ICS 13.310
A 91

GA

中华人民共和国公共安全行业标准

GA/T 1326—2017

安全防范　人脸识别应用
程序接口规范

Security protection—Face recognition applications—
API specifications

2017-10-08 发布　　2017-12-01 实施

中华人民共和国公安部　发布

目　　次

前　言

本标准按照 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由全国安全防范报警系统标准化技术委员会人体生物特征识别应用分技术委员会(SAC/TC 100/SC 2)提出并归口。

本标准起草单位:清华大学、公安部第一研究所、杭州海康威视数字技术股份有限公司、中国科学院自动化研究所、国防科技大学、中国科学院计算技术研究所、北京海鑫科金高科技股份有限公司、广州像素数据技术股份有限公司、上海银晨智能识别科技有限公司、浙江大华技术股份有限公司、四川川大智胜软件股份有限公司、山西省公安厅、深圳市中控生物识别有限公司、广东铂亚信息技术有限公司、江苏省公安厅、武汉市公安局、深圳市飞瑞斯科技有限公司。

本标准主要起草人:苏楠、陈健生、王生进、苏光大、侯鸿川、田青、刘君平、毛芳党、叶挺群、李子青、雷震、谢剑斌、山世光、王贤良、姚若光、张杰、汪海洋、曾文斌、赵军、陈书楷、简伟明、胡雷地、刘军、李璐。

安全防范　人脸识别应用 程序接口规范

1　范围

本标准规定了安全防范系统人脸识别应用程序接口方面的人脸采集、人脸识别算法、应用服务接口的文件格式与接口规范，规范了安防人脸识别应用系统人脸采集接口、人脸识别算法接口以及人脸识别服务接口。

本标准适用于安全防范系统人脸识别应用中的图像采集、数据处理、网络服务等接口的技术方案设计及系统的研发与应用等方面。

2　规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GA/T 893—2010　安防生物特征识别应用术语

3　术语和定义

3.1　术语和定义

GA/T 893—2010 界定的以及下列术语和定义适用于本文件。

3.1.1

人脸采集设备　face capture device

用于采集人脸图像或视频的设备。

3.1.2

名单型人脸识别　watch list based face recognition

判别一个未知身份的待测人脸样本是否在监视名单上。如果判断待测人脸样本在监视名单上，则将确定该待测人脸样本的身份。

4　基本数据类型

4.1　基本数据类型描述

基本数据类型用于规范接口传递数据过程中的数据形式，采用 8 字节对齐方式，当可无效成员[1]无效时，整型类、浮点数类赋值为－1，指针类赋值为 NULL。

4.2　图像数据

用于传递单张图像相关信息，采用结构体形式。

1)　在接口传递数据过程中，可以不生效的数据结构成员。当可无效成员无效时，该成员所赋值无实际意义。

C 语言描述：

```
typedef struct tagONEIMAGE{
long long id;
int image_width;
int image_height;
int image_depth;
int image_width;
int image_height;
int image_depth;
int image_widthstep;
char format[20];
unsigned char *data;
int data_len;
int custom_len;
void *custom;
}ONEIMAGE;
```

图像数据结构体成员说明见表 1。

表 1 图像数据类型结构体成员

成员名	数据类型	说　　明
id	64 位整型	图像编号，可无效
image_width	32 位整型	图像水平像素数
image_height	32 位整型	图像垂直像素数
image_depth	32 位整型	图像深度
image_widthstep	32 位整型	存储一行像素需要的字节数
format	字符型数组	图像格式
data	无符号字符型指针	图像数据段
data_len	32 位整型	图像数据段长度
custom_len	32 位整型	自定义数据长度，可无效
custom	无类型指针	自定义数据，可无效

4.3 多图像数据

用于传递多个图像数据，采用结构体形式。

C 语言描述：

```
typedef struct tagMULTIIMAGE{
int img_num;
ONEIMAGE *img;
}MULTIIMAGE;
```

多图像数据结构体成员说明见表 2。

表 2　多图像数据类型结构体成员

成员名	数据类型	说　　明
img_num	32 位整型	图像数据数量
img	图像数据类型指针	多个图像数据

4.4　人脸位置数据

用于传递一个人脸位置信息，采用结构体形式。

C 语言描述：

```
typedef struct tagFACEROI{
long long id;
int xleft;
int yleft;
int xright;
int yright;
float quality;
}FACEROI;
```

人脸位置数据结构成员说明见表 3。

表 3　人脸位置数据类型结构体成员

成员名	数据类型	说　　明
id	64 位整型	人脸位置数据编号，可无效
xleft	32 位整型	人脸所在区域左上角横坐标数值（以观测者视角为标准）
yleft	32 位整型	人脸所在区域左上角纵坐标数值
xright	32 位整型	人脸所在区域右下角横坐标数值
yright	32 位整型	人脸所在区域右下角纵坐标数值
quality	32 位浮点数	人脸质量，分值越高人脸质量越好[0,1]，可无效

4.5　多人脸位置数据

用于传递多个人脸位置数据，采用结构体形式。

C 语言描述：

```
typedef struct tagMULTIROI{
introi_num;
FACEROI *roi;
}MULTIROI;
```

多人脸位置数据结构成员说明见表 4。

表 4　多人脸位置数据类型结构体成员

成员名	数据类型	说　　明
roi_num	32 位整型	人脸位置数据数量
roi	人脸位置数据类型指针	多个人脸位置数据

4.6　人脸关键点数据

用于传递一个人脸关键点位置信息，采用结构体形式。

C 语言描述：

```
typedef struct tagORGANPOS{
long long id;
int xleft;
int yleft;
int xright;
int yright;
int xchin;
int ychin;
int point_len;
int * point;
int custom_len;
void * custom;
}ORGANPOS;
```

人脸关键点数据结构成员说明见表 5。

表 5　人脸关键点数据类型结构体成员

成员名	数据类型	说　　明
id	64 位整型	人脸关键点编号，可无效
xleft	32 位整型	左眼睛中心横坐标数值
yleft	32 位整型	左眼睛中心纵坐标数值
xright	32 位整型	右眼睛中心横坐标数值
yright	32 位整型	右眼睛中心纵坐标数值
xchin	32 位整型	下颌横坐标数值，可无效
ychin	32 位整型	下颌纵坐标数值，可无效
point_len	32 位整型	其他人脸关键点数据长度
point	32 位整型指针	其他人脸关键点数据，按每个点先横坐标后纵坐标形式顺序排列
custom_len	32 位整型	自定义数据长度，可无效
custom	无类型指针	自定义数据，可无效

4.7 多人脸关键点数据

用于传递多个人脸关键点数据，采用结构体形式。

C 语言描述：

```
typedef struct tagMULTIPOS{
int pos_num;
ORGANPOS * pos;
}MULTIPOS;
```

多人脸关键点数据结构成员说明见表 6。

表 6 多人脸关键点数据类型结构体成员

成员名	数据类型	说　　明
pos_num	32 位整型	人脸关键点数据数量
pos	人脸关键点数据指针	多个人脸关键点数据

4.8 人脸模板数据

人脸模板数据类型用于传递一个人脸模板信息，采用结构体形式。

C 语言描述：

```
typedef struct tagFACETEMPLATE{
longlong id;
int homology_id;
int feature_len;
void * feature;
int custom_len;
void * custom;
}FACETEMPLATE;
```

人脸模板数据结构成员说明见表 7。

表 7 人脸模板数据类型结构体成员

成员名	数据类型	说　　明
id	64 位整型	人员编号，用于区分不同人，可无效
homology_id	32 位整型	同一身份的人对应不同模板的编号，可无效
feature_len	32 位整型	人脸特征长度
feature	无类型指针	人脸特征数据
custom_len	32 位整型	自定义数据长度，可无效
custom	无类型指针	自定义数据，可无效

4.9 人脸相似度数据

用于传递一个人脸相似度信息，采用结构体形式。

C 语言描述：

```
typedef struct tagFACESIM{
longlong probe_id;
int probe_homology_id;
long longgallery_id;
int gallery_homology_id;
long long rank;
float result;
float probability;
int custom_len;
void * custom;
}FACESIM;
```

人脸相似度数据结构成员说明见表 8。

表 8　人脸相似度数据类型结构体成员

成员名	数据类型	说　　明
probe_id	64 位整型	探针模板编号，可无效
probe_homology_id	32 位整型	同一身份的人对应不同模板的编号，可无效
gallery_id	64 位整型	已知人员编号，可无效
gallery_homology_id	32 位整型	同一已知人对应不同模板的编号，可无效
rank	64 位整型	相似度排名（在同一多人脸相似度数据中按相似度降序排序的位置），可无效
result	32 位浮点数	人脸相似度，取值范围：[0～1]
probability	32 位浮点数	同一人可能性，数值越高为同一人的可能性越大，取值范围：[0～1]，可无效
custom_len	32 位整型	自定义数据长度，可无效
custom	无类型指针	自定义数据，可无效

4.10　多人脸相似度数据

用于传递多个人脸相似度数据，采用结构体形式。

C 语言描述：

```
typedef struct tagMULTISIM{
long long id;
FACESIM * sim;
long long sim_num;
}MULTISIM;
```

多人脸相似度数据结构成员说明见表 9。

表 9　多人脸相似度数据类型结构体成员

成员名	数据类型	说　　明
id	64 位整型	多人脸相似度数据编号，可无效
sim	人脸相似度数据指针	多个人脸相似度数据
sim_num	64 位整型	人脸相似度数据数量

4.11　算法模块版本信息数据

用于传递人脸识别算法模块版本信息数据，采用结构体形式。

C 语言描述：

```
typedef struct tagSDKINFO{
int module_type;
int api_type;
char developer_info[255];
char version_info[255];
}SDKINFO;
```

算法模块版本信息数据结构成员说明见表 10。

表 10　算法模块版本信息数据类型结构体成员

成员名	数据类型	说　　明
module_type	32 位整型	人脸识别算法模块支持的功能，采用标志位形式，1 为有效，0 为无效。从右向左依次为人脸检测接口标志位；独立人脸关键点定位接口标志位；集成人脸关键点定位接口标志位，独立人脸归一化接口标志位，集成人脸归一化接口标志位；独立人脸模板提取接口标志位；集成人脸模板提取接口标志位；独立人脸相似度计算接口标志位；集成人脸相似度计算接口标志位，用户可据此规则自行扩展
api_type	32 位整型	人脸识别应用类型，采用标志位形式，1 为有效，0 为无效。从右向左依次为辨认型人脸识别标志位、确认型人脸识别标志位、名单型人脸识别标志位每种类型占用一位，用户可据此规则自行扩展
developer_info	字符型数组	开发商信息（无则赋零）
version_info	字符型数组	版本信息（无则赋零）

4.12　网络地址信息数据

用于传递网络地址信息数据，采用结构体形式。

C 语言描述：

```
typedef struct tagADDRINFO{
```

```
int ip_type;
charip[64];
shortport;
}ADDRINFO;
```

网络地址信息数据结构成员说明见表 11。

表 11　网络地址信息数据类型结构体成员

成员名	数据类型	说　　明
ip_type	32 位整型	IP 协议类型,0:ipv4;1:ipv6
ip	字符型数组	IP 地址(无则赋零)
port	16 位整型	端口号

4.13　人脸采集设备信息数据

用于传递人脸采集设备的信息数据,采用结构体形式。

C 语言描述:

```
typedef struct tagDEVICEINFO{
char device_name[128];
unsigned char serial_number[48];
ADDRINFO addr_info;
short device_type;
intwork_mode;
long long chan_num;
char description[256];
}DEVICEINFO;
```

人脸采集设备信息数据结构成员说明见表 12。

表 12　人脸采集设备信息数据类型结构体成员

成员名	数据类型	说　　明
device_name	字符型数组	设备名称(节点路径,无则赋零)
serial_number	无符号字符型数组	序列号(无则赋零)
addr_info	网络地址信息数据类型	接入设备网络地址信息,可无效
device_type	无符号字符型	设备类型,0:usb 摄像头;1:网络摄像头;2:模拟摄像头,其余用户可据此规则自行扩展,可无效
work_mode	32 位整型	人脸采集设备工作模式,人脸采集设备采集人脸信息的方式,1:静态采集模式;2:动态采集模式
channel_num	64 位整型	设备通道个数,可无效
description	字符型数组	设备描述(无则赋零)

4.14 人脸采集设备码流参数数据

用于传递码流参数的数据，采用结构体形式。

C 语言描述：

```
typedef struct tagPREVIEWINFO{
long long channel;
int stream_type;
int link_mode;
void * play_wnd;
short blocked;
int preview_mode;
int proto_type;
int custom_len;
void * custom;
}PREVIEWINFO;
```

