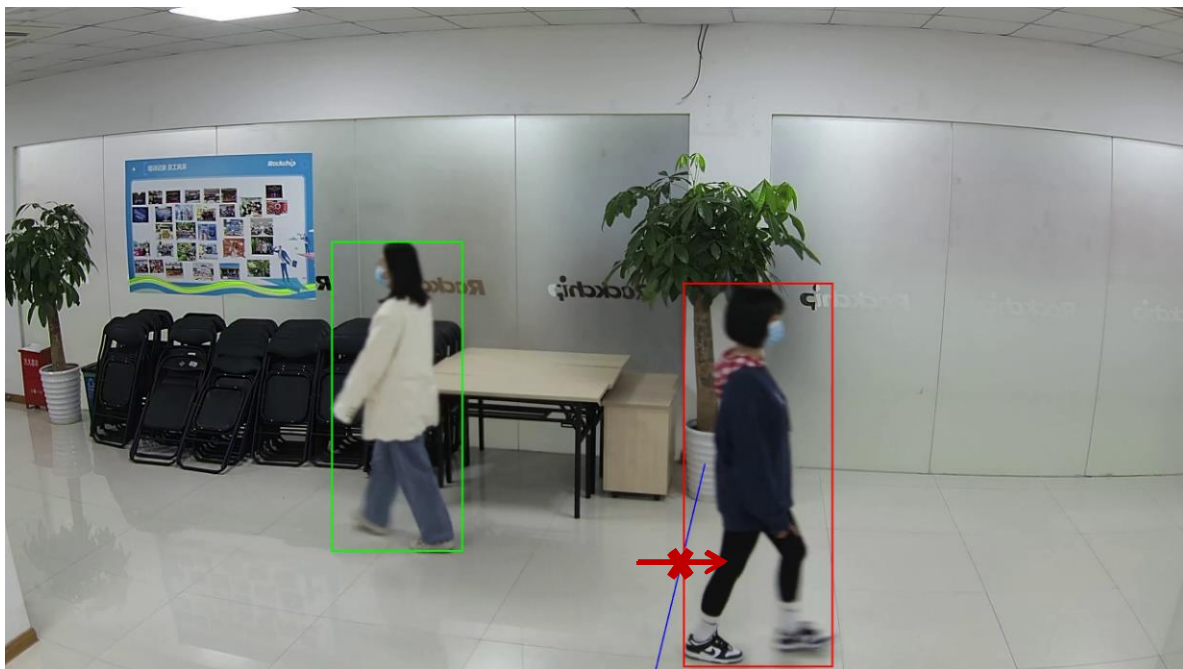


图 4-3-4 越界结果

以下为周界模块相关参考代码

```
RockIvaRetCode ret;  
RockIvaHandle handle;  
  
// 释放回调函数（同通用模块示例代码）  
...  
  
// 周界模块回调函数  
void BaResultCallback(const RockIvaBaResult* result, const RockIvaExecuteStatus status, void* userdata)  
{  
    // 处理结果  
}  
  
// 初始化 IVA 实例  
RockIvaInitParam commonParams;  
memset(&commonParams, 0, sizeof(RockIvaInitParam));  
commonParams.detModel = ROCKIVA_DET_MODEL_CLS7; // 设置检测模型  
ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);  
// 设置回调函数  
ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);  
  
// 初始化周界模块  
RockIvaBaTaskParams baParams;  
memset(&baParams, 0, sizeof(RockIvaBaTaskParams));
```

```
// 构建一个区域入侵规则
baParams.baRules.areaInBreakRule[0].ruleEnable = 1;
baParams.baRules.areaInBreakRule[0].alertTime = 1000; // 目标在区域内触发时间
baParams.baRules.areaInBreakRule[0].ruleID = 1;
baParams.baRules.areaInBreakRule[0].objType |=
    ROCKIVA_OBJECT_TYPE_BITMASK(ROCKIVA_OBJECT_TYPE_PERSON);

// 区域由多个点按顺序连成
baParams.baRules.areaInBreakRule[0].area.pointNum = 4;
baParams.baRules.areaInBreakRule[0].area.points[0].x = 1000;
baParams.baRules.areaInBreakRule[0].area.points[0].y = 1000;
baParams.baRules.areaInBreakRule[0].area.points[1].x = 9000;
baParams.baRules.areaInBreakRule[0].area.points[1].y = 1000;
baParams.baRules.areaInBreakRule[0].area.points[2].x = 9000;
baParams.baRules.areaInBreakRule[0].area.points[2].y = 9000;
baParams.baRules.areaInBreakRule[0].area.points[3].x = 1000;
baParams.baRules.areaInBreakRule[0].area.points[3].y = 9000;

// 构建一个越界规则
baParams.baRules.tripWireRule[0].ruleEnable = 1;
baParams.baRules.tripWireRule[0].event = ROCKIVA_BA_TRIP_EVENT_BOTH;
baParams.baRules.tripWireRule[0].ruleID = 2;
baParams.baRules.tripWireRule[0].objType |=
    ROCKIVA_OBJECT_TYPE_BITMASK(ROCKIVA_OBJECT_TYPE_PERSON);
baParams.baRules.tripWireRule[0].line.head.x = 1000;
baParams.baRules.tripWireRule[0].line.head.y = 5000;
baParams.baRules.tripWireRule[0].line.tail.x = 9000;
baParams.baRules.tripWireRule[0].line.tail.y = 5000;

ret = ROCKIVA_BA_Init(handle, &baParams, BaResultCallback);

// 送帧处理（同通用模块示例代码）
...

// 等待所有帧都处理完成
ROCKIVA_WaitFinish(handle, -1, 5000);
// 销毁周界模块
ROCKIVA_BA_Release(handle)
// 销毁 IVA 实例
ROCKIVA_Release(handle);
```

4.3.2 API 参考

4.3.2.1 函数接口

接口	描述
ROCKIVA_BA_ResultCallback	结果回调函数
ROCKIVA_BA_Init	初始化
ROCKIVA_BA_Destroy	销毁
ROCKIVA_BA_Reset	重置

4.3.2.1.1 ROCKIVA_BA_ResultCallback

【功能】

结果回调函数

【声明】

```
typedef void(*ROCKIVA_BA_ResultCallback)(
    const RockIvaBaResult *result,
    const RockIvaExecuteStatus status,
    void *userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输入	结果
status	输入	状态码
userdata	输入	用户自定义数据

【返回值】

RockIvaRetCode

4.3.2.1.2 ROCKIVA_BA_Init

【功能】

初始化

【声明】

```
RockIvaRetCode ROCKIVA_BA_Init(RockIvaHandle handle,
```

```
const RockIvaBaTaskParams *initParams,
const ROCKIVA_BA_ResultCallback resultCallback);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要初始化的 handle
initParams	输入	初始化参数配置
resultCallback	输入	回调函数

【返回值】

RockIvaRetCode

4.3.2.1.3 ROCKIVA_BA_Destroy

【功能】

销毁

【声明】

```
RockIvaRetCode ROCKIVA_BA_Destroy(RockIvaHandle handle);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要销毁的 handle

【返回值】

RockIvaRetCode

4.3.2.1.4 ROCKIVA_BA_Reset

【功能】

重置

【声明】

```
RockIvaRetCode ROCKIVA_BA_Reset(RockIvaHandle handle, const RockIvaBaTaskParams* params);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要重置的 handle
params	输入	新的初始化参数配置

【返回值】

RockIvaRetCode

4.3.2.2 枚举定义

枚举	描述
RockIvaBaTripEvent	周界规则类型（绊线/区域事件）
RockIvaBaRuleTriggerType	目标规则触发类型

4.3.2.2.1 RockIvaBaTripEvent

【功能】

周界规则绊线事件类型

【声明】

```
typedef enum {  
    ROCKIVA_BA_TRIP_EVENT_BOTH = 0,  
    ROCKIVA_BA_TRIP_EVENT_DEASIL = 1,  
    ROCKIVA_BA_TRIP_EVENT_WIDDERSHINES = 2,  
} RockIvaBaTripEvent;
```

【成员】

成员	描述
ROCKIVA_BA_TRIP_EVENT_BOTH	绊线:双向触发
ROCKIVA_BA_TRIP_EVENT_DEASIL	绊线:顺时针触发
ROCKIVA_BA_TRIP_EVENT_WIDDERSHINES	绊线:逆时针触发

4.3.2.2.2 RockIvaBaRuleTriggerType

【功能】

目标规则触发类型

【声明】

```
typedef enum {
```

```
ROCKIVA_BA_RULE_NONE = 0,  
ROCKIVA_BA_RULE_CROSS = 0x1,  
ROCKIVA_BA_RULE_INAREA = 0x2,  
ROCKIVA_BA_RULE_OUTAREA = 0x4,  
ROCKIVA_BA_RULE_STAY = 0x8,  
} RockIvaBaRuleTriggerType;
```

【成员】

成员	描述
ROCKIVA_BA_RULE_NONE	
ROCKIVA_BA_RULE_CROSS	拌线
ROCKIVA_BA_RULE_INAREA	进入区域
ROCKIVA_BA_RULE_OUTAREA	离开区域
ROCKIVA_BA_RULE_STAY	区域入侵

4.3.2.3 结构体定义

结构体	描述
RockIvaBaWireRule	越界规则
RockIvaBaAreaRule	区域规则
RockIvaBaTaskRule	行为分析类规则配置
RockIvaBaAiConfig	算法配置
RockIvaBaTaskParams	行为分析业务初始化参数配
RockIvaBaTrigger	第一次触发规则信息
RockIvaBaObjectInfo	单个目标检测基本信息
RockIvaBaResult	检测结果全部信息

4.3.2.3.1 RockIvaBaWireRule

【功能】

越界规则

【声明】


```
typedef struct {
    uint8_t ruleEnable;
    uint32_t ruleID;
    RockIvaLine line;
    RockIvaBaTripEvent event;
    RockIvaSize minObjSize[ROCKIVA_OBJECT_TYPE_MAX];
    RockIvaSize maxObjSize[ROCKIVA_OBJECT_TYPE_MAX];
    uint8_t scores[ROCKIVA_OBJECT_TYPE_MAX];
    uint32_t objType;
    uint8_t rulePriority;
    uint8_t sense;
    uint8_t triggerMode;
} RockIvaBaWireRule;
```

【成员】

成员	描述
ruleEnable	规则是否启用，1->启用，0->不启用
ruleID	规则 ID，有效范围[0, 3]
line	越界线配置
event	越界方向
minObjSize	万百分比表示 最小目标: 0 机动车 1 非机动车 2 行人
maxObjSize	万百分比表示 最大目标: 0 机动车 1 非机动车 2 行人
scores	各类别过滤分数阈值，0 为内部自动
objType	配置置触发目标类型：如
rulePriority	规则优先级: 0 高， 1 中， 2 低
sense	灵敏度,1~100
triggerMode	触发模式: 0: 目标仅触发一次; 1: 目标每次越界都触发

4.3.2.3.2 RockIvaBaAreaRule

【功能】

区域规则

【声明】

```
typedef struct {  
    uint8_t ruleEnable;  
    uint32_t ruleID;  
    RockIvaArea area;  
    RockIvaSize minObjSize[ROCKIVA_OBJECT_TYPE_MAX];  
    RockIvaSize maxObjSize[ROCKIVA_OBJECT_TYPE_MAX];  
    uint8_t scores[ROCKIVA_OBJECT_TYPE_MAX];  
    uint32_t objType;  
    uint32_t alertTime;  
    uint8_t sense;  
    uint8_t checkEnter;  
} RockIvaBaAreaRule;
```

【成员】

成员	描述
ruleEnable	规则是否启用，1->启用，0->不启用
ruleID	规则 ID，有效范围[1, 256]
area	区域配置
minObjSize	万百分比表示 最小目标: 0 机动车 1 非机动车 2 行人
maxObjSize	万百分比表示 最大目标: 0 机动车 1 非机动车 2 行人
scores	各类别过滤分数阈值，0 为内部自动
objType	配置触发目标类型
alertTime	区域入侵规则告警时间设置
sense	灵敏度,1~100
checkEnter	区域入侵规则是否需要检查目标有进入 [0: 不启用, 1: 启用]

4.3.2.3.3 RockIvaBaTaskRule**【功能】**

周界规则配置

【声明】

```
typedef struct {  
    RockIvaBaWireRule tripWireRule[ROCKIVA_BA_MAX_RULE_NUM];  
    RockIvaBaAreaRule areaInRule[ROCKIVA_BA_MAX_RULE_NUM];  
    RockIvaBaAreaRule areaOutRule[ROCKIVA_BA_MAX_RULE_NUM];  
    RockIvaBaAreaRule areaInBreakRule[ROCKIVA_BA_MAX_RULE_NUM];  
} RockIvaBaTaskRule;
```

【成员】

成员	描述
tripWireRule	越界事件
areaInRule	进入区域
areaOutRule	离开区域
areaInBreakRule	区域入侵

4.3.2.3.4 RockIvaBaAiConfig

【功能】

算法配置

【声明】

```
typedef struct {
    uint8_t filterPersonMode;
    uint8_t detectResultMode;
} RockIvaBaAiConfig;
```

【成员】

成员	描述
filterPersonMode	过滤非机动车/机动车驾驶员人形结果 0: 不过滤; 1: 过滤
detectResultMode	上报目标检测结果模式 0: 不上报没有触发规则的检测目标; 1: 上报没有触发规则的检测目标

