Classification status: Top secret ( ) Secret ( ) Internal ( ) Public (ÿ)

Intelligent Video Analysis (ROCKIVA) SDK Development Guide

(Technology Department, Graphics Computing Platform Center)

File Status:	Current version: V1.4.2	
[ ] Editing	author:	HPC&AI Team
[ÿ] Officially released	Completion date: 2022-09-06	
	Audit:	YHC
	Completion date: 2022-09-06	

Rockchip Electronics Co., Ltd.

Rockchips Semiconductor Co . , Ltd

Update Record

Version	Modified by	Modified Date	Modification Notes	Approved by
V1.0	YHC	2021-12-30 Initial version		XW/ZHT
V1.1	YHC	2022-01-13	1. The face module supports obtaining associated human shape results 2. The face module results support obtaining face quality information breath	ZHT
V1.2	XDC	2022-04-06 1. Modify some API, structure names and parameters YHC		
V1.3	XDC	2022-06-06	1. Modify the name of the face capture API and structure. parameter	YHC
V1.4.0	XDC	2022-07-18	1. Add vehicle license plate recognition module 2. Add non-motor vehicle detection module 3. Modify some API, structure names and parameters number	YHC
V1.4.2	XDC	2022-09-06	1. The face module supports early reporting and configuration of masks Face capture image non-zoom mode 2. Added battery vehicle alarm to the elevator control part of the object detection module Warning sign 3. General module detection categories increased to include battery vehicles and bicycles Cars and license plates, remove non-motorized vehicles from object detection module Car enumeration type	YHC

目录

Table of contents

1 Overview

ROCKIVA SDK is an intelligent video analysis (IVA) algorithm SDK that can be embedded

AI intelligent algorithm support is provided in product applications such as NVR and IPC camera video streams. The main functions of the SDK include target detection

Measurement, perimeter function, face capture and recognition function, vehicle license plate detection and recognition function and non-motor vehicle detection function:

ÿ Target detection function: detect human figures, faces, motor vehicles, non-motor vehicles and pet cats and dogs in the picture, and return the target position

Place;

ÿ Perimeter function: supports perimeter functions such as area intrusion, cross-border detection, area entry, and area exit;

ÿ Face capture and recognition function: Track and select faces in the picture according to the capture rules, and make

Attribute analysis and feature extraction comparison search;

ÿ Vehicle license plate recognition detection function: detect the motor vehicle and its license plate in the picture, and return the license plate number, vehicle attributes and

License plate attributes;

ÿ Non-motor vehicle detection function: detect the position of non-motor vehicles in the picture, support flat view (road, corridor, etc.) and bird's-eye view

There are two types of scenes: corner (elevator, etc.);

2 Authorization Description

The algorithm SDK requires authorization before use. Users first need to contact the business window to apply for an authorization account, and then use the authorization

After the tool obtains the license key, it can pass the license key string through the initialization interface.

The authorization tool supports running on PC Windows/Linux or devices to obtain authorization.

2.1 Device Authorization

You need to ensure that the device can connect to the Internet, and then deploy the rkauth_tool_bin program of the corresponding platform to the device.

Take the RV1109 platform as an example, use adb to push to the device:

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

Then execute the following command on the device to authorize

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

After successful execution, the license key will be saved to the "key.lic" file. Repeated authorization of the same device will not reduce the number of authorizations.

If key.lic is lost, you can retrieve it again.

2.2 Authorization on PC

If the device cannot connect to the Internet, you can authorize it on the PC. First, you need to obtain device information from the device, and then use the PC's authorization

The tool generates a license key based on the device information.

First, deploy the rkdevice_info executable of the corresponding platform to the device

Take RV1109 as an example, push to the device via adb

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

Execute the rkdevice_info command on the device. If successful, the device.inf file will be generated in the current directory. Copy the file to

Back to PC.

Make sure the PC is connected to the Internet and execute the authorization command on the PC

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

After successful execution, the license key will be saved in the "key.lic" file.

3 Demo usage instructions

3.1 Image Demo Usage Instructions

The SDK provides a command line demo that allows you to easily compile and run the SDK to test its functions by reading an image.

Demo is located in the demo directory of the SDK package. Users only need to execute the compilation script of the corresponding platform.

Take the RV1109 platform as an example

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

After the compilation is complete, a Demo executable file will be generated in the install directory and pushed to the device

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

Next, you need to push the model file to the device. The model directory can be specified in the initialization parameters of the Demo.

Defaults to "/userdata/rockiva_data".

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

Next, you need to prepare test images. The demo runs by reading consecutive image files in the folder one by one.

Demos of streaming functions (detection, perimeter, face, vehicle, etc.) are available

The demo/rockiva_demo/tools/video_to_images.py script extracts images from every three frames (by default) of the video and displays them in order.

0000.jpg, 0001.jpg, 0002.jpg … are put into the image folder in the current directory.

```
cd /userdata/rockiva_rv1109_linux /rockiva_demo ./rockiva_demo /userdata/xxxx/
100
```

The first parameter of Demo is the directory of the image, and the second parameter is the number of images to run. After execution, it will be generated in the current directory.

out folder, which can be copied to a PC to view the running results.

4 SDK Development Instructions

ROCKIVA SDK first obtains human shapes, faces, motor vehicles, non-motor vehicles, etc. from the image based on the target detection and tracking algorithm.

Target information, and then further intelligent analysis based on the target detection and tracking results. Currently supported intelligent analysis functions include weekly

The intelligent algorithms include interfaces, faces, vehicles (motor vehicles and non-motor vehicles) and license plates. More intelligent algorithms will be added based on demand.

Each function that can be analyzed can be turned on or off independently, and can also be used simultaneously.

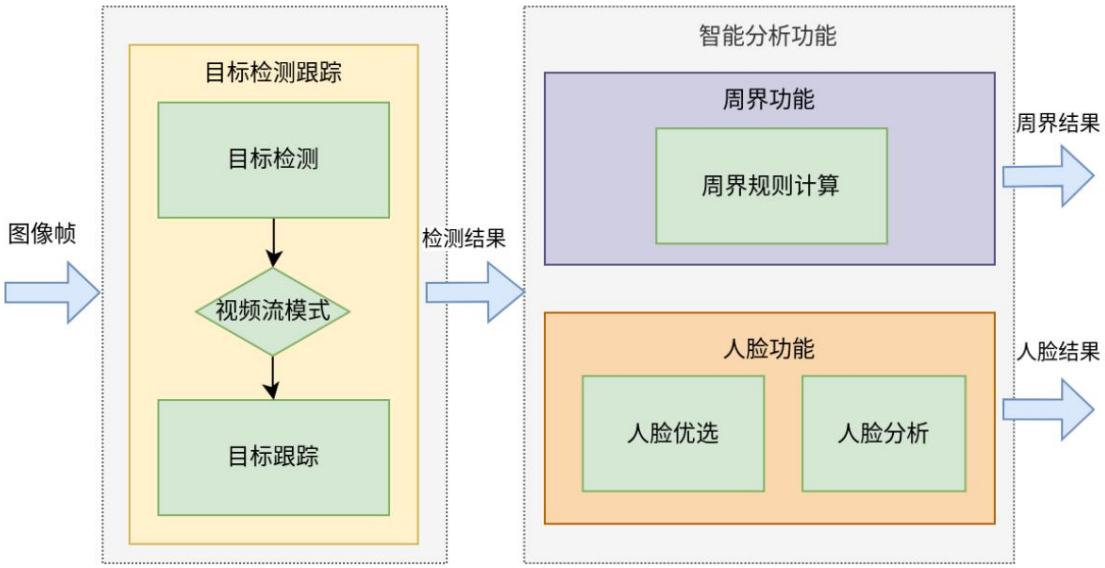

Figure 4-1 RovkIva SDK overall process

4.1 Common Modules

4.1.1 Functional Description

The interface of ROCKIVA SDK is in asynchronous mode. The basic operation is to configure initialization and then send

Image, get the result and release the frame memory in the callback function, and finally release it if it is not needed.

The basic calling process of SDK is shown in Figure 4-1-1.

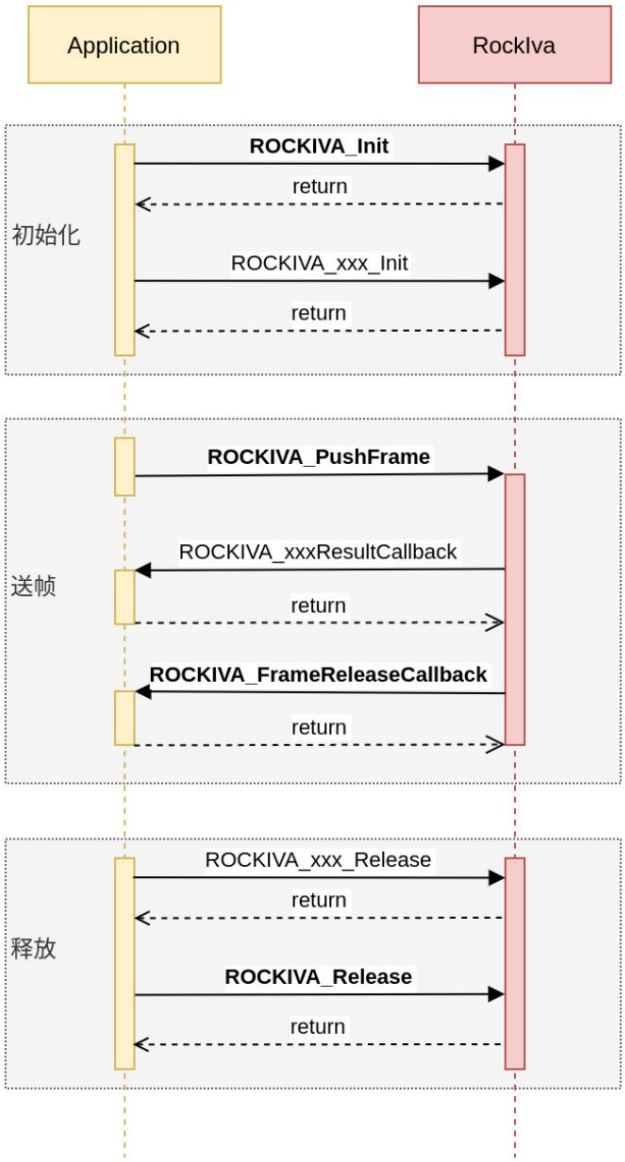

Figure 4-1-1 SDK basic calling process

The callback function ROCKIVA_FrameReleaseCallback is used to notify the application that the image frame can be released. The SDK will internally judge

If the image frame is no longer needed, notify the external application to release it.

For functions that require internal cache frames, such as face capture, it is necessary to release the frames through this callback function;

For functions such as perimeter measurement or other functions that do not require internal frame caching, after obtaining the corresponding results for each frame, the SDK does not need to cache the frames again.

The image frame has been operated, so it can also be released directly in the result callback function.