人脸采集设备码流参数数据类型结构成员说明见表 13。

表 13 人脸采集设备码流参数数据类型结构体成员

成员名	数据类型	说　　明
channel	64 位整型	通道号(起始编号为 1)，可无效
stream_type	32 位整型	码流类型，0：主码流；1：子码流，可无效
link_mode	32 位整型	连接方式，0：TCP(私有协议)；1：UDP(私有协议)；2：多播；3：RTP/RTSP (554 端口)；4：RTSP/HTTP(80 端口)，可无效
play_wnd	无类型指针	播放窗口句柄，可无效
blocked	16 位整型	取流方式 0：非阻塞取流；1：阻塞取流，可无效
preview_mode	32 位整型	预览模式 0：正常预览；1：延迟预览，可无效
proto_type	32 位整型	应用层取流协议 0：rtsp 协议；1：sip 协议，可无效
custom_len	32 位整型	自定义数据长度，可无效
custom	无类型指针	自定义数据，可无效

4.15 人脸信息数据

用于传递一个人脸位置信息，采用结构体形式。

C 语言描述：

```
typedef struct tagFACEINFO{
int id;
char abstime[32];
int channel;
```

```
FACEROI face_roi;
ONEIMAGE * face_image;
int bkg_length;
unsigned char * bkg_image;
}FACEINFO;
```

人脸信息数据结构成员说明见表 14。

表 14 人脸信息数据类型结构体成员

成员名	数据类型	说　　明
id	32 位整型	人脸编号
abstime	字符型数组	绝对时标，格式 yyyy－mm－dd hh:mm:ss
channel	32 位整型	通道号
face_roi	人脸位置数据类型	人脸位置数据
face_image	图像数据	人脸图像数据
bkg_length	32 位整型	图像长度，0：无图像
bkg_image	无符号字符型指针	原始图像数据

5 应用程序接口

5.1 应用程序接口描述

应用程序接口由人脸采集、人脸识别算法、人脸识别引用服务三部分组成。人脸采集接口用于规范调用人脸采集设备的函数，人脸识别算法接口用于规范调用人脸识别算法模块的函数，人脸识别应用服务接口用于规范对外部应用提供人脸识别服务的方法。应用程序接口整体框图如图 1 所示。

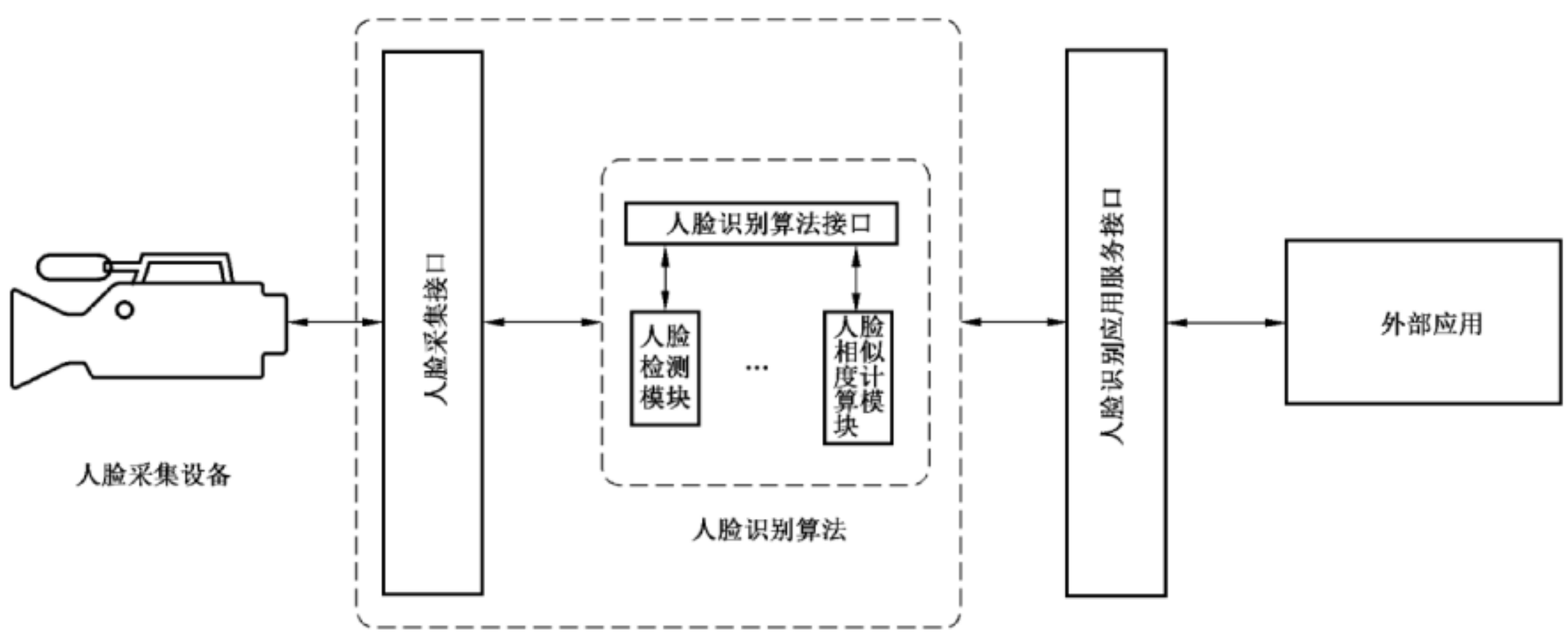

图 1 应用程序接口整体框图

5.2 人脸采集接口

5.2.1 人脸采集接口描述

人脸采集接口函数用 C 语言进行描述。人脸采集接口函数采用动态链接库形式发布，接口函数于动态链接库中导出，支持多线程，如无特殊说明非自定义输出内容的内存管理于函数外部进行、自定义输出内容的内存由函数内部分配，于函数外部使用“free”释放。当可无效参数[2]无效时，整型类、浮点数类赋值为－1，指针类赋值为 NULL。函数原型中各参数均为形式参数。对于不同的平台，应该分别编译成不同的动态链接库进行发布。函数返回值代码见附录 A。动态链接库文件名称见附录 B。人脸采集接口示例代码参见附录 D 中 D.1。

5.2.2 人脸采集接口列表

人脸采集接口如表 15 所示。

表 15 人脸采集接口

名称	说明	章节号
FR_Device_Init	设备初始化函数	5.2.4.1
FR_Device_DeviceList	设备枚举函数	5.2.4.2
FR_Device_Login	设备连接函数	5.2.4.3
FR_Device_SetMode	设备模式设置函数	5.2.4.4
FR_Device_Preview	码流传输函数	5.2.4.5
CALLBACK * REALDATACALLBACK	码流回调函数	5.2.4.6
FR_Device_StopPreview	停止传输函数	5.2.4.7
FR_Device_SetFrameUpload	图像主动上传设置函数	5.2.4.8
CALLBACK * FRAMECALLBACK	图像主动上传回调函数	5.2.4.9
FR_Device_Logout	设备注销函数	5.2.4.10
FR_Device_Cleanup	采集设备释放函数	5.2.4.11

5.2.3 人脸采集接口调用流程

人脸采集接口调用流程如图 2 所示。

2） 在调用人脸识别应用程序接口时，可以不生效的函数参数。当可无效参数无效时，该参数无实际意义。

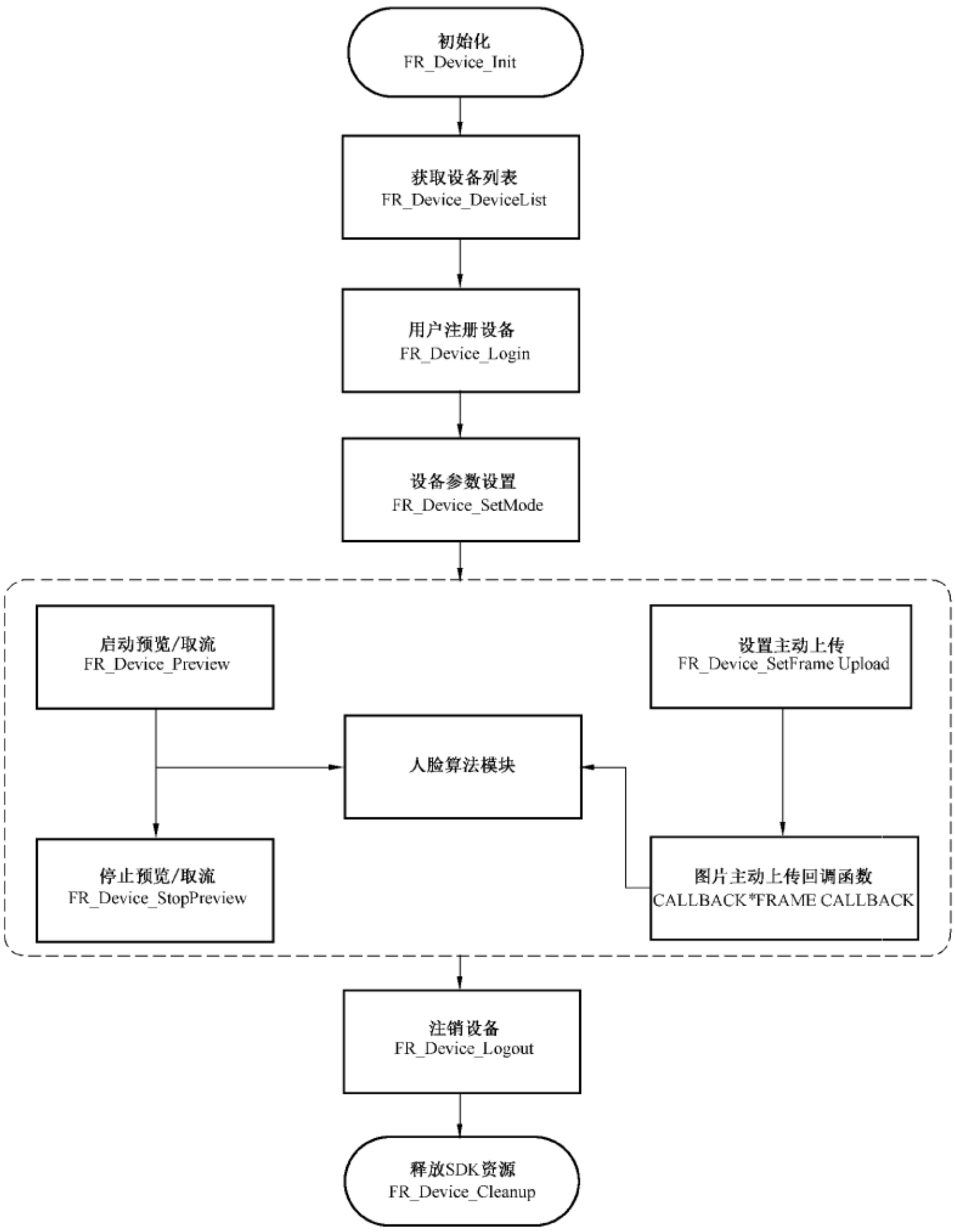

图 2　脸采集设备接口调用流程图

5.2.4　人脸采集接口函数

5.2.4.1　设备初始化函数

用于初始化人脸采集设备的函数，其应被最先调用。

函数原型：int FR_Device_Init(void * custom_input, int custom_len_input);

FR_Device_Init 调用参数如表 16 所示。

表 16 FR_Device_Init 调用参数

参数	数据类型	函数调用类型	参数说明
custom_input	无类型指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度

5.2.4.2 设备枚举函数

用于搜索当前可用的人脸采集设备的函数。

函数原型:int FR_Device_DeviceList(DEVICEINFO * device_info, int * device_num);

FR_Device_Device_List 调用参数如表 17 所示。

表 17 FR_Device_Device_List 调用参数

参数	数据类型	函数调用类型	参数说明
device_info	人脸采集设备信息数据类型指针	输出参数	人脸采集设备信息
device_num	32 位整型指针	输出参数	DEVICEINFO 数组长度

5.2.4.3 设备连接函数

用于连接人脸采集设备的函数。

函数原型:intFR_Device_Login(int api_type,char * username, char * password, DEVICEINFO * device_info, void * custom_input, int custom_len_input, long long * user_id);

FR_Device_Login 调用参数如表 18 所示。

表 18 FR_Device_Login 调用参数

参数名称	数据类型	函数调用类型	参数说明
api_type	32 位整型	输入参数	应用类型
username	字符型指针	输入参数	设备登录用户名
password	字符型指针	输入参数	设备登录密码
device_info	人脸采集设备信息数据类型指针	输入参数	设备信息
custom_input	无类型指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
user_id	64 位整型指针	输出参数	用户 id

5.2.4.4 设备模式设置函数

用于设置人脸采集设备工作模式的函数。

函数原型:int FR_Device_SetMode(long longuser_id, int work_mode, void * custom_input, int custom_len_input);

FR_Device_SetMode 调用参数如表 19 所示。

表 19 FR_Device_SetMode 调用参数

参数名称	数据类型	函数调用类型	参数说明
user_id	64 位整型	输入参数	用户 id
work_mode	32 位整型	输入参数	人脸采集设备工作模式
custom_input	无类型指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度