4.3.2.3.5 RockIvaBaTaskParams

【功能】

周界初始化参数配置

【声明】

```
typedef struct {
    RockIvaBaTaskRule baRules;
    RockIvaBaAiConfig aiConfig;
} RockIvaBaTaskParams;
```

【成员】

成员	描述
baRules	周界规则参数配置初始化
aiConfig	算法配置

4.3.2.3.6 RockIvaBaTrigger

【功能】

目标第一次触发规则信息，可以用于抓拍

【声明】

```
typedef struct {  
    int32_t ruleID;  
    RockIvaBaRuleTriggerType triggerType;  
} RockIvaBaTrigger;
```

【成员】

成员	描述
ruleID	触发规则 ID
triggerType	触发规则类型

4.3.2.3.7 RockIvaBaObjectInfo

【功能】

触发周界规则的单个目标信息

【声明】

```
typedef struct {  
    RockIvaObjectInfo objInfo;  
    uint32_t triggerRules;  
    uint32_t triggerRulesNum;  
    uint32_t triggerRulesID[ROCKIVA_BA_MAX_RULE_NUM];  
    RockIvaBaTrigger firstTrigger;  
} RockIvaBaObjectInfo;
```

【成员】

成员	描述
objInfo	目标检测结果信息
triggerRules	目标触发的规则
triggerRulesNum	目标触发规则数量
triggerRulesID	目标触发的所有规则 ID
firstTrigger	目标第一次触发的规则

4.3.2.3.8 RockIvaBaResult

【功能】

检测结果全部信息

【声明】

```
typedef struct {  
    uint32_t frameId;  
    RockIvaImage frame;  
    uint32_t channelId;  
    uint32_t objNum;  
    RockIvaObjectInfo triggerObjects [ROCKIVA_MAX_OBJ_NUM];  
} RockIvaBaResult;
```

【成员】

成员	描述
frameId	输入图像帧 ID
frame	对应的输入图像帧
channelId	通道号
objNum	目标个数
triggerObjects	触发周界规则的目标

4.4 人脸模块

4.4.1 功能描述

人脸功能根据设置的人脸抓拍规则抓拍人脸图像，抓拍模式分为最优抓拍和快速抓拍，并可对抓拍的人脸进行属性分析包括年龄、性别、口罩等。抓拍模式包括：

- 最优抓拍：抓拍一张目标人从出现到超时时间内或消失时满足规则的最优人脸
- 快速抓拍：抓拍一张目标人出现后满足规则的第一张人脸

- 手动抓拍：可以由应用自定义设置需要触发抓拍的人脸

1) 人脸抓拍模式

人脸抓拍必须要工作在视频流模式下，对于图像中的检测到的人脸会进行人脸质量的计算，然后多帧比较选择最优人脸，当满足抓拍条件（根据配置的抓拍规则：最优抓拍/快速抓拍）将触发一次抓拍，回调返回结果。

人脸抓拍功能接口调用流程如图 4-4-1 所示。

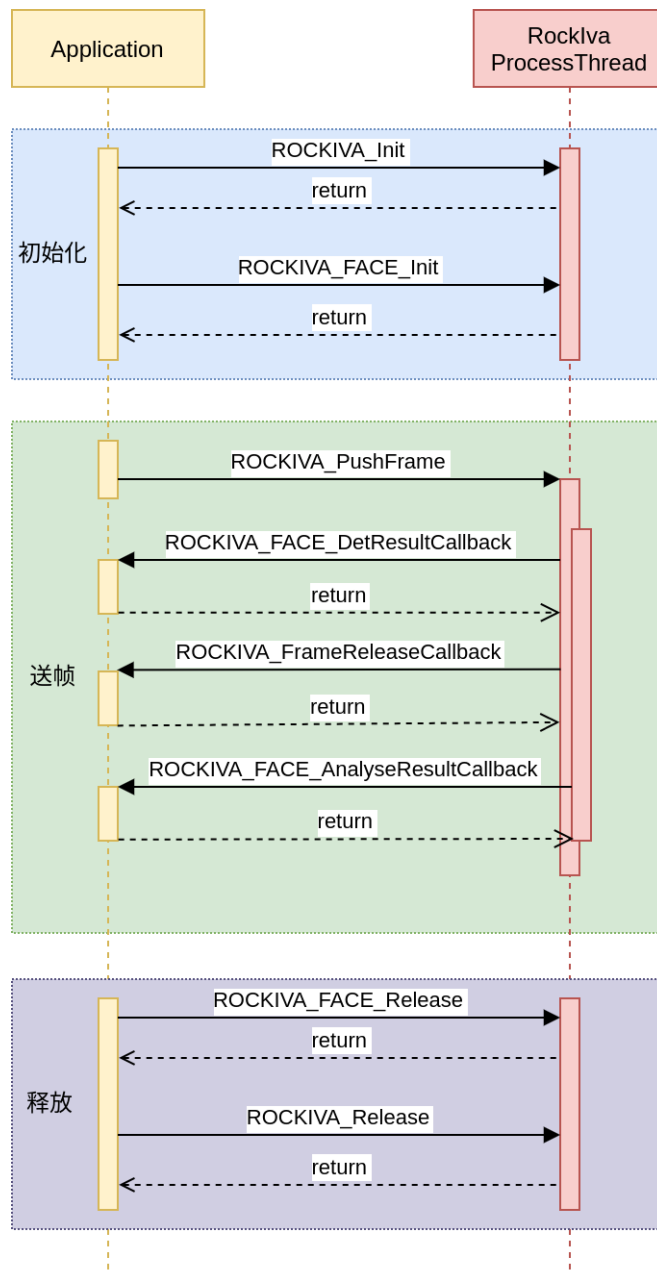


图 4-4-1 人脸抓拍接口调用流程

在从图像中检测到人脸后，对人脸进行质量计算后将通过通过回调函数

ROCKIVA_FACE_DetResultCallback 返回目标检测和质量的结果信息。目标人脸满足触发条件后触发抓拍规则后根据配置的人脸分析、人脸特征提取将在另外线程中异步执行，执行完后通过回调函数 ROCKIVA_FACE_AnalyseResultCallback 返回。

人脸抓拍参考代码如下：

a) 最优抓拍

```
RockIvaRetCode ret;
RockIvaHandle handle;

// 释放回调函数（同通用模块示例代码）
...

// 人脸模块回调函数
void FaceDetResultCallback(const RockIvaFaceDetResult* result, const RockIvaExecuteStatus status, void*
userdata)
{
    // 处理人脸检测结果
}

void FaceAnalyseResultCallback(const RockIvaFaceCapResults* result, const RockIvaExecuteStatus status,
void* userdata)
{
    // 处理人脸抓拍结果
}

// 初始化 IVA 实例
RockIvaInitParam commonParams;
memset(&commonParams, 0, sizeof(RockIvaInitParam));
commonParams.detModel = ROCKIVA_DET_MODEL_PFP; // 设置检测模型
ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);
// 设置回调函数
ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);

// 初始化人脸模块
RockIvaFaceTaskParams faceParams;
memset(&faceParams, 0, sizeof(RockIvaFaceTaskParams));
faceParams->faceTaskType.faceCaptureEnable = 1;
faceParams->faceCaptureRule.optType = ROCKIVA_FACE_OPT_BEST;
faceParams->faceCaptureRule.optBestOverTime = 10000;
faceParams->faceCaptureRule.faceCapacity.maxCaptureNum = 10; // 支持同时抓拍 10 个人脸
// 人脸过滤配置
// 最低人脸质量分阈值，小于阈值将过滤
faceParams->faceCaptureRule.qualityConfig.minScore = 50;
// 遮挡阈值，小于阈值将过滤
faceParams->faceCaptureRule.qualityConfig.minEyescore = 60;
faceParams->faceCaptureRule.qualityConfig.minOcclusionScore = 85;
// 人脸角度，大于阈值将过滤
```

```
faceParams->faceCaptureRule.qualityConfig.maxAngle.pitch = 60;
faceParams->faceCaptureRule.qualityConfig.maxAngle.yaw = 60;
faceParams->faceCaptureRule.qualityConfig.maxAngle.roll = 60;

// 抓拍小图配置
faceParams->faceCaptureRule.captureImageConfig.mode = 1;
faceParams->faceCaptureRule.captureImageConfig.originImageType = 1;
faceParams->faceCaptureRule.captureImageConfig.resizeMode = 1;
faceParams->faceCaptureRule.captureImageConfig.imageInfo.width = 240;
faceParams->faceCaptureRule.captureImageConfig.imageInfo.height = 320;
faceParams->faceCaptureRule.captureImageConfig.imageInfo.format =
    ROCKIVA_IMAGE_FORMAT_RGB888;
faceParams->faceCaptureRule.captureImageConfig.alignWidth = 16;

faceParams->faceCaptureRule.captureImageConfig.expand.up = 0.6;
faceParams->faceCaptureRule.captureImageConfig.expand.down = 0.6;
faceParams->faceCaptureRule.captureImageConfig.expand.left = 0.6;
faceParams->faceCaptureRule.captureImageConfig.expand.right = 0.6;

ret = ROCKIVA_FACE_Init(ctx->handle, faceParams, callback);

// 送帧处理（同通用模块示例代码）
...

// 等待所有帧都处理完成
ROCKIVA_WaitFinish(handle, -1, 5000);
// 销毁检测模块
ROCKIVA_FACE_Release(handle)
// 销毁 IVA 实例
ROCKIVA_Release(handle);
```

a) 快速抓拍

快速抓拍模式除了初始化配置有差异，其他代码和最优抓拍一致

```
...
// 初始化人脸模块
RockIvaFaceTaskParams faceParams;
memset(&faceParams, 0, sizeof(RockIvaFaceTaskParams));
faceParams->faceTaskType.faceCaptureEnable = 1;
faceParams->faceCaptureRule.optType = ROCKIVA_FACE_OPT_FAST;
faceParams->faceCaptureRule.faceQualityThreshold = 60;    //人脸质量超过设定阈值就触发抓拍
...
```

a) 手动抓拍

手动抓拍需要有应用自定义过滤条件来触发。

```
RockIvaRetCode ret;
```



```

RockIvaHandle handle;

// 释放回调函数（同通用模块示例代码）
...

// 人脸模块回调函数
// 人脸检测结果回调
void FaceDetResultCallback(const RockIvaFaceDetResult* result, const RockIvaExecuteStatus status, void*
userdata)
{
    // 手动筛选条件触发抓拍
    for (int i = 0; i < result->objNum; i++) {
        RockIvaFaceInfo* face = &result->faceInfo[i];
        if (face->faceQuality.score < 60 ) {
            continue;
        }
        if (face->faceQuality.angle.pitch > 45 || face->faceQuality.angle.roll > 45 ||
face->faceQuality.angle.yaw > 45) {
            continue;
        }
        if (face->faceQuality.pd < 40) {
            continue;
        }
        ROCKIVA_FACE_SetAnalyseFace(iva_ctx->handle, face->objId);
    }
}

// 人脸抓拍结果回调
void FaceAnalyseResultCallback(const RockIvaFaceCapResults* result, const RockIvaExecuteStatus status,
void* userdata)
{
    ...
}

// 初始化 IVA 实例
RockIvaInitParam commonParams;
memset(&commonParams, 0, sizeof(RockIvaInitParam));
commonParams.detModel = ROCKIVA_DET_MODEL_PFP; // 设置检测模型
ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);
// 设置回调函数
ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);

// 初始化人脸模块
RockIvaFaceTaskParams faceParams;
memset(&faceParams, 0, sizeof(RockIvaFaceTaskParams));
faceParams->faceTaskType.faceCaptureEnable = 0; // 不使能抓拍
faceParams->faceCaptureRule.faceCapacity.maxCaptureNum = 10; // 支持同时抓拍 10 个人脸

// 抓拍人脸小图配置
faceParams->faceCaptureRule.captureImageConfig.mode = 1;
faceParams->faceCaptureRule.captureImageConfig.originImageType = 1;

```

```
faceParams->faceCaptureRule.captureImageConfig.resizeMode = 1;
faceParams->faceCaptureRule.captureImageConfig.imageInfo.width = 240;
faceParams->faceCaptureRule.captureImageConfig.imageInfo.height = 320;
faceParams->faceCaptureRule.captureImageConfig.imageInfo.format =
    ROCKIVA_IMAGE_FORMAT_RGB888;
faceParams->faceCaptureRule.captureImageConfig.alignWidth = 16;

faceParams->faceCaptureRule.captureImageConfig.expand.up = 0.6;
faceParams->faceCaptureRule.captureImageConfig.expand.down = 0.6;
faceParams->faceCaptureRule.captureImageConfig.expand.left = 0.6;
faceParams->faceCaptureRule.captureImageConfig.expand.right = 0.6;

ret = ROCKIVA_FACE_Init(ctx->handle, faceParams, callback);

// 送帧处理（同通用模块示例代码）
...

// 等待所有帧都处理完成
ROCKIVA_WaitFinish(handle, -1, 5000);
// 销毁检测模块
ROCKIVA_FACE_Release(handle)
// 销毁 IVA 实例
ROCKIVA_Release(handle);
```