4.1.2 API Reference

4.1.2.1 Interface

interface	describe
ROCKIVA_Init algorithm	Algorithm SDK initialization configuration
ROCKIVA_Release SDK destruction and release	
ROCKIVA_SetFrameReleaseCallback sets the frame release callback function	
ROCKIVA_FrameReleaseCallback frame release callback function	
ROCKIVA_PushFrame inputs a specific	Input image frame
ROCKIVA_PushFrameWithRoiGet SDK version	valid area of the image frame
ROCKIVA_GetVersion	

4.1.2.1.1 ROCKIVA_Heat

ÿFunctionÿ

Initialize the algorithm SDK configuration and initialize the incoming handle

ÿstatementÿ

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

ÿparameterÿ

Parameter	I/O Input	describe
Namehandle		instance handle to initialize
mode		Enter working mode
userdata		Enter initialization parameters
param		Enter user-defined data

ÿReturn valueÿ

RockIvaRetCode

4.1.2.1.2 ROCKIVA_Release

ÿFunctionÿ

Algorithm SDK Destruction Release

ÿstatementÿ

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

ÿparameterÿ

Parameter	Input/Output	describe
Name handle	Input instance	handle to be released

ÿReturn valueÿ

RockIvaRetCode

4.1.2.1.3 ROCKIVA_SetFrameReleaseCallback

ÿFunctionÿ

Set the frame release callback function

ÿstatementÿ

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

ÿparameterÿ

Parameter	Input/Output	describe
Name	Input instance	handle
handle callback	Input callback function	

ÿReturn valueÿ

RockIvaRetCode

4.1.2.1.4 ROCKIVA_FrameReleaseCallback

ÿFunctionÿ

Frame release callback function

ÿstatementÿ

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

ÿparameterÿ

Parameter Name Input/Output		describe
RockIvaReleaseFrames Input list of releasable frames		
userdata	Enter user-defined data	

ÿReturn valueÿ

RockIvaRetCode

4.1.2.1.5 ROCKIVA_PushFrame

ÿFunctionÿ

Input image frame

ÿstatementÿ

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

ÿparameterÿ

Parameter	Input/Output	describe
Namehandle	Input instance handle	
inputImg	Input Input image frame	

ÿReturn valueÿ

RockIvaRetCode

4.1.2.1.6 ROCKIVA_PushFrameWithRoi

ÿFunctionÿ

Input image frame (supports setting the effective area)

ÿstatementÿ

```
RockIvaRetCode ROCKIVA_PushFrameWithRoi(RockIvaHandle handle,
const RockIvaImage* inputImg, const RockIvaAreas* roiAreas);
```

ÿparameterÿ

Parameter	Input/Output	describe
Namehandle	Input instance handle	
inputImg	Input Input image frame	

ÿReturn valueÿ

RockIvaRetCode

4.1.2.1.7 ROCKIVA_GetVersion

ÿFunctionÿ

Get SDK version number

ÿstatementÿ

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

ÿparameterÿ

Parameter	Input/Output Description
name maxLen	Enter the size of the buffer storing the version number
version	Enter the version number buffer address

ÿReturn valueÿ

RockIvaRetCode

4.1.2.2 Enumeration

enumerate	describe
RockIvaImageFormat	Image pixel format
RockIvaImageTransform	Rotation mode of input image
RockIvaExecuteStatus	Execution callback result status code
RockIvaRetCode	Function returns error code
RockIvaLogLevel	Log printing level
RockIvaObjectType	Target Type
RockIvaWorkMode	Working Mode
RockIvaMemType	Memory type

4.1.2.2.1 RockIvaImageFormat

ÿFunctionÿ

Image pixel format type

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
ROCKIVA_IMAGE_FORMAT_GRAY8	Single channel grayscale image
ROCKIVA_IMAGE_FORMAT_RGB888	Three-channel color image
ROCKIVA_IMAGE_FORMAT_BGR888	Four-channel color image
ROCKIVA_IMAGE_FORMAT_RGBA8888	Four-channel color image
ROCKIVA_IMAGE_FORMAT_BGRA8888	Four-channel color image
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 image (this format is not supported for input images)

ÿNotesÿ

none

4.1.2.2.2 RockIvaImageTransform

ÿFunctionÿ

Rotation mode of input image

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
ROCKIVA_IMAGE_TRANSFORM_NONE	Normal
ROCKIVA_IMAGE_TRANSFORM_FLIP_H	horizontal
ROCKIVA_IMAGE_TRANSFORM_FLIP_V	flip vertical flip
ROCKIVA_IMAGE_TRANSFORM_ROTATE_90 90 degrees clockwise	
ROCKIVA_IMAGE_TRANSFORM_ROTATE_180 180 degrees clockwise	
ROCKIVA_IMAGE_TRANSFORM_ROTATE_270 270 degrees clockwise	

4.1.2.2.3 RockIvaExecuteStatus

ÿFunctionÿ

Execution callback result status code

ÿstatementÿ

```
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;
```

ÿmemberÿ

The	describe
ROCKIVA_SUCCESS member operation results are normal	
ROCKIVA_UNKNOWN Unknown error type	
ROCKIVA_NULL_PTR Failed on null pointer	
ROCKIVA_ALLOC_FAILED Internal to apply for space	
ROCKIVA_INVALID_INPUT Internal input	
ROCKIVA_EXECUTE_FAILED execution error	
ROCKIVA_NOT_CONFIGURED The service has Unconfigured type	
ROCKIVA_NO_CAPACITY The been fully configured.	
ROCKIVA_BUFFER_FULL license cache area is full	
ROCKIVA_LICENSE_ERROR exception	
ROCKIVA_JPEG_DECODE_ERROR Decoding error	
ROCKIVA_DECODER_EXIT	Decoder internal exit

4.1.2.2.4 RockIvaRetCode

ÿFunctionÿ

Function call returns error code

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
ROCKIVA_RET_SUCCESS	success
ROCKIVA_RET_FAIL	fail
ROCKIVA_RET_NULL_PTR	Null Pointer
ROCKIVA_RET_INVALID_HANDLE Invalid handle	
ROCKIVA_RET_LICENSE_ERROR	License Error
ROCKIVA_RET_UNSUPPORTED Not supported	
ROCKIVA_RET_STREAM_SWITCH stream switching	
ROCKIVA_RET_BUFFER_FULL	Cache area full

4.1.2.2.5 RockIvaLogLevel

ÿFunctionÿ

Log printing level

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_LOG_ERROR	Error level (default)
ROCKIVA_LOG_WARN	Warning Level
ROCKIVA_LOG_DEBUG	Debug Level
ROCKIVA_LOG_INFO	Information level
ROCKIVA_LOG_TRACE	Trace call level

ÿillustrateÿ

The larger the log level value, the more logs are printed.

4.1.2.2.6 RockIvaObjectType

ÿFunctionÿ

Detected target categories

ÿstatementÿ

```
typedef enum {
    ROCKIVA_OBJECT_TYPE_NONE = 0x0,
    ROCKIVA_OBJECT_TYPE_PERSON = 0x1,
    ROCKIVA_OBJECT_TYPE_VEHICLE = 0x2,
    ROCKIVA_OBJECT_TYPE_NON_VEHICLE = 0x4,
    ROCKIVA_OBJECT_TYPE_FACE = 0x8,
    ROCKIVA_OBJECT_TYPE_HEAD = 0x10,
    ROCKIVA_OBJECT_TYPE_PET = 0x20,
    ROCKIVA_OBJECT_TYPE_MOTORCYCLE = 0x40,
    ROCKIVA_OBJECT_TYPE_BICYCLE = 0x80,
    ROCKIVA_OBJECT_TYPE_PLATE = 0X100
} RockIvaObjectType;
```

ÿmemberÿ

member	describe
ROCKIVA_OBJECT_TYPE_NONE	unknown
ROCKIVA_OBJECT_TYPE_PERSON	doll
ROCKIVA_OBJECT_TYPE_VEHICLE	Motor Vehicles
ROCKIVA_OBJECT_TYPE_NON_VEHICLE	Non-motorized vehicles
ROCKIVA_OBJECT_TYPE_FACE	Face
ROCKIVA_OBJECT_TYPE_HEAD	Head
ROCKIVA_OBJECT_TYPE_PET	Cats and dogs
ROCKIVA_OBJECT_TYPE_MOTORCYCLE	Battery car
ROCKIVA_OBJECT_TYPE_BICYCLE	bike
ROCKIVA_OBJECT_TYPE_PLATE	License Plate

4.1.2.2.7 RockIvaWorkMode

ÿFunctionÿ

Working Mode

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_MODE_VIDEO	Video streaming mode (enable tracking)
ROCKIVA_MODE_PICTURE	Picture mode (no tracking)

ÿillustrateÿ

The perimeter function must work in video stream mode; the detection and face functions can use video stream or picture mode;

4.1.2.2.8 RockIvaMemType

ÿFunctionÿ

Memory type

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_MEM_TYPE_CPU	Virtual Address Memory
ROCKIVA_MEM_TYPE_DMA	DMA BUF Memory (RV1106)

ÿillustrateÿ

The input image memory space type pre-opened when developing the Demo. For the RV1106 platform RGA, DMA BUF memory is required.

type.

4.1.2.3 Structure

Structure	describe
RockIvaPoint	Point coordinates
RockIvaLine	Line Coordinates
RockIvaRectangle	Regular quadrilateral coordinates
RockIvaQuadrangle	Quadrilateral Coordinates
RockIvaArea	Polygonal area
RockIvaAreas	List of polygonal regions
RockIvaSize	Target size
RockIvaAngle	Angle information
RockIvaRectExpandRatio	The ratio of the detection frame to expand to the surrounding area
RockIvaImageInfo	Image information
RockIvaImage	image
RockIvaObjectInfo	Single target detection result information
RockIvaDetectResult	Target detection results
RockIvaReleaseFrames	List of frames that need to be released
RockIvaInitParam	SDK global parameter configuration

4.1.2.3.1 RockIvaPoint

ÿFunctionÿ

Point coordinates

ÿstatementÿ

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

} RockIvaPoint;
```

ÿmemberÿ

member	Describes
x	the horizontal axis, expressed in ten thousandths, valid from 0 to 9999.
and	The vertical axis, expressed in ten thousandths, valid from 0 to 9999.

4.1.2.3.2 RockIvaLine

ÿFunctionÿ

Line Coordinates

ÿstatementÿ

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

ÿmemberÿ

member	describe
head tail	Head coordinate
	tail coordinate

4.1.2.3.3 RockIvaRectangle

ÿFunctionÿ

Regular quadrilateral coordinates

ÿstatementÿ

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

ÿmemberÿ

member	describe
topLeft	Upper left corner coordinates
bottomRight	Lower right corner coordinates

4.1.2.3.4 RockIvaQuadrangle

ÿFunctionÿ

Arbitrary quadrilateral coordinates

ÿstatementÿ

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

ÿmemberÿ

member	describe
topLeft	Upper left corner coordinates
topRight	Upper right corner coordinates
bottomLeft	Lower left corner coordinates
bottomRight	Lower right corner coordinates

4.1.2.3.5 RockIvaArea

ÿFunctionÿ

Polygon area coordinates

ÿstatementÿ

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

ÿmemberÿ

	describe
Member	The coordinates
pointNum points	of all points in the area enclosed by the number of points

4.1.2.3.6 RockIvaAreas

ÿFunctionÿ

Multi-region

ÿstatementÿ

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

ÿmemberÿ

	describe
Member areaNum	Area number
areas	area array

4.1.2.3.7 RockIvaSize

ÿFunctionÿ

Target size around

ÿstatementÿ

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

} RockIvaSize;
```

ÿmemberÿ

member	describe
width	Width (per-millionth coordinate, value range 0~9999)
height	Height (per-million coordinate, value range 0~9999)

4.1.2.3.8 RockIvaRectExpandRatio

ÿFunctionÿ

The ratio of the detection frame to expand to the surrounding area

ÿstatementÿ

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

ÿmemberÿ

member	describe
up	The detection frame expands upwards in proportion to the frame height
down	The size of the detection frame that expands downward according to the frame height
left	The detection frame expands to the left in proportion to the frame width
right	The detection frame expands to the right in proportion to the frame width

4.1.2.3.9 RockIvaAngle

ÿFunctionÿ

Face angle information

ÿstatementÿ

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

ÿmemberÿ

member	describe
pitch	Pitch angle, which indicates the rotation angle around the x-axis
yaw	Yaw angle, which indicates the rotation angle around the y axis
roll	Roll angle, which indicates the rotation angle around the z axis

4.1.2.3.10 RockIvaImageInfo

ÿFunctionÿ

Image information

ÿstatementÿ

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

ÿmemberÿ

member	describe
width	Image Width
height	Image Height
format	Image pixel format
transformMode	Rotation Mode

4.1.2.3.11 RockIvaImage

ÿFunctionÿ

image

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
frameId	Original acquisition frame number (application customized)
channelId info	Channel number (customized by application, single channel is set to 0)
	Image information
size	Image data size, required when the image format is JPEG
dataAddr	Input image data virtual address
dataPhyAddr	Input image data physical address (set to NULL if not used)
dataFd	Input image data fd (set to 0 if not used)
extData	User-defined extended data

4.1.2.3.12 RockIvaObjectInfo

ÿFunctionÿ

Single target detection result information

ÿstatementÿ

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

ÿmemberÿ

member	describe
objId	Target ID [0~232)
frameId	Frame ID
score	Object detection score [1-100]
rect	Target area frame (per 10,000)
type	Target Category