5.2.4.5 码流传输函数

用于传输人脸采集设备码流的函数

函数原型:intFR_Device_Preview(long longuser_id, PREVIEWINFO preview_info, REALDATACALLBACK cbfunc, void * user, long long * preview_handle);

FR_Device_Preview 调用参数如表 20 所示。

表 20 FR_Device_Preview 调用参数

参数名称	数据类型	函数调用类型	参数说明
user_id	64 位整型	输入参数	用户 id,设备连接函数调用的返回值
preview_info	人脸采集设备码流参数数据类型	输入参数	预览参数
cbfunc	码流回调函数	输入参数	码流数据回调函数
user	无类型指针	输入参数	用户数据
preview_handle	64 位整型指针	输出参数	句柄

5.2.4.6 码流回调函数

用于获取人脸采集设备码流数据的函数

函数原型:typedef void(CALLBACK * REALDATACALLBACK)(long long preview_handle, int data_type, unsigned char * buffer, int buf_size, void * user);

码流回调函数调用参数如表 21 所示。

表 21 码流回调函数调用参数

参数名称	数据类型	函数调用类型	参数说明
preview_handle	64 位整型	输入参数	句柄,码流传输函数返回值
data_type	32 位整型	输出参数	数据类型,0:文件头;1:数据
buffer	无符号字符型指针	输出参数	存放数据缓冲区
buf_size	32 位整型	输入参数	数据长度
user	无类型指针	输入参数	用户数据

5.2.4.7 停止传输函数

用于停止人脸采集设备码流传输的函数

函数原型：int FR_Device_StopPreview(long long preview_handle)；

FR_Device_StopRealPlay 调用参数如表 22 所示。

表 22　FR_Device_StopRealPlay 调用参数

参数名称	数据类型	函数调用类型	参数说明
preview_handle	64 位整型	输入参数	句柄，码流传输函数返回值

5.2.4.8　图像主动上传设置函数

用于设置人脸采集设备图像主动上传回调函数参数的函数

函数原型：int FR_Device_SetFrameUpload(long long user_id，int upload_type，int interval，FRAMECALLBACK frame_callback，void * user)；

FR_Device_SetFrameUpload 调用参数如表 23 所示。

表 23　FR_Device_SetFrameUpload 调用参数

参数名称	数据类型	函数调用类型	参数说明
user_id	64 位整型	输入参数	用户 id，设备连接函数调用的返回值
upload_type	32 位整型	输入参数	主动上传图像类型，0：抽帧上传原始图像；1：人脸图像
interval	32 位整型	输入参数	人脸图像上传时，同一目标上传帧数；抽帧上传时，均匀采样数值
frame_callback	图像主动上传回调函数	输入参数	图像主动上传回调函数
user	无类型指针	输入参数	用户数据

5.2.4.9　图像主动上传回调函数

用于获取人脸采集设备主动上传的图像的函数

函数原型：typedef void(CALLBACK * FRAMECALLBACK)(long long user_id，DEVICEINFO device_info，FACEINFO faceinfo，void * user)；

图像主动上传回调函数调用参数如表 24 所示。

表 24　图像主动上传回调函数调用参数

参数名称	数据类型	函数调用类型	参数说明
user_id	64 位整型	输入参数	用户 id，设备连接函数调用的返回值
device_info	人脸采集设备信息数据类型	输出参数	人脸采集设备信息
faceinfo	人脸信息数据数据类型	输入参数	人脸信息数据
user	无类型指针	输入参数	用户数据

5.2.4.10　设备注销函数

用于断开设备的函数

函数原型：int FR_Device_Logout(long long user_id)；

FR_Device_Logout 调用如表 25 所示。

表 25　FR_Device_Logout 调用参数

参数名称	数据类型	函数调用类型	参数说明
user_id	64 位整型	输入参数	登录用户 id

5.2.4.11　采集设备释放函数

用于释放人脸采集设备接口的函数。

函数原型：int FR_Device_Cleanup()；

5.3　人脸识别算法接口

5.3.1　人脸识别算法接口描述

人脸识别算法接口函数用 C 语言进行描述。人脸识别算法接口函数采用动态链接库形式发布，接口函数于动态链接库中导出，支持多线程。如无特殊说明非自定义输出内容的内存管理于函数外部进行、自定义输出内容的内存由函数内部分配，于函数外部使用“free”释放。当可无效参数无效时，整型类、浮点数类赋值为－1，指针类赋值为 NULL。函数原型中各参数均为形式参数。对于不同的平台，应该分别编译成不同的动态链接库进行发布。动态链接库文件名称见附录表 B。人脸识别算法接口示例代码参见 D.2。函数返回值代码见附录 A。

5.3.2　人脸识别算法接口列表

人脸识别算法接口函数列表如表 26 所示。

表 26　人脸识别算法接口列表

名称	说明	章条号
FR_Getinfo	人脸识别算法模块版本信息获取函数	5.3.4.1
FR_Initialize	人脸识别算法初始化函数	5.3.4.2
FR_Getsize	人脸特征数据长度获取函数	5.3.4.3
FR_Facedetect	人脸检测函数	5.3.4.4
FR_location_A	独立人脸关键点定位函数	5.3.4.5
FR_Location_C	集成人脸关键点定位函数	5.3.4.5
FR_Normalize_A	独立人脸归一化的函数	5.3.4.6
FR_Normalize_C	集成人脸归一化函数	5.3.4.6
FR_Extract_A	独立人脸模板提取函数	5.3.4.7
FR_Extract_C	集成人脸模板提取函数	5.3.4.7
FR_Match	人脸相似度计算函数	5.3.4.8
FR_Match_A	独立人脸相似度计算函数	5.3.4.8
FR_Match_C	集成人脸相似度计算函数	5.3.4.8
FR_Release	人脸识别算法释放函数	5.3.4.9

5.3.3 人脸识别算法接口调用流程

人脸识别算法接口调用流程如图 3 所示。

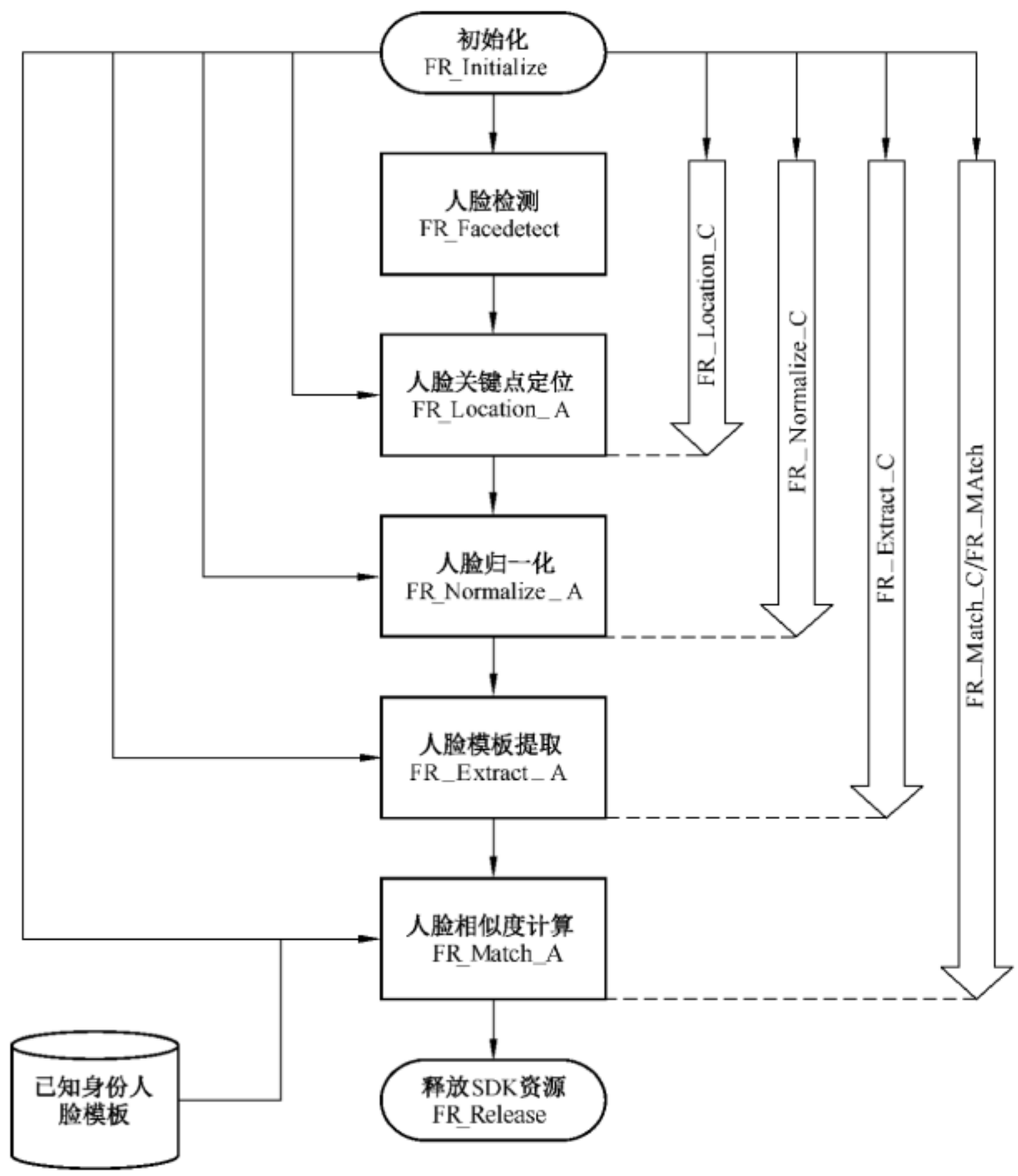

图 3 人脸识别算法接口调用流程图

5.3.4 人脸识别算法接口函数

5.3.4.1 人脸识别算法模块版本信息获取函数

用于获得算法模块信息的函数。

函数原型：int FR_Getinfo(int api_type，SDKINFO * sdk_info_output，int * info_num_output)；

FR_Getinfo 调用参数如表 27 所示。

表 27 FR_Getinfo 调用参数

参数	数据类型	函数调用类型	参数说明
api_type	32 位整型	输入参数	待获取信息模块的 API 类型
sdk_info_output	算法模块版本信息数据指针	输出参数	算法模块版本信息数据数组
info_num_output	32 位整型指针	输出参数	算法模块版本信息数据数组长度

5.3.4.2 人脸识别算法初始化函数

用于初始化人脸识别算法模块的函数，其应先于 5.3.3～5.3.9 调用。

函数原型：int FR_Initialize(SDKINFO sdk_info_input, char configuration_dir[255], void * custom_input, int custom_len_input)；

FR_Initialize 调用参数如表 28 所示。

表 28 FR_Initialize 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
configuration_dir	字符型数组	输入参数	配置文件路径
custom_input	无符号指针	输入参数	自定义输入项
custom_len_input	32 位整型	输入参数	自定义输入项长度

5.3.4.3 人脸特征数据长度获取函数

用于获得人脸特征长度的函数。

函数原型：int FR_Getsize(SDKINFO sdk_info_input, int * template_size)；

FR_Getsize 调用参数如表 29 所示。

表 29 FR_Getsize 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
template_size	32 位整型指针	输出参数	人脸识别单个人脸特征最大长度

返回值：成功返回 0，否则返回错误代码。

5.3.4.4 人脸检测函数

用于在给定的图像中，判断其是否存在人脸。如果存在，确定人脸位置和大小，并输出人脸图像与位置坐标。

函数原型：int FR_Facedetect(SDKINFO sdk_info_input, MULTIIMAGE img_input, void * custom_input, int custom_len_input, MULTIROI * roi_output)；

FR_Facedetect 调用参数如表 30 所示。

表 30 FR_Facedetect 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
img_input	多图像数据类型	输入参数	多图像数据集，传递待定位的多人脸图像数据集
custom_input	无符号指针	输入参数	自定义输入内容

表 30（续）

参数	数据类型	函数调用类型	参数说明
custom_len_input	32 位整型	输入参数	自定义输入内容长度
roi_output	多人脸位置数据类型指针	输出参数	多人脸位置数据集，传递输入图像中人脸位置坐标数据

5.3.4.5 人脸关键点定位函数

用于确定人脸关键点位置的函数，包含两种形式

a） 独立人脸关键点定位函数，在人脸图像中确定关键点位置。

函数原型：int FR_location_A（SDKINFO sdk_info_input，MULTIIMAGE img_input，MULTIROI roi_input，void *custom_input，int custom_len_input，MULTIPOS *orgpos_output）；