2) 单张人脸图像分析模式

单张人脸图像分析模式下，将对图像中检测出的每个目标做人脸质量，对于人脸质量良好的人脸将根据配置开关做人脸属性分析与人脸特征提取。单张人脸图像分析的模式下将不会做目标跟踪和人脸质量优选。代码示例可以参考：`demo/rockiva_demo/face_recog_picture_demo.c`。

3) 人脸导库模式

人脸导库模式和单张人脸图像分析模式类似，不过在人脸导库所送入图像分辨率差异很大，因此内部会选择一个更合适的人脸检测模型，因此导库模式下无法和其他功能一起混用。人脸导库模式下只会选择一个最大的人脸做人脸特征提取，外面拿到人脸特征结果后入库。代码示例可以参考：`demo/rockiva_demo/face_recog_import_demo.c`。

4.4.2 API 参考

4.4.2.1 函数接口

接口	描述
ROCKIVA_FACE_DetResultCallback	人脸检测结果回调函数
ROCKIVA_FACE_AnalyseResultCallback	人脸抓拍分析结果回调函数
ROCKIVA_FACE_PostureResultCallback	坐姿检测结果回调函数
ROCKIVA_FACE_Init	初始化
ROCKIVA_FACE_Release	释放
ROCKIVA_FACE_Reset	运行时重新配置
ROCKIVA_FACE_SetAnalyseFace	指定需要抓拍分析识别的人脸
ROCKIVA_FACE_SetFilteredFace	指定不需要抓拍的人脸
ROCKIVA_FACE_FeatureCompare	1:1 人脸特征比对接口
ROCKIVA_FACE_FeatureLibraryControl	人脸特征布控接口,用于对某个人脸库进行增加、删除、修改
ROCKIVA_FACE_SearchFeature	人脸特征库检索接口,用于对某个人脸库的特征进行检索

4.4.2.1.1 ROCKIVA_FACE_DetResultCallback

【功能】

人脸检测结果回调函数

【声明】

```
typedef void(*ROCKIVA_FACE_DetResultCallback) (  
    const RockIvaFaceDetResult *result,  
    const RockIvaExecuteStatus status,  
    void *userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输入	人脸检测结果
status	输入	执行状态码
userdata	输入	用户自定义数据

【返回值】

RockIvaRetCode

4.4.2.1.2 ROCKIVA_FACE_AnalyseResultCallback

【功能】

人脸抓拍分析结果回调函数

【声明】

```
typedef void(*ROCKIVA_FACE_AnalyseResultCallback)(  
    const RockIvaFaceCapResult *result,  
    const RockIvaExecuteStatus status,  
    void *userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输入	人脸抓拍分析结果
status	输入	执行状态码
userdata	输入	用户自定义数据

【返回值】

RockIvaRetCode

4.4.2.1.3 ROCKIVA_FACE_PostureResultCallback

【功能】

坐姿检测结果回调函数

【声明】

```
typedef void(*ROCKIVA_FACE_PostureResultCallback)(  
    const RockIvaFacePostureResult *result,  
    const RockIvaExecuteStatus status,  
    void *userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输入	结果
status	输入	执行状态码
userdata	输入	用户自定义数据

【返回值】

RockIvaRetCode

4.4.2.1.4 ROCKIVA_FACE_Init

【功能】

人脸功能初始化

【声明】

```
RockIvaRetCode ROCKIVA_FACE_Init(RockIvaHandle handle,  
                                   RockIvaFaceWorkType workType,  
                                   const RockIvaFaceTaskParams *initParams);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入输出	需要初始化的 handle
workType	输入	人脸任务模式
initParams	输入	初始化参数

【返回值】

RockIvaRetCode

4.4.2.1.5 ROCKIVA_FACE_Reset

【功能】

重新设置人脸功能配置参数

【声明】

```
RockIvaRetCode ROCKIVA_FACE_Reset(RockIvaHandle handle, const RockIvaFaceTaskParams*  
params);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要重置的 handle
params	输入	新的初始化参数配置

【返回值】

RockIvaRetCode

4.4.2.1.6 ROCKIVA_FACE_Release

【功能】

销毁人脸模块

【声明】

```
RockIvaRetCode ROCKIVA_FACE_Release(RockIvaHandle handle);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要释放的句柄

【返回值】

RockIvaRetCode

4.4.2.1.7 ROCKIVA_FACE_SetAnalyseFace

【功能】

指定需要抓拍分析识别的人脸，可以调用多次设定多个人脸（需要在检测结果回调中使用，仅 faceCaptureEnable==0 时候可用）

【声明】

```
RockIvaRetCode ROCKIVA_FACE_SetAnalyseFace (  
    RockIvaHandle handle,  
    int trackId);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	handle
trackId	输入	人脸跟踪 id（RockIvaFaceInfo 的 objId）

【返回值】

RockIvaRetCode

4.4.2.1.8 ROCKIVA_FACE_SetFilteredFace

【功能】

指定不需要的人脸，可以调用多次设定多个人脸（只可在检测结果回调 ROCKIVA_FACE_DetResultCallback 中使用）

【声明】

```
RockIvaRetCode ROCKIVA_FACE_SetFilteredFace (  
    RockIvaHandle handle,  
    int trackId);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	handle
trackId	输入	人脸跟踪 id（RockIvaFaceInfo 的 objId）

【返回值】

RockIvaRetCode

4.4.2.1.9 ROCKIVA_FACE_FeatureCompare

【功能】

1:1 人脸特征比对接口

【声明】

```
RockIvaRetCode ROCKIVA_FACE_FeatureCompare(  
    const void* feature1,  
    const void* feature2,  
    float* score);
```

【输入参数】

参数名称	输入/输出	描述
feature1	输入	人脸特征 1
feature2	输入	人脸特征 2
score	输出	人脸 1:1 比对相似度(范围 0.00-1.00)

【返回值】

RockIvaRetCode

4.4.2.1.10 ROCKIVA_FACE_FeatureLibraryControl

【功能】

人脸特征布控接口,用于对某个人脸库进行增、删、改

【声明】

```
RockIvaRetCode ROCKIVA_FACE_FeatureLibraryControl(  
    const char* libName,  
    ROCKIVAFaceLibraryAction action,  
    RockIvaFaceIdInfo *faceIdInfo,  
    uint32_t faceIdNum,  
    const void* faceIdData,  
    int featureSize);
```

【输入参数】

参数名称	输入/输出	描述
libName	输入	人脸库名称
action	输入	人脸库操作类型：增加、删除、修改、查找、清除
faceIdInfo	输入	人脸 ID
faceIdNum	输入	人脸 ID 数量
faceIdData	输入	人脸特征数据
featureSize	输入	人脸特征数据大小

【返回值】

RockIvaRetCode

4.4.2.1.11 ROCKIVA_FACE_SearchFeature

【功能】

人脸特征库检索接口,用于对某个人脸库的特征进行检索

【声明】

```
RockIvaRetCode ROCKIVA_FACE_SearchFeature(  
    const char *libName,  
    const void *featureData,  
    uint32_t featureSize,  
    uint32_t num,  
    uint32_t topK,  
    RockIvaFaceSearchResults *results);
```


【输入参数】

参数名称	输入/输出	描述
libName	输入	人脸库名称
featureData	输入	人脸特征
featureSize	输入	人脸特征数据大小
num	输入	对比特征的个数
topK	输入	前 K 个最相似的特征值
results	输出	比对结果

【返回值】

RockIvaRetCode

4.4.2.2枚举定义

枚举	描述
RockIvaFaceOptType	人脸优选类型
RockIvaFacePriorityMode	多人脸选择顺序
RockIvaFaceWorkMode	人脸业务类型
RockIvaFaceStatus	人脸状态
RockIvaFaceGenderType	性别
RockIvaFaceAgeType	年龄
RockIvaFaceGlassesType	眼镜
RockIvaFaceSmileType	微笑
RockIvaFaceMaskType	佩戴口罩
RockIvaFaceBeardType	胡子
RockIvaFaceCapFrameType	抓拍类型
RockIvaFaceQualityResultCode	人脸质量结果
RockIvaFaceLibraryAction	人脸库特征更新操作类型

4.4.2.2.1 RockIvaFaceOptType

【功能】

人脸优选类型

【声明】

```
typedef enum {  
    ROCKIVA_FACE_OPT_BEST,  
    ROCKIVA_FACE_OPT_FAST,  
} RockIvaFaceOptType;
```

【成员】

成员	描述
ROCKIVA_FACE_OPT_BEST	效果优先模式，目标从出现到消失的最优人脸
ROCKIVA_FACE_OPT_FAST	快速优先模式，目标满足设定质量阈值的人脸

4.4.2.2.2 RockIvaFacePriorityMode

【功能】

多人脸时优先选择顺序

【声明】

```
typedef enum {
    ROCKIVA_FACE_PRIO_SIZE,
    ROCKIVA_FACE_PRIO_CENTER,
} RockIvaFacePriorityMode;
```

【成员】

成员	描述
ROCKIVA_FACE_PRIO_SIZE	按人脸从大到小顺序选择
ROCKIVA_FACE_PRIO_CENTER	按人脸靠近中心顺序选择

4.4.2.2.3 RockIvaFaceWorkMode

【功能】

人脸业务类型

【声明】

```
typedef enum {
    ROCKIVA_FACE_MODE_NORMAL = 0,
    ROCKIVA_FACE_MODE_IMPORT = 1,
    ROCKIVA_FACE_MODE_SEARCH = 2,
    ROCKIVA_FACE_MODE_PANEL = 3,
} RockIvaFaceWorkMode;
```

【成员】

成员	描述
ROCKIVA_FACE_MODE_NORMAL	正常模式(根据 RockIvaWorkMode 配置)
ROCKIVA_FACE_MODE_IMPORT	导库模式(底图特征提取)
ROCKIVA_FACE_MODE_SEARCH	以图搜图模式（暂未实现）
ROCKIVA_FACE_MODE_PANEL	人脸面板机模式（近距离人脸识别分析，支持多摄）

4.4.2.2.4 RockIvaFaceState

【功能】

人脸状态

【声明】

```
typedef enum {  
    ROCKIVA_FACE_STATE_NONE,  
    ROCKIVA_FACE_STATE_FIRST,  
    ROCKIVA_FACE_STATE_TRACKING,  
    ROCKIVA_FACE_STATE_CAPTURING,  
    ROCKIVA_FACE_STATE_ANALYZING,  
    ROCKIVA_FACE_STATE_CAPTURED,  
    ROCKIVA_FACE_STATE_LAST  
} RockIvaFaceStatus;
```

【成员】

成员	描述
ROCKIVA_FACE_STATE_NONE	人脸状态未知
ROCKIVA_FACE_STATE_FIRST	人脸第一次出现
ROCKIVA_FACE_STATE_TRACKING	人脸检测跟踪过程中
ROCKIVA_FACE_STATE_CAPTURING	人脸进入抓拍处理
ROCKIVA_FACE_STATE_ANALYZING	人脸触发抓拍上报，进行人脸分析中
ROCKIVA_FACE_STATE_CAPTURED	人脸抓拍处理完成（完成抓拍回调函数结束后进入该状态，或没有满足质量且不需上报则直接进入该状态）
ROCKIVA_FACE_STATE_LAST	目标最后一次出现

4.4.2.2.5 RockIvaFaceGenderType

【功能】

性别

【声明】

```
typedef enum {  
    ROCKIVA_GENDER_TYPE_UNKNOWN    = 0,  
    ROCKIVA_GENDER_TYPE_MALE      = 1,  
    ROCKIVA_GENDER_TYPE_FEMALE    = 2  
} RockIvaFaceGenderType;
```

【成员】

成员	描述
ROCKIVA_GENDER_TYPE_UNKNOWN	未知
ROCKIVA_GENDER_TYPE_MALE	男性
ROCKIVA_GENDER_TYPE_FEMALE	女性

4.4.2.2.6 RockIvaFaceAgeType**【功能】**

年龄

【声明】

```
typedef enum {  
    ROCKIVA_AGE_TYPE_UNKNOWN      = 0,  
    ROCKIVA_AGE_TYPE_CHILD        = 1,  
    ROCKIVA_AGE_TYPE_EARLYYOUTH  = 2,  
    ROCKIVA_AGE_TYPE_YOUTH        = 3,  
    ROCKIVA_AGE_TYPE_MIDLIFE      = 4,  
    ROCKIVA_AGE_TYPE_OLD          = 5  
} RockIvaFaceAgeType;
```

【成员】

成员	描述
ROCKIVA_AGE_TYPE_UNKNOWN	未知
ROCKIVA_AGE_TYPE_CHILD	儿童
ROCKIVA_AGE_TYPE_EARLYYOUTH	少年
ROCKIVA_AGE_TYPE_YOUTH	青年
ROCKIVA_AGE_TYPE_MIDLIFE	中年
ROCKIVA_AGE_TYPE_OLD	老年

4.4.2.2.7 RockIvaFaceGlassesType**【功能】**

佩戴眼镜类型

【声明】

```
typedef enum {  
    ROCKIVA_GLASSES_TYPE_UNKNOWN    = 0,  
    ROCKIVA_GLASSES_TYPE_NOGLASSES = 1,  
    ROCKIVA_GLASSES_TYPE_GLASSES   = 2,  
    ROCKIVA_GLASSES_TYPE_SUNGLASSES = 3  
} RockIvaFaceGlassesType;
```