4.1.2.3.13 RockIvaDetectResult

ÿFunctionÿ

Target detection results

ÿstatementÿ

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

ÿmemberÿ

member	describe
frameId	Frame ID
frame	The corresponding input image frame
channelId	Channel ID
objNum	Number of targets
objInfo	Target detection information

4.1.2.3.14 RockIvaReleaseFrames

ÿFunctionÿ

List of frames that can be released

ÿstatementÿ

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

ÿmemberÿ

	describe
Member channelId	Channel ID
count	Number
frameId	of frames that can be released

4.1.2.3.15 RockIvaInitParam

ÿFunctionÿ

Algorithm global parameter configuration

ÿstatementÿ

```
typedef struct {
    RockIvaLogLevel RockIvaLogLevel; char
    logPath[ROCKIVA_PATH_LENGTH]; char
    modelPath[ROCKIVA_PATH_LENGTH];
    RockIvaMemInfo license; uint32_t
    coreMask; uint32_t
    channelId;
    RockIvaImageInfo imageInfo;
    RockIvaAreas roiAreas; uint32_t
    detObjectType;
} RockIvaInitParams;
```

ÿmemberÿ

member	describe
RockIvaLogLevel	Log Level
logPath	Log output path
modelPath	Path to store algorithm models
license	License Information
coreMask	Specify which NPU core to use (only valid for RK3588 platform)
channelId	Channel Number
imageInfo	Input image information
roiAreas	Valid area
detObjectType	Configure the target to detect For example, to detect people\motor vehicles\non-motor vehicles: ROCKIVA_OBJECT_TYPE_PERSON| ROCKIVA_OBJECT_TYPE_VEHICLE| ROCKIVA_OBJECT_TYPE_NON_VEHICLE

4.2 Detection Module

4.2.1 Functional Description

The detection function can be configured through the RockIvaInitParam parameter to detect the target type including face, human shape, motor vehicle,

For non-motor vehicles and pet cats and dogs, set the detection result callback. The basic calling process is shown in Figure 4-2-1.

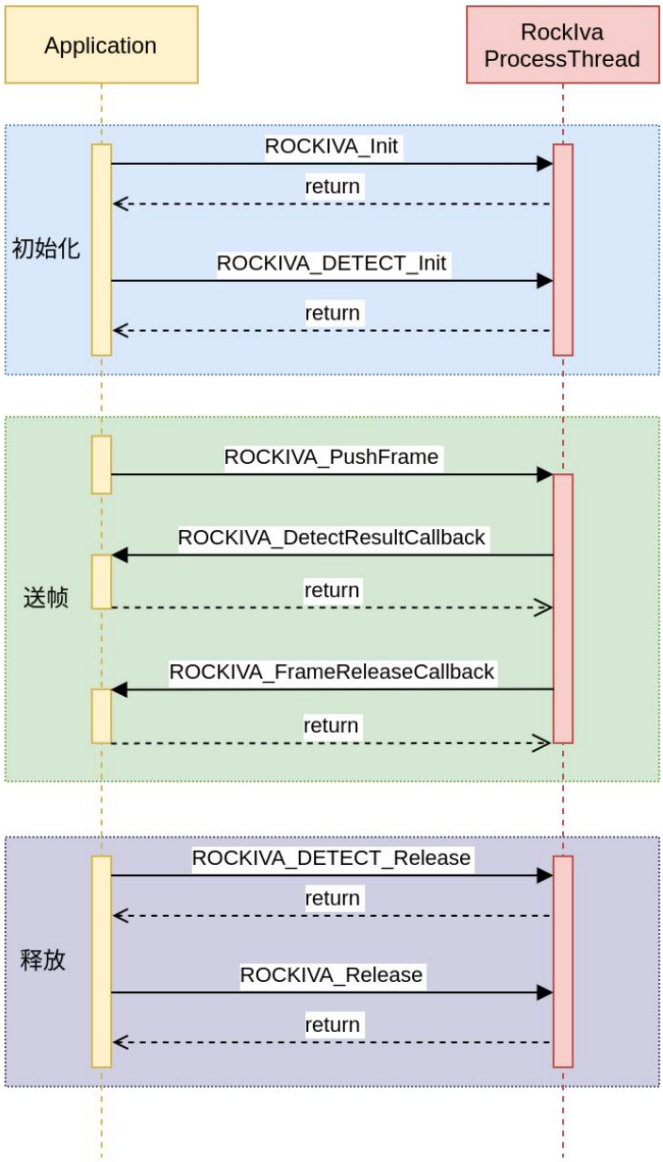

Figure 4-2-1 Detection function call process

After the configuration is initialized, the SDK will create a thread to asynchronously process each frame. Each image frame pushed by the application will be placed in a

The result callback function ROCKIVA_DetectResultCallback runs in

This internal processing thread, so try not to do operations that take too long, otherwise it will affect the processing frame rate.

The detection module can work in video streaming mode or image mode. In video streaming mode, target tracking is enabled and the detection callback

The objId of each detected target is valid; in image mode, target tracking is turned off and the objId value of the detected target is invalid.

For each frame pushed by the user, the result returned by the ROCKIVA_FrameReleaseCallback callback function needs to be

Line release.

4.2.2 API Reference

4.2.2.1 Function Interface

interface	describe
ROCKIVA_DetectResultCallback result callback function	
ROCKIVA_DETECT_Init	initialization
ROCKIVA_DETECT_Release	
ROCKIVA_DETECT_Reset	Reset

4.2.2.1.1 ROCKIVA_DetectResultCallback

ÿFunctionÿ

Result callback function

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/Output	describe
result		result
status		Status Code
userdata		User-defined data

ÿReturn valueÿ

RockIvaRetCode

4.2.2.1.2 ROCKIVA_DETECT_Init

ÿFunctionÿ

initialization

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/Output	describe
Namehandle	Input instance	handle
resultCallback	Input result callback function	

ÿReturn valueÿ

RockIvaRetCode

4.2.2.1.3 ROCKIVA_DETECT_Release

ÿFunctionÿ

release

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/Output	describe
Name handle	Input instance	handle

ÿReturn valueÿ

RockIvaRetCode

4.2.2.1.4 ROCKIVA_DETECT_Reset

ÿFunctionÿ

Reset

ÿstatementÿ

```
RockIvaRetCode ROCKIVA_DETECT_Reset(RockIvaHandle handle);
```

ÿInput parametersÿ

Parameter Name	The handle of the	describe
handle	input/output input	to be reset, reconfigured at runtime (reconfiguration will cause some internal records to be cleared and reset, but the model will not be reinitialized)

ÿReturn valueÿ

RockIvaRetCode

4.3 Perimeter Module

4.3.1 Functional Description

The process of perimeter function is similar to detection. Initialize the perimeter related configuration and configure the perimeter through ROCKIVA_BA_Init.

Boundary result callback.

The perimeter function module code calling process is shown in Figure 4-3-1.

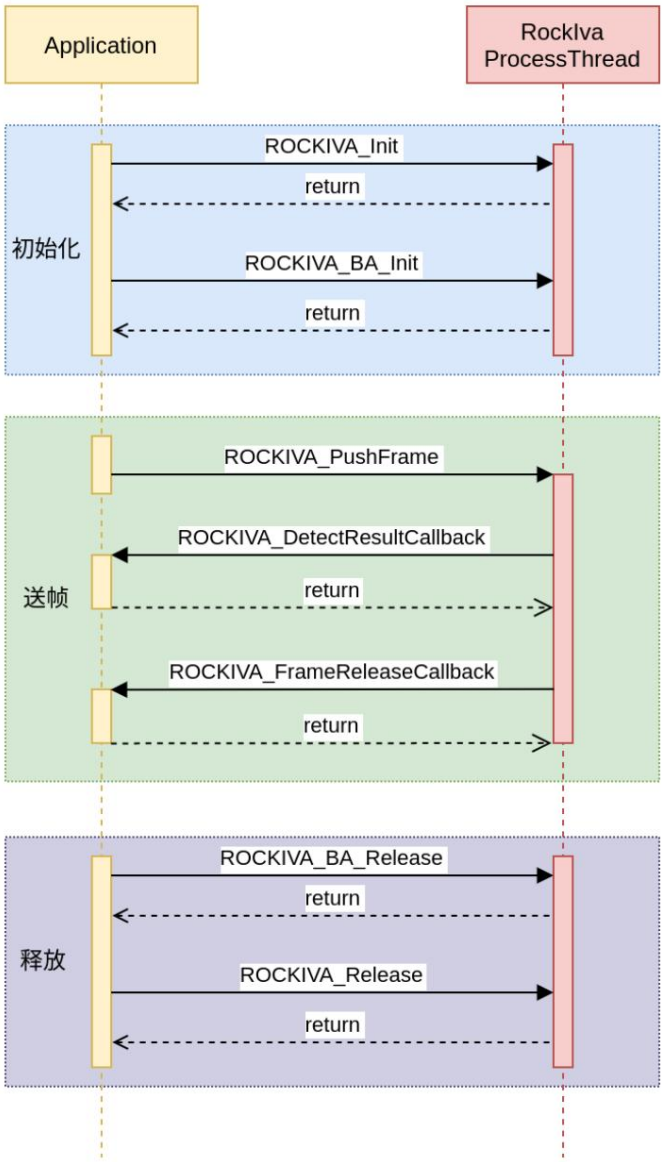

Figure 4-3-1 Perimeter function call process

Note that the perimeter must work in video streaming mode (ROCKIVA_MODE_VIDEO), because the perimeter rules

The judgment needs to be calculated based on the coordinate changes of the target's previous and next frames.

There are four types of perimeter functions:

ÿ Area Invasion: Triggered when the target enters the set area and stays there for more than the set time

ÿ Area entry: Triggered when the target enters the area from outside the set area

ÿ Area Leaving: Triggered when the target leaves the set area.

ÿ Cross-border detection: The target crosses the set line segment (in accordance with the set direction) and is triggered

The main configurable rules are:

Rule configuration	illustrate
area	Set the regular area: a polygon consisting of up to 6 points in order
boundaries	Set rule boundary: a line segment formed by two points, direction can be set
Target type filter sets the triggered target type: human, vehicle, non-motor vehicle	
Target size filter Set target size: minimum and maximum width and height	

The results of area intrusion are shown in Figure 4-3-2 and Figure 4-3-3. When the target enters the restricted area, an event will be reported.

The cross-border result is shown in Figure 4-3-4. When the target crosses the boundary line from a prohibited direction, an event will be reported.

