北京华捷艾米软件科技有限公司

保密程度	
绝密	( )
机密	( )
秘密	( )
内部资料	( )
公开资料	(√)

ImiNI SDK 规格说明书

文 件 编 号：HJNJ-A100M-RJ-002

版　　本：1.7.2

编　　制：软件部

审　　核：部门经理

批　　准：系统级负责人

受 控 状 态：

2015 年 7 月 9 日颁布　　2015 年 7 月 9 日实施

修订历史:

版本号	日期	修订内容	作者
0.6.6	2018-09-14	初版	闫国启
1.6.0	2018-09-28	修改 ImiSetFrameSync 名字为 ImiCamSetFrameSync	李伟
1.6.6	2018-11-27	增加 C++接口说明	闫国启
1.7.2	2019-04-29	增加 reset 属性	邓福
1.7.3	2019-07-12	删除 C++接口说明	邓福
1.7.8	2019-09-10	增加设备电流切换属性 增加设置曝光区域接口 imiCamSetExposureArea 增加清除曝光区域接口 imiCamClearExposureArea	邓福
1.8.0	2019-10-22	增加获取设备支持的能力接口	邓福
1.8.1	2019-11-15	增加红外 8Bit 输出接口 imiSetOutput8BitIrData 增加 RGB 转灰度输出接口 imiCamSetRGB2Gray	闫国启
1.8.2	2020-01-20	增加旋转角度设置接口 setRotationAngle	李伟
1.8.7	2020-03-02	增加设置 FrameMode 接口 imiCamSetFrameMode 增加打开彩色流接口 imiCamStartStream2	邓福
1.8.8	2020-03-26	增加彩色设置属性包括增益，对比度，亮度，白平衡等（详见 4.7.1 章节）	邓福
1.8.9	2021-02-02	增加红外 AE 功能开关 imiSetIRFaceAEEnable	孙鹏
1.8.10	2021-07-01	增加旋转平移参数，红外人脸框参数，用户相机参数属性	刘宗泽

目录

目录

1. 概述

ImiNI SDK 是基于华捷艾米体感设备私有 USB 通信协议开发的软件开发套件。套件封装了对华捷艾米体感设备底层功能的设置与操作，这些底层功能包括 Sensor 的设置、彩色/深度视频流、设备的控制操作等，用于使用华捷艾米体感设备的二次开发，可广泛用于 3D 建模、机器视觉、体感游戏开发等领域。

1. 1. 术语与简称

Table 1.1-1 列举了本文档中所提及的专业术语与简称。.

Table 1.1-1 术语定义与简称列表

名 称	描 述

名 称	描 述
Imi	华捷艾米公司的简称
ImiNI	艾米体感接口(Imi Natural Interface)
Iminect	华捷艾米公司的体感设备名称

2. 功能说明

2.1. 功能列表

Table 2.1-2 功能列表

功 能	描 述
深度图像功能	- 视野范围（水平视野弧度 60°、垂直视野弧度 47°） - 深度图最大深度 10 米 - 用户位置 - 绝对坐标与相对坐标的转换
彩色图像功能	- 图像输出格式：H264、RGB24、MJPEG 和 YUV420SP（不同产品支持的图像输出格式会有差别），帧率最大支持 30fps
骨架功能	- 同时跟踪 2 人以上骨骼数据，20 个关节点 （仅全功能版本 SDK 支持此功能）

3. 接口说明

3.1. C 接口说明

3.1.1. ImiNect 接口定义

3.1.1.1. API 说明

3.1.1.1.1. imiInitialize

[功能]

ImiSDK 初始化函数；

[格式]

int32_t imiInitialize()

[参数]

无

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

程序开始时调用。
可能的错误码：
0x8030010d 不支持的驱动类型
0x80300301 USB 初始化失败

3.1.1.1.2. imiInitialize2

[功能]

ImiSDK 初始化函数 2；支持 USB、FILE、NET 三种驱动类型设定

[格式]

int32_t imiInitialize2(ImiDriverInfo* mDriverInfo)

[参数]

Type	Name	Description	IN/OUT
ImiDriverInfo	mDriverInfo	数据驱动类型信息，见 ImiDriverInfo	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

需要使用文件模式或网络模式时，在程序开始时调用。文件模式使用 Record 接口记录的文件。
可能的错误码：
0x8030010d 不支持的驱动类型
0x80300301 USB 初始化失败

3.1.1.1.3. imiDestroy

[功能]

销毁函数，释放所有资源

[格式]

int32_t imiDestroy()

[参数]

无。

[返回值]

Value	Description
0	操作成功
非零值	操作失败，imiGetLastError()获取详细错误码

[说明]

程序退出前调用。

可能的错误码：

0x80300101　设备未初始化

3.1.1.1.4. imiGetDeviceList

[功能]

获取系统中 Iminect 设备列表和个数。

[格式]

int32_t imiGetDeviceList(ImiDeviceAttribute** pDeviceList, int32_t* pDeviceCount)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceAttribute**	pDeviceList	指向设备列表的指针	IN
int32_t*	pDeviceCount	指向设备数量的指针	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101　设备未初始化

3.1.1.1.5. imiReleaseDeviceList

[功能]

释放设备列表内存资源。

[格式]

int32_t imiReleaseDeviceList(ImiDeviceAttribute** pDeviceList)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceAttribute**	pDeviceList	指向设备列表的指针	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化

3.1.1.1.6. imiOpenDevice

[功能]

打开设备。

[格式]

int32_t imiOpenDevice (const char* pDeviceUri, ImiDeviceHandle* pDevice, int32_t reserve)

[参数]

Type	Name	Description	IN/OUT
const char*	pDeviceUri	设备 URI，当此值为 NULL 时，默认打开设备列表中的第一个设备	IN
ImiDeviceHandle*	pDevice	指向 ImiDeviceHandle 的指针	OUT
int32_t	reserve	保留，填 0	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300102 设备未打开
0x80300103 打开的设备数量超过上限
0x80300202 第二个参数为空指针
0x80300206 参数取值非法
0x80300302 USB 打开设备失败

3.1.1.1.7. imiCloseDevice

[功能]

关闭设备。

[格式]

int32_t imiCloseDevice(ImiDeviceHandle device)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN

[返回值]

Value	Description

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

先关闭和释放打开的数据流，见 imiStopStream(),imiDestroyChanelInstance()。
可能的错误码：
0x80300101 设备未初始化
0x80300102 设备未打开

3.1.1.1.8. imiSetDeviceProperty

[功能]

修改设备属性。

[格式]

int32_t imiSetDeviceProperty(ImiDeviceHandle device, uint32_t propertyId, const void* pData, uint32_t dataSize)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN
uint32_t	propertyId	属性标识符，见 enum IMI_PROP	IN
const void*	pData	指向属性内容的指针	IN
uint32_t dataSize	dataSize	属性内容长度	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：
0x80300101 设备未初始化
0x80300102 设备未打开
0x80300109 空指针异常
0x8030010b 设置属性值失败
0x80300203 第三个参数为空指针
0x80300206 参数取值非法

3.1.1.1.9. imiGetDeviceProperty

[功能]

获取设备属性函数。

[格式]

int32_t imiGetDeviceProperty(ImiDeviceHandle device, uint32_t propertyId, void* pData, uint32_t* pDataSize)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN
uint32_t	propertyId	属性标识符，见 enum IMI_PROP	IN
void*	pData	指向属性内容的指针	OUT
uint32_t*	pDataSize	指向属性内容长度的指针	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300102 设备未打开
0x80300109 空指针异常
0x8030010c 获取属性值失败
0x80300203 第三个参数为空指针
0x80300206 参数取值非法

3.1.1.1.10.imiSetDeviceStateCallback

[功能]

设置设备状态变化通知回调。

[格式]

int32_t imiSetDeviceStateCallback(ImiDeviceStateCallback callback, void* pData)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceStateCallback	callback	Imi 设备状态变化通知函数	IN
void*	pData	用户数据，在 ImiDeviceStateCallback 做入参传回	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300201 第一个参数为空指针

3.1.1.1.11.imiGetSupportFrameMode

[功能]

获取设备支持的帧模式，帧模式是像素格式、分辨率、帧率、每像素多少位数据的组合。

[格式]

int32_t imiGetSupportFrameMode(ImiDeviceHandle device, ImiFrameType frameType, const ImiFrameMode** pMode, uint32_t* pNumber)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN
ImiFrameType	frameType	Frame type，见 enum ImiFrameType	IN

const ImiFrameMode**	pMode	所有支持的帧模式。指针数组，内存由 SDK 管理。用法：
		const ImiFrameMode* pMode = NULL;
		uint32_t number = 0;
		int32_t ret = imiGetSupportFrameMode(device1, type1, &pMode, &number);
		当支持的帧模式个数大于 0 时，取值如下：pMode[0]…pMode[number-1]
			OUT

uint32_t*	pNumber	指向支持的帧模式的个数的指针	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300102 设备未打开
0x80300203 第三个参数为空指针
0x80300204 第四个参数为空指针
0x80300207 非法的帧类型

3.1.1.1.12.imiSetFrameMode

[功能]

设置帧模式，帧模式是像素格式、分辨率、帧率、每像素多少位数据的组合。

[格式]

int32_t imiSetFrameMode(ImiDeviceHandle device, ImiFrameType frameType, ImiFrameMode* pMode)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN
ImiFrameType	frameType	Frame type，见 enum ImiFrameType	IN
ImiFrameMode*	pMode	指向帧模式的指针，见 ImiFrameMode。设置时只需要填写像素格式和分辨率。	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300102 设备未打开
0x80300109 空指针异常
0x8030010b 设置属性值失败
0x80300206 参数取值非法
0x80300207 非法的帧类型

3.1.1.1.13.imiGetCurrentFrameMode

[功能]

获取当前帧模式。

[格式]

const ImiFrameMode* imiGetCurrentFrameMode(ImiDeviceHandle device, ImiFrameType frameType)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN
ImiFrameType	frameType	Frame type，见 enum ImiFrameType	IN

[返回值]

Value	Description
非空指针	操作成功，指针中存储了 frameType 的帧当前模式，见 ImiFrameMode
空指针	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300102 设备未打开
0x80300207 非法的帧类型

3.1.1.1.14.imiOpenStream

[功能]

打开指定数据类型的数据流。

[格式]

int32_t imiOpenStream(ImiDeviceHandle device, ImiFrameType frameType, ImiNewFrameCallback callback, void* pData, ImiStreamHandle* pStream)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN
ImiFrameType	frameType	Frame type，见 enum ImiFrameType	IN
ImiNewFrameCallback	callback	新数据帧到达通知回调函数，可为空	IN
void*	pData	用户数据，做 ImiNewFrameCallback 入参	IN
ImiStreamHandle*	pStream	用于保存 Stream 句柄的内存地址	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300102 设备未打开
0x80300104 设备已断开
0x80300105 流未找到
0x80300106 打开流的数量超过上限
0x8030010f 加载 h264 解码库失败
0x80300110 导入 h264 解码库 API 失败
0x80300205 第五个参数为空指针
0x80300207 非法的帧类型

3.1.1.1.15.imiCloseStream

[功能]

关闭指定的数据流。

[格式]

int32_t imiCloseStream(ImiStreamHandle Stream)

[参数]

Type	Name	Description	IN/OUT
ImiStreamHandle	Stream	Stream 句柄，由 OpenStream 得到	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300105 流未找到

3.1.1.1.16.imiReadNextFrame

[功能]

读取对应数据流下一帧数据。

[格式]

int32_t imiReadNextFrame(ImiStreamHandle Stream, ImiImageFrame** ppFrame, int32_t timeout)

[参数]

Type	Name	Description	IN/OUT
ImiStreamHandle	stream	数据流句柄	IN
ImiImageFrame**	ppFrame	保存 ImiImageFrame 指针的地址	OUT
int32_t	timeout	等待时长，毫秒（ms）	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

当前数据流没有数据，将阻塞等待到有新帧到来或超时。
ImiImageFrame 使用结束后，使用 imiReleaseFrame()，释放引用。
可能的错误码：

0x80300001 系统调用失败
0x80300101 设备未初始化
0x80300105 流找不到
0x80300107 没有帧数据
0x8030010a 等待超时
0x80300201 第一个参数为空指针
0x80300202 第二个参数为空指针
0x80300206 参数取值非法

3.1.1.1.17.imiReleaseFrame

[功能]

释放 frame 引用。

[格式]

int32_t imiReleaseFrame(ImiImageFrame** pFrame)

[参数]

Type	Name	Description	IN/OUT
ImiImageFrame**	pFrame	指向 ImiImageFrame 指针的地址	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