【成员】

成员	描述
ROCKIVA_GLASSES_TYPE_UNKNOWN	未知
ROCKIVA_GLASSES_TYPE_NOGLASSES	不戴眼镜
ROCKIVA_GLASSES_TYPE_GLASSES	戴眼镜
ROCKIVA_GLASSES_TYPE_SUNGLASSES	太阳眼镜（预留）

4.4.2.2.8 RockIvaFaceMaskType**【功能】**

是否戴口罩

【声明】

```
typedef enum {  
    ROCKIVA_MASK_TYPE_UNKNOWN    = 0,  
    ROCKIVA_MASK_TYPE_YES       = 1,  
    ROCKIVA_MASK_TYPE_NO        = 2  
} RockIvaFaceMaskType;
```

【成员】

成员	描述
ROCKIVA_MASK_TYPE_UNKNOWN	未知
ROCKIVA_MASK_TYPE_YES	戴口罩
ROCKIVA_MASK_TYPE_NO	不戴口罩

4.4.2.2.9 RockIvaFaceBeardType**【功能】**

是否有胡子

【声明】

```
typedef enum {
```

```
ROCKIVA_BEARD_TYPE_UNKNOWN    = 0,  
ROCKIVA_BEARD_TYPE_YES       = 1,  
ROCKIVA_BEARD_TYPE_NO        = 2  
} RockIvaFaceBeardType;
```

【成员】

成员	描述
ROCKIVA_BEARD_TYPE_UNKNOWN	未知
ROCKIVA_BEARD_TYPE_YES	有胡子
ROCKIVA_BEARD_TYPE_NO	没有胡子

4.4.2.2.10 RockIvaFaceCapFrameType

【功能】

抓拍类型

【声明】

```
typedef enum {  
    ROCKIVA_FACE_CAP_TYPE_UNKNOWN,  
    ROCKIVA_FACE_CAP_TYPE_IMPORT,  
    ROCKIVA_FACE_CAP_TYPE_PICTURE,  
    ROCKIVA_FACE_CAP_TYPE_DISPEAR,  
    ROCKIVA_FACE_CAP_TYPE_TIMEOUT,  
    ROCKIVA_FACE_CAP_TYPE_QUALITY,  
    ROCKIVA_FACE_CAP_TYPE_FORENOTICE,  
    ROCKIVA_FACE_CAP_TYPE_CUSTOM,  
} RockIvaFaceCapFrameType;
```

【成员】

成员	描述
ROCKIVA_FACE_CAP_TYPE_UNKNOWN	未知
ROCKIVA_FACE_CAP_TYPE_IMPORT	人脸导库模式结果帧
ROCKIVA_FACE_CAP_TYPE_PICTURE	单图运行模式结果帧
ROCKIVA_FACE_CAP_TYPE_DISPEAR	目标消失抓拍帧
ROCKIVA_FACE_CAP_TYPE_TIMEOUT	超时抓拍帧
ROCKIVA_FACE_CAP_TYPE_QUALITY	满足阈值抓拍
ROCKIVA_FACE_CAP_TYPE_FORENOTICE	提前上报人脸分析结果（如口罩佩戴判断等）
ROCKIVA_FACE_CAP_TYPE_CUSTOM	用户指定

4.4.2.2.11 RockIvaFaceQualityResultCode

【功能】

人脸质量结果

【声明】

```
typedef enum {  
    ROCKIVA_FACE_QUALITY_OK,  
    ROCKIVA_FACE_QUALITY_NO_FACE,  
    ROCKIVA_FACE_QUALITY_SCORE_FAIL,  
    ROCKIVA_FACE_QUALITY_SIZE_FAIL,  
    ROCKIVA_FACE_QUALITY_CLARITY_FAIL,  
    ROCKIVA_FACE_QUALITY_ANGLE_FAIL,  
    ROCKIVA_FACE_QUALITY_MASK_FAIL,  
    ROCKIVA_FACE_QUALITY_USER_ABANDON  
} RockIvaFaceQualityResultCode;
```

【成员】

成员	描述
ROCKIVA_FACE_QUALITY_OK	人脸质量合格
ROCKIVA_FACE_QUALITY_NO_FACE	没有检测到人脸
ROCKIVA_FACE_QUALITY_SCORE_FAIL	人脸质量分过低
ROCKIVA_FACE_QUALITY_SIZE_FAIL	人脸过小
ROCKIVA_FACE_QUALITY_CLARITY_FAIL	人脸模糊
ROCKIVA_FACE_QUALITY_ANGLE_FAIL	人脸角度过大
ROCKIVA_FACE_QUALITY_MASK_FAIL	人脸遮挡
ROCKIVA_FACE_QUALITY_USER_ABANDON	用户丢弃人脸

4.4.2.2.12 RockIvaFacePoseState

【功能】

人脸坐姿结果

【声明】

```
typedef enum {  
    ROCKIVA_FACE_POSE_STATE_NONE,  
    ROCKIVA_FACE_POSE_STATE_NORMAL,  
    ROCKIVA_FACE_POSE_STATE_HUNCHBACK,  
    ROCKIVA_FACE_POSE_STATE_UP,  
    ROCKIVA_FACE_POSE_STATE_SIDE,  
    ROCKIVA_FACE_POSE_STATE_TILT  
} RockIvaFacePoseState;
```

【成员】

成员	描述
ROCKIVA_FACE_POSE_STATE_NONE	人脸坐姿未知
ROCKIVA_FACE_POSE_STATE_NORMAL	人脸坐姿正常
ROCKIVA_FACE_POSE_STATE_HUNCHBACK	人脸坐姿驼背、趴下、低头
ROCKIVA_FACE_POSE_STATE_UP	人脸坐姿仰头
ROCKIVA_FACE_POSE_STATE_SIDE	人脸坐姿侧脸
ROCKIVA_FACE_POSE_STATE_TILT	人脸坐姿歪头

4.4.2.2.13 RockIvaFaceLibraryAction

【功能】

特征更新类型

【声明】

```
typedef enum {  
    ROCKIVA_FACE_FEATURE_INSERT = 0,  
    ROCKIVA_FACE_FEATURE_DELETE = 1,  
    ROCKIVA_FACE_FEATURE_UPDATE = 2,  
    ROCKIVA_FACE_FEATURE_RETRIEVAL = 3,  
    ROCKIVA_FACE_FEATURE_CLEAR = 4,  
} RockIvaFaceLibraryAction;
```

【成员】

成员	描述
ROCKIVA_FACE_FEATURE_INSERT	添加特征
ROCKIVA_FACE_FEATURE_DELETE	删除特征
ROCKIVA_FACE_FEATURE_UPDATE	更新特征
ROCKIVA_FACE_FEATURE_RETRIEVAL	查找标签
ROCKIVA_FACE_FEATURE_CLEAR	清除库信息

4.4.2.3 结构体定义

结构体	描述
RockIvaFaceTaskType	算法业务类型
RockIvaFaceCapacity	人脸 SDK 处理能力
RockIvaFaceTaskParams	人脸分析业务初始化参数配置
RockIvaFaceRule	人脸抓拍规则设置
RockIvaFacePoseConfig	人脸坐姿检测配置
RockIvaFaceQualityConfig	人脸质量过滤配置
RockIvaCaptureImageConfig	抓拍上报图像配置
RockIvaFaceAttribute	人脸属性结构体
RockIvaFaceQualityInfo	人脸质量信息

RockIvaFaceInfo	单个目标人脸检测基本信息
RockIvaFaceAnalyseInfo	单个目标人脸检测基本信息
RockIvaFaceDetResult	人脸检测处理结果
RockIvaFaceCapResults	人脸抓拍处理结果
RockIvaFaceCapResult	单个人脸抓拍结果
RockIvaFacePostureResult	人脸坐姿处理结果
RockIvaHeadInfo	单个目标坐姿结果
RockIvaFaceIdInfo	入库特征对应的详细信息，用户输入
RockIvaFaceSearchResult	特征比对返回结果
RockIvaFaceSearchResults	特征比对返回结果列表
RockIvaFaceCallback	人脸功能回调函数

4.4.2.3.1 RockIvaFaceTaskType

【功能】

算法业务类型

【声明】

```
typedef struct {  
    uint8_t faceCaptureEnable;  
    uint8_t faceRecognizeEnable;  
    uint8_t faceAttributeEnable;  
    uint8_t relatedPersonEnable;  
    uint8_t faceLandmarkEnable;  
    uint8_t facePoseEnable;  
} RockIvaFaceTaskType;
```

【成员】

成员	描述
faceCaptureEnable	人脸抓拍业务 1：有效
faceRecognizeEnable	人脸识别业务 1：有效
faceAttributeEnable	人脸属性分析业务 1：有效
relatedPersonEnable	是否关联人体 1：有效
faceLandmarkEnable	人脸关键点 1：使能 5 点 2：使能 5 点和 106 点
facePoseEnable	人脸坐姿业务 1：有效

4.4.2.3.2 RockIvaFaceCapacity

【功能】

人脸 SDK 处理能力

【声明】

```
typedef struct {  
    uint32_t maxDetectNum;  
    uint32_t maxCaptureNum;  
    uint32_t maxRecogNum;  
} RockIvaFaceCapacity;
```

【成员】

成员	描述
maxDetectNum	最大的检测个数
maxCaptureNum	最大的抓拍个数
maxRecogNum	最大的识别个数

4.4.2.3.3 RockIvaFaceTaskParams

【功能】

人脸分析业务初始化参数配置

【声明】

```
typedef struct {  
    RockIvaFaceWorkMode mode;  
    RockIvaFaceRule faceCaptureRule;  
    RockIvaFacePoseConfig facePoseConfig;  
    RockIvaFaceTaskType faceTaskType;  
} RockIvaFaceTaskParams;
```

【成员】

成员	描述
mode	人脸任务模式
faceCaptureRule	人脸抓拍规则
facePoseConfig	人脸坐姿检测参数配置
faceTaskType	人脸业务类型：人脸抓拍业务/人脸识别业务

4.4.2.3.4 RockIvaFaceRule

【功能】

人脸抓拍规则设置

【声明】

```
typedef struct {
```

```
uint8_t sensitivity;
uint8_t detectAreaEn;
RockIvaArea detectArea;
uint8_t qualityFilterMode;
RockIvaFaceQualityConfig qualityConfig;
RockIvaFaceCapacity faceCapacity;
RockIvaCaptureImageConfig captureImageConfig;
RockIvaFaceOptType optType;
uint32_t optBestNum;
uint32_t optBestOverTime;
uint32_t faceQualityThreshold;
uint8_t captureWithMask;
uint8_t faceIouThreshold;
RockIvaFacePriorityMode prioMode;
} RockIvaFaceRule;
```

【成员】

成员	描述
sensitivity	检测灵敏度[1,100]
detectAreaEn	是否设置检测区域[0 关 1 开]
detectArea	检测区域
qualityFilterMode	人脸质量过滤模式 [0 不满足不上报, 1 不满足也上报但不进行人脸分析]
qualityConfig	人脸质量过滤配置
faceCapacity	人脸最大检测、抓拍和识别个数配置, 目前只实现设置最大抓拍个数, 0[无限制]
captureImageConfig	抓拍上报人脸图像配置
optType	人脸优选类型
optBestNum	ROCKIVA_FACE_OPT_BEST: 人脸优选张数(未实现) 范围: 1-3
optBestOverTime	人脸质量最优抓拍超时时间设置 ms, 若人脸在此时段内未消失则上报此时段内的质量最优人脸; 若为 0, 则在消失后才上报质量最优人脸
faceQualityThreshold	快速抓拍时满足抓拍的人脸质量分阈值
captureWithMask	支持抓拍戴口罩的人脸并上报人脸是否佩戴口罩[0: 关; 1: 开; 2: 开且口罩提前上报(满足 qualityConfig 配置)], 若打开则 qualityConfig 的 minMouthScore 和 minNoseScore 过滤失效
faceIouThreshold	多摄模式下 RGB 和 IR 图像中人脸需要满足的 IOU 阈值, 百分比范围[0, 100], 默认 0 为 50%
prioMode	多人脸时选择顺序

4.4.2.3.5 RockIvaFacePoseConfig

【功能】

人脸坐姿检测功能配置

【声明】

```
typedef struct {  
    uint32_t maxFrame;  
    uint32_t maxStep;  
    uint16_t minSize;  
    uint16_t faceWidth;  
    uint16_t faceHeight;  
    uint16_t headY0;  
} RockIvaFacePoseConfig;
```

【成员】

成员	描述
maxFrame	常状态持续帧数，超过阈值进行告警，实际应用需设大一点避免频繁告警
maxStep	当前帧间隔检出有效人脸和头肩帧超过阈值后，不再进行告警
minSize	最小人脸大小（万分比[0-10000]）
faceWidth	正常坐姿状态下人脸宽度（万分比[0-10000]）
faceHeight	正常坐姿状态下人脸高度（万分比[0-10000]）
headY0	正常坐姿状态下头肩检测框 y0 坐标（万分比[0-10000]）

4.4.2.3.6 RockIvaFaceQualityConfig**【功能】**

人脸质量过滤配置

【声明】

```
typedef struct {  
    uint16_t minScore;  
    uint16_t minSize;  
    uint16_t minPd;  
    uint16_t minClarity;  
    RockIvaAngle maxAngle;  
    uint16_t minEyescore;  
    uint16_t minNoseScore;  
    uint16_t minMouthScore;  
    uint16_t minOcclusionScore;  
} RockIvaFaceQualityConfig;
```