Figure 4-3-2 Area invasion before entering restricted area

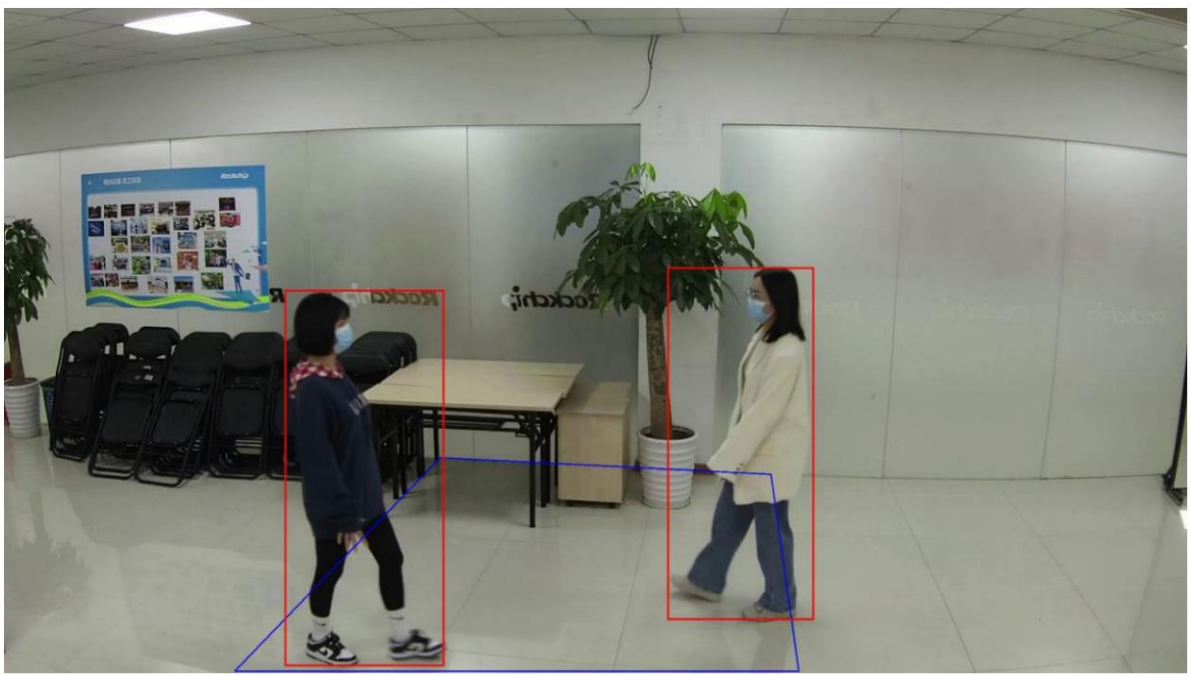

Figure 4-3-3 After entering the restricted area

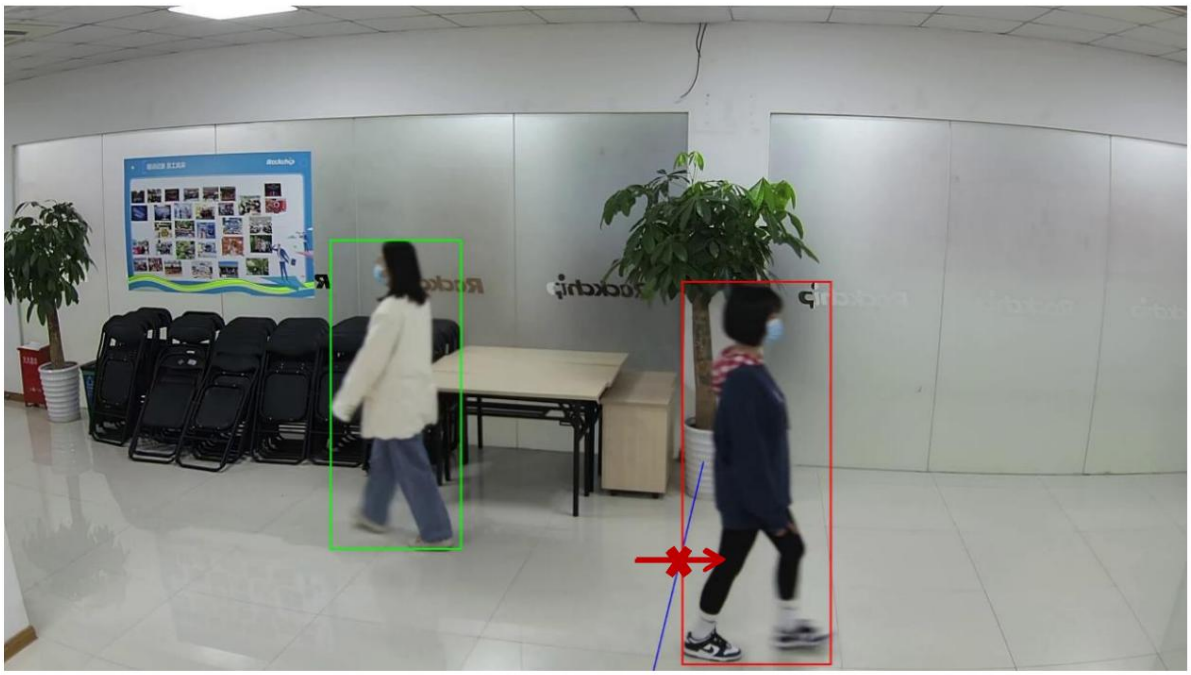

Figure 4-3-4 Out-of-bounds results

4.3.2 API Reference

4.3.2.1 Function Interface

interface	describe
ROCKIVA_BA_ResultCallback result callback function	
ROCKIVA_BA_InitDestruction	initialization
ROCKIVA_BA_DestroyReset	
ROCKIVA_BA_Reset	

4.3.2.1.1 ROCKIVA_BA_ResultCallback

ÿFunctionÿ

Result callback function

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/Output	describe
result		result
status		Status Code
userdata		User-defined data

ÿReturn valueÿ

RockIvaRetCode

4.3.2.1.2 ROCKIVA_BA_Heat

ÿFunctionÿ

initialization

ÿstatementÿ

```
RockIvaRetCode ROCKIVA_BA_Init(RockIvaHandle handle,
                               const RockIvaBaTaskParams *initParams,
                               const ROCKIVA_BA_ResultCallback resultCallback);
```

ÿInput parametersÿ

Parameter	Input/output	describe
Name	handle to be initialized	
handle	Enter initialization parameter configuration	
initParams resultCallback	Input callback function	

ÿReturn valueÿ

RockIvaRetCode

4.3.2.1.3 ROCKIVA_BA_Destroy

ÿFunctionÿ

destroy

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/output	describe
Name handle		handle to be destroyed

ÿReturn valueÿ

RockIvaRetCode

4.3.2.1.4 ROCKIVA_BA_Reset

ÿFunctionÿ

Reset

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	I/O input	describe
Name handle		handle to be reset
params		Enter new initialization parameter configuration

ÿReturn valueÿ

RockIvaRetCode

4.3.2.2 Enumeration Definition

Enumeration Description	
RockIvaBaTripEvent	Perimeter Rule Type (Tripwire/Area Event)
RockIvaBaRuleObjectFilter triggers rule target type filtering	
RockIvaBaRuleTriggerTypeTarget rule trigger type	

4.3.2.2.1 RockIvaBaTripEvent

ÿFunctionÿ

Perimeter Rule Type (Tripwire/Area Event)

ÿstatementÿ

```
typedef enum {
    ROCKIVA_BA_TRIP_EVENT_BOTH = 0,
    ROCKIVA_BA_TRIP_EVENT_DEASIL = 1,
    ROCKIVA_BA_TRIP_EVENT_WIDDERSHINES = 2,
    ROCKIVA_BA_TRIP_EVENT_IN = 3,
    ROCKIVA_BA_TRIP_EVENT_OUT = 4,
    ROCKIVA_BA_TRIP_EVENT_STAY = 5,
} RockIvaBaTripEvent;
```

ÿmemberÿ

Member	describe
ROCKIVA_BA_TRIP_EVENT_BOTH Trip line: trigger the bidirectional trigger	
ROCKIVA_BA_TRIP_EVENT_DEASIL trigger clockwise	
ROCKIVA_BA_TRIP_EVENT_WIDDERSHINES Trip line: trigger the entry area counterclockwise	
ROCKIVA_BA_TRIP_EVENT_IN Area Invasion	
ROCKIVA_BA_TRIP_EVENT_STAY	

4.3.2.2.2 RockIvaBaRuleObjectFilter

ÿFunctionÿ

Trigger rule target type filtering

ÿstatementÿ

```
typedef enum {
    ROCKIVA_BA_RULE_OBJ_NONE = 0,
    ROCKIVA_BA_RULE_OBJ_VEHICLE = 1,
    ROCKIVA_BA_RULE_OBJ_NONVEHICLE = 2,
    ROCKIVA_BA_RULE_OBJ_PERSON = 4,
    ROCKIVA_BA_RULE_OBJ_FULL = 7,
} RockIvaBaRuleObjectFilter;
```

ÿmemberÿ

member	describe
ROCKIVA_BA_RULE_OBJ_NONE	No goals
ROCKIVA_BA_RULE_OBJ_VEHICLE	Enabling Motor Vehicle Targets
ROCKIVA_BA_RULE_OBJ_NONVEHICLE	Enable non-motorized vehicle targets
ROCKIVA_BA_RULE_OBJ_PERSON	Enable humanoid target
ROCKIVA_BA_RULE_OBJ_FULL	Enable all

4.3.2.2.3 RockIvaBaRuleTriggerType

ÿFunctionÿ

Target rule trigger type

ÿstatementÿ

```
typedef enum {
    ROCKIVA_BA_RULE_NONE = 0b000000000000,
    ROCKIVA_BA_RULE_CROSS = 0b000000000001,
    ROCKIVA_BA_RULE_INAREA = 0b000000000010,
    ROCKIVA_BA_RULE_OUTAREA = 0b000000000100,
    ROCKIVA_BA_RULE_STAY = 0b000000001000,
} RockIvaBaRuleTriggerType;
```

ÿmemberÿ

member	describe
ROCKIVA_BA_RULE_NONE	
ROCKIVA_BA_RULE_CROSS	Mixing line
ROCKIVA_BA_RULE_INAREA	Enter the area
ROCKIVA_BA_RULE_OUTAREA	Leaving the area
ROCKIVA_BA_RULE_STAY	Area Invasion

4.3.2.3 Structure Definition

Structure	describe
RockIvaBaWireRule	Cross-border rules
RockIvaBaAreaRule	Regional Rules
RockIvaBaTaskRule	Behavior analysis rule configuration
RockIvaBaAiConfig	Algorithm Configuration
RockIvaBaTaskParams	Behavior analysis service initialization parameter configuration
RockIvaBaTrigger	First trigger rule information
RockIvaBaObjectInfo	Basic information of single target detection
RockIvaBaResult	All information of test results

4.3.2.3.1 RockIvaBaWireRule

ÿFunctionÿ

Cross-border rules

ÿstatementÿ

```
typedef struct {
    uint8_t ruleEnable;
    uint32_t ruleID;
    RockIvaLine line;
    RockIvaLine directLine;
    RockIvaBaTripEvent event;
    RockIvaSize minObjSize[3];
    RockIvaSize maxObjSize[3];
    uint32_t objType; uint8_t
    rulePriority; uint8_t sense;

    uint8_t triggerMode;
} RockIvaBaWireRule;
```

ÿmemberÿ

	describe
Member	Whether the rule is enabled, 1->enabled, 0->disabled
ruleEnable	Rule ID, valid range: [0, 3]
ruleID	Out of bounds configuration
line directLine	Direction line configuration
event	Cross-border direction
minObjSize	The ratio of 10,000 indicates the smallest target: 0 Motor vehicle 1 Non-motor vehicle 2 Pedestrian
maxObjSize	The percentage indicates the largest target: 0 Motor vehicle 1 Non-motor vehicle 2 Pedestrian
objType	Set the trigger target type: RockIvaBaRuleObjectFilter Example: car, person: RULE_OBJ_VEHICLE | RULE_OBJ_PERSON
rulePriority	Rule priority: 0 high, 1 medium, 2 low
sense	Sensitivity, 1~100
trigerFashion	Trigger mode: 0: The target is triggered only once; 1: The target is triggered every time it crosses the boundary

4.3.2.3.2 RockIvaBaAreaRule

ÿFunctionÿ

Regional Rules

ÿstatementÿ

```
typedef struct {
    uint8_t ruleEnable;
    uint32_t ruleID;
    RockIvaArea area;
    RockIvaBaTripEvent event;
    RockIvaSize minObjSize[3];
    RockIvaSize maxObjSize[3];
    uint32_t objType;
    uint32_t alertTime;
    uint8_t sense;
    uint8_t checkEnter;
} RockIvaBaAreaRule;
```

ÿmemberÿ

	describe
Member	Whether the rule is enabled, 1->enabled, 0->disabled
ruleEnable ruleID	Rule ID, valid range: [1, 256]
area	Regional Configuration
event	Regional Events
minObjSize	The ratio of 10,000 indicates the smallest target: 0 Motor vehicle 1 Non-motor vehicle 2 Pedestrian
maxObjSize	The percentage indicates the largest target: 0 Motor vehicle 1 Non-motor vehicle 2 Pedestrian
objType	Configure the trigger target type: RockIvaBaRuleObjectFilter Example: car, person: RULE_OBJ_VEHICLE | RULE_OBJ_PERSON
alertTime	Alarm time setting
sense	Sensitivity, 1~100
checkEnter	Whether the area intrusion rule needs to check whether the target has entered [0: Disable, 1: Enable]