ImiImageFrame 使用结束后，应尽快使用 imiReleaseFrame()，释放引用。默认读取 5 帧。
可能的错误码：

0x80300101 设备未初始化
0x80300201 第一个参数为空指针

3.1.1.1.18.imiCreateRecorder

[功能]

创建记录器，并指定记录保存的文件路径

[格式]

int32_t imiCreateRecorder(const char* pFileToSave, ImiRecordHandle* pRecorder)

[参数]

Type	Name	Description	IN/OUT
const char*	pFileToSave	指向文件保存路径的指针，文件路径包含文件名。如”c:/1.data”。	IN
ImiRecordHandle*	pRecorder	用于保存 Recorder 句柄的内存地址	OUT

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

[说明]

录制回放使用方式：

1） 调用 imiInitialize2，指定驱动类型为 FILE，指定文件路径
或
修改配置文件 imi.ini,确认[Common]Driver=Dummy，然后配置 Data 文件路径，
[File_Driver] Record_File_Path="c:/1.data"

2）调用 openStream 等接口回放，流程与正常使用 iminect 设备流程一致。

可能的错误码：

0x80300002 打开文件失败
0x80300101 设备未初始化
0x80300102 设备未打开
0x80300104 设备已断开
0x80300106 打开的记录器数量超过上限

3.1.1.1.19.imiRecorderAttachStream

[功能]

将已打开的需要记录的流附加到记录器

[格式]

int32_t imiRecorderAttachStream(ImiRecordHandle recorder, ImiStreamHandle stream, ImiDataType dataType, ImiBOOL bCompress)

[参数]

Type	Name	Description	IN/OUT
ImiRecordHandle	recorder	Recorder 句柄	IN
ImiStreamHandle	stream	stream 由 imiOpenStream()获得	IN
ImiDataType	dataType	Data type， 见 enum ImiDataType	IN
ImiBOOL	bCompress	是否压缩	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

[说明]

只是将流附加到记录器，并未开始记录

可能的错误码：

0x80300101 设备未初始化
0x80300105 记录流未找到
0x80300108 记录器未找到
0x80300209 非法的数据类型

3.1.1.1.20.imiRecorderStart

[功能]

开启记录器

[格式]

int32_t imiRecorderStart(ImiRecordHandle recorder)

[参数]

Type	Name	Description	IN/OUT
ImiRecordHandle	recorder	Recorder 句柄，由 imiCreateRecorder 得到	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300108 记录器未找到

3.1.1.1.21.imiRecorderStop

[功能]

关闭记录器;

[格式]

int32_t imiRecorderStop(ImiRecordHandle recorder)

[参数]

Type	Name	Description	IN/OUT
ImiRecordHandle	recorder	Recorder 句柄，由 imiCreateRecorder 得到	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

[说明]

只会暂停记录，可以再次调用 imiRecorderStart 继续记录

可能的错误码：

0x80300101 设备未初始化
0x80300108 记录器未找到

3.1.1.1.22.imiDestroyRecorder

[功能]

销毁记录器;

[格式]

int32_t imiDestroyRecorder(ImiRecordHandle recorder)

[参数]

Type	Name	Description	IN/OUT
ImiRecordHandle	recorder	Recorder 句柄，由 imiCreateRecorder 得到	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300108 记录器未找到

3.1.1.1.23.imiGetVersion

[功能]

获取版本信息;

[格式]

int32_t imiGetVersion(ImiDeviceHandle device, ImiVersions* pImiVersion)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	Devicehandle 由 imiOpenDevice()获得	IN
ImiVersions*	pImiVersion	版本信息	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300101 设备未初始化
0x80300102 设备找不到
0x8030010c 获取属性值失败
0x80300202 第二个参数为空指针
0x80300206 参数取值非法
0x8030020a 空指针异常

3.1.1.1.24.imiGetLastError

[功能]

获取最近一次操作的失败原因码;

[格式]

int32_t imiGetLastError()

[参数]

无

[返回值]

Value	Description
整型值	详细错误码

[说明]

无。

3.1.1.1.25.imiGetErrorString

[功能]

获取错误码对应的文字说明。

[格式]

const char* imiGetErrorString(int32_t nErrorCode)

[参数]

Type	Name	Description	IN/OUT
int32_t	nErrorCode	nErrorCode 由 imiGetLastError 获得	IN

[返回值]

Value	Description
字符串	错误码对应的文字说明
空	获取失败，不存在该错误码

[说明]

可能的错误码：

0x80300206　参数取值非法

3.1.1.1.26.imiSetLogOutputDir

[功能]

设置日志生成路径;

[格式]

int32_t imiSetLogOutputDir(const char* pOutputDir)

[参数]

Type	Name	Description	IN/OUT
const char*	pOutputDir	目标路径	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300002　打开文件失败
0x80300201　第一个参数为空指针
0x80300206　参数取值非法

3.1.1.1.27.imiSetLogLevel

[功能]

设置日志级别;

[格式]

int32_t imiSetLogLevel(uint32_t level)

[参数]

Type	Name	Description	IN/OUT
uint32_t	level	日志级别： 0:Verbose, 1:Information, 2:Warning, 3:Error	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300206　参数取值非法

3.1.1.1.28.imiConvertCoordinateDepthToColor

[功能]

深度图坐标转彩色图坐标，输入深度图坐标位置，输出对应彩色图坐标位置。

[格式]

int32_t imiConvertCoordinateDepthToColor (ImiCoordinateConvertMode convertMode, uint32_t depthX, uint32_t depthY, uint16_t depthZ, uint32_t* pImageX, uint32_t* pImageY)

[参数]

Type	Name	Description	IN/OUT
ImiCoordinateConvertMode	convertMode	深度坐标到彩色坐标的转换模式	IN
uint32_t	depthX	深度图 X 坐标	IN
uint32_t	depthY	深度图 Y 坐标	IN
uint32_t	depthZ	深度图 Z 坐标，深度值	IN
uint32_t*	pImageX	指向彩色图 X 坐标的指针	OUT
uint32_t*	pImageY	指向彩色图 Y 坐标的指针	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300206　参数取值非法

3.1.1.1.29.imiConvertCoordinateDepthToWorld

[功能]

深度图坐标转世界坐标坐标，输入深度图坐标位置，输出对应世界坐标系位置。

[格式]

int32_t imiConvertCoordinateDepthToWorld(ImiVector4* pDst, const ImiVector4I* pSrc, int32_t height, int32_t width)

[参数]

Type	Name	Description	IN/OUT
ImiVector4	pDst	指向输出的世界坐标的指针	OUT
ImiVector4I	pSrc	指向输入的深度图坐标的指针	IN
int32_t	height	深度图高	IN
int32_t	widht	深度图宽	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

可能的错误码：

0x80300206 参数取值非法

3.1.1.1.30.imiConvertCoordinateWorldToDepth

[功能]

世界坐标转深度图坐标，输入世界坐标位置，输出对应深度图坐标位置。

[格式]

int32_t imiConvertCoordinateWorldToDepth(ImiVector4I* pDst, const ImiVector4* pSrc, int32_t height, int32_t width)

[参数]

Type	Name	Description	IN/OUT
ImiVector4I	pDst	指向输出的深度图坐标的指针	OUT
ImiVector4	pSrc	指向输入的世界坐标的指针	IN
int32_t	height	深度图高	IN
int32_t	widht	深度图宽	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

3.1.1.1.31.imiConvertDepthToPointCloud

[功能]

深度转点云，输入深度数据，输出点云数据。

[格式]

int32_t imiConvertDepthToPointCloud(const ImiImageFrame* pDepth, float factor, float fx, float fy, float cx, float cy, ImiPoint3D* pPointClouds)

[参数]

Type	Name	Description	IN/OUT
const ImiImageFrame*	pDepth	指向深度数据的指针	IN
float	factor	camera_factor	IN
float	fx	camera_fx	IN
float	fy	camera_fy	IN
float	cx	camera_cx	IN
float	cy	camera_cy	IN
ImiPoint3D*	pPointClouds	输出的点云数据（buffer 由调用者申请）	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

factor：若输出点云坐标以 m 为单位，传入 1000.0
fx：相机内部参考，x 轴归一化焦距，以像素单位表示
fy：相机内部参考，y 轴归一化焦距，以像素单位表示
cx：相机内部参考，主点（通常在图像中心）的 x 轴坐标值
cy：相机内部参考，主点（通常在图像中心）的 y 轴坐标值
ImiPoint3D：参考结构体 ImiPoint3D 定义
若 factor、fx、fy、cx、cy 其中任何一个小于等于 0，这几个参数将使用默认值

3.1.1.1.32.imiSetUpgradeChannelNo

[功能]

设置是否升级设备固件，当用户注册的升级回调函数被调用时，用此函数进行设置。

[格式]

int32_t imiSetUpgradeChannelNo(const char* pChannelNo)

[参数]

Type	Name	Description	IN/OUT
const char*	pChannelNo	设备升级的渠道号	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

[说明]

为支持批量升级，升级服务器会分配设备渠道号提供给商家，需在初始化后打开设备前进行设置
可能的错误码：
0x80300206　参数取值非法

3.1.1.1.33.imiDeviceRequestUpgrade

[功能]

判断是否需要在线升级固件。

[格式]

int32_t imiDeviceRequestUpgrade(ImiDeviceHandle pDevice)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	pDevice	设备 handle	IN

[返回值]

Value	Description
0	成功，需要升级固件
小于 0	失败，或者不需要升级固件

3.1.1.1.34.imiDeviceStartUpgrade

[功能]

开始在线升级固件，并设置升级状态回调。

[格式]

int32_t imiDeviceStartUpgrade(ImiDeviceHandle pDevice, const ImiUpgradeCallbacks* pCallbacks)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	pDevice	设备 handle	IN
const ImiUpgradeCallbacks*	pCallbacks	升级状态的回调	IN

[返回值]

Value	Description
0	成功
小于 0	失败

3.1.1.1.35.imiDeviceStartUpgradeOffLine

[功能]

开始离线升级固件，并设置升级状态回调。

[格式]

int32_t imiDeviceStartUpgradeOffLine(ImiDeviceHandle pDevice, const ImiUpgradeCallbacks* pCallbacks, const ImiUpgradeRomPath* pUpgradeRomPath)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	pDevice	设备 handle	IN
const ImiUpgradeCallbacks*	pCallbacks	升级状态的回调	IN
const ImiUpgradeRomPath*	pUpgradeRomPath	设置固件路径	IN

[返回值]

Value	Description
0	成功
小于 0	失败

3.1.1.1.36.imiSelectUser

[功能]

选择优先跟踪骨架的用户

[格式]

int32_t imiSelectUser(const ImiDeviceHandle device, uint32_t userId)

[参数]

Type	Name	Description	IN/OUT
const ImiDeviceHandle	device	设备	IN
uint32	userId	优先跟踪骨架的用户 ID	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

3.1.1.1.37.imiUnSelectUser

[功能]

取消选择优先跟踪骨架的用户

[格式]

int32_t imiUnSelectUser (const ImiDeviceHandle device, uint32_t userId)

[参数]

Type	Name	Description	IN/OUT
const ImiDeviceHandle	device	设备	IN
uint32	userId	优先跟踪骨架的用户 ID	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

3.1.1.1.38.imiTakePhoto

[功能]

拍照并保存为 bmp 格式的图片

[格式]

int32_t imiTakePhoto (const char* pBsmpImagePath)

[参数]

Type	Name	Description	IN/OUT
const char*	pBsmpImagePath	指向图片保存路径的指针	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

3.1.1.1.39.imiSetImageRegistration

[功能]

设置设备配准功能

[格式]

int32_t imiSetImageRegistration(ImiDeviceHandle device, ImiBOOL bEnable)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	设备	IN
ImiBOOL	bEnable	IMI_TRUE:打开配准 IMI_FALSE:关闭配准	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败，imiGetLastError()获取详细错误码

3.1.1.1.40.imiIsImageRegistrationEnable

[功能]

查询设备配准是否打开

[格式]

ImiBOOL imiIsImageRegistrationEnable(ImiDeviceHandle device)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	设备	IN

[返回值]

Value	Description
IMI_TRUE	配准已打开
IMI_FALSE	配准未打开

3.1.1.1.41.imiGetSupportCapacity

[功能]

获取设备支持的能力

[格式]

int32_t imiGetSupportCapacity(ImiDeviceHandle device, ImiSupportCapacity* pImiSupportCapacity)

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	设备	IN
ImiSupportCapacity*	pImiSupportCapacity	设备能力信息结构体	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

3.1.1.1.42.imiSetOutput8BitIrData

[功能]