【成员】

成员	描述
minScore	最小质量分(默认 0 不过滤)
minSize	最小人脸大小 (万分比[0-10000], 默认 0 为 30 像素)
minPd	最小瞳距 (像素值, 默认 0 不限制)
minClarity	最小清晰度 (默认 0 不过滤)
maxAngle	最大人脸角度 (默认都设为 0 时不过滤)
minEyescore	眼睛遮挡阈值, 低于该阈值将被当作遮挡过滤, 值范围[0,100]
minNoseScore	鼻子遮挡阈值, 低于该阈值将被当作遮挡过滤, 值范围[0,100]
minMouthScore	嘴巴遮挡阈值, 低于该阈值将被当作遮挡过滤, 值范围[0,100]
minOcclusionScore	脸部遮挡阈值, 低于该阈值将被当作遮挡过滤, 值范围[0,100]

4.4.2.3.7 RockIvaCaptureImageConfig

【功能】

抓拍上报图像配置

【声明】

```
typedef struct {  
    uint8_t mode;  
    uint8_t originImageType;  
    uint8_t resizeMode;  
    uint8_t alignWidth;  
    RockIvaRectExpandRatio expand;  
    RockIvaImageInfo imageInfo;  
} RockIvaCaptureImageConfig;
```

【成员】

成员	描述
mode	上报抓拍图像模式, [0: 无; 1: 小图; 2: 大图; 3: 大小图]
originImageType	上报大图格式 [0: 原始图像; 1: JPEG 编码图像] 注: 为了节省内存, 如果需要抓拍上报背景大图, 建议配置为 1, 即抓拍上报 JPEG 编码图像, 同时应用需要注册媒体回调函数 ROCKIVA_Yuv2Jpeg
resizeMode	抓拍图像缩放方式 [0: 保持目标大小, 除非目标大小超过 imageInfo 设定大小; 1: 固定缩放到 imageInfo 设定大小]
alignWidth	抓拍图像的对齐宽度
expand	抓拍图像扩展人脸框上下左右的比例大小配置
imageInfo	抓拍图像信息设置, 宽高需要设置 4 对齐。人脸小图开辟的内存空间与设置的宽、高、图像格式和最大人脸抓拍个数有关

4.4.2.3.8 RockIvaFaceAttribute

【功能】

人脸属性结构体

【声明】

```
typedef struct {  
    RockIvaFaceGenderType gender;  
    RockIvaFaceAgeType age;  
    RockIvaFaceEmotionType emotion;  
    RockIvaFaceGlassesType eyeGlass;  
    RockIvaFaceSmileType smile;  
    RockIvaFaceMaskType mask;  
    RockIvaFaceBeardType beard;  
    uint32_t attractive;  
} RockIvaFaceAttribute;
```

【成员】

成员	描述
gender	性别
age	年龄
emotion	情感
eyeGlass	眼镜
smile	笑容
mask	口罩
beard	胡子
attractive	颜值

4.4.2.3.9 RockIvaFaceQualityInfo

【功能】

人脸质量信息

【声明】

```
typedef struct {  
    uint16_t score;  
    uint16_t clarity;  
    RockIvaAngle angle;  
} RockIvaFaceQualityInfo;
```

【成员】

成员	描述
score	人脸质量分数(值范围 0~100)
clarity	人脸清晰度(值范围 0~100, 100 表示最清晰)
angle	人脸角度

4.4.2.3.10 RockIvaFaceInfo

【功能】

单个目标人脸检测基本信息

【声明】

```
typedef struct {  
    uint32_t objId;  
    uint32_t frameId;  
    RockIvaRectangle faceRect;  
    RockIvaFaceQualityInfo faceQuality;  
    RockIvaFaceState faceState;  
    RockIvaObjectInfo person;  
} RockIvaFaceInfo;
```

【成员】

成员	描述
objId	目标 ID[0,2^32)
frameId	人脸所在帧序号
faceRect	人脸区域原始位置
faceQuality	人脸质量信息
faceState	人脸状态
person	关联的人体检测信息（需要设置 relatedPersonEnable 为 1）

4.4.2.3.11 RockIvaFaceAnalyseInfo

【功能】

单个目标人脸分析结果

【声明】

```
typedef struct {  
    uint32_t featureSize;  
    char feature[ROCKIVA_FACE_FEATURE_SIZE_MAX];  
    RockIvaFaceAttribute faceAttr;  
} RockIvaFaceAnalyseInfo;
```

【成员】

成员	描述
featureSize	特征真实长度，单位字节
feature	半结构化特征信息
faceAttr	人脸属性

4.4.2.3.12 RockIvaFaceDetResult

【功能】

人脸检测处理结果

【声明】

```
typedef struct {  
    uint32_t frameId;  
    uint32_t channelId;  
    uint32_t objNum;  
    RockIvaFaceInfo faceInfo[ROCKIVA_FACE_MAX_FACE_NUM];  
    uint32_t recordNum;  
    RockIvaFaceInfo faceRecord[ROCKIVA_FACE_MAX_FACE_NUM];  
} RockIvaFaceDetResult;
```

【成员】

成员	描述
frameId	帧序号
channelId	通道号
objNum	人脸个数
faceInfo	各目标检测信息
recordNum	人脸记录数
faceRecord	人脸记录信息

4.4.2.3.13 RockIvaFaceCapResults

【功能】

人脸抓拍处理结果

【声明】


```
typedef struct {  
    uint32_t channelId;  
    uint32_t frameId;  
    RockIvaImage frame;  
    RockIvaFaceCapFrameType faceCapFrameType;  
    uint32_t num;  
    RockIvaFaceCapResult faceResults[ROCKIVA_FACE_MAX_FACE_NUM];  
} RockIvaFaceCapResults;
```

【成员】

成员	描述
channelId	通道号
frameId	帧序号
frame	对应的输入图像帧, 如果没有配置缓存大图此时图像缓存数据不能访问
faceCapFrameType	抓拍帧类型
num	人脸结果数量
faceResults	人脸结果

4.4.2.3.14 RockIvaFaceCapResult**【功能】**

人脸抓拍处理结果

【声明】

```
typedef struct {  
    RockIvaFaceInfo faceInfo;  
    RockIvaFaceQualityResultCode qualityResult;  
    RockIvaFaceAnalyseInfo faceAnalyseInfo;  
    RockIvaImage captureImage;  
    RockIvaImage originImage;  
    RockIvaRectangle faceRectOnCaptureImage;  
} RockIvaFaceCapResult;
```

【成员】

成员	描述
faceInfo	人脸基本检测信息
qualityResult	人脸质量结果，如果结果不是 OK 则没有人脸分析信息
faceAnalyseInfo	人脸分析信息
captureImage	人脸抓拍小图
originImage	抓拍帧背景大图
faceRectOnCaptureImage	人脸抓拍小图上的人脸位置

4.4.2.3.15 RockIvaFacePostureResult

【功能】

人脸坐姿处理结果

【声明】

```
typedef struct {  
    uint32_t frameId;  
    uint32_t channelId;  
    RockIvaImage frame;  
    uint32_t objNum;  
    RockIvaHeadInfo headInfo[ROCKIVA_FACE_MAX_FACE_NUM];  
} RockIvaFacePostureResult;
```

【成员】

成员	描述
faceInfo	人脸基本检测信息
qualityResult	人脸质量结果，如果结果不是 OK 则没有人脸分析信息
faceAnalyseInfo	人脸分析信息
captureImage	人脸抓拍小图
originImage	抓拍帧背景大图
faceRectOnCaptureImage	人脸抓拍小图上的人脸位置

4.4.2.3.16 RockIvaHeadInfo

【功能】

单个目标坐姿结果

【声明】

```
typedef struct {  
    uint32_t objId;  
    uint32_t frameId;  
    uint32_t detScore;  
    RockIvaRectangle headRect;  
    RockIvaFaceQualityInfo faceQuality;  
    RockIvaFacePoseState facePoseState;  
    RockIvaObjectInfo face;  
} RockIvaHeadInfo;
```

【成员】

成员	描述
objId	目标跟踪 ID
frameId	帧序号
detScore	目标检测分数 [1-100]
headRect	目标头肩区域原始位置
faceQuality	人脸质量信息
facePoseState	脸坐姿状态
face	关联的人脸检测信息

4.4.2.3.17 RockIvaFaceIdInfo**【功能】**

入库特征对应的详细信息，用户输入

【声明】

```
typedef struct {  
    char faceIdInfo[ROCKIVA_FACE_INFO_SIZE_MAX];  
} RockIvaFaceIdInfo;
```

【成员】

成员	描述
faceIdInfo	人脸标签信息

4.4.2.3.18 RockIvaFaceSearchResult**【功能】**

特征比对返回结果

【声明】

```
typedef struct {  
    char faceIdInfo[ROCKIVA_FACE_INFO_SIZE_MAX];  
    float score;  
} RockIvaFaceSearchResult;
```

【成员】

成员	描述
faceIdInfo	人脸标签信息
score	比对分数

4.4.2.3.19 RockIvaFaceSearchResults**【功能】**

特征比对返回结果列表

【声明】

```
typedef struct {  
    RockIvaFaceSearchResult faceIdScore[ROCKIVA_FACE_ID_MAX_NUM];  
    int num;  
} RockIvaFaceSearchResults;
```

【成员】

成员	描述
faceIdScore	人脸详细信息
num	比对分数

4.4.2.3.20 RockIvaFaceCallback**【功能】**

人脸结果回调函数

【声明】

```
typedef struct {  
    ROCKIVA_FACE_DetResultCallback detCallback;  
    ROCKIVA_FACE_AnalyseResultCallback analyseCallback;  
} RockIvaFaceCallback;
```

【成员】

成员	描述
detCallback	人脸检测回调函数指针
analyseCallback	人脸抓拍分析回调函数指针

4.5 车辆车牌模块

4.5.1 功能描述

车辆车牌检测识别功能可以检测图像中车辆位置、车辆属性、车牌号码和车牌属性，其通过 ROCKIVA_PLATE_Init 初始化车辆车牌相关规则配置并配置车辆车牌结果回调函数。