FR_Location_A 调用参数如表 31 所示。

表 31 FR_Location_A 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
img_input	多图像数据类型	输入参数	多图像数据集，传递待定位的图像数据
roi_input	多人脸位置数据类型	输入参数	多人脸位置数据集，传递输入图像中人脸位置坐标数据
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
orgpos_output	多人脸关键点数据类型指针	输出参数	多人脸关键点数据集，传递输入图像中人脸关键点位置信息

b） 集成人脸关键点定位函数，在图像中确定人脸关键点的位置。

函数原型：int FR_Location_C（SDKINFO sdk_info_input，MULTIIMAGE img_input，void *custom_input，int custom_len_input，MULTIPOS *orgpos_output，void *custom_output，int *custom_len_output）；

FR_Location_C 调用参数如表 32 所示。

表 32 FR_Location_C 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
img_input	多图像数据类型	输入参数	多图像数据集，传递待定位的图像数据
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度

表 32（续）

参数	数据类型	函数调用类型	参数说明
orgpos_output	多人脸关键点数据类型指针	输出参数	多人脸关键点数据集，传递输入图像中人脸关键点位置信息
custom_output	无符号指针	输出参数	自定义输出项
custom_len_output	32 位整型指针	输出参数	自定义输出内容长度

5.3.4.6 人脸归一化函数

进行人脸归一化的函数。包含两种形式

a） 独立人脸归一化的函数，用于对输入人脸图像进行处理，形成最适于进行人脸识别的人脸图像。

函数原型：int FR_Normalize_A(SDKINFO sdk_info_input，MULTIIMAGE img_input，MULTIPOS orgpos_input，void * custom_input，int custom_len_input，MULTIIMAGE * normal_img_output，MULTIPOS * normal_pos_output)；

FR_Normalize_A 调用参数如表 33 所示。

表 33 FR_Normalize_A 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
img_input	多图像数据类型	输入参数	多图像数据集，传递输入的多图像数据集
orgpos_input	多人脸关键点数据类型	输入参数	多人脸关键点数据集，传递输入图像对应的人脸关键点
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
normal_img_output	多图像数据类型指针	输出参数	多图像数据集，传递归一化图像
normal_pos_output	多人脸关键点数据类型指针	输出参数	多人脸关键点数据集，传递归一化图像对应人脸关键点位置信息

b） 集成人脸归一化函数，用于对输入人脸图像进行处理，形成最适于进行人脸识别的人脸图像。

函数原型：int FR_Normalize_C(SDKINFO sdk_info_input，MULTIIMAGE img_input，void * custom_input，int custom_len_input，MULTIIMAGE * normal_img_output，MULTIPOS * normal_pos_output，void * custom_output，int * custom_len_output)；

FR_Normalize_C 调用参数如表 34 所示。

表 34 FR_Normalize_C 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
img_input	多图像数据类型	输入参数	多图像数据集,传递待归一化的图像数据集
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
normal_img_output	多图像数据类型指针	输出参数	多图像数据集,传递归一化图像
normal_pos_output	多人脸关键点数据类型指针	输出参数	多人脸关键点数据集,传递归一化图像中人脸关键点位置信息
custom_output	无符号指针	输出参数	自定义输出项
custom_len_output	32 位整型指针	输出参数	自定义输出内容长度

5.3.4.7 人脸模板提取函数

用于从图像中提取人脸模板。包含两种形式。

a) 独立人脸模板提取函数,从人脸几何尺寸归一化图像中提取人脸模板。

函数原型:int FR_Extract_A(SDKINFO sdk_info_input, MULTIIMAGE normal_img_input, MULTIPOS normal_pos_input, void * custom_input, int custom_len_input, FACETEMPLATE * template_output)。

FR_Extract_A 调用参数如表 35 所示。

表 35 FR_Extract_A 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
normal_img_input	多图像数据类型	输入参数	多图像数据集,传递归一化的多人脸图像数据集
normal_pos_input	多人脸关键点数据类型	输入参数	多人脸关键点数据集,传递归一化图像对应的人脸关键点位置
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
template_output	人脸模板数据类型指针	输出参数	模板,传递人脸模板

b) 集成人脸模板提取函数,从图像中提取人脸模板。

函数原型:int FR_Extract_C(SDKINFO sdk_info_input, MULTIIMAGE Img_input, void * custom_input, int custom_len_input, FACETEMPLATE * template_output, void * custom_output, int * custom_len_output);

FR_Extract_C 调用参数如表 36 所示。

表 36　FR_Extract_C 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
img_input	多图像数据类型	输入参数	多图像数据集，传递待提取人脸模板的多图像数据集
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入项长度
template_output	人脸模板数据类型指针	输出参数	人脸模板数据，传递人脸模板
custom_output	无符号指针	输出参数	自定义输出项
custom_len_output	32 位整型指针	输出参数	自定义输出内容长度

5.3.4.8　人脸相似度计算函数

用于计算人脸相似度的函数，支持一对一、一对多、多对多 3 种方式。包含 3 种形式。

a)　人脸相似度计算函数

函数原型：int FR_Match(SDKINFO sdk_info_input, MULTIIMAGE * probe_img_input, long long multiimg_num, MULTIIMAGE * gallery_img_input, long long multiimg_num, void * custom_input, int custom_len_input, MULTISIM * sim_output, long long sim_num_output_output, void * custom_output, int * custom_len_output)；

FR_Match 调用参数如表 37 所示。

表 37　FR_Match 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
probe_img_input	多图像数据类型	输入参数	多图像数据集，传递待计算的图像
multiimg_num	64 位整型	输入参数	多图像数据集数量
gallery_img_input	多图像数据类型	输入参数	多图像数据集，传递目标的图像
multiimg_num	64 位整型	输入参数	多图像数据集数量
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
sim_output	多人脸相似度数据类型指针	输出参数	多人脸相似度数据，传递计算结果，以降序排列
sim_num_output	64 位整型	输入参数	多人脸相似度数据数量
custom_output	无符号指针	输出参数	自定义输出项
custom_len_output	32 位整型指针	输出参数	自定义输出内容长度

b） 独立人脸相似度计算函数。

函数原型：int FR_Match_A(SDKINFO sdk_info_input, FACETEMPLATE * probe_template_input, long long face_num, FACETEMPLATE * gallery_template_input, long long target_num, void * custom_input, int custom_len_input, MULTISIM * sim_output, long long * sim_num_output);

FR_Match_A 调用参数如表 38 所示。

表 38 FR_Match_A 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
probe_template_input	人脸模板数据类型指针	输入参数	人脸模板数据，传递待计算人脸模板
face_num	64 位整型	输入参数	待计算人脸模板数量
gallery_template_input	人脸模板数据类型指针	输入参数	目标人脸模板数据，传递目标人脸模板
target_num	64 位整型	输入参数	目标人脸模板数量
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
sim_output	多人脸相似度数据类型指针	输出参数	多人脸相似度数据，传递计算结果，每个多人脸相似度数据内以降序排列
sim_num_output	64 位整型指针	输出参数	多人脸相似度数据数量

c） 集成人脸相似度计算函数。

函数原型：int FR_Match_C(SDKINFO sdk_info_input, MULTIIMAGE * probe_img_input, long long multiimg_num, FACETEMPLATE * gallery_template_input, long long target_num, void * custom_input, int custom_len_input, MULTISIM * sim_output, long long sim_num_output, void * custom_output, int * custom_len_output);

FR_Match_C 调用参数如表 39 所示。

否则返回错误代码。

表 39 FR_Match_C 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据	输入参数	API 类型
probe_img_input	多图像数据类型指针	输入参数	多图像数据集，传递待计算的图像
multiimg_num	64 位整型	输入参数	多图像数据集数量
gallery_template_input	人脸模板数据类型指针	输入参数	目标人脸模板数据集合，传递目标人脸模板

表 39（续）

参数	数据类型	函数调用类型	参数说明
target_num	64 位整型	输入参数	目标人脸模板数量
custom_input	无符号指针	输入参数	自定义输入内容
custom_len_input	32 位整型	输入参数	自定义输入内容长度
sim_output	多人脸相似度数据类型指针	输出参数	多人脸相似度数据，传递计算结果，以降序排列
sim_num_output	64 位整型	输入参数	多人脸相似度数据数量
custom_output	无符号指针	输出参数	自定义输出项
custom_len_output	32 位整型指针	输出参数	自定义输出内容长度

5.3.4.9 人脸识别算法释放函数

用于释放人脸识别模块所占用资源的函数。

函数原型：int FR_Release(SDKINFO sdk_info_input)；

FR_Release 调用参数如表 40 所示。

表 40 FR_Release 调用参数

参数	数据类型	函数调用类型	参数说明
sdk_info_input	算法模块版本信息数据指针	输入参数	待释放的算法模块版本信息

5.4 人脸识别应用服务接口

5.4.1 人脸识别应用服务接口描述

人脸识别应用服务通过 REST 架构风格的 Web 服务方式对外提供相关功能，所涉及的数据结构采用 XML 格式进行封装，全局结构见附录 C。返回值代码见附录 A。

5.4.2 服务资源模型表述

系统的协议资源模型采用属性层次结构表示。系统的顶级节点定义为 FR。资源模型如图 4 所示。

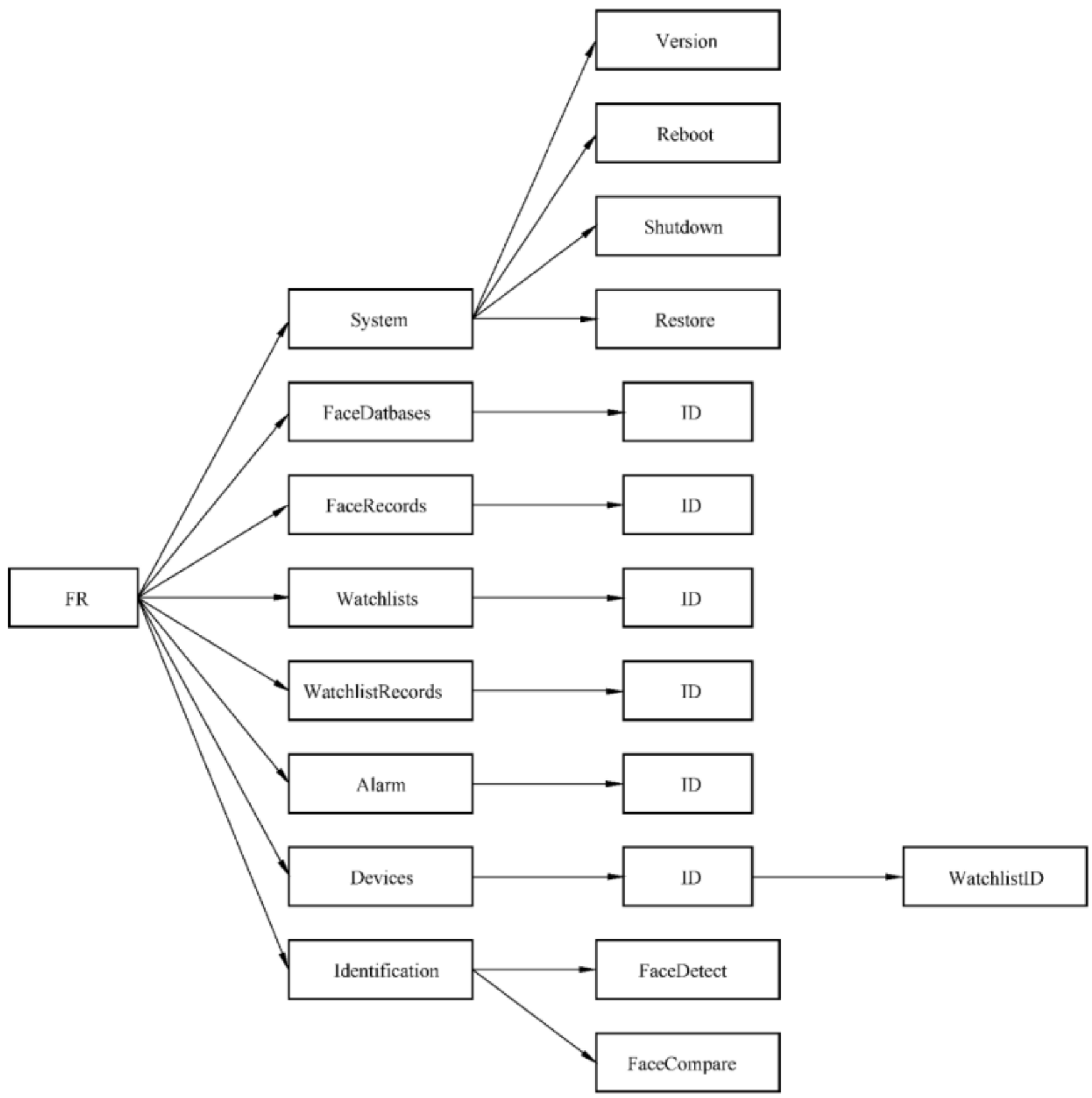

图 4　资源模型图

人脸识别应用服务可提供的数据服务接口列表如表 41 所示。

表 41　接口列表

资源 URI	定义	方法			
		Put	Get	Post	Delete
/FR	顶级资源名				
/FR/System	系统相关信息				
/FR/System/Version	版本		√		
/FR/System/Reboot	重启服务			√	
/FR/System/Shutdown	关闭服务			√	
/FR/System/Restore	恢复默认值			√	