设置输出红外 8Bit

[格式]

int32_t imiSetOutput8BitIrData(ImiDeviceHandle device, ImiBOOL bEnable);

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	设备	IN

Type	Name	Description	IN/OUT
ImiBOOL	bEnable	设置是否输出红外 8Bit	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

3.1.1.1.43.imiSetRotationAngle

[功能]

设置图像的顺时针旋转角度

[格式]

int32_t imiSetRotationAngle(ImiDeviceHandle device, ImiRotationAngle rotationAngle);

[参数]

Type	Name	Description	IN/OUT
ImiDeviceHandle	device	设备	IN
ImiRotationAngle	rotationAngle	顺时针旋转角度: IMI_ROTATION_ANGLE_0: 顺时针旋转 0 度 IMI_ROTATION_ANGLE_90：顺时针旋转 90 度 IMI_ROTATION_ANGLE_180：顺时针旋转 180 度 IMI_ROTATION_ANGLE_270：顺时针旋转 270 度	IN

[返回值]

Value	Description
0	操作成功
非零值	操作失败, imiGetLastError()获取详细错误码

3.1.1.2. 支持的帧模式列表

FrameType	Platform	PixelFormat	Resolution	FPS
IMI_COLOR_FRAME	Windows	IMI_PIXEL_FORMAT_IMAGE_H264	1920*1080	30
		IMI_PIXEL_FORMAT_IMAGE_H264	1280*720	30
		IMI_PIXEL_FORMAT_IMAGE_H264	640*480	30
		IMI_PIXEL_FORMAT_RGB_24	1920*1080	30
		IMI_PIXEL_FORMAT_RGB_24	1280*720	30
		IMI_PIXEL_FORMAT_RGB_24	640*480	30
		IMI_PIXEL_FORMAT_YUV420SP	640*480	30
		IMI_PIXEL_FORMAT_YUV422	320*240	30
		IMI_PIXEL_FORMAT_YUV422	640*480	30
		IMI_PIXEL_FORMAT_YUV422	1280*960	30
	Android	IMI_PIXEL_FORMAT_IMAGE_H264	1920*1080	30
		IMI_PIXEL_FORMAT_IMAGE_H264	1280*720	30
		IMI_PIXEL_FORMAT_IMAGE_H264	640*480	30
		IMI_PIXEL_FORMAT_RGB_24	640*480	30
		IMI_PIXEL_FORMAT_YUV420SP	640*480	30
		IMI_PIXEL_FORMAT_YUV422	320*240	30
		IMI_PIXEL_FORMAT_YUV422	640*480	30
		IMI_PIXEL_FORMAT_YUV422	1280*960	30
	Linux	IMI_PIXEL_FORMAT_IMAGE_H264	1920*1080	30
		IMI_PIXEL_FORMAT_IMAGE_H264	1280*720	30
		IMI_PIXEL_FORMAT_IMAGE_H264	640*480	30
		IMI_PIXEL_FORMAT_YUV420SP	640*480	30

		IMI_PIXEL_FORMAT_YUV422	320*240	30
		IMI_PIXEL_FORMAT_YUV422	640*480	30
		IMI_PIXEL_FORMAT_YUV422	1280*960	30
IMI_DEPTH_FRAME	Windows/Android/Linux	IMI_PIXEL_FORMAT_DEP_16BIT	640*480	30
		IMI_PIXEL_FORMAT_DEP_16BIT	320*240	30
		IMI_PIXEL_FORMAT_DEP_16BIT	640*480	25
		IMI_PIXEL_FORMAT_DEP_16BIT	640*480	15
		IMI_PIXEL_FORMAT_DEP_16BIT	640*400	25
		IMI_PIXEL_FORMAT_DEP_16BIT	640*400	15
		IMI_PIXEL_FORMAT_DEP_16BIT	320*200	25
		IMI_PIXEL_FORMAT_DEP_16BIT	320*200	15
IMI_DEPTH_SKELETON_FRAME	Windows/Android/Linux	IMI_PIXEL_FORMAT_DEP_16BIT	640*480	30
		IMI_PIXEL_FORMAT_DEP_16BIT	320*240	30
IMI_USER_INDEX_SKELETON_FRAME	Windows/Android/Linux	IMI_PIXEL_FORMAT_DEP_16BIT	640*480	30
		IMI_PIXEL_FORMAT_DEP_16BIT	320*240	30
IMI_SKELETON_FRAME	Windows/Android/Linux	IMI_PIXEL_FORMAT_DEP_16BIT	640*480	30
		IMI_PIXEL_FORMAT_DEP_16BIT	320*240	30
IMI_DEPTH_IR_FRAME	Windows/Android/Linux	IMI_PIXEL_FORMAT_DEP_IR_16BIT	640*400	25
		IMI_PIXEL_FORMAT_DEP_IR_16BIT	640*480	25
IMI_IR_FRAME	Windows/Android/Linux	IMI_PIXEL_FORMAT_IR_16BIT	640*488	30
		IMI_PIXEL_FORMAT_IR_16BIT	640*480	25
		IMI_PIXEL_FORMAT_IR_16BIT	640*480	15
		IMI_PIXEL_FORMAT_IR_16BIT	640*400	25
		IMI_PIXEL_FORMAT_IR_16BIT	640*400	15

3.1.1.3. 设备属性参考

```
//General,
IMI_PROPERTY_GENERAL_VERSION       = 0x00,//ImiVersions Read Only
IMI_PROPERTY_GENERAL_SERIAL_NUMBER = 0x01,//String, Read Only
IMI_PROPERTY_GENERAL_FRAME_SYNC    = 0x02, //Not Support Yet
IMI_PROPERTY_IMAGE_REGISTRATION    = 0x03, //value type ImiBOOL, IMI_TRUE:open registration,
IMI_FALSE: close registration
IMI_PROPERTY_DEVICE_ATTRIBUTE  = 0x04,   //ImiDeviceAttribute, Read Only
IMI_PROPERTY_DEVICE_LIGHT_MODE = 0x05, //0 all close  1 open point light  2 open flood light
IMI_PROPERTY_RESET_DEVICE      = 0x06,   //value type uint8_t, 1 reset Device
IMI_PROPERTY_IR_DIST           = 0x07,//value type uint8_t, 0-255
IMI_PROPERTY_DEVICE_SWITCH_POWER = 0x09, //valuie type uint8_t, 1 :power is 1.3A   0: power is 0.8A
default: 0

// Color,
IMI_PROPERTY_COLOR_MIRROR     = 0x13, //value type uint8_t, 1:mirror, 0: not mirror
IMI_PROPERTY_COLOR_SHARPNESS  = 0x14, //Not Support Yet
IMI_PROPERTY_COLOR_BRIGHTNESS = 0x15, //Not Support Yet
IMI_PROPERTY_COLOR_CONTRAST   = 0x16, //Not Support Yet
IMI_PROPERTY_COLOR_SATURATION = 0x17, //Not Support Yet
IMI_PROPERTY_COLOR_GAIN       = 0X18, //Not Support Yet
IMI_PROPERTY_COLOR_AUTO_WHITE_BALANCE_MODE = 0x19, //Not Support Yet
IMI_PROPERTY_COLOR_AUTO_EXPOSURE_MODE      = 0x1a, //Not Support Yet
IMI_PROPERTY_COLOR_ANTIFLICKER             = 0x1b, //Not Support Yet
IMI_PROPERTY_COLOR_INTRINSIC_PARAMS      = 0x1d, // camera intrinsic parameter, value type float
params[9]: ImiCameraIntrinsic Struct

//Depth,
IMI_PROPERTY_DEPTH_HOLE_FILTER = 0x33,//value type uint8_t, 1: open(default), 0: close
IMI_PROPERTY_DEPTH_MIRROR      = 0x34, //value type uint8_t, 1:mirror, 0: not mirror
IMI_PROPERTY_DEPTH_DECIMATION  = 0x35,//Not Support Yet
IMI_PROPERTY_DEPTH_DENOISE     = 0x37,  // value type uint8_t, 1: denoising, 0: not denoising, default 1
IMI_PROPERTY_DEPTH_INTRINSIC_PARAMS = 0x36, // depth intrinsic parameter, value type float params[9]:
ImiCameraIntrinsic Struct

//IR,
IMI_PROPERTY_IR_MIRROR       = 0x1045, // Bool
IMI_PROPERTY_IR_INTRINSIC_PARAMS = 0x60, // ir intrinsic parameter, value type float params[9]:
ImiCameraIntrinsic Struct

//Skeleton,
IMI_PROPERTY_SKELETON_SMOOTH    = 0x40,
```

```
IMI_PROPERTY_SKELETON_MIRROR    = 0x41, //value type uint8_t, 1:mirror, 0: not mirror
IMI_PROPERTY_SKELETON_USER_SELECTOR_MODE     = 0x53,
IMI_PROPERTY_SKELETON_SELECT_TRACK_USER      = 0x54,
IMI_PROPERTY_SKELETON_UNSELECT_TRACK_USER    = 0x55,
IMI_PROPERTY_SKELETON_CALIBRATION          = 0x57,  // value type uint8_t, 1:use calibration, 0: don't
use calibration, default 0

IMI_PROPERTY_GROUND_EQUATION    = 0x70,          // value type uint8_t, 1:calculate ground equation, 0:
don't calculate the ground equation, default 0
IMI_PROPERTY_GROUND_CLEANUP             = 0x71,  // value type uint8_t, 1:clear the ground, 0: don't
clear the ground, default 0

IMI_PROPERTY_LD_OPERATE        = 0x80,  // value type uint8_t, 1: close the projector, 0: open the
projector, default 0
IMI_PROPERTY_FLOODLIGHT        = 0x90,  // value type uint8_t, 1: open the floodlight, 0: close the
floodlight, default 0

IMI_PROPERTY_LASER_SAFETY_MODE    = 0x99,  // value type uint8_t, 1: open the laser safety mode, 0:
close the laser safety mode, default 1

IMI_PROPERTY_SAFETY_DIST         = 0x100,   // value type int16_t,
IMI_PROPERTY_LIGHT_THRESHOLD     = 0x101,   // value type int32_t,
IMI_PROPERTY_AMBIENT_LIGHT_MODE = 0x102 //value type uint8_t, 1:open, 0:close, default 0
IMI_PROPERTY_REAL_SAFETY_DIST     = 0x103,  // value type int16_t, only get
IMI_PROPERTY_REAL_LIGHT_THRESHOLD = 0x104,  // value type int32_t, only get

IMI_PROPERTY_DEPTH_IR_MIRROR      = 0x105, //value type uint8_t, 1:mirror, 0: not mirror

//1.8.10 增加属性
IMI_PROPERTY_ROTATE_TRANS_INTRINSIC_PARAMS  =  0x106,  //  rt  intrinsic  parameter,  value  type
float params[12]:float rotate[9] + float trans[3]  ImiCameraIntrinsic Struct
IMI_PROPERTY_IR_RECT                    = 0x107,    //ImiRect
IMI_PROPERTY_CAMERA_PARAMS_4_USER = 0x109, //Inquiry

IMI_PROPERTY_IR_ENABLE= 0x10A,
IMI_PROPERTY_PARAMS_IR2DEPTH_RT = 0x10B, //Inquiry RT Array IR Camera 2 DepthCamera
IMI_PROPERTY_DEPTH_IR_COLOR_MIRROR = 0x10C,//value type uint8_t, 1:mirror, 0: not mirror
```

3.1.1.4. 头文件说明

sTable 3.1.4-3 头文件列表

文件名	说 明
include/ImiNect.h	ImiNI API 函数申明
include/ImiDefines.h	数据结构定义
include/ImiPlatform.h	平台相关定义
Include/ImiProperties.h	ImiNI 属性相关定义
Include/ImiSkeleton.h	骨架相关定义，仅全功能版本 SDK 支持骨架跟踪功能
Include/ImiUpgrade.h	设备升级相关定义