车辆车牌检测识别模块代码调用流程如图 4-5-1 所示。

车辆车辆属性检测识别结果如图 4-5-2 所示。

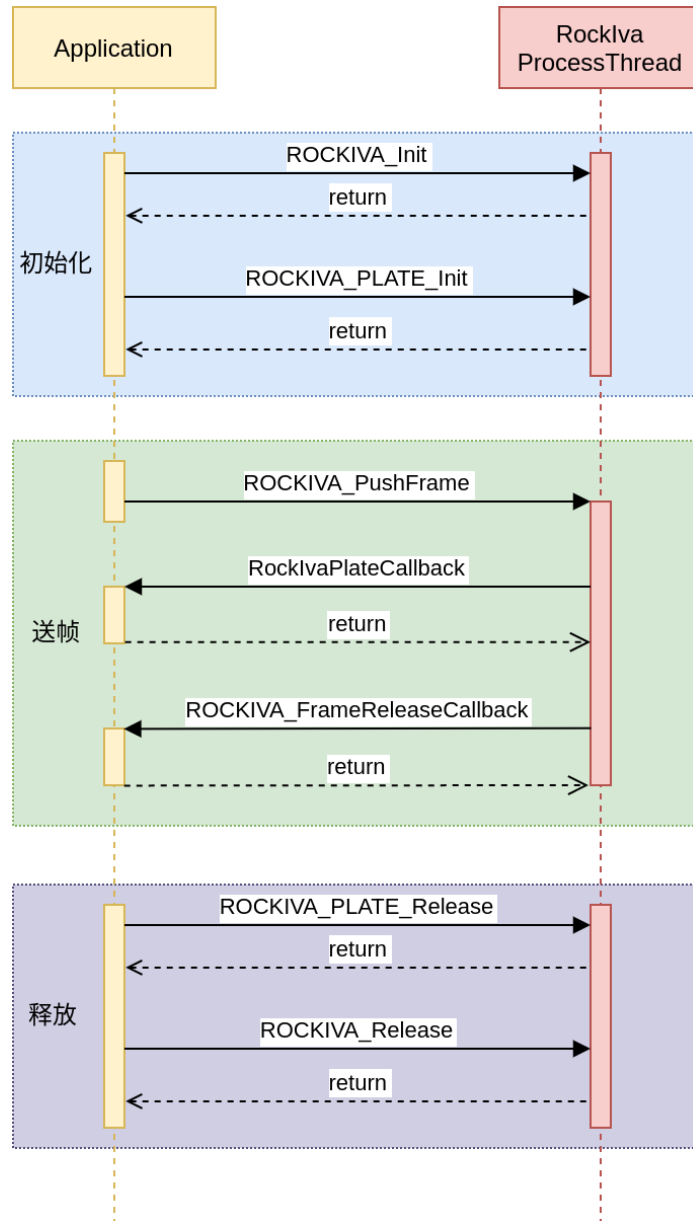


图 4-5-1 车辆车牌识别功能调用流程

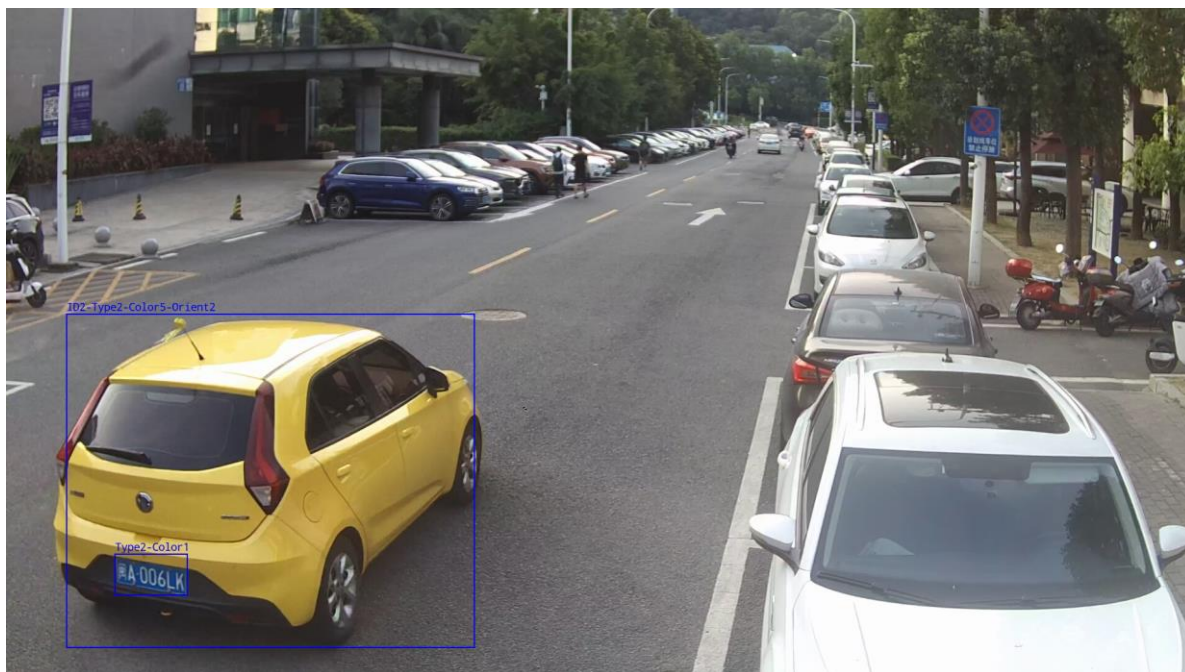


plate number=闽A006LK scores=(99,99,99,99,99,99,99) len=7 type=2 color=1

图 4-5-2 车辆车牌检测识别结果

车辆车牌识别参考代码如下：

```
RockIvaRetCode ret;
RockIvaHandle handle;

// 释放回调函数
void FrameReleaseCallback(const RockIvaReleaseFrames* releaseFrames, void* userdata)
{
    // 处理释放
}

// 车牌识别模块回调函数
void PlateResultCallback(const RockIvaPlateResult* result, const RockIvaExecuteStatus status, void*
userdata)
{
    // 处理结果
}

// 初始化 IVA 实例
RockIvaInitParam commonParams;
memset(&commonParams, 0, sizeof(RockIvaInitParam));
commonParams.detModel = ROCKIVA_DET_MODEL_CLS7; // 设置检测模型，车牌识别必须为 CLS7 或 CLS8
ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);
// 设置回调函数
ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);

// 初始化车牌识别模块
```

```
RockIvaPlateTaskParam plateParams;
memset(plateParams, 0, sizeof(RockIvaPlateTaskParam));

// 机动车车身最小宽度（这里设置为 1920 下 200 像素宽度）
plateParams->vehicleMinSize = ROCKIVA_PIXEL_RATIO_CONVERT(1920, 200);
// 车牌最小像素宽度（这里设置为 1920 下 80 像素宽度）
plateParams->plateMinSize = ROCKIVA_PIXEL_RATIO_CONVERT(1920, 70);
// 车牌识别字符最小分数
plateParams->plateMinScore = 95;
// 设置识别模式（0: 单帧模式; 1: 跟踪模式,内部会记录车辆 id 的识别,已有识别结果的车辆不会再返回结果）
plateParams->mode = 1;

ret = ROCKIVA_PLATE_Init(handle, & plateParams, PlateResultCallback);

// 处理图像帧（同通用模块示例代码）

// 等待所有帧都处理完成
ROCKIVA_WaitFinish(handle, -1, 5000);
// 销毁车牌识别模块
ROCKIVA_PLATE_Release(handle)
// 销毁 IVA 实例
ROCKIVA_Release(handle);
```

4.5.2 API 参考

4.5.2.1 函数接口

接口	描述
RockIvaPlateCallback	车辆车牌检测识别结果回调函数
ROCKIVA_PLATE_Init	车牌功能初始化
ROCKIVA_PLATE_Reset	运行时重新配置(重新配置会导致内部的一些记录清空复位，但是模型不会重新初始化)
ROCKIVA_PLATE_Release	车牌功能销毁

4.5.2.1.1 RockIvaPlateCallback

【功能】

车辆车牌结果回调函数

【声明】

```
typedef void (*RockIvaPlateCallback)(
    const RockIvaPlateResult* result,
```



```
const RockIvaExecuteStatus status,  
void* userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输出	车辆车牌结果
status	输出	状态码
userdata	输出	用户自定义数据

【返回值】

RockIvaRetCode

4.5.2.1.2 ROCKIVA_PLATE_Init

【功能】

车辆车牌检测识别功能初始化

【声明】

```
RockIvaRetCode ROCKIVA_PLATE_Init(RockIvaHandle handle,  
const RockIvaPlateTaskParam* initParams,  
const RockIvaPlateCallback callback);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入输出	需要初始化的 handle
initParams	输入	初始化参数
callback	输入	回调函数

【返回值】

RockIvaRetCode

4.5.2.1.3 ROCKIVA_PLATE_Reset

【功能】

重置

【声明】

```
RockIvaRetCode ROCKIVA_PLATE_Reset(RockIvaHandle handle, const RockIvaPlateTaskParams*  
initParams);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要重置的 handle
initParams	输入	新的初始化参数配置

【返回值】

RockIvaRetCode

4.5.2.1.4 ROCKIVA_PLATE_Release

【功能】

销毁

【声明】

```
RockIvaRetCode ROCKIVA_PLATE_Release(RockIvaHandle handle);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要释放的句柄

【返回值】

RockIvaRetCode

4.5.2.2枚举定义

枚举	描述
RockIvaPlateType	车牌类型
RockIvaPlateColor	车牌颜色
RockIvaVehicleType	机动车类型
RockIvaVehicleColor	车身颜色
RockIvaVehicleOrient	机动车朝向

4.5.2.2.1 RockIvaPlateType

【功能】

车牌类型

【声明】

```
typedef enum {  
    PLATE_TYPE_NONE = 0,  
    PLATE_TYPE_LARGE_CAR,  
    PLATE_TYPE_SMALL_CAR,  
    PLATE_TYPE_EMBASSY_CAR,  
    PLATE_TYPE_CONSULATE_CAR,  
    PLATE_TYPE_TRAILER,
```

```
PLATE_TYPE_COACH_CAR,  
PLATE_TYPE_POLICE_CAR,  
PLATE_TYPE_HONGKONG,  
PLATE_TYPE_MACAO,  
PLATE_TYPE_ARMED_POLICE,  
PLATE_TYPE_PLA,  
PLATE_TYPE_NEW_ENGERY,  
PLATE_TYUPE_OTHER,  
} RockIvaPlateType;
```

【成员】

成员	描述
PLATE_TYPE_NONE	无车牌
PLATE_TYPE_LARGE_CAR	大型汽车号牌
PLATE_TYPE_SMALL_CAR	小型汽车号牌
PLATE_TYPE_EMBASSY_CAR	使馆汽车号牌
PLATE_TYPE_CONSULATE_CAR	领馆汽车号牌
PLATE_TYPE_TRAILER	挂车号牌
PLATE_TYPE_COACH_CAR	教练车号牌
PLATE_TYPE_POLICE_CAR	警车号牌
PLATE_TYPE_HONGKONG	香港出入境号牌
PLATE_TYPE_MACAO	澳门出入境号牌
PLATE_TYPE_ARMED_POLICE	武警号牌
PLATE_TYPE_PLA	军队号牌
PLATE_TYPE_NEW_ENGERY	新能源号牌
PLATE_TYUPE_OTHER	其他号牌

4.5.2.2.2 RockIvaPlateColor

【功能】

车牌颜色

【声明】

```
typedef enum {  
    PLATE_COLOR_UNKNOWN = 0,  
    PLATE_COLOR_BLUE,  
    PLATE_COLOR_YELLOW,  
    PLATE_COLOR_GREEN,  
    PLATE_COLOR_BLACK,  
    PLATE_COLOR_WHITE,  
} RockIvaPlateColor;
```

【成员】

成员	描述
PLATE_COLOR_UNKNOWN	车牌颜色未知
PLATE_COLOR_BLUE	蓝牌
PLATE_COLOR_YELLOW	黄牌
PLATE_COLOR_GREEN	绿牌
PLATE_COLOR_BLACK	黑牌
PLATE_COLOR_WHITE	白牌

4.5.2.2.3 RockIvaVehicleType

【功能】

机动车类型

【声明】

```
typedef enum {
    VEHICLE_TYPE_UNKNOWN = 0,
    VEHICLE_TYPE_BUS,
    VEHICLE_TYPE_SEDAN,
    VEHICLE_TYPE_VAN,
    VEHICLE_TYPE_SUV,
    VEHICLE_TYPE_PICKUP,
```

```
VEHICLE_TYPE_TRUCK,  
} RockIvaVehicleType;
```

【成员】

成员	描述
VEHICLE_TYPE_UNKNOWN	未知
VEHICLE_TYPE_BUS	客车（大客车、中客车、公交车）
VEHICLE_TYPE_SEDAN	轿车（小轿车、掀背车、微型车、跑车）
VEHICLE_TYPE_VAN	面包车（面包车、MPV）
VEHICLE_TYPE_SUV	SUV 运动型多用途汽车
VEHICLE_TYPE_PICKUP	皮卡车
VEHICLE_TYPE_TRUCK	货车（大货车、中货车、小货车、厢式货车）

4.5.2.2.4 RockIvaVehicleColor

【功能】

车身颜色

【声明】

```
typedef enum {  
    VEHICLE_COLOR_UNKNOWN = 0,  
    VEHICLE_COLOR_BLACK,  
    VEHICLE_COLOR_BLUE,  
    VEHICLE_COLOR_BROWN,  
    VEHICLE_COLOR_GREY,  
    VEHICLE_COLOR_YELLOW,
```

```
VEHICLE_COLOR_GREEN,
VEHICLE_COLOR_PURPLE,
VEHICLE_COLOR_RED,
VEHICLE_COLOR_WHITE,
} RockIvaVehicleColor;
```

【成员】

成员	描述
VEHICLE_COLOR_UNKNOWN	未知
VEHICLE_COLOR_BLACK	黑色
VEHICLE_COLOR_BLUE	蓝色（蓝、青蓝）
VEHICLE_COLOR_BROWN	棕色
VEHICLE_COLOR_GREY	灰色（灰、银、深灰）
VEHICLE_COLOR_YELLOW	黄色（黄、橘、金）
VEHICLE_COLOR_GREEN	绿色
VEHICLE_COLOR_PURPLE	紫色
VEHICLE_COLOR_RED	红色
VEHICLE_COLOR_WHITE	白色

4.5.2.2.5 RockIvaVehicleOrient

【功能】

机动车朝向

【声明】

```
typedef enum {
    VEHICLE_ORIENT_UNKNOWN = 0,
    VEHICLE_ORIENT_FRONT,
    VEHICLE_ORIENT_BACK,
    VEHICLE_ORIENT_SIDE,
} RockIvaVehicleOrient;
```

【成员】

成员	描述
VEHICLE_ORIENT_UNKNOWN	未知
VEHICLE_ORIENT_FRONT	正面
VEHICLE_ORIENT_BACK	背面
VEHICLE_ORIENT_SIDE	侧面

4.5.2.3 结构体定义

结构体	描述
RockIvaPlateTaskParam	车牌识别业务初始化参数配置
RockIvaVehicleAttribute	机动车属性
RockIvaPlateInfo	车牌识别信息
ROCKIVAPlateResult	车牌识别处理结果

4.5.2.3.1 RockIvaPlateTaskParam

【功能】

车牌识别业务初始化参数配置

【声明】

```
typedef struct {  
    uint16_t vehicleMinSize;  
    uint16_t plateMinSize;  
    uint16_t plateMinScore;  
    uint8_t mode;  
    RockIvaAreas detectAreas;  
} RockIvaPlateTaskParam;
```

【成员】

成员	描述
vehicleMinSize	机动车车身最小宽度（万分比[0-10000]）
plateMinSize	车牌最小宽度（万分比[0-10000]）
plateMinScore	车牌识别字符最小分数(0-99)
mode	运行模式（0：单帧模式；1：跟踪模式,内部会记录车辆id的识别,已有识别结果的车辆不会再返回结果）
detectAreas	检测区域（对每个检测区域内最大的车辆进行识别）

4.5.2.3.2 RockIvaVehicleAttribute

【功能】

机动车属性

【声明】

```
typedef struct {  
    RockIvaVehicleType type;
```



```
RockIvaVehicleColor color;  
RockIvaVehicleOrient orient;  
} RockIvaVehicleAttribute;
```