4.3.2.3.3 RockIvaBaTaskRule

ÿFunctionÿ

Perimeter rule configuration

ÿstatementÿ

```
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;
```

ÿmemberÿ

ÿÿ	describe
tripWireRule	Cross-border
areaInRule	events Entering a
areaOutRule	region Leaving a
areaInBreakRule	region Region intrusion

4.3.2.3.4 RockIvaBaAiConfig

ÿFunctionÿ

Algorithm Configuration

ÿstatementÿ

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

} RockIvaBaAiConfig;
```

ÿmemberÿ

member	
DescriptionfilterPersonMode Filter non-motor vehicle/motor vehicle driver human results0: Do not filter; 1: FilterdetectResultMode Report	
target detection result mode0: Do not report detection targets that do not trigger rules; 1: Report detection targets that do not trigger rules	

4.3.2.3.5 RockIvaBaTaskParams

ÿFunctionÿ

Perimeter initialization parameter configuration

ÿstatementÿ

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

ÿmemberÿ

member	describe
baRules	Perimeter rule parameter configuration Initialization
aiConfig	algorithm configuration

4.3.2.3.6 RockIvaBaTrigger

ÿFunctionÿ

The target triggers the rule information for the first time, which can be used for snapshot

ÿstatementÿ

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

ÿmemberÿ

member	describe
ruleID	Trigger rule ID
triggerType	Trigger rule type

4.3.2.3.7 RockIvaBaObjectInfo

ÿFunctionÿ

Single target information that triggers perimeter rules

ÿstatementÿ

```
typedef struct {
    RockIvaObjectInfo objInfo; uint32_t triggerRules;

    RockIvaBaTrigger firstTrigger;
} RockIvaBaObjectInfo;
```

ÿmemberÿ

member	describe
objInfo	Target detection result information
triggerRules firstTrigger	Target triggered rules
	The rule that the target triggers for the first time

4.3.2.3.8 RockIvaBaResult

ÿFunctionÿ

All information of test results

ÿstatementÿ

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

ÿmemberÿ

member	describe
frameId	Input image frame ID
frame	The corresponding input image frame
channelId	Channel Number
objNum	Number of targets
triggerObjects	Targets that trigger perimeter rules

4.4 Face Module

4.4.1 Functional Description

The face function captures face images according to the set face capture rules. The capture modes are divided into optimal capture and fast capture.

Perform attribute analysis on captured faces, including age, gender, masks, etc.

ÿ Optimal capture: Capture the optimal face of the target person that meets the rules from the time the target person appears to the time when the target person times out or disappears

ÿ Quick capture: capture the first face of the target person that meets the rules after it appears

1) Face capture mode

Face capture must work in video streaming mode, and the face quality will be calculated for the detected faces in the image.

Then multiple frames are compared to select the best face. When the capture conditions are met (according to the configured capture rules: optimal capture/fast capture), the capture will be triggered.

Send a snapshot and the callback returns the result.

The face capture function interface calling process is shown in Figure 4-4-1.

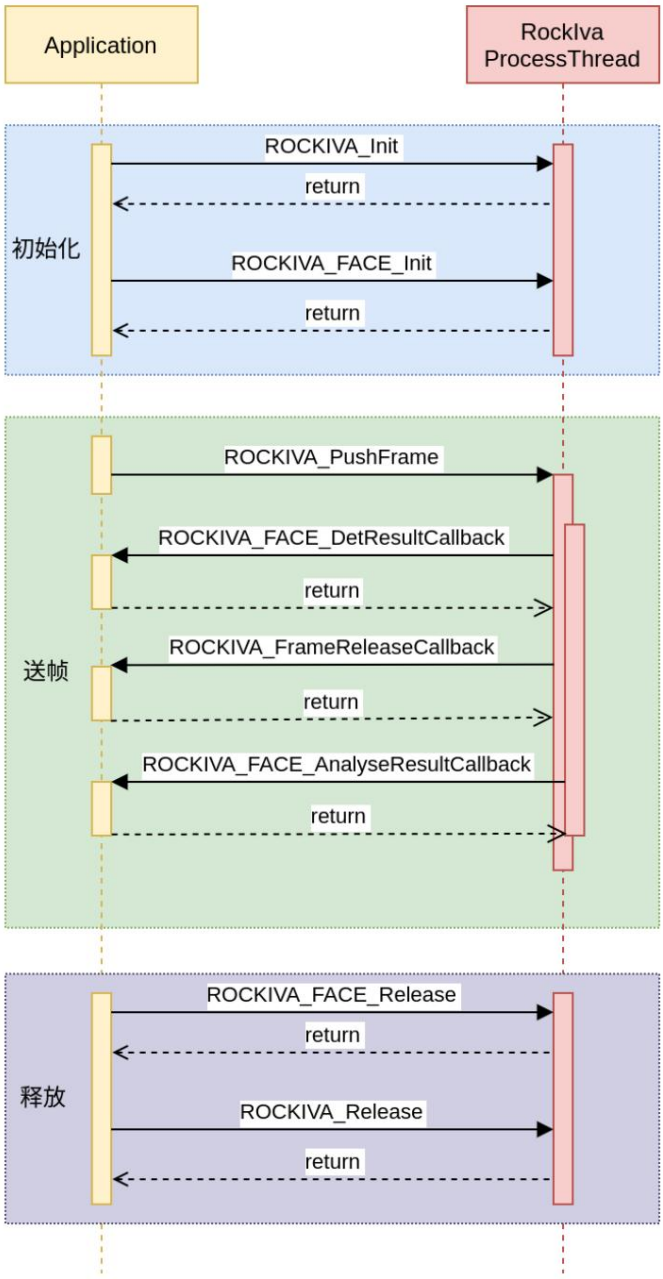

Figure 4-4-1 Face capture interface calling process

After detecting a face from an image, the face quality is calculated and then passed to the callback function.

ROCKIVA_FACE_DetResultCallback returns the result information of target detection and quality.

After the snapshot rule is triggered, the face analysis and face feature extraction will be executed asynchronously in another thread according to the configuration.

Call function ROCKIVA_FACE_AnalyseResultCallback to return.

2) Single face image analysis mode

In single face image analysis mode, each target detected in the image will be evaluated for face quality.

The face will be analyzed for attributes and features based on the configuration switch.

Tracking and face quality optimization.

3) Face library import mode

The face library mode is similar to the single face image analysis mode, but the resolution of the image sent to the face library is very different.

Therefore, a more appropriate face detection model will be selected internally, so the library import mode cannot be used together with other functions.

In library mode, only the largest face will be selected for facial feature extraction, and the facial feature results will be stored in the library after being obtained from outside.

4.4.2 **API** Reference

4.4.2.1 Function Interface

interface	describe
ROCKIVA_FACE_DetResultCallback face detection result callback function	
ROCKIVA_FACE_AnalyseResultCallback face capture analysis result callback function initialization	
ROCKIVA_FACE_Init Release	
ROCKIVA_FACE_Release 1:1 face feature	
ROCKIVA_FACE_FeatureCompare comparison interface	
ROCKIVA_FACE_FeatureLibraryControl Face feature control interface, used to add, delete, modify, search, and clear a face library Face feature	library retrieval interface, used to retrieve
ROCKIVA_FACE_SearchFeature	features of a face library

4.4.2.1.1 ROCKIVA_FACE_DetResultCallback

ÿFunctionÿ

Face detection result callback function

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/Output	describe
result		Face detection results
status		Execution status code
userdata		User-defined data

ÿReturn valueÿ

RockIvaRetCode

4.4.2.1.2 ROCKIVA_FACE_AnalyseResultCallback

ÿFunctionÿ

Face capture analysis result callback function

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/Output	describe
result		Face capture analysis results
status		Execution status code
userdata		User-defined data

ÿReturn valueÿ

RockIvaRetCode

4.4.2.1.3 ROCKIVA_FACE_Init

ÿFunctionÿ

Face function initialization

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/output	describe
Name	handle that needs to be initialized	
handle	Input face task mode	
workType initParams	Enter initialization parameters	

ÿReturn valueÿ

RockIvaRetCode

4.4.2.1.4 ROCKIVA_FACE_Reset

ÿFunctionÿ

Reset

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	I/O input	describe
Name handle	handle to be reset	
params	Enter new initialization parameter configuration	

ÿReturn valueÿ

RockIvaRetCode

4.4.2.1.5 ROCKIVA_FACE_Release

ÿFunctionÿ

destroy

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/Output	describe
Name handle	Input	handle to be released

ÿReturn valueÿ

RockIvaRetCode

4.4.2.1.6 ROCKIVA_FACE_FeatureCompare

ÿFunctionÿ

1:1 facial feature comparison interface

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/Output	describe
feature1	Input	face feature 1
feature2	Input	facial features 2
score	Output	face 1:1 comparison similarity (range 0.00-1.00)

ÿReturn valueÿ

RockIvaRetCode

4.4.2.1.7 ROCKIVA_FACE_FeatureLibraryControl

ÿFunctionÿ

Face feature control interface, used to add, delete, and modify a face database

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/Output	describe
libName	Input face	database name
action	Input face	database operation type: add, delete, modify, search, clear
faceIdInfo	Enter Face	ID
faceIdNum	Enter the F	ace ID number
faceIdData	Input facial	feature data
featureSize	Input facial	feature data size

ÿReturn valueÿ

RockIvaRetCode

4.4.2.1.8 ROCKIVA_FACE_SearchFeature

ÿFunctionÿ

Face feature library retrieval interface, used to retrieve features of a face library

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/Output	describe
libName	Input	face database name
featureData	Input	facial features
featureSize	Input	facial feature data size
in a	Input	Enter the number of features to compare
topK	Input	the top K most similar feature values
results	Output	comparison results

ÿReturn valueÿ

RockIvaRetCode

4.4.2.2 Enumeration Definition

enumerate	describe
RockIvaFaceOptType	Face selection type
RockIvaFaceWorkMode	Face service type
RockIvaFaceStatus	Face status
RockIvaFaceGenderType	gender
RockIvaFaceAgeType	age
RockIvaFaceGlassesType	Glasses
RockIvaFaceSmileType	Smile
RockIvaFaceMaskType	Wear a mask
RockIvaFaceBeardType	beard
RockIvaFaceCapFrameType Capture type	
RockIvaFaceQualityResultCode face quality result	
RockIvaFaceLibraryAction	Face database feature update operation type

4.4.2.2.1 RockIvaFaceOptType

ÿFunctionÿ

Face selection type

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_FACE_OPT_BEST Effect priority mode, the best face from the appearance to the disappearance of the target	
ROCKIVA_FACE_OPT_FAST Fast priority mode, targeting faces that meet the set quality threshold	

4.4.2.2.2 RockIvaFaceWorkMode

ÿFunctionÿ

Face service type

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_FACE_MODE_NORMAL	Normal mode (according to)
ROCKIVA_FACE_MODE_IMPORT	Library guide mode (base map feature extraction)
ROCKIVA_FACE_MODE_SEARCH	Image search mode (not yet implemented)

4.4.2.2.3 RockIvaFaceState

ÿFunctionÿ

Face status

ÿstatementÿ