3.1.1.5. 使用说明

使用时序如图：

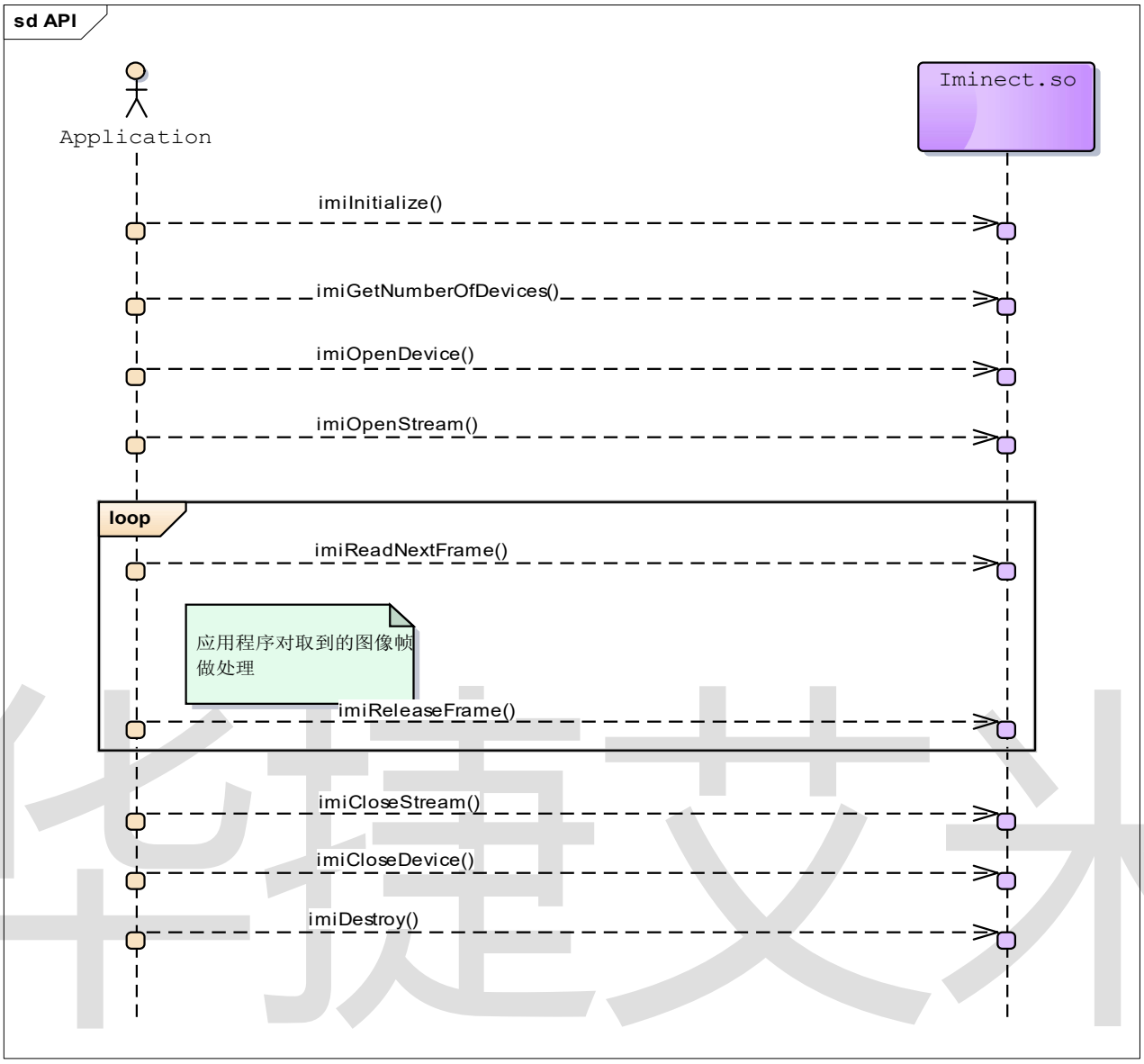


Figure 6.2.1 API 调用时序图

1. 获取图像可也可基于事件驱动，即注册数据流数据通知事件，流程如：
imiRegisterStreamCallback→imiStartStream，Callback 触发→imiReadNextFrame；

2. 同时打开多个数据流，可使用 imiWaitForStream，同时等待多个数据流数据；

3. 对图像帧处理完后，应及时调用 imiReleaseFrame 释放帧；

4. 所有 API 函数是线程安全的；

5. 返回值异常时，调用 imiGetLastError，获取错误码，更多信息请联系 IMI 工程师；

6. 当 imiOpenStream 中的 frameType 为 IMI_DEPTH_SKELETON_FRAME、
IMI_USER_INDEX_SKELETON_FRAME 或 IMI_SKELETON_FRAME 时（即需要打开骨架流）：

Windows 环境：需要将 ImiSDK 安装目录下 Redist 中的 NiTE2 文件夹拷贝至程序运行目录下；

Android 环境： 如果在未使用 ImiSDK.jar 时需要将发布包中 libs 目录下 Data 文件夹中的所有文件打包至 apk 安装目录下的 files 路径下。

3.1.1.6. 错误码描述

错误码值(0x)	错误码描述
80300000	No Error! System Base Code
80300001	System calls failed!
80300002	Failed to open the file!
80300100	No Error! Common Base Code
80300101	Device has not been initialized, please call imiInitialize first!
80300102	The device is closed, please open it first!
80300103	Too many devices has been opened, can not open another one!
80300104	The device is disconnected, please check the device
80300105	Can not find the valid stream!
80300106	Too many streams has been opened, can not open another one!
80300107	No frame data right now, try reading it later!
80300108	Can not find the record stream!
80300109	NULL pointer error!
8030010A	Wait timeout!
8030010B	Set property value failed!
8030010C	Get property value failed!
8030010D	Unsupported driver type!
8030010E	Can not open this type of stream!
8030010F	Load library failed, please check whether avcodec-56.dll, avutil-54.dll″ ″swresample-1.dll and swscale-3.dll are all exist!
80300110	Get proc address failed!
80300111	Common operation not supported!
80300200	No Error! Param Base Code
80300201	The first pointer parameter is NULL!
80300202	The second pointer parameter is NULL!
80300203	The third pointer parameter is NULL!
80300204	The fourth pointer parameter is NULL!
80300205	The fifth pointer parameter is NULL!
80300206	The value of the parameter is invalid!
80300207	The parameter frame type is invalid!
80300208	Device URI is invalid!
80300209	Data type is invalid!
80300300	No Error! USB Base Code
80300301	USB initialize failed!
80300302	USB open device failed!
80300303	libusb_cancel_transfer usb_cancel_async failed!
80300304	ReapTransfer libusbemu_reap invalid
80300305	libusbemu reap failure
80300306	Hot plug init failed!
80300307	Wait Object failed
80300308	GetOverlappedResult failed

80300309	pDataHead->nMagic, data is mess, trying to correction
8030030A	No Mess data
8030030B	read Mess data end
8030030C	USBK device list init failed!
8030030D	Device path is not in the device list!
8030030E	usbK API init failed!
8030030F	usbk claim interface failed!
80300310	usbK reset device failed!
80300311	Interface/alt setting number failed!
80300312	buffer size is zero, please check it!
80300313	USB wrong send control type
80300314	USB wrong receive control type
80300315	async thread error!
80300316	RegisterClass failure!
80300317	RegisterDeviceNotification failure!
80300318	Endpoint.failed to submit asynch I/O transfer!
80300319	Libusb resetDevice open failed.
8030031A	Libusb resetDevice claim interface failed
8030031B	Libusb resetDevice failed
8030031C	imiUSBOpenDeviceImplResetDevice failed!
8030031D	Libusb open failed
8030031E	Libusb claim interface failed
8030031F	filepath is empty!
80300320	imiUSBUpdateAP Open File Failed
80300321	Calculate MD5 failed!
80300322	Device getHubNumber failed!
80300400	No Error! SDK Errcode Code Base
80300401	Separate URL with , failed!
80300402	Segment do not match !
80300403	Separate URL is part, not 4(code,msg,data,url)
80300404	code is not 0!
80300405	msg is not ok!
80300406	url's format in data is invalid!
80300407	md5's format in data is invalid!
80300408	Format http get url error!
80300409	HTTP get nothing!
8030040A	dodownload file is invalid
8030040B	Invalid program file!
8030040C	Fork program failed!
8030040D	Calculate CRC failed due to open file failed!
8030040E	MD5File Apply memory faield!
8030040F	MD5String Apply memory faield!
80300410	thread_ not created
80300411	Unable to Set thread's priority!
80300412	Start Thread Failed

80300500	No Error! SDK error code base
80300501	Cannot locate reference to Direct3DCreate9 ABI in DLL
80300502	Direct3DCreate9 failed
80300503	Create d3d device failed!
80300504	IDirect3D9_GetAdapterIdentifier failed
80300505	IDirectXVideoAccelerationService_CreateSurface failed
80300506	IDirectXVideoDecoderService_GetDecoderConfigurations failed
80300507	Failed to find a supported decoder configuration
80300508	IDirectXVideoDecoderService_CreateVideoDecoder failed
80300509	cannot load DXVA2CreateDirect3DDeviceManager9 function
8030050A	OurDirect3DCreateDeviceManager9 failed
8030050B	IDirect3DDeviceManager9_ResetDevice failed
8030050C	cannot load function DXVA2CreateVideoService
8030050D	OpenDeviceHandle failed
8030050E	GetVideoService failed
8030050F	VlcVaDxva2 cannot load d3d9.dll
80300510	VlcVaDxva2 cannot load dxva2.dll
80300511	VlcVaDxva2 create Direct3D device Failed
80300512	d3dCreateDeviceManager failed
80300513	dxCreateVideoService failed
80300514	dxFindVideoServiceConversion failed
80300515	IDirect3DDeviceManager9_TestDevice failed
80300516	vlc_va_setVlcVaDxva2 failed
80300517	VaGrabSurface failed
80300518	Error hanpped in Waiting new frame event
80300519	Invalid parameter: prop is NULL
8030051A	Invalid parameter: props is NULL
8030051B	ImiDevice_setIntPropertyCallback Type is not int!
8030051C	setIntPropertyCallback Invalid length
8030051D	Invalid value,sharpness must between 0 and 100
8030051E	getIntPropertyCallback Type is not int!
8030051F	getIntPropertyCallback Invalid len
80300520	setResolutionCallback Property Id invalid!
80300521	setResolutionCallback Resolution Invalid
80300522	setResolutionCallback Not Support Resolution
80300523	resetDeviceCallback Invalid parameter
80300524	setCmosRigisterCallback Invalid parameter
80300525	getCmosRigisterCallback Invalid parameter
80300526	getVersionCallback Invalid parameter
80300527	setTecPointCallback Invalid parameter
80300528	getTecStatusCallback Invalid parameter
80300529	getFirmwareLog Invalid parameter
8030052A	ImiDevice_writeAHBCallback Invalid parameter
8030052B	readAHBCallback Invalid parameter
8030052C	setFlashDataCallback Invalid parameter

8030052D	getFlashDataCallback Invalid parameter
8030052E	convert void* to ImiDevice* failed!
8030052F	safety params value pointer is NULL
80300530	setSafetyParams safety params size is 0
80300531	convert void* to ImiDevice* failed!
80300532	safety params value pointer is NULL
80300533	setRegistrationParams safety params size is 0
80300534	Get File list Failed
80300535	UploadFile Open File Failed
80300536	Init Upload File Failed
80300537	Write Upload File Failed
80300538	Finish Upload File Failed
80300539	setRegistration Invalid registration mode
8030053A	setRegistrationCallback Invalid parameter
8030053B	getRegistrationCallback Invalid parameter
8030053C	imiUSBShutdown fail!
8030053D	imiUSBRegisterToConnectivityEvents failed!
8030053E	Handlehotplugevent Invalid paramter!
8030053F	openDevice->imiUSBOpenDeviceByfd failed!
80300540	openDevice Invalid device mode.
80300541	closeDevice Failed device is null.
80300542	Ut-ExecuteCMD:invalid parameter,null==pSend
80300543	Ut-ExecuteCMD:invalid parameter,null==pRecv
80300544	Ut-ExecuteCMD:invalid parameter,cmd length error
80300545	Ut-ExecuteCMD:Invalid command head
80300546	Ut-ExecuteCMD:send control msg fail!
80300547	Ut-ExecuteCMD:Receive Timeout
80300548	Ut-ExecuteCMD:return value length error
80300549	imiProtocolSetProperty pValue is null
8030054A	imiProtocolGetProperty return value length error
8030054B	imiProtocolGetSerialNo return value length error
8030054C	imiProtocolGetSafetyParam return value length error
8030054D	imiProtocolSetCmosRigister Invalid nValue, can not be NULL
8030054E	imiProtocolGetCmosRigister return value length error
8030054F	imiProtocolPrivateSet len error
80300550	imiProtocolPrivateGet return value length error
80300551	GetVersion return length value Error
80300552	getLog Open File Failed
80300553	ExecuteCMD sync time failed!
80300554	Invalid pAHBData, can not be NULL
80300555	FileDownload Open File Failed
80300556	imiProtocolReadAHB return value length error
80300557	imiProtocolGetTecData return value length error
80300558	SetFirmwareData Init Upload File error
80300559	SetFirmwareData Write Upload File error