【成员】

成员	描述
type	机动车类型
color	机动车车身颜色
orient	机动车朝向

4.5.2.3.3 RockIvaPlateInfo

【功能】

车牌识别信息

【声明】

```
typedef struct {  
    uint32_t vehicleId;  
    int plateCode[ROCKIVA_PLATE_MAX_CHAR_NUM];  
    uint16_t plateScore[ROCKIVA_PLATE_MAX_CHAR_NUM];  
    int plateLen;  
    RockIvaRectangle plateRect;  
    RockIvaPlateType plateType;  
    RockIvaPlateColor plateColor;  
    RockIvaRectangle vehicleRect;  
    RockIvaVehicleAttribute vehicleAttr;  
} RockIvaPlateInfo;
```

【成员】

成员	描述
vehicleId	车牌对应的机动车跟踪 Id
plateCode	车牌字符，车牌最大长度为 20
plateScore	车牌字符分数[0-100]，车牌最大长度为 20
plateLen	车牌字符长度
plateRect	车牌坐标
plateType	车牌类型
plateColor	车牌颜色
vehicleRect	机动车坐标
vehicleAttr	机动车属性

4.5.2.3.4 RockIvaPlateResult

【功能】

车牌识别处理结果

【声明】

```
typedef struct {  
    uint32_t frameId;  
    uint32_t channelId;  
    RockIvaImage frame;  
    uint32_t objNum;  
    RockIvaPlateInfo plateInfo[ROCKIVA_PLATE_MAX_NUM];  
} RockIvaPlateResult;
```

【成员】

成员	描述
frameId	帧 ID
channelId	通道号
frame	对应的输入图像帧
objNum	车牌个数
plateInfo	各车牌识别结果，车牌最大数目为 10

4.6 客流统计模块

4.6.1 功能描述

客流统计模块支持越界人流统计和区域人数统计两种类型。客流统计模块对应接口头文件位于 rockiva_ts_api.h。

客流统计参考示例代码如下：

```
// 释放回调函数（同通用模块示例代码）
...

// 客流模块回调函数
void TsResultCallback(const RockIvaTsResult* result, const RockIvaExecuteStatus status, void* userdata)
{
    // 处理结果
}

// 初始化 IVA 实例
RockIvaInitParam commonParams;
memset(&commonParams, 0, sizeof(RockIvaInitParam));
commonParams.detModel = ROCKIVA_DET_MODEL_PHCP; // 设置检测模型，带头肩类别
ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);
// 设置回调函数
ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);

// 初始化客流模块
RockIvaTsTaskParams tsParams;
memset(&tsParams, 0, sizeof(RockIvaTsTaskParams));
tsParams.lines[0].ruleEnable = 1;
tsParams.lines[0].ruleID = 1;
tsParams.lines[0].objType = ROCKIVA_OBJECT_TYPE_HEAD;
tsParams.lines[0].line.head.x = ROCKIVA_PIXEL_RATION_CONVERT(1920, 100);
tsParams.lines[0].line.head.y = ROCKIVA_PIXEL_RATION_CONVERT(1080, 500);
tsParams.lines[0].line.tail.x = ROCKIVA_PIXEL_RATION_CONVERT(1920, 1800);
tsParams.lines[0].line.tail.y = ROCKIVA_PIXEL_RATION_CONVERT(1080, 500);
tsParams.lines[0].direct = ROCKIVA_LINE_DIRECT_CW;

tsParams.areas[0].ruleEnable = 1;
tsParams.areas[0].ruleID = 2;
tsParams.areas[0].area.pointNum = 4;
tsParams.areas[0].objType = ROCKIVA_OBJECT_TYPE_HEAD;
tsParams.areas[0].area.points[0].x = 1000;
tsParams.areas[0].area.points[0].y = 1000;
tsParams.areas[0].area.points[1].x = 9000;
tsParams.areas[0].area.points[1].y = 1000;
tsParams.areas[0].area.points[2].x = 9000;
tsParams.areas[0].area.points[2].y = 9000;
tsParams.areas[0].area.points[3].x = 1000;
tsParams.areas[0].area.points[3].y = 9000;
ret = ROCKIVA_TS_Init(ctx->handle, &tsParams, TsResultCallback);

// 送帧处理（同通用模块示例代码）
...

// 等待所有帧都处理完成
ROCKIVA_WaitFinish(handle, -1, 5000);
// 销毁客流模块
ROCKIVA_TS_Release(handle)
```

```
// 销毁 IVA 实例  
ROCKIVA_Release(handle);
```

4.6.2 API 参考

4.6.2.1 函数接口

接口	描述
ROCKIVA_TS_ResultCallback	客流统计结果回调函数
ROCKIVA_TS_Init	初始化客流统计模块
ROCKIVA_TS_Release	销毁客流统计模块
ROCKIVA_TS_Reset	复位（重设规则，并清除统计信息）
ROCKIVA_TS_ResetCount	复位规则计数

4.6.2.1.1 ROCKIVA_TS_ResultCallback

【功能】

结果回调函数

【声明】

```
typedef void (*ROCKIVA_TS_ResultCallback)(  
    const RockIvaTsResult* result,  
    const RockIvaExecuteStatus status,  
    void* userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输出	客流统计结果
status	输出	状态码
userdata	输出	用户自定义数据

【返回值】

RockIvaRetCode

4.6.2.1.2 ROCKIVA_TS_Init

【功能】

客流统计功能初始化

【声明】

```
RockIvaRetCode ROCKIVA_TS_Init(RockIvaHandle handle,  
                                const RockIvaTsTaskParams* initParams,  
                                const ROCKIVA_TS_ResultCallback resultCallback);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	需要初始化的 handle
initParams	输入	初始化参数
resultCallback	输入	回调函数

【返回值】

RockIvaRetCode

4.6.2.1.3 ROCKIVA_TS_Release

【功能】

销毁

【声明】

```
RockIvaRetCode ROCKIVA_TS_Release(RockIvaHandle handle);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要释放的句柄

【返回值】

RockIvaRetCode

4.6.2.1.4 ROCKIVA_TS_Reset

【功能】

复位（重设规则，并清除统计信息）

【声明】

```
RockIvaRetCode ROCKIVA_TS_Reset(RockIvaHandle handle, const RockIvaTsTaskParams* params);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要释放的句柄

【返回值】

RockIvaRetCode

4.6.2.1.5 ROCKIVA_TS_ResetCount

【功能】

复位规则计数

【声明】

```
RockIvaRetCode ROCKIVA_TS_ResetCount(RockIvaHandle handle, int rule_id);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要释放的句柄

【返回值】

RockIvaRetCode

4.6.2.2枚举定义

枚举	描述
RockIvaLineDirect	越界方向

4.6.2.2.1 RockIvaLineDirect

【功能】

越界方向

【声明】

```
typedef enum {  
    ROCKIVA_LINE_DIRECT_CW = 0,  
    ROCKIVA_LINE_DIRECT_CCW = 1,  
} RockIvaLineDirect;
```

【成员】

成员	描述
ROCKIVA_LINE_DIRECT_CW	顺时针方向，假设越界线方向从下往上，越界方向为从左往右
ROCKIVA_LINE_DIRECT_CCW	逆时针方向，假设越界线方向从下往上，越界方向为从右往左

4.6.2.3 结构体定义

结构体	描述
RockIvaTsLineRule	越界人流量统计配置
RockIvaTsAreaRule	区域人流量统计配置
RockIvaTsTaskParams	人流统计初始化参数配置
RockIvaLineTrafficStatic	越界人数统计结果
RockIvaAreaTrafficStatic	区域人数统计结果
RockIvaTrafficResult	人数统计结果
RockIvaTsResult	人流统计结果全部信息

4.6.2.3.1 RockIvaTsLineRule

【功能】

越界人流配置

【声明】

```
typedef struct  
{  
    uint8_t ruleEnable;  
    uint32_t ruleID;  
    RockIvaLine line;  
    RockIvaLineDirect direct;  
    RockIvaSize minObjSize;  
    RockIvaSize maxObjSize;  
    uint8_t sense;  
    RockIvaObjectType objType;  
} RockIvaTsLineRule;
```

【成员】

成员	描述
ruleEnable	规则是否启用，1->启用，0->不启用
ruleID	规则 ID（不能重复）
line	越界线配置
direct	越界方向
minObjSize	万百分比表示 最小目标
maxObjSize	万百分比表示 最大目标
sense	灵敏度,1~100
objType	目标类型

4.6.2.3.2 RockIvaTsAreaRule

【功能】

区域人流配置

【声明】

```
typedef struct
{
    uint8_t ruleEnable;
    uint32_t ruleID;
    RockIvaArea area;
    RockIvaSize minObjSize;
    RockIvaSize maxObjSize;
    uint8_t sense;
    RockIvaObjectType objType;
} RockIvaTsAreaRule;
```

【成员】

成员	描述
ruleEnable	规则是否启用，1->启用，0->不启用
ruleID	规则 ID（不能重复）
area	区域配置
minObjSize	万百分比表示 最小目标
maxObjSize	万百分比表示 最大目标
sense	灵敏度,1~100
objType	目标类型

4.6.2.3.3 RockIvaTsTaskParams

【功能】

人流统计初始化参数配置

【声明】


```
typedef struct
{
    RockIvaTsLineRule lines[ROCKIVA_TS_MAX_RULE_NUM];
    RockIvaTsAreaRule areas[ROCKIVA_TS_MAX_RULE_NUM];
} RockIvaTsTaskParams;
```

【成员】

成员	描述
lines	区域人数统计
areas	拌线人数统计

4.6.2.3.4 RockIvaLineTrafficStatic**【功能】**

越界人数统计结果

【声明】

```
typedef struct
{
    uint32_t count;
} RockIvaLineTrafficStatic;
```

【成员】

成员	描述
count	拌线人数

4.6.2.3.5 RockIvaAreaTrafficStatic**【功能】**

区域人数统计

【声明】

```
typedef struct
{
    uint32_t count;
    uint32_t in;
    uint32_t out;
} RockIvaAreaTrafficStatic;
```

【成员】

成员	描述
count	区域内人数
in	进区域人数
out	出区域人数

4.6.2.3.6 RockIvaTrafficResult

【功能】

人数统计结果

【声明】

```
typedef struct
{
    RockIvaLineTrafficStatic lines[ROCKIVA_TS_MAX_RULE_NUM];
    RockIvaAreaTrafficStatic areas[ROCKIVA_TS_MAX_RULE_NUM];
} RockIvaTrafficResult;
```

【成员】

成员	描述
lines	绊线人数统计结果
areas	区域人数统计结果

4.6.2.3.7 RockIvaTsResult

【功能】

人流统计结果全部信息

【声明】