```
typedef enum {
    ROCKIVA_FACE_STATE_NONE,                    /* Face status unknown*/
    ROCKIVA_FACE_STATE_FIRST,                   /* The face appears for the first time */
    ROCKIVA_FACE_STATE_TRACKING,                /* Face detection and tracking process*/
    ROCKIVA_FACE_STATE_CAPTURING,               /* Face capture processing*/
    ROCKIVA_FACE_STATE_ANALYZING, face analysis in   /* Face triggers capture and reporting,
progress*/
    ROCKIVA_FACE_STATE_CAPTURED,                /* Face capture processing completed (capture completed
Enter this state after the callback function is completed, or directly enter this state if the quality is not met and no report is required) */
    ROCKIVA_FACE_STATE_LAST                     /* The last time the target appeared*/
} RockIvaFaceStatus;
```

ÿmemberÿ

member	describe
ROCKIVA_FACE_STATE_NONE	Face status unknown
ROCKIVA_FACE_STATE_FIRST	The first appearance of a human face
ROCKIVA_FACE_STATE_TRACKING	Face detection and tracking process
ROCKIVA_FACE_STATE_CAPTURING	Face capture processing
ROCKIVA_FACE_STATE_ANALYZING	Face triggers capture and reporting, and face analysis is in progress
ROCKIVA_FACE_STATE_CAPTURED	Face capture processing completed (capture completion callback letter Enter this state after the number of times has ended, or the quality is not met If no report is required, the system will directly enter this state.)
ROCKIVA_FACE_STATE_LAST	The target was last seen

4.4.2.2.4 RockIvaFaceGenderType

ÿFunctionÿ

gender

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_GENDER_TYPE_UNKNOWN	unknown
ROCKIVA_GENDER_TYPE_MALE	male
ROCKIVA_GENDER_TYPE_FEMALE	woman

4.4.2.2.5 RockIvaFaceAgeType

ÿFunctionÿ

age

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
ROCKIVA_AGE_TYPE_UNKNOWN	unknown
ROCKIVA_AGE_TYPE_CHILD	child
ROCKIVA_AGE_TYPE_EARLYYOUTH	juvenile
ROCKIVA_AGE_TYPE_YOUTH	youth
ROCKIVA_AGE_TYPE_MIDLIFE	middle aged
ROCKIVA_AGE_TYPE_OLD	elderly

4.4.2.2.6 RockIvaFaceGlassesType

ÿFunctionÿ

Type of glasses

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_GLASSES_TYPE_UNKNOWN	unknown
ROCKIVA_GLASSES_TYPE_NOGLASSES	Not wearing glasses
ROCKIVA_GLASSES_TYPE_GLASSES	Wearing glasses
ROCKIVA_GLASSES_TYPE_SUNGLASSES	Sunglasses (reserved)

4.4.2.2.7 RockIvaFaceSmileType

ÿFunctionÿ

Smile or not

ÿstatementÿ

```
typedef enum {
    ROCKIVA_SMILE_TYPE_UNKNOWN = 0,
    ROCKIVA_SMILE_TYPE_YES = 1,
    ROCKIVA_SMILE_TYPE_NO = 2
} RockIvaFaceSmileType;
```

ÿmemberÿ

member	describe
ROCKIVA_SMILE_TYPE_UNKNOWN	unknown
ROCKIVA_SMILE_TYPE_YES	Smile
ROCKIVA_SMILE_TYPE_NO	Not smiling

4.4.2.2.8 RockIvaFaceMaskType

ÿFunctionÿ

Wear a mask?

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_MASK_TYPE_UNKNOWN	unknown
ROCKIVA_MASK_TYPE_YES	Wear a mask
ROCKIVA_MASK_TYPE_NO	Not wearing a mask

4.4.2.2.9 RockIvaFaceBeardType

ÿFunctionÿ

Whether there is a beard

ÿstatementÿ

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

ÿmemberÿ

member	describe
ROCKIVA_BEARD_TYPE_UNKNOWN	
ROCKIVA_BEARD_TYPE_YES	Unknown
ROCKIVA_BEARD_TYPE_NO	With beard Without beard

4.4.2.2.10 RockIvaFaceCapFrameType

ÿFunctionÿ

Capture type

ÿstatementÿ

```
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,
} RockIvaFaceCapFrameType;
```

ÿmemberÿ

member	describe
ROCKIVA_FACE_CAP_TYPE_UNKNOWN unknown	
ROCKIVA_FACE_CAP_TYPE_IMPORT face import mode result frame	
ROCKIVA_FACE_CAP_TYPE_PICTURE Single picture operation mode result frame	
ROCKIVA_FACE_CAP_TYPE_DISPEAR Target disappears and is captured	
ROCKIVA_FACE_CAP_TYPE_TIMEOUT Timeout capture frame	
ROCKIVA_FACE_CAP_TYPE_QUALITY captures the face when the threshold is met	
ROCKIVA_FACE_CAP_TYPE_FORENOTICE Report face analysis results in advance (such as mask wearing judgment, etc.)	

4.4.2.2.11 RockIvaFaceQualityResultCode

ÿFunctionÿ

Face quality results

ÿstatementÿ

```
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
} RockIvaFaceQualityResultCode;
```

ÿmemberÿ

	describe
ROCKIVA_FACE_QUALITY_OK	Member: face quality is qualified
ROCKIVA_FACE_QUALITY_NO_FACE No face detected	
ROCKIVA_FACE_QUALITY_SCORE_FAIL The face quality score is too low	
ROCKIVA_FACE_QUALITY_SIZE_FAIL The face is too small	
ROCKIVA_FACE_QUALITY_CLARITY_FAIL The face is blurred	
ROCKIVA_FACE_QUALITY_ANGLE_FAIL The face angle is too large	
ROCKIVA_FACE_QUALITY_MASK_FAIL Face occlusion	

4.4.2.2.12 RockIvaFaceLibraryAction

ÿFunctionÿ

Feature Update Type

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
ROCKIVA_FACE_FEATURE_INSERT	Add Features
ROCKIVA_FACE_FEATURE_DELETE	Deleting Features
ROCKIVA_FACE_FEATURE_UPDATE	Update Features
ROCKIVA_FACE_FEATURE_RETRIEVAL	Find Tags
ROCKIVA_FACE_FEATURE_CLEAR	Clear library information

4.4.2.3 Structure Definition

Structural	describe
RockIvaFaceTaskType Face SDK	Algorithm business type
RockIvaFaceCapacity	processing capabilities
RockIvaFaceQualityConfig face quality filter	configuration
Initialization parameter	Face capture rule settings
RockIvaFaceParams face attribute	configuration of RockIvaFaceRule face analysis service
RockIvaFaceAttribute face angle	structure
RockIvaAngle	information
RockIvaFaceQualityInfo face quality information	
RockIvaFaceInfo Basic	Basic information of single target face detection
RockIvaFaceAnalyseInfo face detection	information of single target face detection
RockIvaFaceDetResult Face capture	processing results
Detailed information corresponding to	processing result
RockIvaFaceIdInfo feature	the RockIvaFaceCapResult stored features, user input
RockIvaFaceSearchResult	comparison returns the result
RockIvaFaceSearchResults feature comparison returns	a list of results
RockIvaFaceCallback callback function	
RockIvaCaptureImageConfig capture and report image	configuration

4.4.2.3.1 RockIvaFaceTaskType

ÿFunctionÿ

Algorithm business type

ÿstatementÿ

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

ÿmemberÿ

	describe
Member faceCaptureEnable Face capture	service 1: valid faceRecognizeEnable
Face recognition service 1: valid faceAttributeEnable Face attribute analysis	
service 1: valid relatedPersonEnable Whether to associate with human body 1: valid	

4.4.2.3.2 RockIvaFaceCapacity

ÿFunctionÿ

Face SDK processing capabilities

ÿstatementÿ

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

} RockIvaFaceCapacity;
```

ÿmemberÿ

The	describe
maxDetectNum maximum number of detections of a member	
maxCaptureNum The maximum number of captures The maximum	
maxRecogNum	number of recognitions

4.4.2.3.3 RockIvaFaceQualityConfig

ÿFunctionÿ

Face quality filter configuration

ÿstatementÿ

```
typedef struct { uint16_t
    minScore; uint16_t minSize;
    uint16_t minClarity;

    RockIvaAngle maxAngle;
} RockIvaFaceQualityConfig;
```

ÿmemberÿ

ÿÿ	describe
minScore	Minimum quality score (default 0, no
minSize	filtering) Minimum face size (per ten thousand [0-10000], default 0 is 30 pixels)
minClarity	Minimum clarity (default 0, no filtering)
maxAngle	Maximum face angle (default 0, no filtering)

4.4.2.3.4 RockIvaFaceRule

ÿFunctionÿ

Face capture rule settings

ÿstatementÿ

```
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 faceQualityThrehold; uint8_t
    captureWithMask;
} RockIvaFaceRule;
```

ÿmemberÿ

	describe
Member	Detection sensitivity [1,100]
sensitivity	Whether to set the detection area [0 Off 1 On]
detectAreaEn	Detection area
detectArea qualityFilterMode Face quality filter mode [0: no report if not satisfied, 1: report if not satisfied but no face detection]	analyze]
qualityConfig	Face quality filter configuration
faceCapacity	The maximum number of faces to be detected, captured, and recognized is currently only available for setting the maximum number of faces to be captured. Number, 0 [unlimited]
captureImageConfig: capture and report face image configuration	
optType	Face selection type
optBestNum 1-3	ROCKIVA_FACE_OPT_BEST: The number of faces that are preferred (not implemented) Range:
optBestOverTime The optimal face quality capture timeout setting in ms. If the face does not disappear within this period,	Report the best quality face in this period; if it is 0, the best quality face will be reported after it disappears. Face
faceQualityThrehold The face quality threshold that meets the requirements for fast capture	
captureWithMask supports capturing faces of people wearing masks and reporting whether the faces are wearing masks [0: off; 1: on;	2: Open and masks are reported in advance (satisfying qualityConfig configuration)], if opened then The minMouthScore and minNoseScore filters of qualityConfig are invalid

4.4.2.3.5 RockIvaFaceTaskParams

ÿFunctionÿ

Initialization parameter configuration of face analysis service

ÿstatementÿ

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

ÿmemberÿ

member	describe
mode	Face task mode
faceCaptureRule face capture rule	
faceTaskType	Face service type: face capture service/face recognition service

4.4.2.3.6 RockIvaFaceAttribute

ÿFunctionÿ

Face attribute structure

ÿstatementÿ

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

ÿmemberÿ

member	describe
gender	gender
age	age
emotion	emotion
eyeGlass	Glasses
smile	Smile
mask	Face mask
beard	beard
attractive	Appearance

4.4.2.3.7 RockIvaFaceQualityInfo

ÿFunctionÿ

Face quality information

ÿstatementÿ

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

ÿmemberÿ

member	describe
score	Face quality score (value range 0~100)
clarity	Face clarity (value range 0~100, 100 means the clearest)
angle	Face angle

4.4.2.3.8 RockIvaFaceInfo

ÿFunctionÿ

Basic information of single target face detection

ÿstatementÿ

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

ÿmemberÿ

member	describe
objId	Target ID [0,2^32)
frameId	The frame number where the face is located
faceRect	Original position of face area
faceQuality	Face quality information
faceState	Face status
person	Related human detection information (relatedPersonEnable needs to be set to 1)

4.4.2.3.9 RockIvaFaceAnalysisInfo

ÿFunctionÿ

Single target face analysis results

ÿstatementÿ

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

ÿmemberÿ

	describe
Member featureSize	The actual length of the feature, in bytes
feature	Semi-structured feature information
faceAttr	Face attributes

4.4.2.3.10 RockIvaFaceDetResult

ÿFunctionÿ

Face detection processing results

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
frameId	Frame number
channelId	Channel Number
objNum	Number of faces
faceInfo	Target detection information
recordNum	Number of face records
faceRecord	Face record information

4.4.2.3.11 RockIvaFaceCapResult

ÿFunctionÿ

Face capture processing results

ÿstatementÿ

```
typedef struct {
    uint32_t frameId;
    uint32_t channelId;
    RockIvaImage frame;
    RockIvaFaceCapFrameType faceCapFrameType;
    RockIvaFaceInfo faceInfo;
    RockIvaFaceAnalysisInfo faceAnalysisInfo;
    RockIvaImage captureImage;
} RockIvaFaceCapResult;
```

ÿmemberÿ

member	describe
frameId	Frame number
channelId frame	Channel Number
	The corresponding input image frame, if the cached image is not configured, the image cache data cannot be accessed ask
faceCapFrameType capture frame type	
faceInfo	Basic face detection information
faceAnalyseInfo	Face analysis information
captureImage	Face capture thumbnail

4.4.2.3.12 RockIvaFaceIdInfo

ÿFunctionÿ

Detailed information corresponding to the incoming features, entered by the user

ÿstatementÿ

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

ÿmemberÿ

	describe
Member faceIdInfo	Face tag information

4.4.2.3.13 RockIvaFaceSearchResult

ÿFunctionÿ

Feature comparison returns results

ÿstatementÿ

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

} RockIvaFaceSearchResult;
```

ÿmemberÿ

	describe
Member faceIdInfo	Face label information
score	comparison score

4.4.2.3.14 RockIvaFaceSearchResults

ÿFunctionÿ

Feature comparison returns a list of results

ÿstatementÿ

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

} RockIvaFaceSearchResults;
```

ÿmemberÿ

	describe
Member faceIdScore	Face details
in a	comparison score

4.4.2.3.15 RockIvaFaceCallback

ÿFunctionÿ

Face result callback function

ÿstatementÿ

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

ÿmemberÿ

	describe
Member	Face detection callback function pointer
detCallback analyseCallback	Face capture analysis callback function pointer

4.4.2.3.16 RockIvaCaptureImageConfig

ÿFunctionÿ

Capture and report image configuration

ÿstatementÿ

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

ÿmemberÿ

member	describe
mode	Reporting snapshot image mode, [0: none; 1: small image; 2: large image; 3: large and small images]
resizeMode	Capture image scaling method [0: Keep the target size unless the target size exceeds imageInfo sets the size; 1: Fixed scaling to the size set by imageInfo]
alignWidth expand	Alignment width of the captured image
imageInfo	Configuration of the ratio of the upper, lower, left, and right sides of the expanded face frame in the captured image
	The captured image information is set, and the width and height need to be set to 4 to align. The time is related to the width, height, image format and the maximum number of face captures.