8030055A	SetFirmwareData Finish Upload File error
8030055B	SetFixedParams Init Upload File error
8030055C	SetFixedParams Write Upload File error
8030055D	SetFixedParams Finish Upload File error
8030055E	Open EndPoint Failed
8030055F	Start EndPoint Failed
80300560	Usb Not Started!
80300561	ImiShortProperty_getMemberValue Invalid parameter
80300562	ImiShortProperty_checkIfChanged Invalid parameter
80300563	ImiShortProperty_setMemberValue Invalid parameter
80300564	checkValidOfMode FrameMode not support
80300565	setCurrentAndFirmwareFrameMode FrameMode not support
80300566	imiResolution2Number Invalid resolution
80300567	startImpl fail
80300568	close sensorHW fail
80300569	Stop stream fail!
8030056A	imiProtocolReset error
8030056B	ExecuteCMD Open error
8030056C	ExecuteCMD Close error
8030056D	allocOneFrame fail!
8030056E	Frame Buffer OverFlow
8030056F	m_imiFrame is NULL, Maybe StartBuffer is missing
80300570	pDataHead->nMagic(0x) != 0x Error!
80300571	nBufferSize() != pCurrHeader->nBufSize() error
80300572	processChunk Parameter Error!
80300573	pDataHead->nMagic(0x) != 0x Error!
80300574	processChunk Buffer size() error
80300575	processChunk BufferHead error
80300576	processChunk BufferHead size() error
80300577	Decoder initialize failed!
80300578	setFrameMode error closeSensorHW closed
80300579	ImiStreamImplColor Packet buf ID check error
8030057A	ImiStreamImplColor H264 Packet buf ID check error
8030057B	ImiStreamImplColor m_imiFrame is NULL
8030057C	ImiStreamImplDepth m_imiFrame is NULL
8030057D	ImiStreamImplDepth_setFrameMode error closeSensorHW closed
8030057E	ImiStreamImplDepthSkeleton m_imiFrame is NULL
8030057F	ImiStreamImplIR Packet end ID check error
80300580	ImiStreamImplIR_setFrameMode error closeSensorHW close
80300581	No Use Now
80300582	ImiStreamImplSkeleton_processEndFrame m_imiFrame is NULL
80300583	ImiStreamImplSkeleton Frame Buffer OverFlow frameBuffer is error
80300584	ImiStreamImplSkeleton_setFrameMode error closeSensorHW closed
80300585	ImiStreamImplUserIndexSkeleton m_imiFrame is NULL

80300586	ImiStreamImplUserIndexSkeleton Packet buf ID check error
80300587	ImiStreamImplUserIndexSkeleton_setFrameMode error
80300588	DummyDevice_addProperty Invalid parameter: prop is NULL
80300589	DummyDevice_addPropertys Invalid parameter: props is NULL
8030058A	DummyDevice_setIntPropertyCallback Type is not int!
8030058B	DummyDevice_getIntPropertyCallback Type is not int!
8030058C	DummyDevice_getIntPropertyCallback Invalid len
8030058D	DummyDevice_setCmosRigisterCallback Invalid parameter
8030058E	DummyDevice_getCmosRigisterCallback Invalid parameter
8030058F	DummyDevice_getFirmwareLog Invalid parameter
80300590	DummyDevice_getVersionCallback Invalid parameter
80300591	DummyDevice_writeAHBCallback Invalid parameter
80300592	DummyDevice_readAHBCallback Invalid parameter
80300593	DummyDevice_getDeviceList the pointer is NULL!
80300594	DummyDevice Handlehotplugevent INVALIDPARAM
80300595	Open Device Failed
80300596	openDeviceByFd Open Device Failed
80300597	Device_isConflictWithOpenedStreams Invalid ImiFrameType
80300598	Did not Initialized! Please Initialize First!
80300599	Context Open Device Failed
8030059A	Context Open2 Device Failed
8030059B	Context_imiGetDeviceAttributeByUri Param can't been null
8030059C	CreateStream failed!
8030059D	Start Stream failed
8030059E	Failed to alloc memory size!!!
8030059F	No available Frame!
803005A0	allocOneFrame failed!
803005A1	HTTP get nothing!
803005A2	jsonxx parse no data found!
803005A3	jsonxx parse failed!
803005A4	Failed to createDirectory
803005A5	update ap, init cmd failed!
803005A6	StreamRecorder delete file error
803005A7	StreamRecorder File opened error
803005A8	get current Framamode failed!
803005A9	Start Driver Stream Failed,retcode
803005AA	Error hanpped in Waiting new frame event
803005AB	SensorFrameSync_addSensor NullPointer Input Parameter!
803005AC	SensorFrameSync_removeSensor NullPointer Input Parameter!
803005AD	Error hanpped in Waiting new frame event
803005AE	Device is already opened!

3.1.2. ImiCamera 接口定义

3.1.2.1. API 说明

3.1.2.1.1. imiCamOpen

[功能]

打开 UVC Camera, 仅支持在 Windows 上使用。

[格式]

int32_t imiCamOpen (ImiCameraHandle* pCameraDevice)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle *	pCameraDevice	指向 CameraDevice 的指针	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.2. imiCamOpen2

[功能]

打开 UVC Camera, 仅支持在 Android 平台上使用。

[格式]

int32_t imiCamOpen2(int32_t vid, int32_t pid, int32_t fd, int32_t busnum, int32_t devaddr, const char *usbfs, ImiCameraHandle* pCameraDevice)

[参数]

Type	Name	Description	IN/OUT
int32_t	vid	设备 VendorID	IN
int32_t	pid	设备 ProductID	IN
int32_t	fd	UVC Camera 在 android 上挂在的设备节点描述符	IN
int32_t	busnum	UVC Camera 在 android 上的设备节点编号	IN
int32_t	devaddr	UVC Camera 在 android 上的设备地址	IN
const char *	usbfs	UVC Camera 在 android 上的设备 URI	IN
ImiCameraHandle *	pCameraDevice	指向 CameraDevice 的指针	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

[说明]

Android： 打开 Android Native Camera

3.1.2.1.3. imiCamClose

[功能]

关闭打开的 Camera。

[格式]

int32_t imiCamClose(ImiCameraHandle cameraDevice)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.4. getCamAttrList

[功能]

获取 camera 属性列表。

[格式]

int32_t getCamAttrList(ImiCamAttribute** pList, int32_t* nCount)

[参数]

Type	Name	Description	IN/OUT	
ImiCamAttribute**	pList	Camera 属性列表	OUT	
Int32_t*	nCount	列表的个数	OUT	

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.5. ImiCamOpenURI

[功能]

打开 UVC Camera。

[格式]

int32_t imiCamOpenURI(const char* pURI, ImiCameraHandle* pCameraDevice)

[参数]

Type	Name	Description	IN/OUT	
const char*	pURI	Camera 的 URI	IN	
ImiCameraHandle*	pCameraDevice	Camera 的句柄指针	OUT	

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.6. ImiCamCloseURI

[功能]

关闭 UVC Camera。

[格式]

int32_t imiCamCloseURI(ImiCameraHandle cameraDevice)

[参数]

Type	Name	Description	IN/OUT	
ImiCameraHandle	cameraDevice	Camera 的句柄	IN	

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.7. imiCamGetSupportFrameModes

[功能]

获取相机支持的 Camera frame mode 列表

[格式]

int32_t imiCamGetSupportFrameModes(ImiCameraHandle cameraDevice, const ImiCameraFrameMode** pModes, uint32_t* pNumber)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
const ImiCameraFrameMode**	pModes	支持的 Camera frame mode 列表	OUT
uint32_t*	pNumber	支持的模式个数	OUT

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.8. imiCamGetCurrentFrameMode

[功能]

获取相机当前的 Camera frame mode

[格式]

const ImiCameraFrameMode* imiCamGetCurrentFrameMode(ImiCameraHandle cameraDevice)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN

[返回值]

Value	Description
非空指针	操作成功，指针中存储了相机当前的 Camera frame mode，见 ImiCameraFrameMode
空指针	操作失败

3.1.2.1.9. imiCamSetFrameMode

[功能]

设置相机当前的 Camera frame mode

[格式]

int32_t imiCamSetFrameMode(ImiCameraHandle cameraDevice, ImiCameraFrameMode* pMode)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
ImiCameraFrameMode**	pMode	指向 Camera frame mode 的指针	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.10. imiCamStartStream

[功能]

开启 Camera 数据流。

[格式]

int32_t imiCamStartStream(ImiCameraHandle cameraDevice, const ImiCameraFrameMode* pMode)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
const ImiCameraFrameMode*	pMode	指向 Camera frame mode 的指针	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.11. imiCamStartStream2

[功能]

开启 Camera 数据流。

[格式]

int32_t imiCamStartStream2(ImiCameraHandle cameraDevice)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

[说明]

此时打开的是默认的分辨率或者通过接口 imiCamSetFrameMode 设置的分辨率

3.1.2.1.12. imiCamStopStream

[功能]

关闭 Camera 数据流。

[格式]

int32_t imiCamStopStream(ImiCameraHandle cameraDevice)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.13. imiCamReadNextFrame

[功能]

读取 Camera 数据。

[格式]

int32_t imiCamReadNextFrame(ImiCameraHandle cameraDevice, ImiCameraFrame** pFrame, int32_t timeout)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
ImiCameraFrame**	pFrame	Camera 数据指针	OUT
int32_t	timeout	读取 Camera 数据的超时时间	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.14. imiCamReleaseFrame

[功能]

释放读取的 Camera 数据。

[格式]

int32_t imiCamReleaseFrame(ImiCameraFrame** pFrame)

[参数]

Type	Name	Description	IN/OUT
ImiCameraFrame**	pFrame	Camera 数据指针	IN

[返回值]

Value	Description

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.15. imiCamSetMirror

[功能]

设置 Camera 数据的 Mirror 属性。

[格式]

int32_t imiCamSetMirror(ImiCameraHandle cameraDevice, ImiCAMBOOL bMirror)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
ImiCAMBOOL	bMirror	要设置的 Mirror 状态值	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.10.imiCamSetFramesSync

[功能]

设置深度流和彩色流数据同步开关。

[格式]

int32_t imiCamSetFramesSync(ImiCameraHandle cameraDevice, bool bSync)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
Bool	bSync	开启关闭同步功能	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

[说明]

打开深度流和彩色流之后，设置同步，关闭深度流和彩色流之前，关闭同步

3.1.2.1.11.imiCamSetExposureArea

[功能]

设置曝光区域。

[格式]

int32_t imiCamSetExposureArea(ImiCameraHandle cameraDevice, uint32_t startX, uint32_t startY, uint32_t areaWidth, uint32_t areaHeight)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
uint32	startX	曝光区域起始位置 X 坐标	IN
uint32	startY	曝光区域起始位置 Y 坐标	IN

Type	Name	Description	IN/OUT
uint32	areaWidth	曝光区域宽度	IN
uint32	areaHeight	曝光区域高度	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.12.imiCamClearExposureArea

[功能]

清除设置的曝光区域。

[格式]

int32_t imiCamClearExposureArea(ImiCameraHandle cameraDevice)

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.13.imiCamSetRGB2Gray

[功能]

设置 RGB 彩色是否转灰度输出接口。

[格式]

int32_t imiCamSetRGB2Gray(ImiCameraHandle cameraDevice, bool bEnable);

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN
bool	bEnable	RGB 彩色是否转灰度输出	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.1.14.imiCamSetRotationAngle

[功能]

设置彩色图像的顺时针旋转角度。

[格式]

int32_t imiCamSetRotationAngle(ImiCameraHandle cameraDevice, ImiCameraRotationAngle rotationAngle);

[参数]

Type	Name	Description	IN/OUT
ImiCameraHandle	cameraDevice	Camera Device 句柄	IN

Type	Name	Description	IN/OUT
ImiCameraRotationAngle	rotationAngle	顺时针旋转角度: CAMERA_ROTATION_ANGLE_0: 顺时针旋转 0 度 CAMERA_ROTATION_ANGLE_90： 顺时针旋转 90 度 CAMERA_ROTATION_ANGLE_180： 顺时针旋转 180 度 CAMERA_ROTATION_ANGLE_270 ：顺时针旋转 270 度	IN

[返回值]