表 41（续）

资源 URI	定义	方法			
		Put	Get	Post	Delete
/FR/Identification	人脸识别				
/FR/Identification/FaceDetect	人脸检测			√	
/FR/Identification/FaceCompare	人脸比对			√	
/FR/FaceDatabases	人脸数据库		√		
/FR/FaceDatabases/〈ID〉	人脸数据库相关资源	√		√	√
/FR/FaceRecords	人脸记录		√		
/FR/FaceRecords/〈ID〉	人脸记录相关资源	√	√	√	√
/FR/Watchlists	关注名单		√		
/FR/Watchlists/〈ID〉	关注名单相关资源	√	√	√	√
/FR/Watchlists/〈ID〉/〈ReocrdID〉	关注名单记录相关资源			√	√
/FR/Alarm	关注名单命中		√		
/FR/Alarm/〈ID〉	关注名单命中相关资源				√
/FR/Devices	采集设备		√		
/FR/Devices/〈ID〉	采集设备相关资源		√		
/FR/Devices/〈ID〉/〈WatchlistID〉	采集设备关联的关注名单			√	√

描述数据的资源定义如表 42 所示。

表 42　资源定义

资源 URI	定义	章条号
/FR/System/Version	获取版本信息	5.4.3.1
/FR/Identification/FaceDetect	人脸检测	5.4.3.2
/FR/Identification/FaceCompare	人脸相似度计算	5.4.3.3
/FR/FaceDatabases	获取人脸数据库列表	5.4.3.4
/FR/FaceDatabases/〈ID〉	添加/修改人脸数据库	5.4.3.5
/FR/FaceDatabases/〈ID〉	删除人脸数据库	5.4.3.6
/FR/FaceRecords	获取人脸记录列表	5.4.3.7
/FR/FaceRecords/〈ID〉	添加/修改人脸记录	5.4.3.8
/FR/FaceRecords/〈ID〉	删除人脸记录	5.4.3.9

表 42（续）

资源 URI	定义	章条号
/FR/Identification/FaceCompare	辨认型人脸识别	5.4.3.10
/FR/Watchlist	获取关注名单列表	5.4.3.11
/FR/Watchlist/〈ID〉	添加/修改关注名单	5.4.3.12
/FR/Watchlist/〈ID〉	删除关注名单	5.4.3.13
/FR/Watchlist/〈ID〉	获取关注名单记录	5.4.3.14
/FR/Watchlist/〈ID〉/〈RecordID〉	添加/修改关注名单记录	5.4.3.15
/FR/Watchlist/〈ID〉/〈RecordID〉	删除关注名单记录	5.4.3.16
/FR/Devices	获得人脸采集设备列表	5.4.3.17
/FR/Device/〈ID〉/〈WatchlistID〉	添加关注名单关联设备	5.4.3.18
/FR/Device/〈ID〉/〈WatchlistID〉	删除关注名单关联设备	5.4.3.19
/FR/Alarm	获取关注名单命中列表	5.4.3.20
/FR/Alarm/〈ID〉	删除关注名单命中记录	5.4.3.21
/FR/System/Reboot	人脸识别应用服务重启	5.4.3.22
/FR/System/Shutdown	关闭人脸识别应用服务	5.4.3.23
/FR/System/Restore	人脸识别应用服务恢复默认设置	5.4.3.24

5.4.3 接口定义

5.4.3.1 获取版本信息

用于获取人脸识别应用服务版本信息。

获取版本信息方法如表 43 所示。

表 43 获取版本信息方法

URI	/FR/System/Version		
功能	获取版本信息		
方法	查询字符串	传递数据	返回结果
GET	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素 〈ServiceInfo〉
注释			

5.4.3.2 人脸检测

用于在给定的图像中，判断其是否存在人脸。如果存在，确定人脸位置和大小，并输出人脸图像与位置坐标。

人脸检测方法如表 44 所示。

表 44　人脸检测方法

URI	/FR/Identification/FaceDetect		
功能	用于在给定的图像中，判断其是否存在人脸，并确定人脸位置与大小		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素 〈Image〉	根元素〈Response〉 子元素 〈 FaceInfoList〉
注释			

5.4.3.3　人脸相似度计算

用于计算两张输入图像中的人脸相似度。

人脸相似度计算方法如表 45 所示。

表 45　人脸相似度计算方法

URI	/FR/Identification/FaceCompare		
功能	用于计算两张输入图像中的人脸相似度		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素 〈Image〉 〈Image〉	根元素〈Response〉 子元素 〈CopmareResult〉
注释			

5.4.3.4　获取人脸数据库列表

用于获取人脸数据库列表。

获取人脸数据库列表方法如表 46 所示。

表 46　获取人脸数据库列表方法

URI	/FR/FaceDatabases		
功能	获取人脸数据库列表		
方法	查询字符串	传递数据	返回结果
GET	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素 〈 FaceDBList〉
注释			

5.4.3.5　添加/修改人脸数据库

用于添加/修改人脸数据库。

添加/修改人脸数据库方法如表 47 所示。

表 47　添加/修改人脸数据库方法

URI	/FR/FaceDatabases/〈ID〉		
功能	用于添加/修改人脸数据库		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素 〈FaceInfo〉	根元素〈Response〉 子元素
PUT	N/A	根元素〈Request〉 子元素 〈FaceInfo〉	根元素〈Response〉 子元素
注释			

5.4.3.6　删除人脸数据库

用于删除人脸数据库。

人脸数据库删除方法如表 48 所示。

表 48　删除人脸数据库方法

URI	/FR/FaceDatabases/〈ID〉		
功能	用于删除人脸数据库		
方法	查询字符串	传递数据	返回结果
DELETE	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.7　获取人脸记录列表

用于获取人脸数据库中的人脸记录数据。

获取人脸记录列表方法如表 49 所示。

表 49　获取人脸记录列表方法

URI	/FR/FaceRecords		
功能	获取人脸数据库中的人脸记录数据		
方法	查询字符串	传递数据	返回结果
GET		根元素〈Request〉 子元素 〈RecordInquireType〉 〈SingleInquireCond〉 〈BatchInquireCond〉	根元素〈Response〉 子元素 〈FaceRecordList〉
注释			

5.4.3.8 添加/修改人脸记录

用于添加/修改人脸数据库中的人脸记录。

添加/修改人脸记录方法如表 50 所示。

表 50 添加/修改人脸记录

URI	/FR/FaceRecords/〈ID〉		
功能	用于添加/修改人脸数据库中的人脸记录		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素 〈FaceRecord〉	根元素〈Response〉 子元素
PUT	N/A	根元素〈Request〉 子元素 〈FaceRecord〉	根元素〈Response〉 子元素
注释			

5.4.3.9 删除人脸记录

用于删除人脸数据库中的人脸记录。

删除人脸记录方法如表 51 所示。

表 51 删除人脸记录方法

URI	/FR/FaceRecords/〈ID〉		
功能	用于删除人脸数据库中的人脸记录		
方法	查询字符串	传递数据	返回结果
DELETE	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.10 辨认型人脸识别

用于辨认型人脸识别。

辨认型人脸识别方法如表 52 所示。

表 52　辨认型人脸识别方法

URI	/FR/Identification/FaceCompare		
功能	查询数据库中的人脸，与输入的人脸进行比对查询最相似的人		
方法	查询字符串	传递数据	返回结果
POST		根元素〈Request〉 子元素 〈IdentificationCond〉	根元素〈Response〉 子元素 〈IdentificationResultList〉
注释			

5.4.3.11　获取关注名单列表

用于获得关注名单列表。

获取关注名单列表方法如表 53 所示。

表 53　获取关注名单列表方法

URI	/FR/Watchlist		
功能	获得关注名单列表		
方法	查询字符串	传递数据	返回结果
GET	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素 〈WatchLists〉
注释			

5.4.3.12　添加/修改关注名单

用于添加/修改关注名单。

添加/修改关注名单方法如表 54 所示。

表 54　添加/修改关注名单方法

URI	/FR/Watchlist/〈ID〉		
功能	添加/修改和删除关注名单		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素 〈Watchlist〉	根元素〈Response〉 子元素
PUT	N/A	根元素〈Request〉 子元素 〈Watchlist〉	根元素〈Response〉 子元素
注释			

5.4.3.13 删除关注名单

用于删除关注名单。

删除关注名单方法如表 55 所示。

表 55 删除关注名单方法

URI	/FR/Watchlist/〈ID〉		
功能	删除关注名单		
方法	查询字符串	传递数据	返回结果
DELETE	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.14 获取关注名单记录

用于获取关注名单记录。

获取关注名单记录方法如表 56 所示。

表 56 获取关注名单记录方法

URI	/FR/Watchlist/〈ID〉		
功能	获取关注名单记录		
方法	查询字符串	传递数据	返回结果
GET		根元素〈Request〉 子元素 〈WatchListInquireCond〉	根元素〈Response〉 子元素 〈WatchListRecordList〉
注释			

5.4.3.15 添加/修改关注名单记录

用于添加/修改关注名单内人脸记录。

添加/修改关注名单记录方法如表 57 所示。

表 57 添加/修改关注名单记录方法

URI	/FR/Watchlist/〈ID〉/〈RecordID〉		
功能	添加/修改关注名单记录		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素 〈WatchListRecord〉	根元素〈Response〉 子元素
注释			

5.4.3.16 删除关注名单记录

用于删除关注名单内人脸记录。

删除关注名单记录方法如表 58 所示。

表 58 删除关注名单记录方法

URI	/FR/Watchlist/〈ID〉/〈RecordID〉		
功能	删除关注名单记录		
方法	查询字符串	传递数据	返回结果
DELETE	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.17 获得人脸采集设备列表

用于获取人脸采集设备列表。

获取人脸采集设备列表方法如表 59 所示。

表 59 获取人脸采集设备列表

URI	/FR/Devices		
功能	获取人脸采集设备列表		
方法	查询字符串	传递数据	返回结果
GET	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素 〈DeviceList〉
注释			

5.4.3.18 添加/修改关注名单关联设备

用于添加/修改关注名单关联设备。

添加/修改关注名单关联设备方法如表 60 所示。

表 60 添加/修改关注名单关联设备方法

URI	/FR/Device/〈ID〉/〈WatchlistID〉		
功能	添加/修改关注名单关联设备		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素 〈RelateInfo〉	根元素〈Response〉 子元素

5.4.3.19 删除关注名单关联设备

删除设置关注名单关联设备。

删除关注名单关联设备方法如表 61 所示。

表 61 删除关注名单关联设备方法

URI	/FR/Device/〈ID〉/〈WatchlistID〉		
功能	删除关注名单关联设备		
方法	查询字符串	传递数据	返回结果
DELETE	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.20 获取关注名单命中列表

用于获取关注名单命中列表。

获取关注名单命中列表方法如表 62 所示。

表 62 获取关注名单命中列表方法

URI	/FR/Alarm		
功能	获取关注名单命中列表		
方法	查询字符串	传递数据	返回结果
GET		根元素〈Request〉 子元素 〈AlarmInquireCond〉	根元素〈Response〉 子元素 〈 AlarmList〉
注释			

5.4.3.21 删除关注名单命中记录

用于删除监控名单命中记录。

删除关注名单命中记录方法如表 63 所示。

表 63 删除关注名单命中记录方法

URI	/FR/Alarm/〈ID〉		
功能	删除关注名单命中记录		
方法	查询字符串	传递数据	返回结果
DELETE		根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.22 人脸识别应用服务重启

用于重启脸识别应用服务。

人脸识别应用服务重启方法如表 64 所示。

表 64 人脸识别应用服务重启方法

URI	/FR/System/Reboot		
功能	重启人脸识别应用服务		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.23 关闭人脸识别应用服务

用于关闭人脸识别应用服务。

关闭人脸识别应用服务方法如表 65 所示。

表 65 关闭人脸识别应用服务方法

URI	/FR/System/Shutdown		
功能	关闭人脸识别应用服务		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

5.4.3.24 人脸识别应用服务恢复默认设置

用于恢复人脸识别应用服务默认设置。

人脸识别应用服务恢复默认设置方法如表 66 所示。

表 66 人脸识别应用服务恢复默认设置方法

URI	/FR/System/Restore		
功能	恢复人脸识别应用服务默认设置		
方法	查询字符串	传递数据	返回结果
POST	N/A	根元素〈Request〉 子元素	根元素〈Response〉 子元素
注释			