4.5 Vehicle License Plate Module

4.5.1 Functional Description

The vehicle license plate detection and recognition function can detect the vehicle position, vehicle attributes, license plate number and license plate attributes in the image.

ROCKIVA_PLATE_Init Initializes the vehicle license plate related rule configuration and configures the vehicle license plate result callback function.

The code calling process of the vehicle license plate detection and recognition module is shown in Figure 4-5-1.

The vehicle attribute detection and recognition results are shown in Figure 4-5-2.

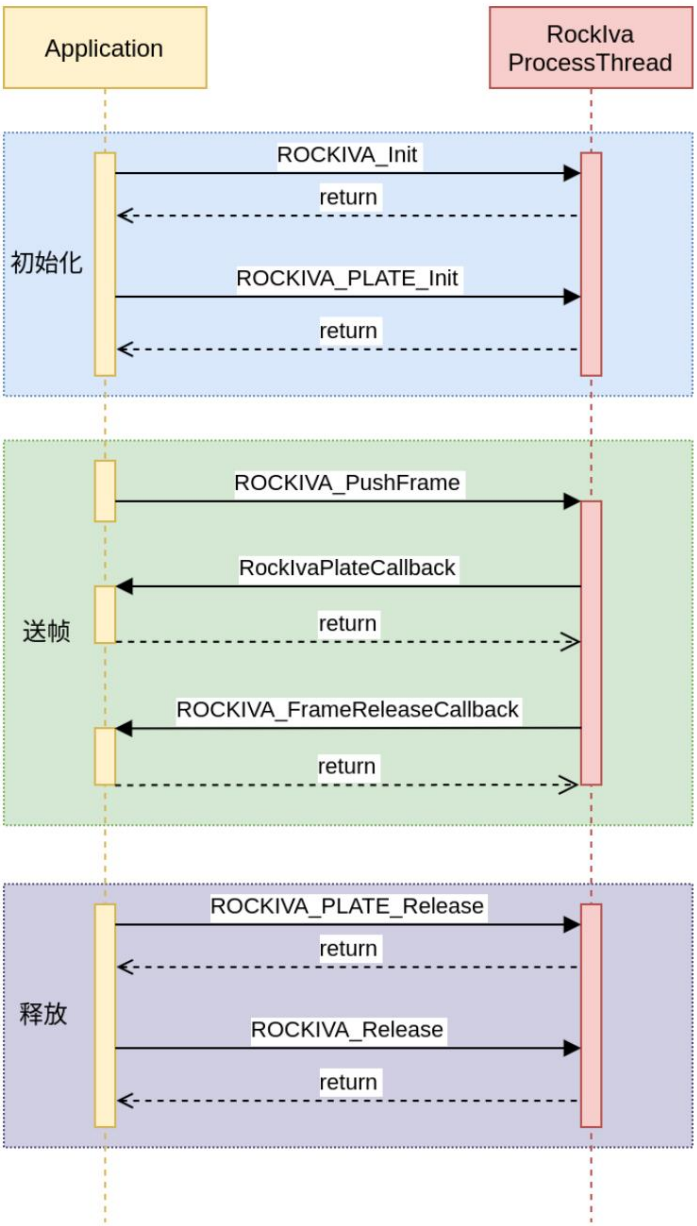

Figure 4-5-1 Vehicle license plate recognition function call process

Figure 4-5-2 Vehicle license plate detection and recognition results

4.5.2 API Reference

4.5.2.1 Function Interface

interface	describe
RockIvaPlateCallback	Vehicle license plate detection and recognition result callback function
ROCKIVA_PLATE_Init	initialization
ROCKIVA_PLATE_Reset	Reset
ROCKIVA_PLATE_Release	release

4.5.2.1.1 RockIvaPlateCallback

ÿFunctionÿ

Vehicle license plate result callback function

ÿstatementÿ

```
typedef void (*RockIvaPlateCallback)(
                                        const RockIvaPlateResult* result,
                                        const RockIvaExecuteStatus status,
                                        void* userdata);
```

ÿInput parametersÿ

Parameter name	Input/Output	describe
result	Output	vehicle license plate results
status	Output	status code
userdata	Output	user-defined data

ÿReturn valueÿ

RockIvaRetCode

4.5.2.1.2 ROCKIVA_PLATE_Init

ÿFunctionÿ

Vehicle license plate detection and recognition function initialization

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/output	describe
name	handle	that needs to be initialized
handle initParams	Enter	initialization parameters
callback	Input	callback function

ÿReturn valueÿ

RockIvaRetCode

4.5.2.1.3 ROCKIVA_PLATE_Reset

ÿFunctionÿ

Reset

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	I/O input handle	describe
Namehandle	to be reset	
initParams	Enter new initialization parameter configuration	

ÿReturn valueÿ

RockIvaRetCode

4.5.2.1.4 ROCKIVA_PLATE_Release

ÿFunctionÿ

destroy

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/Output	describe
Name handle	Input handle to be released	

ÿReturn valueÿ

RockIvaRetCode

4.5.2.2 Enumeration Definition

enumerate	describe
RockIvaPlateType	License Plate Type
RockIvaPlateColor	License plate color
RockIvaVehicleType	Vehicle type
RockIvaVehicleColor	Body color
RockIvaVehicleOrient	Vehicle direction

4.5.2.2.1 RockIvaPlateType

ÿFunctionÿ

License Plate Type

ÿstatementÿ

```
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;
```

ÿmemberÿ

Members	describe
PLATE_TYPE_NONE Large car without license plates	
PLATE_TYPE_LARGE_CAR Small car license plate	
PLATE_TYPE_SMALL_CAR license plate	
PLATE_TYPE_EMBASSY_CAR Embassy car license plate	
PLATE_TYPE_CONSULATE_CAR Consulate car license plate	
PLATE_TYPE_TRAILER Training	Trailer license plate
PLATE_TYPE_COACH_CAR Police car vehicle license plate	
PLATE_TYPE_POLICE_CAR Hong Kong license plate	
PLATE_TYPE_HONGKONG Macau entry entry and exit license plate	
PLATE_TYPE_MACAO and exit license plate	
PLATE_TYPE_ARMED_POLICE Armed Police license plate	
PLATE_TYPE_PLA	Military license plate
PLATE_TYPE_NEW_ENGERY New energy license plate	
PLATE_TYUPE_OTHER	Other license plates

4.5.2.2.2 RockIvaPlateColor

ÿFunctionÿ

License plate color

ÿstatementÿ

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

ÿmemberÿ

member	describe
PLATE_COLOR_UNKNOWN	License plate color unknown
PLATE_COLOR_BLUE	Blue Card
PLATE_COLOR_YELLOW	Yellow Card
PLATE_COLOR_GREEN	Green Card
PLATE_COLOR_BLACK	Black Card
PLATE_COLOR_WHITE	White Label

4.5.2.2.3 RockIvaVehicleType

ÿFunctionÿ

Vehicle type

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
VEHICLE_TYPE_UNKNOWN	unknown
VEHICLE_TYPE_BUS	Passenger vehicles (large buses, medium buses, buses)
VEHICLE_TYPE_SEDAN	Passenger cars (sedans, hatchbacks, subcompacts, sports cars)
VEHICLE_TYPE_VAN	Van (van, MPV)
VEHICLE_TYPE_SUV	SUV Sports Utility Vehicle
VEHICLE_TYPE_PICKUP	Pickup Truck
VEHICLE_TYPE_TRUCK	Trains (large trucks, medium trucks, small trucks, vans)

4.5.2.2.4 RockIvaVehicleColor

ÿFunctionÿ

Body color

ÿstatementÿ

```
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;
```

ÿmemberÿ

member	describe
VEHICLE_COLOR_UNKNOWN Unknown	
VEHICLE_COLOR_BLACK Blue	black
VEHICLE_COLOR_BLUE (blue, cyan)	
VEHICLE_COLOR_BROWN Gray	
VEHICLE_COLOR_GREY (gray, silver, dark gray)	
VEHICLE_COLOR_YELLOW Yellow (yellow, orange, gold)	
VEHICLE_COLOR_GREEN Purple	
VEHICLE_COLOR_PURPLE Red	
VEHICLE_COLOR_RED White	
VEHICLE_COLOR_WHITE	

4.5.2.2.5 RockIvaVehicleOrient

ÿFunctionÿ

Vehicle direction

ÿstatementÿ

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

ÿmemberÿ

member	describe
VEHICLE_ORIENT_UNKNOWN Unknown	
VEHICLE_ORIENT_FRONT Back	front
VEHICLE_ORIENT_BACK	
VEHICLE_ORIENT_SIDE	

4.5.2.3 Structure Definition

Structure	describe
RockIvaPlateTaskParam	License plate recognition service initialization parameter configuration
RockIvaVehicleAttribute	Motor vehicle attributes
RockIvaPlateInfo	License Plate Recognition Information
ROCKIVAPlateResult	License plate recognition processing results

4.5.2.3.1 RockIvaPlateTaskParam

ÿFunctionÿ

License plate recognition service initialization parameter configuration

ÿstatementÿ

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

ÿmemberÿ

ÿÿ	describe
vehicleMinSize	Minimum width of motor vehicle body (per ten thousand [0-10000])
plateMinSize	Minimum license plate width (percentage [0-10000])
plateMinScore	Minimum score for license plate recognition characters (0-99)
mode	Operation mode (0: single frame mode; 1: tracking mode, the vehicle ID will be recorded internally. Vehicles with identification results will not return results)
detectAreas	Detection area (identify the largest vehicle in each detection area)

4.5.2.3.2 RockIvaVehicleAttribute

ÿFunctionÿ

Motor vehicle attributes

ÿstatementÿ

```
typedef struct {
    RockIvaVehicleType type;
    RockIvaVehicleColor color;
    RockIvaVehicleOrient orient;
} RockIvaVehicleAttribute;
```

ÿmemberÿ

member	describe
type	Vehicle type
color	Vehicle body color
orient	Vehicle orientation

4.5.2.3.3 RockIvaPlateInfo

ÿFunctionÿ

License Plate Recognition Information

ÿstatementÿ

```
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;
```

ÿmemberÿ

ÿÿ	describe
vehicleId	Vehicle tracking ID corresponding to the license plate
plateCode	License plate characters. The maximum length of a license plate is 20
plateScore	License plate character score [0-100], the maximum length of the license plate is 20
plateLen	License plate character length
plateRect	License plate coordinates
plateType	License Plate Type
plateColor	License plate color
vehicleRect	Vehicle coordinates
vehicleAttr	Motor vehicle attributes

4.5.2.3.4 RockIvaPlateResult

ÿFunctionÿ

License plate recognition processing results

ÿstatementÿ

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

ÿmemberÿ

member	describe
frameId	Frame ID
channelId	Channel Number
frame	The corresponding input image frame
objNum	Number of license plates
plateInfo	The maximum number of license plate recognition results is 10.