Value	Description
0	操作成功
小于 0	操作失败

3.1.2.2. 支持的帧模式列表

Camera Type	Platform	PixelFormat	Resolution	FPS
UVC Camera	Windows	CAMERA_PIXEL_FORMAT_RGB888	1920*1080	30
		CAMERA_PIXEL_FORMAT_RGB888	1280*720	30
		CAMERA_PIXEL_FORMAT_RGB888	640*480	30
		CAMERA_PIXEL_FORMAT_RGB888	320*240	30
	Android	CAMERA_PIXEL_FORMAT_RGB888	1920*1080	30
		CAMERA_PIXEL_FORMAT_RGB888	1280*720	30
		CAMERA_PIXEL_FORMAT_RGB888	640*480	30
		CAMERA_PIXEL_FORMAT_RGB888	320*240	30
		CAMERA_PIXEL_FORMAT_MJPEG	1920*1080	30
		CAMERA_PIXEL_FORMAT_MJPEG	1280*720	30
		CAMERA_PIXEL_FORMAT_MJPEG	640*480	30
		CAMERA_PIXEL_FORMAT_MJPEG	320*240	30

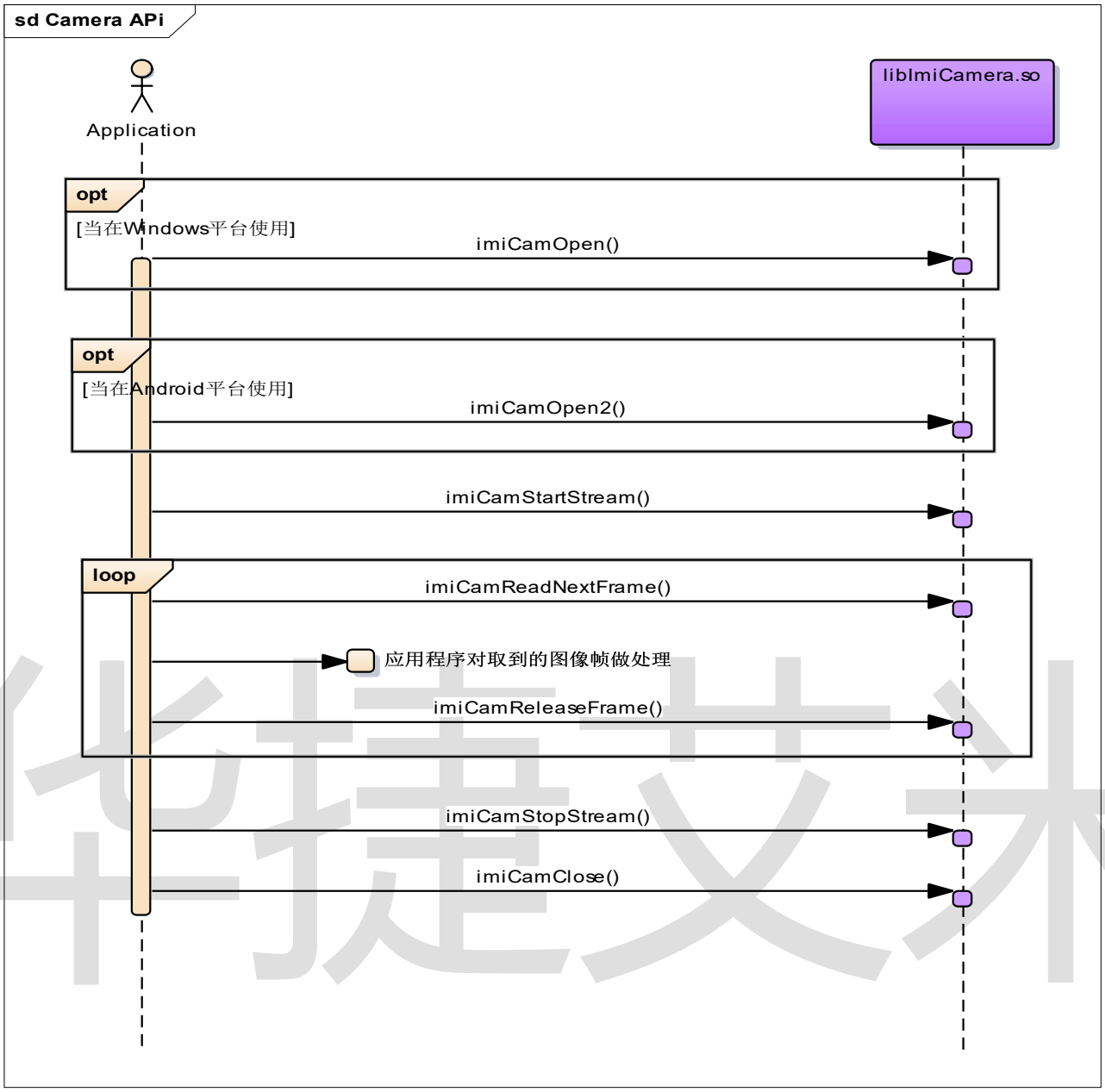
3.1.2.3. 头文件说明

sTable 3.12.3-4 头文件列表

文件名	说 明
include/ ImiCamera.h	Camera API 函数申明
include/ ImiCameraDefines.h	数据结构定义

3.1.2.4. 使用说明

使用 ImiCamera 接口打开 IMI UVC Camera, 使用时序如下图：



4. C 接口编程指引

4.1. 设备介绍

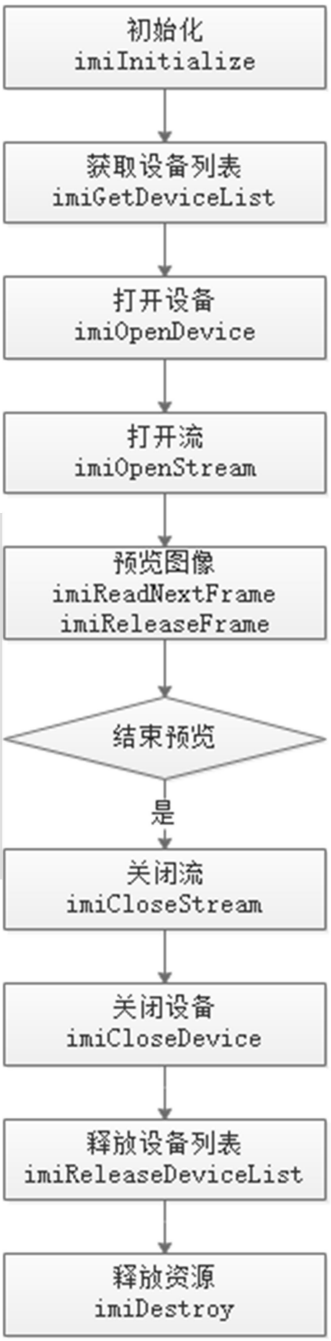
型号	深度流	彩色流		红外流
		UVC	非 UVC	
A100	√		√	
A100M	√	√		
A100M-	√	√		
A100S+	√	√		
A100S	√		√	
A200BL55	√	√		√
A200BL40	√	√		√
A200BL25	√	√		√

4.2. 打开单路流

4.2.1. 打开普通彩色、深度、红外摄像头

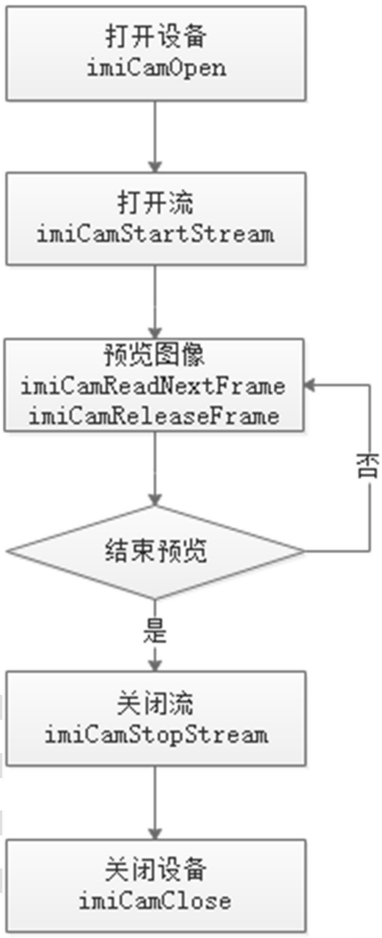
不同的流（color，depth，IR）通过设置 imiOpenStream 的第二个参数（IMI_COLOR_FRAME，IMI_DEPTH_FRAME，IMI_IR_FRAME）来实现。