6 接口安全策略要求

接口安全策略要求如下：

a) 本标准涉及之内部功能模块的各种接口和对外服务接口应保证其接入与传输的安全性和数据的完整性；

b) 本标准涉及接口在不同应用场景和链接方式中，应制定相对应的接口安全策略，保证数据的兼容性以及传输与交换的安全性和数据的完整性；

c) 接口安全策略宜采用 SSOIS 安全策略，在逻辑上应至少包含：系统安全评估、系统访问控制、系统入侵检测、系统口令认证、系统安全审计、防范恶意代码、系统加密等内容；

d) 本标准中接口涉及之系统的运行日志应在 SSOIS 安全策略的管理下做到对系统安全事件的“可知、可控、可预测”；

e) 本标准中接口涉及之系统的网络边界接入点应实施安全策略控制。

附　录　A
（规范性附录）
接口返回值代码

接口返回值定义见表 A.1。

表 A.1　接口返回值定义

错误代码数值	宏定义	说明
0	SUCCES	成功
1	FR_INDENTIFY_FAIL	人脸确认失败
2	FR_NO_ALARM	待测人脸样本不在关注名单上
3	FR_NO_FACE	没有检测出人脸
−1	NO_PERMISSION	没有使用权限
−2	FR_APITYPE_ERROR	API 类型错误
−3	FR_UNKNOWN_ERROR	未知错误
−4	FR_NO_INITIALIZE	没有对人脸识别模块进行初始化
−5	FR_CONFIGURATION FILE_ERROR	配置文件错误
−6	FR_GET_SDKVISION_ERROR	获取版本信息失败
−7	FR_GET_TEMPLATE_SIZE_ERROR	获取模板长度失败
−8	FR_INITIALIZE_ERROR	初始化失败
−9	FR_RELEASE_ERROR	资源释放失败
−10	FR_LOCATION_ERROR	定位失败
−11	FR_NORMALIZE_ERROR	人脸归一化失败
−12	FR_EXTRACT_ERROR	模板提取失败
−13	FR_MATCH_ERROR	相似度计算失败
−14	DEC_PASSWORD_ERROR	用户名密码错误。注册时输入的用户名或者密码错误
−15	DEC_CHANNEL_ERROR	通道号错误。设备没有对应的通道号
−16	DEC_OVER_MAXLINK	设备总的连接数超过最大
−17	DEC_VERSIONNOMATCH	SDK 版本不匹配
−18	DEC_NETWORK_FAIL_CONNECT	连接设备失败。设备不在线或网络原因引起的连接超时等
−19	DEC_NETWORK_SEND_ERROR	向设备发送失败
−20	DEC_NETWORK_RECV_ERROR	从设备接收数据失败
−21	DEC_NETWORK_RECV_TIMEOUT	从设备接收数据超时
−22	DEC_NETWORK_ERRORDATA	传送的数据有误。发送给设备或者从设备接收到的数据错误，如远程参数配置时输入设备不支持的值

表 A.1（续）

错误代码数值	宏定义	说明
－23	DEC_ORDER_ERROR	调用次序错误
－24	DEC_COMMANDTIMEOUT	设备命令执行超时
－25	DEC_PARAMETER_ERROR	参数错误。SDK 接口中给入的输入或输出参数为空，或者参数格式或值不符合要求
－26	DEC_CHAN_EXCEPTION	设备通道处于错误状态
－27	DEC_NOSUPPORT	设备不支持
－28	DEC_BUSY	设备忙
－29	DEC_PASSWORD_FORMAT_ERROR	密码输入格式不正确
－30	DEC_DVRNORESOURCE	设备资源不足
－31	DEC_DVROPRATEFAILED	设备操作失败
－32	DEC_GETPLAYTIMEFAIL	获取当前播放的时间出错
－33	DEC_PLAYFAIL	播放出错
－34	DEC_ALLOC_RESOURCE_ERROR	SDK 资源分配错误
－35	DEC_NOENOUGH_BUF	缓冲区太小。接收设备数据的缓冲区或存放图片缓冲区不足
－36	DEC_CREATESOCKET_ERROR	创建 SOCKET 出错
－37	DEC_SETSOCKET_ERROR	设置 SOCKET 出错
－38	DEC_MAX_NUM	个数达到最大。分配的注册连接数、预览连接数超过 SDK 支持的最大数
－39	DEC_USERNOTEXIST	用户不存在。注册的用户 ID 已注销或不可用
－40	DEC_MAX_USERNUM	登录设备的用户数达到最大
－41	DEC_NOENCODEING	设备该通道没有启动编码
－42	DEC_MAX_PLAYERPORT	播放器路数达到最大
－43	DEC_CREATEDIR_ERROR	建立日志文件目录失败
－44	DEC_BINDSOCKET_ERROR	绑定套接字失败
－45	DEC_SOCKETCLOSE_ERROR	socket 连接中断，此错误通常是由于连接中断或目的地不可达
－46	DEC_USERID_ISUSING	注销时用户 ID 正在进行某操作
－47	DEC_SOCKETLISTEN_ERROR	监听失败
－48	DEC_PROGRAM_EXCEPTION	程序异常
－49	WS_DATABASE_ERROR	数据库操作失败
－50	WS_RESOLUTION_UNSUPPORT	分辨率不支持
－51	WS_SUBWS_EXIST	子服务重复添加
－52	WS_IMAGE_ENCODE_ERROR	图片编码失败

表 A.1（续）

错误代码数值	宏定义	说明
−53	WS_IMAGE_DECODE_ERROR	图片解码失败
−54	WS_OUT_OF_MEM	内存不足
−55	WS_NO_RESULT	无结果
−56	WS_NO_RESOURCE	达到系统资源上限
−57	WS_DEVICE_EXIST	设备重复添加
−58	WS_NO_DATA	无数据
−59	WS_SUBWS_OFFLINE	子服务不在线
−60	WS_SOCKET_TIMEOUT	网络超时
−61	WS_TOO_MUCH_CONCURRENCY	并发数过多
−62	WS_WATCHLIST_UNLOADED	名单分组未加载
−63	WS_BUSY	服务器正忙
−64	WS_STORAGE_NO_SPACE	存储空间不足

附　录　B
（规范性附录）
动态链接库文件名称

动态链接库文件名称见表 B.1。

表 B.1　动态链接库文件名称

接口	Windows 平台	Linux/Unix 平台
人脸采集设备接口	FRD.dll	FRD.so
人脸识别算法模块版本信息获取接口	FR_PUBLIC.dll	FR_PUBLIC.so
人脸识别初始化接口	同上	同上
人脸模板长度获取接口	同上	同上
人脸识别释放接口	同上	同上
人脸检测模块接口	FR_DETECT.dll	FR_DETECT.so
人脸定位模块接口	FR_LOCATE.dll	FR_LOCATE.so
人脸几何尺寸归一化接口	FR_NORMAL.dll	FR_NORMAL.so
人脸模板提取接口	FR_EXTRACT.dll	FR_EXTRACT.so
人脸相似度计算接口	FR_MATCH.dll	FR_MATCH.so