4.6 Object Detection Module

4.6.1 Functional Description

The object detection function currently supports the detection of non-motor vehicles and flames (not yet open), which is initialized by ROCKIVA_OBJECT_Init.

Initialize the object (non-motor vehicle, flame) related detection rule configuration and configure the object detection result callback function.

Initialization ROCKIVA_Init does not need to specify detObjectType. And for horizontal perspective (road, corridor, etc.) and top perspective (elevator, etc.)

Adaptation and optimization have been made for two types of scenarios.

The object detection module code calling process is shown in Figure 4-6-1.

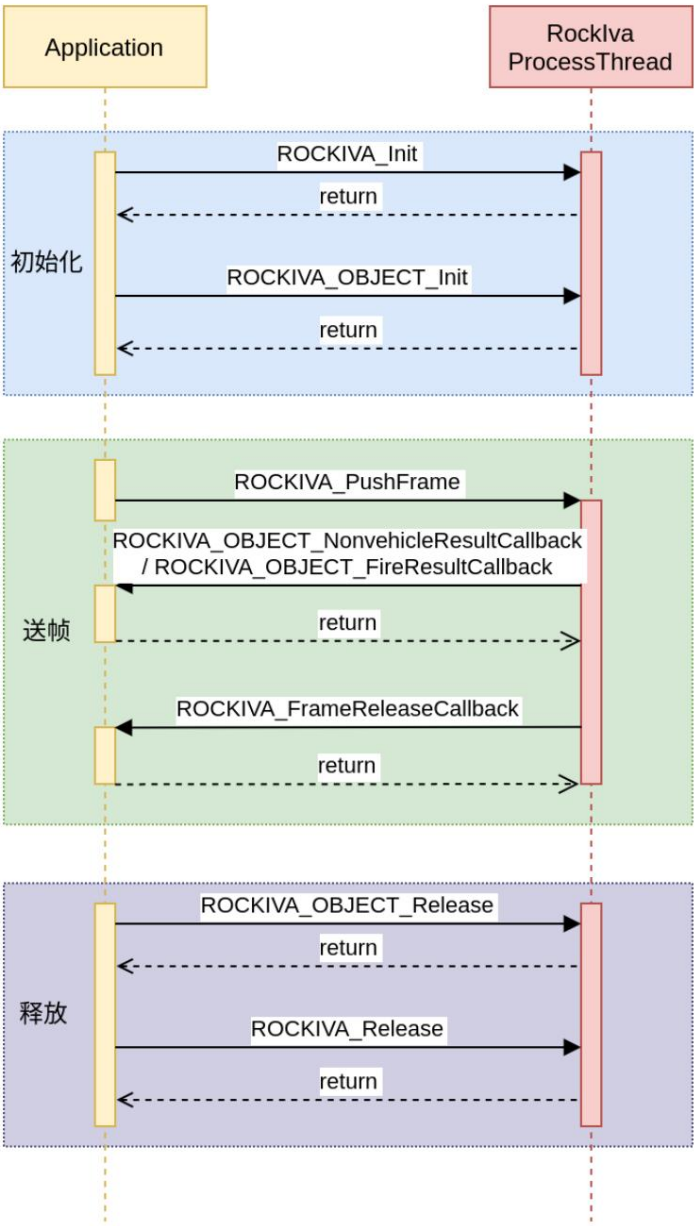

Figure 4-6-1 Object recognition function call process

The non-motor vehicle detection results are shown in Figure 4-6-2 and Figure 4-6-3 below. objID+non-motor vehicle type (1: battery vehicle, 2: automatic vehicle) car).

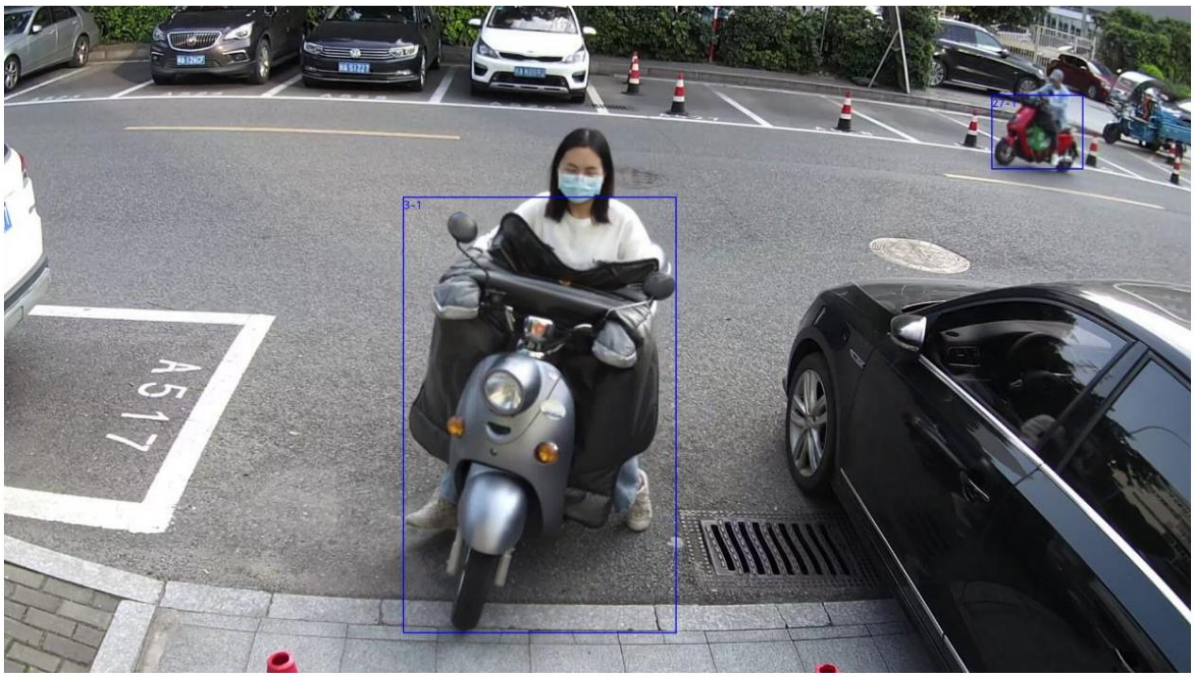

Figure 4-6-2 Electric vehicle detection

Figure 4-6-3 Bicycle detection

4.6.2 API Reference

4.6.2.1 Function Interface

interface	describe
ROCKIVA_OBJECT_NonvehicleResultCallback Non-motor vehicle	detection result callback function
ROCKIVA_OBJECT_FireResultCallback initialization	Flame detection result callback function
ROCKIVA_OBJECT_Init release	
ROCKIVA_OBJECT_Release	

4.6.2.1.1 ROCKIVA_OBJECT_NonvehicleResultCallback

ÿFunctionÿ

Non-motor vehicle detection result callback function

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/output	describe
result	Output	non-motor vehicle test results
status	Output	status code
userdata	Output	user-defined data

ÿReturn valueÿ

RockIvaRetCode

4.6.2.1.2 ROCKIVA_OBJECT_FireResultCallback

ÿFunctionÿ

Flame detection result callback function

ÿstatementÿ

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

ÿInput parametersÿ

Parameter name	Input/output	describe
result	Output	flame detection results
status	Output	status code
userdata	Output	user-defined data

ÿReturn valueÿ

RockIvaRetCode

4.6.2.1.3 ROCKIVA_OBJECT_Init

ÿFunctionÿ

Object recognition function initialization

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/output	describe
name	handle	that needs to be initialized
handle	Enter	initialization parameters
initParams callback	Input	callback function

ÿReturn valueÿ

RockIvaRetCode

4.6.2.1.4 ROCKIVA_OBJECT_Release

ÿFunctionÿ

destroy

ÿstatementÿ

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

ÿInput parametersÿ

Parameter	Input/Output	describe
Name handle	Input handle to be released	

ÿReturn valueÿ

RockIvaRetCode

4.6.2.2 Enumeration Definition

enumerate	describe
None	

4.6.2.3 Structure Definition

Structure Description	
RockIvaNonvehicleRule	Non-motor vehicle detection rule setting
RockIvaFireRule flame detection rule settings	
RockIvaObjectTaskParams non-motor vehicle	Video structured initialization parameter configuration
RockIvaNonvehicleAttribute Single target properties	
RockIvaObjectNonvehicleInfo non-motor vehicle attribute detection basic information	
RockIvaObjectNonvehicleAttrResult Non-motor vehicle detection result full information	
RockIvaObjectFireResult	Flame detection results full information
RockIvaObjectResultCallback	Callback function configuration

4.6.2.3.1 RockIvaNonvehicleRule

ÿFunctionÿ

Non-motor vehicle detection rule setting

ÿstatementÿ

```
typedef struct {

    uint8_t enable;
    uint8_t detScene;
    uint8_t sensitivity;
    uint32_t minSize;
    uint32_t filterType;
    uint32_t frameRate;
} RockIvaNonvehicleRule;
```

ÿmemberÿ

member	describe
enable	Enable 1: On; 0: Off Detection scene
detScene	default 0: Flat view battery vehicle detection; 1: Top view (inside the elevator) battery vehicle detection (not yet available) Alarm
sensitivity	sensitivity [1,100] (not yet available) Minimum
minSize	detection non-motor vehicle attribute size, less than this value is
filterType	filtered Filter type (not yet
frameRate	available) Running frame rate (not yet available)

4.6.2.3.2 RockIvaFireRule

ÿFunctionÿ

Flame detection rule setting

ÿstatementÿ

```
typedef struct {

    uint8_t enable;
    uint8_t sensitivity;
    uint32_t minSize;
    uint32_t frameRate;
} RockIvaFireRule;
```

ÿmemberÿ

member	describe
enable	Enable 1: on; 0: off
sensitivity	Alarm sensitivity [1,100] (not yet available)
minSize	Minimum flame size, filter if the flame size is smaller than this value
frameRate	Running frame rate (not yet available)

4.6.2.3.3 RockIvaObjectTaskParams

ÿFunctionÿ

Video structured initialization parameter configuration

ÿstatementÿ

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

ÿmemberÿ

	describe
Member nonvehicleRule	Non-motor vehicle inspection rules
fireRule	Flame detection rules

4.6.2.3.4 RockIvaNonvehicleAttribute

ÿFunctionÿ

Non-motor vehicle attributes

ÿstatementÿ

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

ÿmemberÿ

member	describe
type	Non-motor vehicle category

4.6.2.3.5 RockIvaObjectNonvehicleInfo

ÿFunctionÿ

Basic information of single target non-motor vehicle attribute detection

ÿstatementÿ

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

ÿmemberÿ

member	describe
objId	Target ID [0,2^32)
frameId	The frame number of the non-motor vehicle
objectRect attr	Original location of non-motorized vehicle area
alert	Non-motor vehicle attribute information
	Elevator control battery car alarm

4.6.2.3.6 RockIvaObjectNonvehicleAttrResult

ÿFunctionÿ

All information about non-motor vehicle test results

ÿstatementÿ

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

ÿmemberÿ

member	describe
frameId	Input image frame ID
channelId	Channel Number
frame	The corresponding input image frame
objNum	Number of targets
objectInfo[ROCKIVA_MAX_OBJ_NUM]	Non-motorized detection information, the maximum number of detection information is 128

4.6.2.3.7 RockIvaObjectFireResult

ÿFunctionÿ

Flame detection results full information

ÿstatementÿ

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

ÿmemberÿ

member	describe
frameId	Input image frame ID
channelId	Channel Number
objNum	Number of targets
objectInfo[ROCKIVA_MAX_OBJ_NUM]	Flame detection information, the maximum number of detection information is 128

4.6.2.3.8 RockIvaObjectResultCallback

ÿFunctionÿ

Callback function configuration

ÿstatementÿ

```
typedef struct {

    ROCKIVA_OBJECT_NonvehicleResultCallback nonvehicleCallback;
    ROCKIVA_OBJECT_FireResultCallback fireCallback;
} RockIvaObjectResultCallback;
```

ÿmemberÿ

Member nonvehicleCallback	Description Battery vehicle detection result
fireCallback	callback function Flame detection result callback function