```
typedef struct
{
    uint32_t channelId;
    uint32_t frameId;
    RockIvaImage frame;
    RockIvaTrafficResult traffic;
    uint32_t objNum;
    RockIvaObjectInfo objInfo[ROCKIVA_MAX_OBJ_NUM];
} RockIvaTsResult;
```

【成员】

成员	描述
channelId	通道号
frameId	输入图像帧 ID
frame	对应的输入图像帧
traffic	人数统计结果
objNum	目标个数
objInfo	各目标检测信息

4.7 物体检测模块

4.7.1 功能描述

物体检测功能目前支持检测非机动车和火焰（暂未开放），通过 ROCKIVA_OBJECT_Init 初始化物体（非机动车、火焰）相关检测规则配置并配置物体检测结果回调函数。使用该功能时初始化 ROCKIVA_Init 无需指定 detObjectType。并针对平视角（马路、走廊等）和俯视角（电梯等）两类场景做了适配优化。

物体检测模块代码调用流程如图 4-6-1 所示。

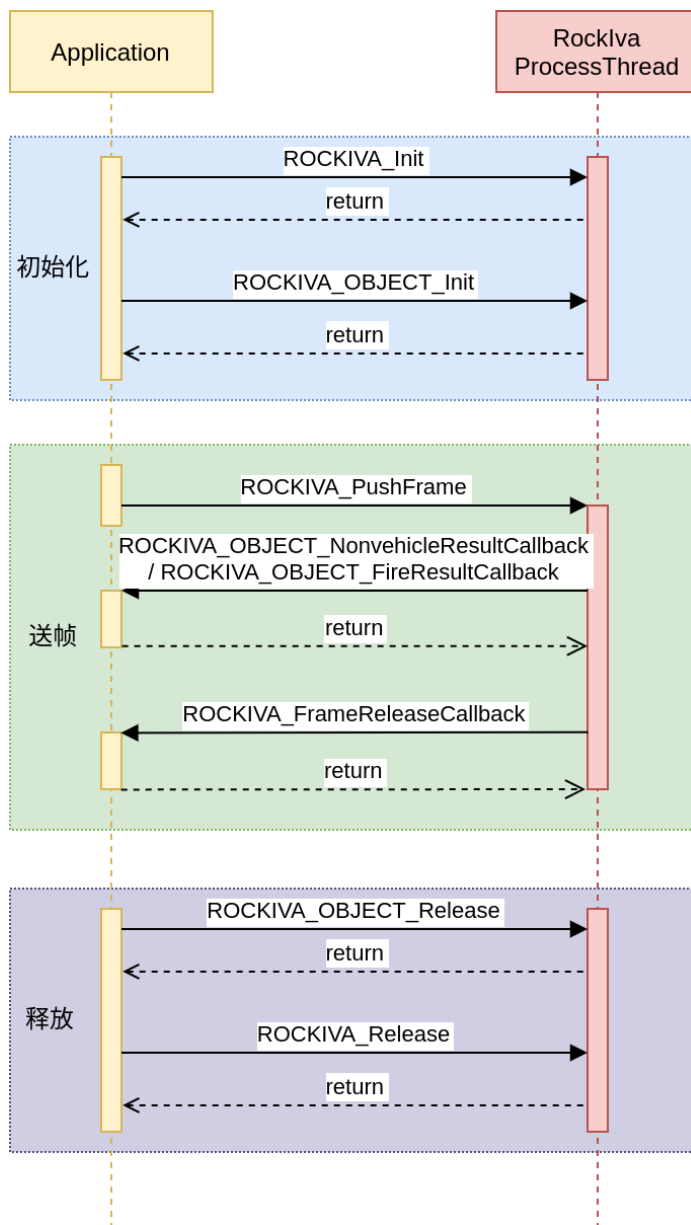


图 4-6-1 物体识别功能调用流程

非机动车检测结果如下图 4-6-2 和图 4-6-3 所示，objID+非机动车类型（1：电瓶车、2：自动车）。

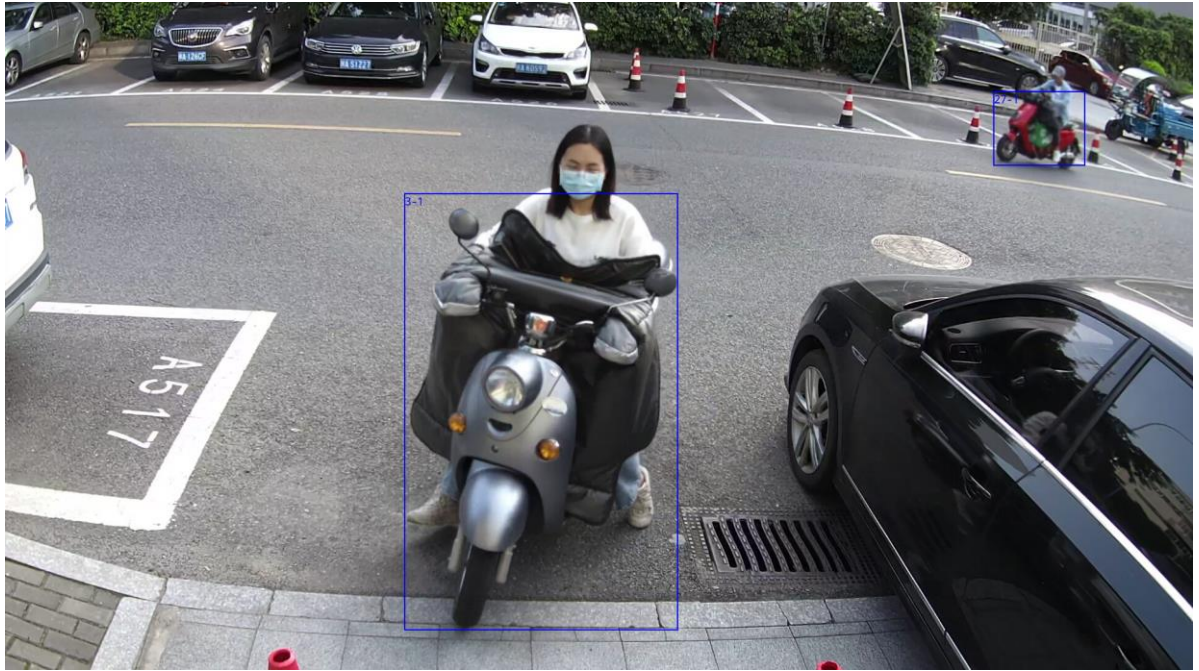


图 4-6-2 电动车检测



图 4-6-3 自行车检测

```
// 释放回调函数（同通用模块示例代码）
...

// 物体检测回调函数
void NonvehicleResultCallback(const RockIvaObjectNonvehicleAttrResult* result, const
RockIvaExecuteStatus status, void* userdata)
{
    // 处理非机动车结果
```

```
}  
void FireResultCallback(const RockIvaObjectFireResult* result, const RockIvaExecuteStatus status,  
                        void* userdata)  
{  
    // 处理火焰检测结果  
}  
// 初始化 IVA 实例  
RockIvaInitParam commonParams;  
memset(&commonParams, 0, sizeof(RockIvaInitParam));  
commonParams.detModel = ROCKIVA_DET_MODEL_CLS7; // 设置检测模型  
ret = ROCKIVA_Init(&handle, ROCKIVA_MODE_VIDEO, &commonParams, NULL);  
// 设置回调函数  
ROCKIVA_SetFrameReleaseCallback(handle, framerelease_callback);  
  
// 初始化物体检测模块  
RockIvaObjectTaskParams objectParams;  
memset(&objectParams, 0, sizeof(RockIvaObjectTaskParams));  
// 使能非机动车检测  
objectParams->nonvehicleRule.enable = 1;  
objectParams->nonvehicleRule.detScene = 0; // 0: 平视角, 1: 俯视角  
// 使能火焰检测  
objectParams->fireRule.enable = 1;  
RockIvaObjectResultCallback callback;  
callback.nonvehicleCallback = NonvehicleResultCallback;  
callback.fireCallback = FireResultCallback;  
ret = ROCKIVA_OBJECT_Init(handle, &objectParams, callback);  
  
// 送帧处理（同通用模块示例代码）  
...  
  
// 等待所有帧都处理完成  
ROCKIVA_WaitFinish(handle, -1, 5000);  
// 销毁模块  
ROCKIVA_OBJECT_Release(handle)  
// 销毁 IVA 实例  
ROCKIVA_Release(handle);
```

4.7.2 API 参考

4.7.2.1 函数接口

接口	描述
ROCKIVA_OBJECT_NonvehicleResultCallback	非机动车检测结果回调函数
ROCKIVA_OBJECT_FireResultCallback	火焰检测结果回调函数
ROCKIVA_OBJECT_Init	初始化

ROCKIVA_OBJECT_Release	释放
ROCKIVA_OBJECT_Reset	运行时重新配置

4.7.2.1.1 ROCKIVA_OBJECT_NonvehicleResultCallback

【功能】

非机动车检测结果回调函数

【声明】

```
typedef void (*ROCKIVA_OBJECT_NonvehicleResultCallback)(
    const RockIvaObjectNonvehicleAttrResult* result,
    const RockIvaExecuteStatus status,
    void* userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输出	非机动车检测结果
status	输出	状态码
userdata	输出	用户自定义数据

【返回值】

RockIvaRetCode

4.7.2.1.2 ROCKIVA_OBJECT_FireResultCallback

【功能】

火焰检测结果回调函数

【声明】

```
typedef void (*ROCKIVA_OBJECT_FireResultCallback)(
    const RockIvaObjectFireResult* result,
    const RockIvaExecuteStatus status,
    void* userdata);
```

【输入参数】

参数名称	输入/输出	描述
result	输出	火焰检测结果
status	输出	状态码
userdata	输出	用户自定义数据

【返回值】

RockIvaRetCode

4.7.2.1.3 ROCKIVA_OBJECT_Init**【功能】**

物体识别功能初始化

【声明】

```
RockIvaRetCode ROCKIVA_OBJECT_Init(RockIvaHandle handle,  
                                     const RockIvaObjectTaskParams* initParams,  
                                     const RockIvaObjectResultCallback callback);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入输出	需要初始化的 handle
initParams	输入	初始化参数
callback	输入	回调函数

【返回值】

RockIvaRetCode

4.7.2.1.4 ROCKIVA_OBJECT_Release**【功能】**

销毁

【声明】

```
RockIvaRetCode ROCKIVA_OBJECT_Release(RockIvaHandle handle);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	要释放的句柄

【返回值】

RockIvaRetCode

4.7.2.1.5 ROCKIVA_OBJECT_Reset

【功能】

运行时重新配置(重新配置会导致内部的一些记录清空复位，但是模型不会重新初始化)

【声明】

```
RockIvaRetCode ROCKIVA_OBJECT_Reset(RockIvaHandle handle, const RockIvaObjectTaskParams*  
initParams);
```

【输入参数】

参数名称	输入/输出	描述
handle	输入	句柄
initParams	输入	配置参数

【返回值】

RockIvaRetCode

4.7.2.2枚举定义

枚举	描述
暂无	

4.7.2.3结构体定义

结构体	描述
RockIvaNonvehicleRule	非机动车检测规则设置
RockIvaFireRule	火焰检测规则设置
RockIvaObjectTaskParams	视频结构化初始化参数配置
RockIvaNonvehicleAttribute	非机动车属性
RockIvaObjectNonvehicleInfo	单个目标非机动车属性检测基本信息
RockIvaObjectNonvehicleAttrResult	非机动车检测结果全部信息
RockIvaObjectFireResult	火焰检测结果全部信息

4.7.2.3.1 RockIvaNonvehicleRule

【功能】

非机动车检测规则设置

【声明】

```
typedef struct
{
    uint8_t enable;
    uint8_t detScene;
    uint8_t sensitivity;
    uint32_t minSize;
    uint32_t filterType;
    uint32_t intervalTime;
    RockIvaAreas roiAreas;
} RockIvaNonvehicleRule;
```

【成员】

成员	描述
enable	使能 1：开启；0：关闭
detScene	检测场景 默认 0：平视角电瓶车检测；1：俯视角（电梯内）电瓶车检测（暂未开放）
sensitivity	报警灵敏度[1,100]（暂未开放）
minSize	最小检测非机动车属性尺寸，小于该值过滤
filterType	过滤类型，例如： ROCKIVA_OBJECT_TYPE_BITMASK(ROCKIVA_OBJECT_TYPE_MOTORCYCLE)
intervalTime	运行时间间隔（单位毫秒）
roiAreas	有效区域

4.7.2.3.2 RockIvaFireRule

【功能】

火焰检测规则设置

【声明】

```
typedef struct
{
    uint8_t enable;
    uint8_t sensitivity;
    uint32_t minSize;
```

```
uint32_t intervalTime;  
} RockIvaFireRule;
```

【成员】

成员	描述
enable	使能 1：开启；0：关闭
sensitivity	报警灵敏度[1,100]（暂未开放）
minSize	最小火焰尺寸，小于该值过滤
intervalTime	运行时间间隔（单位毫秒）

4.7.2.3.3 RockIvaObjectTaskParams

【功能】

视频结构化初始化参数配置

【声明】

```
typedef struct  
{  
    RockIvaNonvehicleRule nonvehicleRule;  
    RockIvaFireRule fireRule;  
} RockIvaObjectTaskParams;
```

【成员】

成员	描述
nonvehicleRule	非机动车检测规则
fireRule	火焰检测规则

4.7.2.3.4 RockIvaNonvehicleAttribute

【功能】

非机动车属性

【声明】

```
typedef struct  
{  
    RockIvaNonvehicleType type;  
} RockIvaNonvehicleAttribute;
```

【成员】

成员	描述
type	非机动车类别

4.7.2.3.5 RockIvaObjectNonvehicleInfo

【功能】

单个目标非机动车属性检测基本信息

【声明】

```
typedef struct
{
    uint32_t objId;
    uint32_t frameId;
    RockIvaRectangle objectRect;
    RockIvaNonvehicleAttribute attr;
    bool alert;
    uint8_t firstTrigger;
} RockIvaObjectNonvehicleInfo;
```

【成员】

成员	描述
objId	目标 ID[0,2^32)
frameId	非机动车所在帧序号
objectRect	非机动车区域原始位置
attr	非机动车属性信息
alert	1: 电瓶车告警
firstTrigger	1: 第一次触发电瓶车告警

4.7.2.3.6 RockIvaObjectNonvehicleAttrResult

【功能】

非机动车检测结果全部信息

【声明】

```
typedef struct
{
    uint32_t frameId;
    uint32_t channelId;
    RockIvaImage frame;
    uint32_t objNum
    RockIvaObjectNonvehicleInfo objectInfo[ROCKIVA_MAX_OBJ_NUM];
}
```

```
} RockIvaObjectNonvehicleAttrResult;
```

【成员】

成员	描述
frameId	输入图像帧 ID
channelId	通道号
frame	对应的输入图像帧
objNum	目标个数
objectInfo[ROCKIVA_MAX_OBJ_NUM]	非机动检测信息，最大检测信息个数为 128

4.7.2.3.7 RockIvaObjectFireResult**【功能】**

火焰检测结果全部信息

【声明】

```
typedef struct
{
    uint32_t frameId;
    uint32_t channelId;
    uint32_t objNum;
    RockIvaObjectInfo objectInfo[ROCKIVA_MAX_OBJ_NUM];
} RockIvaObjectFireResult;
```

【成员】

成员	描述
frameId	输入图像帧 ID
channelId	通道号
objNum	目标个数
objectInfo[ROCKIVA_MAX_OBJ_NUM]	火焰检测信息，最大检测信息个数为 128

4.7.2.3.8 RockIvaObjectResultCallback**【功能】**

回调函数配置

【声明】

```
typedef struct
{
    ROCKIVA_OBJECT_NonvehicleResultCallback nonvehicleCallback;
```

```
ROCKIVA_OBJECT_FireResultCallback fireCallback;  
} RockIvaObjectResultCallback;
```

【成员】

成员	描述
nonvehicleCallback	电瓶车检测结果回调函数
fireCallback	火焰检测结果回调函数