附　录　C
（规范性附录）
人脸识别应用服务结构

人脸识别应用服务全局结构：

```
〈！--基础结构--〉

〈！-- IP 协议类型--〉
〈simpleType name="IPType"〉
〈restriction base="string"〉
〈！--ipv4--〉
〈enumeration value="ipv4" /〉
〈！--ipv6--〉
〈enumeration value="ipv6" /〉
〈/restriction〉
〈/simpleType〉

〈！— 图片--〉
〈complexType name="Image"〉
〈sequence〉
〈！—图片--〉
〈element name="imageData" type="base64Binary" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈！—人脸比对结果--〉
〈complexType name=" CopmareResult"〉
〈sequence〉
〈！-- 相似度--〉
〈element name="similairy" type="float" minOccurs="1" maxOccurs="1"/〉
〈！-- 同一人可能性--〉
〈element name="probability" type="float" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈！-- 人脸采集设备信息--〉
〈complexType name="DeviceInfo"〉
〈sequence〉
〈！-- 人脸采集设备编号--〉
〈element name="id" type="int" minOccurs="1" maxOccurs="1" /〉
〈！-- 名称--〉
〈element name="name" type="string" minOccurs="1" maxOccurs="1" /〉
```

```
〈! -- 地址类型--〉
〈element name="ipType" type="IPType" minOccurs="1" maxOccurs="1" /〉
〈! -- 地址--〉
〈element name="ip" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 端口--〉
〈element name="port" type="short" minOccurs="1" maxOccurs="1" /〉
〈! -- 设备类型,0 usb, 1 webcam 2 模拟--〉
〈element name="device_type" type="short" minOccurs="1" maxOccurs="1" /〉
〈! -- 设备支持人脸工作模式--〉
〈element name="workMode" type="long" minOccurs="1" maxOccurs="1" /〉
〈! -- 设备序列号--〉
〈element name="serialnumber" type="string" minOccurs="1" maxOccurs="1" /〉
〈! --设备通道个数--〉
〈element name="channel_num" type="long" minOccurs="1" maxOccurs="1" /〉
〈! -- 开发商信息--〉
〈element name="developerInfo" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 版本号--〉
〈element name="version" type="string" minOccurs="1" maxOccurs="1" /〉
〈! --描述--〉
〈element name="description" type="string" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 人脸采集设备列表--〉
〈complexType name="DeviceList"〉
〈sequence〉
〈! -- 人脸采集设备--〉
〈element name="devices" type="DeviceInfo" minOccurs="1" maxOccurs ="unbounded" /〉
〈/sequence〉
〈/complexType〉

〈! --人脸识别应用服务信息结构--〉
〈complexType name="ServiceInfo"〉
〈sequence〉
〈! -- 服务名称--〉
〈element name="name" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 开发商信息--〉
〈element name="developerInfo" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 版本号--〉
〈element name="version" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 备注信息--〉
〈element name="note" type="string" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
```

```
</complexType>

<! -- 人员基本属性-->
<complexType name="HumanAttribute">
<sequence>
<! -- 姓名-->
<element name="xm" type="string" minOccurs="1" maxOccurs="1" />
<! -- 性别-->
<element name="xbdm" type="string" minOccurs="1" maxOccurs="1" />
<! -- 生日-->
<element name="csrq" type="string" minOccurs="1" maxOccurs="1" />
<! -- 公民身份号码-->
<element name="gmsfhm" type="string" minOccurs="1" maxOccurs="1" />
<! — 籍贯国家代码-->
<element name="jggjdqdm" type="string" minOccurs="1" maxOccurs="1" />
<! — 籍贯省市县代码-->
<element name="jgssxdm" type="string" minOccurs="1" maxOccurs="1" />
</sequence>
</complexType>

<! -- 人脸区域-->
<complexType name="ROI">
<sequence>
<! -- 人脸区域左上角坐标横坐标-->
<element name="xleft" type="int" minOccurs="1" maxOccurs="1" />
<! -- 人脸区域左上角坐标纵坐标-->
<element name="yleft" type="int" minOccurs="1" maxOccurs="1" />
<! -- 人脸区域右下角坐标横坐标-->
<element name="xright" type="int" minOccurs="1" maxOccurs="1" />
<! -- 人脸区域右下角坐标纵坐标-->
<element name="yright" type="int" minOccurs="1" maxOccurs="1" />
</sequence>
</complexType>

<! -- 人脸信息-->
<complexType name="FaceInfo">
<sequence>
<! -- 人脸编号-->
<element name="id" type="long" minOccurs="1" maxOccurs="1" />
<! -- 人脸区域-->
<element name="faceROI" type="ROI" minOccurs="1" maxOccurs="1" />
<! -- 人脸质量-->
<element name="quality" type="float" minOccurs="1" maxOccurs="1" />
```

```
〈! -- 图片数据--〉
〈element name="imageData" type="base64Binary" minOccurs="0" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 人脸信息列表--〉
〈complexType name="FaceInfoList"〉
〈sequence〉
〈! —人脸信息--〉
〈element name="faceList" type="FaceInfo" minOccurs="0" maxOccurs="unbounded" /〉
〈/sequence〉
〈/complexType〉

〈! -- 图片类型--〉
〈simpleType name="ImageType"〉
〈restriction base="string"〉
〈! -- 无图片--〉
〈enumeration value="none" /〉
〈! -- 人脸图像--〉
〈enumeration value="face" /〉
〈! -- 图像--〉
〈enumeration value="full" /〉
〈/restriction〉
〈/simpleType〉

〈! -- 人脸数据库--〉
〈complexType name="FaceDB"〉
〈sequence〉
〈! -- 人脸数据库编号--〉
〈element name="id" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 人脸数据库名字--〉
〈element name="name" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 人脸数据库备注--〉
〈element name="note" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 人脸数据库总记录数--〉
〈element name="recordsCount" type="long" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 人脸数据库列表--〉
〈complexType name="FaceDBList"〉
〈sequence〉
〈! -- 人脸数据库--〉
```

```
〈element name="faceDBs" type="FaceDB" minOccurs="1" maxOccurs="unbounded" /〉
〈/sequence〉
〈/complexType〉

〈! -- 记录查询类型--〉
〈simpleType name="RecordInquireType"〉
〈restriction base="string"〉
〈! -- 批量--〉
〈enumeration value="batch" /〉
〈! -- 单个--〉
〈enumeration value="single" /〉
〈/restriction〉
〈/simpleType〉

〈! -- 批量查询条件--〉
〈complexType name="BatchInquireCond"〉
〈sequence〉
〈! -- 人脸数据库编号--〉
〈element name="databaseID" type="int" minOccurs="1" maxOccurs="1"/〉
〈! -- 人员基本属性--〉
〈element name="humanAttribute"type="HumanAttribute"minOccurs="1"maxOccurs="1"/〉
〈! -- 起始查询记录编号--〉
〈element name="startID" type="long" minOccurs="1" maxOccurs="1" /〉
〈! -- 最大查询记录数--〉
〈element name="maxRetNum" type="long" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 单个查询条件--〉
〈complexType name="SingleInquireCond"〉
〈sequence〉
〈! -- 人脸数据库编号--〉
〈element name="databaseID" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 人脸记录编号--〉
〈element name="faceID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 同一人对应不同模板编号--〉
〈element name="multiID" type="int" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 人脸记录--〉
〈complexType name="FaceRecord"〉
〈sequence〉
```

```
〈! -- 人脸记录编号--〉
〈element name="faceID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 同一人对应不同模板编号--〉
〈element name="multiID" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 所属的人脸数据库编号--〉
〈element name="databaseID" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 人员基本属性--〉
〈element name="humanAttribute"type="HumanAttribute"minOccurs="1"maxOccurs="1"/〉
〈! --人脸图像--〉
〈element name="faceImageData" type="base64Binary" minOccurs="1" maxOccurs="1" /〉
〈! --图像--〉
〈element name="bkgImageData" type="base64Binary" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 人脸记录列表--〉
〈complexType name="FaceRecordList"〉
〈sequence〉
〈! -- 人脸记录--〉
〈element name="faceRecords"type="FaceRecord"minOccurs="1"maxOccurs="unbounded"/〉
〈/sequence〉
〈/complexType〉

〈! --辨认型人脸识别条件--〉
〈complexType name="IdentificationCond"〉
〈sequence〉
〈! -- 识别类型--〉
〈element name="type" type="IdentificationType" minOccurs="1" maxOccurs="1"/〉
〈! -- 人脸采集设备 ID--〉
〈element name="DevID" type="int" minOccurs="0" maxOccurs="unbounded"/〉
〈! -- 人脸库 ID--〉
〈element name="databaseID" type="int" minOccurs="0" maxOccurs="unbounded"/〉
〈! -- 最小相似度--〉
〈element name="minSimilarity" type="float" minOccurs="1" maxOccurs="1"/〉
〈! -- 最大相似度--〉
〈element name="maxSimilarity" type="float" minOccurs="1" maxOccurs="1"/〉
〈! -- 人员基本属性--〉
〈element name="humanAttribute"type="HumanAttribute"minOccurs="1"maxOccurs="1"/〉
〈! -- 起始辨认人脸记录 ID--〉
〈element name="startID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 总共返回辨认结果数量--〉
〈element name="maxRetNum" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 图片数据--〉
```

```
〈element name="imageData" type="base64Binary" minOccurs="1" maxOccurs="1"/〉
〈! -- 是否返回图片--〉
〈element name="rspWithImage" type="boolean" minOccurs="1" maxOccurs="1"/〉
〈! --返回图片类型--〉
〈element name="imageType" type="ImageType" minOccurs="1" maxOccurs="1"/〉
〈/sequence〉
〈/complexType〉

〈! -- 辨认结果--〉
〈complexType name="IdentificationResult"〉
〈sequence〉
〈! -- 人脸记录-〉
〈element name="faceRecord" type="FaceRecord" minOccurs="1" maxOccurs="1" /〉
〈! -- 相似度--〉
〈element name="similairy" type="float" minOccurs="1" maxOccurs="1"/〉
〈! -- 同一人可能性--〉
〈element name="probability" type="float" minOccurs="1" maxOccurs="1" /〉
〈! -- 触发人脸采集设备编号--〉
〈element name="DevID" type="int" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 辨认结果列表--〉
〈complexType name="IdentificationResultList"〉
〈sequence〉
〈! -- 人脸记录-〉
〈element name="results"type="IdentificationResult"minOccurs="1"maxOccurs="unbounded"/〉
〈/sequence〉
〈/complexType〉

〈! -- 辨认型人脸识别类型--〉
〈simpleType name="IdentificationType"〉
〈restriction base="string"〉
〈! -- 根据人脸采集设备进行辨认--〉
〈enumeration value="byDev" /〉
〈! -- 根据人脸数据库进行辨认--〉
〈enumeration value="byDatabase" /〉
〈/restriction〉
〈/simpleType〉

〈! -- 关注名单类型--〉
〈simpleType name="WatchListType"〉
```

```
〈restriction base="integer"〉
〈! -- 黑名单--〉
〈enumeration value="1" /〉
〈! -- 白名单--〉
〈enumeration value="0" /〉
〈/restriction〉
〈/simpleType〉

〈! -- 关注名单信息--〉
〈complexType name="WatchList"〉
〈sequence〉
〈! -- 关注名单编号--〉
〈element name="id" type="int" minOccurs="1" maxOccurs="1"/〉
〈! -- 关注名单名称--〉
〈element name="name" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 名单记录数--〉
〈element name="num" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 备注--〉
〈element name="note" type="string" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 关注名单列表--〉
〈complexType name="WatchLists"〉
〈sequence〉
〈! -- 关注名单列表--〉
〈element name="watchLists"type="WatchList"minOccurs="1"maxOccurs="unbounded"/〉
〈/sequence〉
〈/complexType〉

〈! -- 关注名单记录--〉
〈complexType name="WatchListRecord"〉
〈sequence〉
〈! -- 人脸记录编号--〉
〈element name="faceID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 同一人对应不同模板编号--〉
〈element name="multiID" type="string" minOccurs="1" maxOccurs="1" /〉
〈! -- 关注名单编号--〉
〈element name="watchListID" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 人员基本属性--〉
〈element name="humanAttribute"type="HumanAttribute"minOccurs="1"maxOccurs="1"/〉
〈! -- 图片数据--〉
〈element name="imageData" type="base64Binary" minOccurs="1" maxOccurs="1" /〉
```

```
〈/sequence〉
〈/complexType〉

〈! -- 关注名单记录列表--〉
〈complexType name="WatchListRecordList"〉
〈sequence〉
〈! -- 关注名单记录--〉
〈element name="records"type="WatchListRecord"minOccurs="1"maxOccurs="unbounded"/〉
〈/sequence〉
〈/complexType〉

〈! -- 关注名单记录查询条件--〉
〈complexType name="WatchListInquireCond"〉
〈sequence〉
〈! —关注名单 ID--〉
〈element name="watchListID" type="int" minOccurs="1" maxOccurs="1"/〉
〈! -- 人员基本属性--〉
〈element name="humanAttribute"type="HumanAttribute"minOccurs="1"maxOccurs="1"/〉
〈! -- 起始 ID--〉
〈element name="startID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 查询最大记录数--〉
〈element name="maxRetNum" type="long" minOccurs="1" maxOccurs="1" /〉
〈! -- 是否返回图片--〉
〈element name="rspWithImage" type="boolean" minOccurs="1" maxOccurs="1"/〉
〈/sequence〉
〈/complexType〉

〈! --关注名单关联人脸采集设备信息--〉
〈complexType name="RelateInfo"〉
〈sequence〉
〈! -- 人脸采集设备编号--〉
〈element name="DevID" type="int" minOccurs="1" maxOccurs="1"/〉
〈! -- 关注名单编号--〉
〈element name="watchListID" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 关注名单类型--〉
〈element name="type" type=" WatchListType " minOccurs="1" maxOccurs="1" /〉
〈! -- 阈值--〉
〈element name="threshold" type="int" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! --关注名单命中查询类型--〉
```

```
〈simpleType name="AlarmInquireType"〉
〈restriction base="string"〉
〈! -- 根据人脸采集设备进行查询--〉
〈enumeration value="byDev" /〉
〈! -- 根据关注名单进行查询--〉
〈enumeration value="byWatchList" /〉
〈/restriction〉
〈/simpleType〉

〈! -- 排序方式--〉
〈simpleType name="SortType"〉
〈restriction base="string"〉
〈! -- 按时间升序排序--〉
〈enumeration value="byTimeAsc" /〉
〈! -- 按时间降序排序--〉
〈enumeration value="byTimeDesc" /〉
〈! -- 按相似度升序排序--〉
〈enumeration value="bySimilarityAsc" /〉
〈! -- 按相似度降序排序--〉
〈enumeration value="bySimilarityDesc" /〉
〈/restriction〉
〈/simpleType〉

〈! -- 命中信息--〉
〈complexType name="AlarmInfo"〉
〈sequence〉
〈! -- 命中编号--〉
〈element name="alarmID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 人脸记录编号--〉
〈element name="faceID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 同一人对应不同模板编号--〉
〈element name="multiID" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 关注名单编号--〉
〈element name="watchListID" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 相似度--〉
〈element name="similairy" type="float" minOccurs="1" maxOccurs="1"/〉
〈! -- 同一人可能性--〉
〈element name="probability" type="float" minOccurs="1" maxOccurs="1" /〉
〈! -- 人员基本属性--〉
〈element name="humanAttribute"type="HumanAttribute"minOccurs="1"maxOccurs="1"/〉
〈! -- 关注名单类型--〉
〈element name="type" type="WatchListType" minOccurs="1" maxOccurs="1" /〉
〈! -- 图片数据--〉
```

```
〈element name="imageData" type="base64Binary" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉

〈! -- 命中信息列表--〉
〈complexType name="AlarmList"〉
〈sequence〉
〈! -- 命中信息--〉
〈element name="alarms" type="AlarmInfo" minOccurs="1" maxOccurs="unbounded"/〉
〈/sequence〉
〈/complexType〉

〈! -- 命中查询条件--〉
〈complexType name="AlarmInquireCond"〉
〈sequence〉
〈! -- 命中查询类型--〉
〈element name="inquireType" type="AlarmInquireType" minOccurs="1" maxOccurs="1"/〉
〈! -- 人脸采集设备 ID--〉
〈element name="DevID" type="int" minOccurs="0" maxOccurs="unbounded"/〉
〈! —关注名单 ID--〉
〈element name="watchListID" type="int" minOccurs="0" maxOccurs="unbounded"/〉
〈! -- 起始时间--〉
〈element name="startTime" type="dateTime" minOccurs="1" maxOccurs="1"/〉
〈! -- 结束时间--〉
〈element name="endTime" type="dateTime" minOccurs="1" maxOccurs="1"/〉
〈! -- 最小相似度--〉
〈element name="minSimilarity" type="float" minOccurs="1" maxOccurs="1"/〉
〈! -- 最大相似度--〉
〈element name="maxSimilarity" type="float" minOccurs="1" maxOccurs="1"/〉
〈! -- 命中关注名单类型--〉
〈element name="alarmType" type="WatchListType" minOccurs="1" maxOccurs="1"/〉
〈! -- 返回结果排序类型--〉
〈element name="sortType" type="SortType" minOccurs="1" maxOccurs="1"/〉
〈! -- 起始 ID--〉
〈element name="startID" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 最大查询返回记录数--〉
〈element name="maxRetNum" type="long" minOccurs="1" maxOccurs="1"/〉
〈! -- 是否返回图片--〉
〈element name="rspWithImage" type="boolean" minOccurs="1" maxOccurs="1"/〉
〈/sequence〉
〈/complexType〉

〈! -- 错误信息--〉
```

```
〈complexType name="Response"〉
〈sequence〉
〈! -- 错误码--〉
〈element name="statusCode" type="int" minOccurs="1" maxOccurs="1" /〉
〈! -- 错误描述--〉
〈element name="statusString" type="string" minOccurs="1" maxOccurs="1" /〉
〈/sequence〉
〈/complexType〉
```