4.2.1.1. 流程图



4.2.2. 打开 UVC 彩色摄像头

4.2.2.1. 流程图

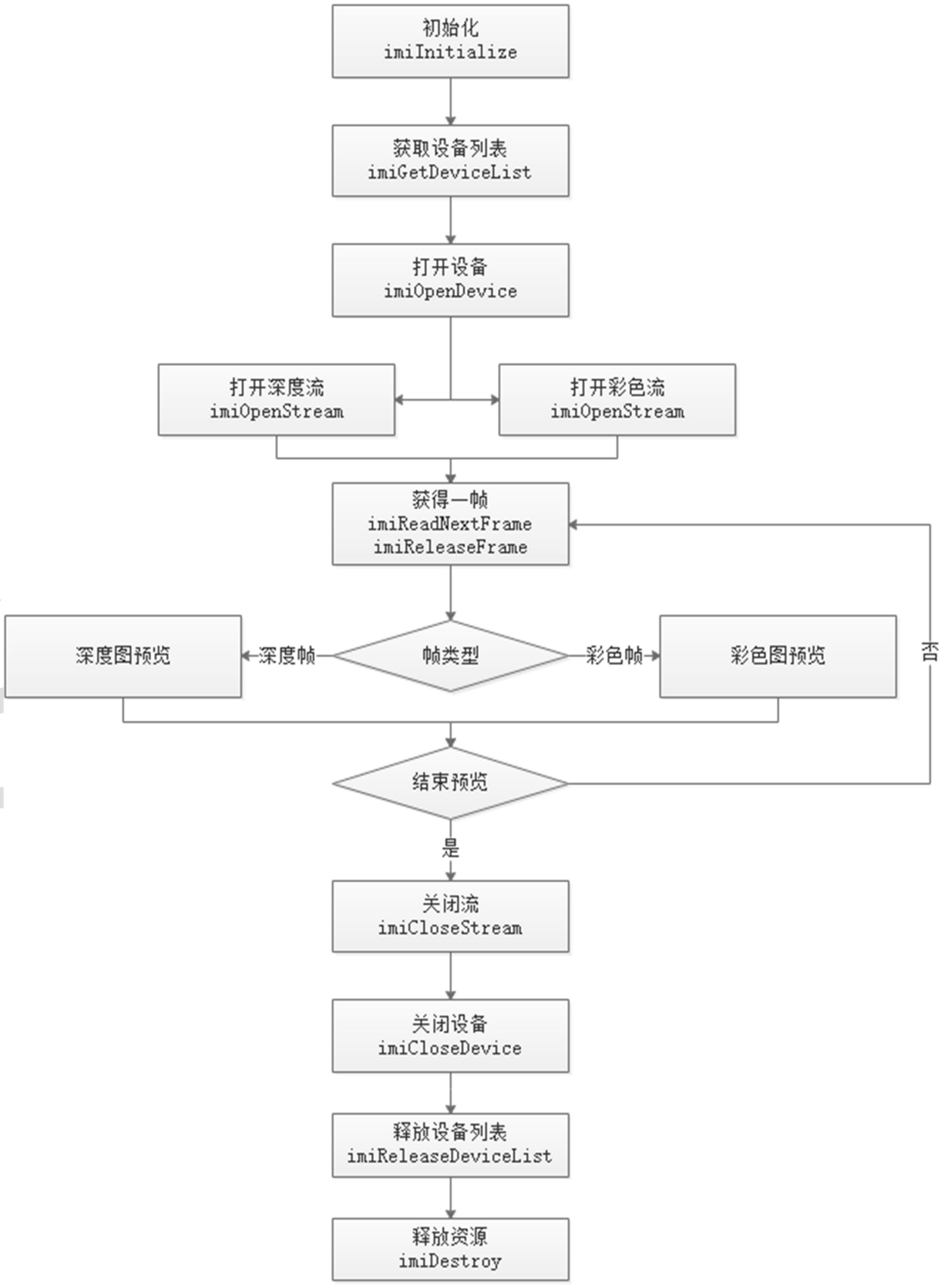


4.3. 打开两路流

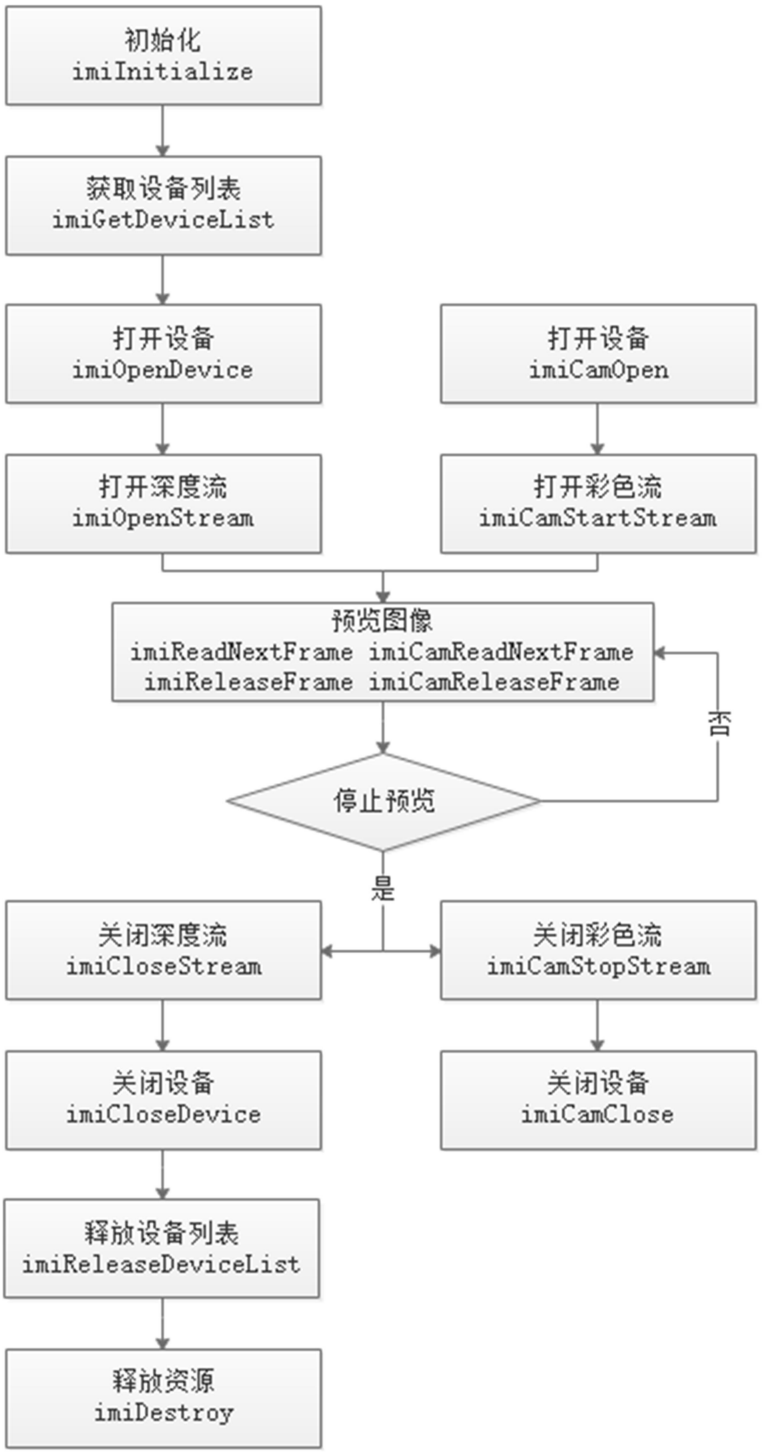
4.3.1. 打开深度流和彩色流

4.3.1.1. 不带 UVC 彩色摄像头的流程图

打开深度图时，imiOpenStream 的第二个参数为 IMI_DEPTH_FRAME；
打开彩色图时，imiOpenStream 的第二个参数为 IMI_COLOR_FRAME。


初始化
imiInitialize
获取设备列表
imiGetDeviceList
打开设备
imiOpenDevice
打开深度流
imiOpenStream
打开彩色流
imiOpenStream
获得一帧
imiReadNextFrame
imiReleaseFrame
深度图预览
深度帧
帧类型
彩色帧
彩色图预览
否
结束预览
是
关闭流
imiCloseStream
关闭设备
imiCloseDevice
释放设备列表
imiReleaseDeviceList
释放资源
imiDestroy

4.3.1.2. 带 UVC 彩色摄像头的流程图

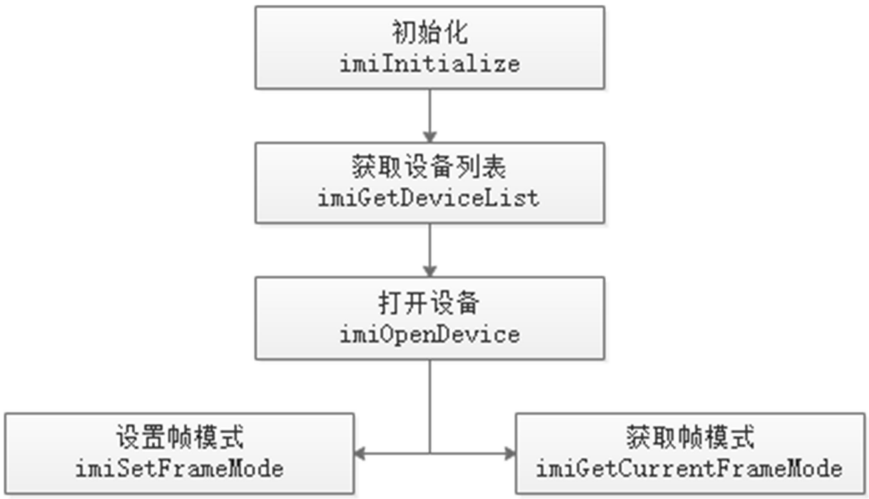


4. 4. 帧模式设置/获取

4.4.1. 不带 UVC 彩色摄像头

通过数据结构 ImiFrameMode 来设置/获取帧模式的像素格式、分辨率、帧率、每像素多少位数据等参数。

4.4.2. 流程图



4.5. 数据处理

4.5.1. 彩色图 YUV 转 RGB

Color 流取到的数据帧如果是 YUV420SP 或 YUV422 格式的数据，可参照下面示例代码进行转换。

4.5.1.1. YUV420SP 转 RGB

```
void YUV420SPToRGB(uint8_t* rgb, uint8_t* yuv420sp, int width, int heigh)
{
	int frameSize = width * heigh;
	int tmp;
	int pos = 0;
	int i, j;
	int uvp;
	int y, u, v, yp;
	int y1192;
	int r, g, b;

	for (j = 0, yp = 0; j < heigh; j++) {
		uvp = frameSize + (j >> 1) * width;
		u = 0;
		v = 0;
		for (i = 0; i < width; i++, yp++) {
			y = (0xff & ((int)yuv420sp[yp])) - 16;
			if (y < 0) y = 0;
			if ((i & 1) == 0) {
				v = (0xff & yuv420sp[uvp++]) - 128;
				u = (0xff & yuv420sp[uvp++]) - 128;
			}

			y1192 = 1192 * y;
			r = (y1192 + 1634 * v);
			g = (y1192 - 833 * v - 400 * u);
			b = (y1192 + 2066 * u);

			if (r < 0) r = 0; else if (r > 262143) r = 262143;
			if (g < 0) g = 0; else if (g > 262143) g = 262143;
			if (b < 0) b = 0; else if (b > 262143) b = 262143;

			tmp = 0xff000000 | ((r << 6) & 0xff0000) | ((g >> 2) & 0xff00) | ((b >> 10) &
```

```
0xff);
                        *(rgb+pos) = (tmp >> 16) & 0xff;
                        pos++;
                        *(rgb+pos) = (tmp >> 8) & 0xff;
                        pos++;
                        *(rgb+pos) = (tmp) & 0xff;
                        pos++;
                }
        }
}
```

4.5.1.2. YUV422 转 RGB

```
void YUV422ToRGB(uint8_t *rgb, uint8_t *yuv422, uint32_t width, uint32_t height)
{
        uint32_t in, out = 0;
        uint32_t pixel_16;
        uint8_t pixel_24[3];
        uint32_t pixel32;
        int y0, u, y1, v;

        for(in = 0; in < width * height * 2; in += 4)
        {
                pixel_16 = yuv422[in + 3] << 24 |
                           yuv422[in + 2] << 16 |
                           yuv422[in + 1] <<  8 |
                           yuv422[in + 0];
                y0 = (pixel_16 & 0x000000ff);
                u  = (pixel_16 & 0x0000ff00) >>  8;
                y1 = (pixel_16 & 0x00ff0000) >> 16;
                v  = (pixel_16 & 0xff000000) >> 24;
                pixel32 = convert_yuv_to_rgb_pixel(y0, u, v);
                pixel_24[0] = (pixel32 & 0x000000ff);
                pixel_24[1] = (pixel32 & 0x0000ff00) >> 8;
                pixel_24[2] = (pixel32 & 0x00ff0000) >> 16;
                rgb[out++] = pixel_24[0];
                rgb[out++] = pixel_24[1];
                rgb[out++] = pixel_24[2];
                pixel32 = convert_yuv_to_rgb_pixel(y1, u, v);
                pixel_24[0] = (pixel32 & 0x000000ff);
                pixel_24[1] = (pixel32 & 0x0000ff00) >> 8;
                pixel_24[2] = (pixel32 & 0x00ff0000) >> 16;
                rgb[out++] = pixel_24[0];
                rgb[out++] = pixel_24[1];
                rgb[out++] = pixel_24[2];
        }
}
```

4.5.2. 深度图数据转换 RGB

```
#define MAX_DEPTH                   10000
#define IMAGE_WIDTH                 640
#define IMAGE_HEIGHT                480

void calculateHistogram(float* pHistogram, int histogramSize, const ImiImageFrame* frame)
{
    const uint16_t* pDepth = (const uint16_t*)frame->pData;
```

```
    memset(pHistogram, 0, histogramSize*sizeof(float));

    int height = frame->height;
    int width = frame->width;

    unsigned int nNumberOfPoints = 0;
    for (int y = 0; y < height; ++y)
    {
        for (int x = 0; x < width; ++x, ++pDepth)
        {
            if (*pDepth != 0)
            {
                pHistogram[*pDepth]++;
                nNumberOfPoints++;
            }
        }
    }

    for (int nIndex=1; nIndex<histogramSize; nIndex++)
    {
        pHistogram[nIndex] += pHistogram[nIndex-1];
    }

    if (nNumberOfPoints)
    {
        for (int nIndex=1; nIndex<histogramSize; nIndex++)
        {
            pHistogram[nIndex] = (256 * (1.0f - (pHistogram[nIndex] / nNumberOfPoints)));
        }
    }
}

void depth2RGB()
{
         static float              g_depthHist[MAX_DEPTH];
         static RGB888Pixel        g_rgbImage[1280 * 1024];
         calculateHistogram(g_depthHist, MAX_DEPTH, pFrame);

         uint32_t rgbSize;
         uint16_t * pde = (uint16_t*)pFrame->pData;
         for (rgbSize = 0; rgbSize < pFrame->size/2; ++rgbSize)
{
                  g_rgbImage[rgbSize].r = g_depthHist[pde[rgbSize]];
                  g_rgbImage[rgbSize].g = g_rgbImage[rgbSize].r;
                  g_rgbImage[rgbSize].b = 0;//display yellow
         }
}
```

4.5.3. 数据帧保存

4.5.3.1. 保存图片

本例用于 UVC 设备，将彩色图的 RGB 裸数据加个 bmp 的头，保存为图片。

```
int32_t takePhoto(const char* bmpImagePath, const ImiCameraFrame* pframe)
{
```

```
    BMPFILEHEADER_T        bmfh;                    // bitmap file header
    BMPINFOHEADER_T        bmih;                    // bitmap info header (windows)

    const int OffBits = 54;

    int32_t imagePixSize = pframe->width * pframe->height;

    bmfh.bfReserved1 = 0;
    bmfh.bfReserved2 = 0;
    bmfh.bfType = 0x4d42;
    bmfh.bfOffBits = OffBits;               // 头部信息 54 字节
    bmfh.bfSize = imagePixSize * 3 + OffBits;

    memset(&bmih, 0, sizeof(BMPINFOHEADER_T));
    bmih.biSize = 40;                                // 结构体大小为 40
    bmih.biPlanes = 1;
    bmih.biSizeImage = imagePixSize * 3;
    bmih.biBitCount = 24;
    bmih.biCompression = 0;
    bmih.biWidth = pframe->width;
    bmih.biHeight = -pframe->height;


    memcpy( (void*)g_bmpColor, pframe->pData, imagePixSize*3);

    //rgb->bgr
    for( int i = 0; i < imagePixSize; ++i )
    {
        char r = g_bmpColor[3*i + 2];
        g_bmpColor[3*i+2] = g_bmpColor[3*i];
        g_bmpColor[3*i] = r;
    }

    char buf[128]= {0};
    std::string fullPath = bmpImagePath;


    FILE* pSaveBmp = fopen(fullPath.c_str(), "wb");
    if( NULL == pSaveBmp)
    {
        return -1;
    }

    fwrite( &bmfh, 8, 1,  pSaveBmp );
    fwrite( &bmfh.bfReserved2, sizeof(bmfh.bfReserved2), 1, pSaveBmp);
    fwrite( &bmfh.bfOffBits, sizeof(bmfh.bfOffBits), 1, pSaveBmp);
    fwrite( &bmih, sizeof(BMPINFOHEADER_T), 1, pSaveBmp );
    fwrite( g_bmpColor, imagePixSize*3, 1, pSaveBmp);

    fclose(pSaveBmp);

    return 0;
}
```

4.5.3.2. 保存文件

本例用于 UVC 设备，将彩色图的 RGB 裸数据保存到文件。

```
bool gStop = false;
void saveFile()
```

```
{
        FILE* pFile = fopen("Color_reg.raw", "wb");
        if (!pFile)
        {
                return;
        }
        for (; ;)
        {
                if (gStop)
                {
                        break;
                }
                fwrite(pCamFrame->pData, pCamFrame->size, 1, pFile);
        }

        fclose(pFile);
}
```

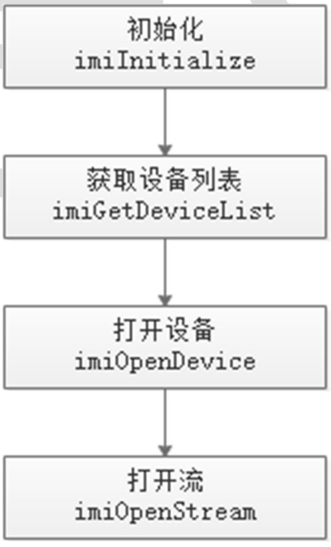
4.6. 资源申请与回收

4.6.1. 用完即回收

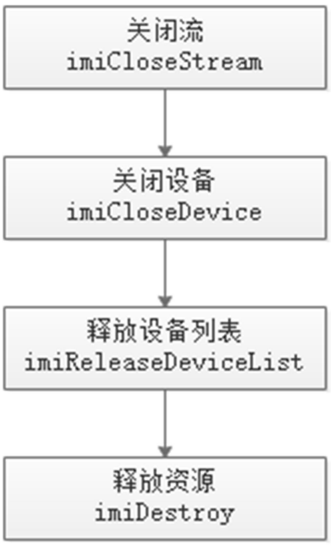
读取一帧数据，处理完后即释放。非 UVC 设备的 imiReadNextFrame 和 imiFrameRelease。UVC 设备的 imiCamReadNextFrame 和 imiCamReleaseFrame。

4.6.2. 程序结束前回收

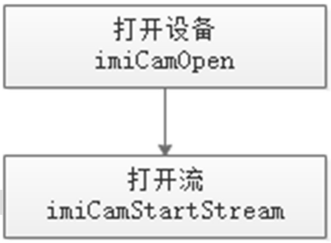
非 UVC 设备程序运行时资源申请的流程



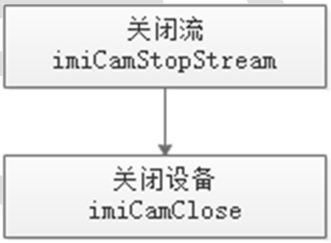
非 UVC 设备程序结束时资源释放的流程



UVC 设备程序运行时资源申请的流程



UVC 设备程序结束时资源释放的流程



4.7. 设备属性

4.7.1. UVC 设备属性 ID 列表

打开设备后再进行操作。UVC 设备通过 imiCamSetProperty 和 imiCamGetProperty 来获取和设置属性。属性 ID 的枚举常量如下：

属性 ID	描述
IMI_CAM_PROPERTY_COLOR_AE_MODE	AE 模式
IMI_CAM_PROPERTY_COLOR_EXPOSURE_ABSOLUTE_TIME	彩色曝光时间
IMI_CAM_PROPERTY_COLOR_BACKLIGHT_COMPENSATION	刷脸/扫码模式切换
IMI_CAM_PROPERTY_COLOR_FOCUS_ABS	彩色相机焦点
IMI_CAM_PROPERTY_COLOR_IRIS_ABS	彩色相机光圈
IMI_CAM_PROPERTY_COLOR_ZOOM_ABS	彩色相机缩放
IMI_CAM_PROPERTY_COLOR_PAN_ABS	彩色相机全景
IMI_CAM_PROPERTY_COLOR_TILT_ABS	彩色相机倾斜
IMI_CAM_PROPERTY_COLOR_ROLL_ABS	彩色相机滚动
IMI_CAM_PROPERTY_COLOR_BRIGHTNESS	彩色图像亮度
IMI_CAM_PROPERTY_COLOR_CONTRAST	彩色图像对比度
IMI_CAM_PROPERTY_COLOR_GAIN	彩色图像增益
IMI_CAM_PROPERTY_COLOR_HUE	彩色图像色调

IMI_CAM_PROPERTY_COLOR_SATURATION	彩色图像饱和度
IMI_CAM_PROPERTY_COLOR_SHARPNESS	彩色图像清晰度
IMI_CAM_PROPERTY_COLOR_GAMMA	彩色图像伽玛
IMI_CAM_PROPERTY_COLOR_WHITE_BALANCE_TEMPERATURE	彩色图像白平衡

4.7.2. 非 UVC 设备属性 ID 列表

初始化并打开设备后再进行操作。非 UVC 设备通过 imiGetDeviceProperty 和 imiSetDeviceProperty 来获取和设置属性。属性 ID 的枚举常量如下：

属性 ID	描述
IMI_PROPERTY_GENERAL_VERSION	版本号
IMI_PROPERTY_GENERAL_SERIAL_NUMBER	序列号
IMI_PROPERTY_IMAGE_REGISTRATION	配准
IMI_PROPERTY_IR_DIST	泛光源红外距离设置
IMI_PROPERTY_COLOR_MIRROR	彩色图图像镜像
IMI_PROPERTY_COLOR_INTRINSIC_PARAMS	彩色相机内参
IMI_PROPERTY_DEPTH_HOLE_FILTER	深度图填洞
IMI_PROPERTY_DEPTH_MIRROR	深度图图像镜像
IMI_PROPERTY_DEPTH_DENOISE	深度图去小块
IMI_PROPERTY_DEPTH_INTRINSIC_PARAMS	深度相机内参
IMI_PROPERTY_IR_MIRROR	红外图像镜像
IMI_PROPERTY_IR_INTRINSIC_PARAMS	红外相机内参
IMI_PROPERTY_SKELETON_MIRROR	骨架图像镜像
IMI_PROPERTY_GROUND_EQUATION	深度图地面方程
IMI_PROPERTY_GROUND_CLEANUP	深度图清地面
IMI_PROPERTY_LD_OPERATE	激光发射器开关
IMI_PROPERTY_FLOODLIGHT	泛光源开关
IMI_PROPERTY_LASER_SAFETY_MODE	安全模式
IMI_PROPERTY_SAFETY_DIST	安全距离
IMI_PROPERTY_LIGHT_THRESHOLD	环境光阈值
IMI_PROPERTY_AMBIENT_LIGHT_MODE	环境光模式
IMI_PROPERTY_REAL_SAFETY_DIST	实时安全距离
IMI_PROPERTY_REAL_LIGHT_THRESHOLD	实时环境光阈值
IMI_PROPERTY_DEPTH_IR_MIRROR	深度+红外图像镜像
IMI_PROPERTY_RESET_DEVICE	重启设备
IMI_PROPERTY_DEVICE_SWITCH_POWER	设备电流 0.8A 和 1.3A 切换
IMI_PROPERTY_ROTATE_TRANS_INTRINSIC_PARAMS	相机外参
IMI_PROPERTY_IR_ENABLE	红外 AE 使能设置接口
IMI_PROPERTY_CAMERA_PARAMS_4_USER	专门获取相机工作参数接口
IMI_PROPERTY_IR_RECT	设置软件 IRAE rect 接口

4.7.3. 获取属性

4.7.3.1. 获取序列号

非 UVC 设备属性 ID 中的值都可以通过 imiGetDeviceProperty 来获取，以获取序列号为例，方法如下：

```
char szSerialNum[64] = {0};
uint32_t serialSize = 64;
imiGetDeviceProperty(m_device, IMI_PROPERTY_GENERAL_SERIAL_NUMBER, szSerialNum, &serialSize);
```

4.7.3.2. 版本号查询

不同于 4.7.1.1，本属性通过独立接口获取。适用于非 UVC 设备，需初始化并打开设备后再进行操作，可查询 firmware（整型）、hardware（整型）、uvcColor（字符串）、bridgefw（字符串）和 sdk（整型）等的版本号。

```
ImiVersions verInfo;
memset(&verInfo, 0, sizeof(verInfo));
imiGetVersion(m_device, & verInfo);
firmware: verInfo.fw_version.major, verInfo.fw_version.minor, verInfo.fw_version.revision,
          verInfo.fw_version.ap_major, verInfo.fw_version.ap_minor, verInfo.fw_version.ap_version,

hardware: verInfo.hw_version.hardware, verInfo.hw_version.chip, verInfo.hw_version.ap_hardware,

uvcColor: verInfo.uvc_color_version,

bridgefw: verInfo.bridge_fw_version,

sdk: verInfo.sdk_version.major, verInfo.sdk_version.minor, verInfo.sdk_version.revision
```

4.7.4. 设置属性

4.7.4.1. 设置镜像

非 UVC 设备设置彩色图图像镜像的方法如下：

```
bool bColorMirror = true;
imiSetDeviceProperty(m_device, IMI_PROPERTY_COLOR_MIRROR, &bColorMirror, 1);
```

UVC 设备设置彩色图图像镜像的方法如下：

```
imiCamSetMirror(cameraDevice, true);
```

4.7.4.2. 设置去小块

```
uint8_t open = 1;
imiSetDeviceProperty(m_device,  IMI_PROPERTY_DEPTH_DENOISE, &open, sizeof(uint8_t));
```

4.7.4.3. 设置帧同步

只针对 55 基线的 A200M 适用，方法如下：

```
imiCamSetFramesSync(cameraDevice, true);
```

4.7.4.4. 打开配准

```
imiSetImageRegistration(m_device, true);
```

4.7.4.5. 重启设备

```
uint8_t value= 1;
imiSetDeviceProperty(m_device,  IMI_PROPERTY_RESET_DEVICE, &value, sizeof(uint8_t));
```

4.7.4.6. 设备电流切换

```
uint8_t value= 0x01;
imiSetDeviceProperty(m_device,  IMI_PROPERTY_DEVICE_SWITCH_POWER, &value, sizeof(uint8_t)); // 1.3A

value= 0x00;
imiSetDeviceProperty(m_device,  IMI_PROPERTY_DEVICE_SWITCH_POWER, &value, sizeof(uint8_t)); // 0.8A
```

4.7.4.7. 打开红外 AE 功能

```
imiSetIRFaceAEEnable(m_device,true);
```

4.8 获取设备支持的能力

```
ImiSupportCapacity pSupportCapacity;
imiGetSupportCapacity(m_device,&pSupportCapacity);
```

5. 帮助

5.1. 注意事项

接口中带 Cam 字样的适用于 UVC 设备。

5.2. 常见问题

平台	问题	解决办法
windows	识别不了 IMI 3D Sensor 设备	先卸载 IMI 设备驱动，然后再安装 IMI 设备驱动，驱动安装和卸载程序在 Hjimi\ImiSDK\Driver 文件夹。
	x64 平台找不到 lib 库	在 SDK 安装目录，以管理员权限执行 Tools\script\copy_x64_libs.bat，所有依赖的库都会自动拷贝到 imix64libs 文件夹，用户可以根据自己需要放置 64 位库。
linux	Can not execute binary file:Exec format error	1.查看系统架构是 arm 还是 x86，与 sdk 版本是否一致 2.查看系统是 32 位还是 64 位，与 sdk 版本是否一致 3.查看命令 uname -a
	找不到运行依赖的库	设置环境变量 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:XXX XXX 代表库路径
	虚拟机下，设备正常打开，但是数据没上来	检查虚拟机的 USB 设置，是 2.0 还是 3.0，使用 A200 系列的摄像头时需要把虚拟机 USB 设置成 3.0
	运行 Color Depth Viewer，设备打开正常，但是没有画面显示	ColorDepthViewer 打开的是 1180 的分辨率的彩色图，查看设备是否支持
	系统安装的 libusb 与 sdk 中的 libusb 冲突	1.确定 sdk 版本，如果是 1.6.5 之前的版本请参看下一条 2.删除 sdk 中的 libusb，libudev 文件