附 录 D
（资料性附录）
示 例 代 码

D.1 人脸采集设备接口示例代码

```
void CALLBACK preview_callback(long preview_handle, int data_type, unsigned char * buffer,
int buf_size, void * user)
{
    // 码流头
    if (data_type == 0)
    {
        // ...
    }
    // 码流数据
    else if (data_type == 1)
    {
        // ...
    }
    // 待扩展
    else
    {
    }
}

void CALLBACK frame_callback(long user_id, DEVICEINFO device_info, FACEINFO
faceinfo, void * user)
{
    // 设备信息
    printf(device_info.device_name);

    // 保存人脸图像
    FILE * fp = fopen("face.jpg", "wb+");
    fwrite(faceinfo.face_image, 1, faceinfo.face_length, fp);
    fclose(fp);

    // 保存图像
    fp = fopen("bkg.jpg", "wb+");
    fwrite(faceinfo.bkg_image, 1, faceinfo.bkg_length, fp);
    fclose(fp);
}
```

```
bool demo()
{
    // 初始化
    int ret = FR_Device_Init(NULL, 0);
    if (ret != SUCCES)
    {
        return false;
    }

    // 获取设备数
    int device_num = 0;
    FR_Device_DeviceList(NULL, &device_num);
    if (device_num == 0)
    {
        return false;
    }

    // 获取所有设备,需要在外部申请内存
    DEVICEINFO * device_list = new DEVICEINFO[device_num];
    ret = FR_Device_DeviceList(device_list, &device_num)
    if (ret != SUCCES)
    {
        return false;
    }

    // 设备连接
    long long user_id = 0;
    DEVICEINFO device_info = device_list[0];
    ret = FR_Device_Login(api_type, "username", "password", &device_info, NULL, 0,
&user_id);
    if (ret != SUCCES)
    {
        return false;
    }

    // 设置设备模式
    int work_mode = 0;
    ret = FR_Device_SetMode(user_id, work_mode, NULL, 0);
    if (ret != SUCCES)
    {
        return false;
    }
```

```
// 预览
PREVIEWINFO preview_info;
preview_info.channel        = 1;        // 通道 1
preview_info.stream_type    = 0;        // 主码流
preview_info.link_mode      = 1;        // tcp 连接方式
preview_info.play_wnd       = NULL;     // 播放窗口
preview_info.blocked        = 0;        // 非阻塞取流
preview_info.preview_mode = 0;          // 正常预览
preview_info.proto_type     = 0;        // rtsp 协议取流
preview_info.custom         = NULL;
preview_info.custom_len     =-1;

long long preview_hanle = NULL;
ret = FR_Device_Preview(user_id, preview_info, preview_callback, NULL, &preview_
hanle);
if (ret != SUCCES)
{
    return false;
}

// 停止预览
ret = FR_Device_StopPreview(preview_handle);
if (ret != SUCCES)
{
    return false;
}

// 开启图像主动上传
int upload_type = 1;     // 人脸图像上传
int upload_interval = 1;     // 同一目标上传 1 次
ret = FR_Device_SetFrameUpload(user_id, upload_type, upload_interval, frame_callback,
NULL);
if (ret != SUCCES)
{
    return false;
}

// 关闭图像主动上传
ret = FR_Device_SetFrameUpload(0, 0, NULL, 0);
if (ret != SUCCES)
{
    return false;
```

```
        }
        // 设备注销
        ret = FR_Device_Logout(user_id);
        if (ret != SUCCES)
        {
            return false;
        }
        FR_Device_Cleanup();
        return true;
    }
```

D.2 人脸识别算法模块接口示例代码

```
void demo()
{
// TODO：在此添加控件通知处理程序代码
int flag,info_num,custom_len_input,feature_len;
int api_type = 1;
void * custom_input = NULL;
custom_len_input = -1;
//获取 sdk 版本信息
SDKINFO *sdkinfo ;
flag = FR_Getinfo(api_type,NULL, &info_num);
if (flag)return;
sdkinfo = (SDKINFO *)malloc(info_num * sizeof(SDKINFO));
flag = FR_Getinfo(api_type,sdkinfo, &info_num);
if (flag)
{
    if(NULL != sdkinfo){free(sdkinfo);sdkinfo =NULL;}
    return;
}
//初始化人脸识别算法模块
flag = FR_Initialize( *sdkinfo, ".\configdir", custom_input, custom_len_input);
if (flag)
{
    if(NULL != sdkinfo){free(sdkinfo);sdkinfo =NULL;}
    return;
}
//获取人脸识别最大单个人脸特征长度
flag = FR_Getsize( *sdkinfo,&feature_len);
if (flag)
{
    if(NULL != sdkinfo){free(sdkinfo);sdkinfo =NULL;}
```

```
    return;
}

MULTIIMAGE probe_img_input ;//假定已赋值
probe_img_input.multiimg_num = 1;
FACETEMPLATE probe_template;
probe_template.feature_len = feature_len;
probe_template.feature = (void * )malloc(probe_template.feature_len);
//人脸模板提取
flag = extract( * sdkinfo, probe_img_input, &probe_template);
if (flag)
{
    flag = FR_Release( * sdkinfo);
    if(NULL != sdkinfo){free(sdkinfo);sdkinfo =NULL;}
    if(NULL != probe_template.feature)
    {free(probe_template.feature);probe_template.feature = NULL;}
    return;
}
MULTIIMAGE gallery_img_input ; //假定已赋值
gallery_img_input.multiimg_num = 1;
FACETEMPLATE gallery_template;
gallery_template.feature_len = feature_len;
gallery_template.feature = (void * )malloc(gallery_template.feature_len);
//人脸模板提取
flag = extract( * sdkinfo, gallery_img_input,&gallery_template);
if (flag)
{
    flag = FR_Release( * sdkinfo);
    if(NULL != sdkinfo){free(sdkinfo);sdkinfo =NULL;}
    if(NULL != probe_template.feature)
    {free(probe_template.feature);probe_template.feature = NULL;}
    if(NULL != gallery_template.feature)
    {free(gallery_template.feature);gallery_template.feature = NULL;}
    return;
}
//人脸相似度计算
long long face_num = 1;
long long target_num = 1;
long long sim_num;
sim_num = face_num;
MULTISIM * sim_output;
sim_output = (MULTISIM * )malloc(sim_num * sizeof(MULTISIM));
for (int i=0;i<sim_num;i++ )
```

```
    {
        sim_output->sim_num = target_num;
        sim_output->sim = (FACESIM * )malloc(sim_output->sim_num * sizeof(FACESIM));
    }
    flag = FR_Match_A( * sdkinfo, &probe_template, face_num, &gallery_template, target_num,
custom_input, custom_len_input, sim_output, &sim_num);

    //释放内存
    flag = FR_Release( * sdkinfo);
    for (int i=0;i<sim_num;i++ )
    {
        if (NULL != sim_output->sim){free(sim_output->sim);sim_output->sim = NULL;}
    }
    if(NULL != sdkinfo){free(sdkinfo);sdkinfo =NULL;}
    if(NULL != probe_template.feature)
        {
        free(probe_template.feature);
        probe_template.feature = NULL;
        }
    if(NULL != gallery_template.feature)
        {
        free(gallery_template.feature);
        gallery_template.feature = NULL;
        }
    if(NULL != sim_output)
        {
        free(sdkinfo);
        sim_output =NULL;
        }
    }

        #define SUPPORTNUM 1 // SUPPORTNUM:支持的最大人脸检出数
        #define SHAPENUM 42 // SHAPENUM:其他人脸关键点个数

    int extract(SDKINFO sdkinfo,MULTIIMAGE img_input,FACETEMPLATE  * template_output)
    {
        //人脸检测
        MULTIROI face_roi;

        face_roi.roi_num = SUPPORTNUM;
        face_roi.roi = (FACEROI * )malloc(5 * sizeof(FACEROI));
        int flag;
        void * custom_input = NULL;
```

```
        int custom_len_input =-1;
        flag = FR_Facedetect(sdkinfo, img_input, custom_input, custom_len_input, &face_roi);
        if (flag)
        {
            if(NULL != face_roi.roi){free(face_roi.roi);face_roi.roi = NULL;}
            return flag;
    }
    //人脸关键点位置
    MULTIPOS orgpos;
    orgpos.pos_num = face_roi.roi_num;
    orgpos.pos = (ORGANPOS *)malloc(orgpos.pos_num * sizeof(ORGANPOS));
    for (int i=0;i<orgpos.pos_num;i++)
    {
        (orgpos.pos+i)->point_len = SHAPENUM * 2;
        (orgpos.pos+i)->point = (int *)malloc((orgpos.pos+i)->point_len * sizeof(int));
    }
    flag = FR_location_A(sdkinfo, img_input, face_roi, custom_input, custom_len_input,
&orgpos);
    if (flag)
    {
        if(NULL != face_roi.roi){free(face_roi.roi);face_roi.roi = NULL;}
        for (int i=0;i<orgpos.pos_num;i++)
        {
            if(NULL != (orgpos.pos+i)->point)
            {
                free((orgpos.pos+i)->point);
            }
        }
        if(NULL != orgpos.pos){free(orgpos.pos);orgpos.pos = NULL;}
        return flag;
    }
    //人脸归一化
    MULTIIMAGE normal_img;
    normal_img.img_num = face_roi.roi_num;
    normal_img.img = (ONEIMAGE *)malloc(normal_img.img_num * sizeof(ONEIMAGE));
    for (int i=0;i<normal_img.img_num;i++)
    {
        (normal_img.img+i)->image_height = Normal_heigh;// Normal_heigh:归一化图像高度
        (normal_img.img+i)->image_width = Normal_width; //Normal_ width:归一化图像宽度
        (normal_img.img+i)->image_depth = Normal_depth;//Normal_ depth:归一化图像深度
        (normal_img.img+i)->data_len = Normal_heigh * Normal_width * Normal_depth;
        (normal_img.img+i)->data = (unsigned char *)malloc(Normal_heigh * Normal_width *
Normal_depth * sizeof(unsigned char));
```

```
}
MULTIPOS normal_pos;
normal_pos.pos_num = orgpos.pos_num;
normal_pos.pos = (ORGANPOS1 * )malloc(orgpos.pos_num * sizeof(ORGANPOS1));
for (int i=0;i<normal_pos.pos_num;i++)
{
    (normal_pos.pos+i)->point_len = SHAPENUM * 2;
    (normal_pos.pos+i)->point = (int * )malloc((normal_pos.pos+i)->point_len * sizeof
(int));
}
flag = FR_Normalize_A(sdkinfo, img_input, orgpos, custom_input, custom_len_input,
&normal_img, &normal_pos);
if (flag)
{
    for (int i=0;i<orgpos.pos_num;i++)
    {
        if(NULL != (orgpos.pos+i)->point)
        {
            free((orgpos.pos+i)->point);
        }
    }
    for (int i=0;i<normal_img.img_num;i++)
    {
        if(NULL != (normal_img.img+i)->data)free((normal_img.img+i)->data);
        (normal_img.img+i)->data = NULL;
    }
    for (int i=0;i<normal_pos.pos_num;i++)
    {
        if(NULL != (normal_pos.pos+i)->point)free((normal_pos.pos+i)->point);
        (normal_pos.pos+i)->point = NULL;
    }
    if(NULL != face_roi.roi){free(face_roi.roi);face_roi.roi = NULL;}
    if(NULL != orgpos.pos){free(orgpos.pos);orgpos.pos = NULL;}
    if(NULL != normal_img.img){free(normal_img.img);normal_img.img = NULL;}
    if(NULL != normal_pos.pos){free(normal_pos.pos);normal_pos.pos = NULL;}
    return 0;
}
////////////////////////////////////////////////////////////////////////////////
//提取人脸模板
flag = FR_Extract_A(sdkinfo, normal_img, normal_pos, custom_input, custom_len_input,
template_output);
//释放内存
for (int i=0;i<orgpos.pos_num;i++)
```

```
{
    if(NULL != (orgpos.pos+i)->point)
    {
        free((orgpos.pos+i)->point);
    }
}
for (int i=0;i<normal_img.img_num;i++)
{
    if(NULL != (normal_img.img+i)->data)free((normal_img.img+i)->data);
    (normal_img.img+i)->data = NULL;
}
for (int i=0;i<normal_pos.pos_num;i++)
{
    if(NULL != (normal_pos.pos+i)->point)free((normal_pos.pos+i)->point);
    (normal_pos.pos+i)->point = NULL;
}
if(NULL != face_roi.roi){free(face_roi.roi);face_roi.roi = NULL;}
if(NULL != orgpos.pos){free(orgpos.pos);orgpos.pos = NULL;}
if(NULL != normal_img.img){free(normal_img.img);normal_img.img = NULL;}
if(NULL != normal_pos.pos){free(normal_pos.pos);normal_pos.pos = NULL;}
return flag;
}
```

参 考 文 献

[1] GB/T 15272—1994 程序设计语言 C
[2] GB/T 28181—2011 安全防范视频监控联网系统 信息传输、交换、控制技术要求
[3] GA/T 922.2—2011 安防人脸识别应用系统 第 2 部分:人脸图像数据
[4] GB/T 20271—2006 信息安全技术 信息系统通用安全技术要求
[5] GB/T 20273—2006 信息安全技术 数据库管理系统安全技术要求
[6] GB/T 30267.1—2013 信息技术 生物特征识别应用程序接口 第 1 部分:BioAPI 规范
[7] NIST_FRVT2012_api_Aug15

GA/T 1326—2017

中华人民共和国公共安全
行　业　标　准
安全防范　人脸识别应用
程序接口规范
GA/T 1326—2017
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100029)
北京市西城区三里河北街16号(100045)
网址:www.spc.org.cn
服务热线:400-168-0010
2018年1月第一版
*
书号:155066·2-32672

GA/T 1326-2017