

华为 HiLens

SDK 参考

文档版本 01
发布日期 2023-05-30



版权所有 © 华为技术有限公司 2023。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <https://www.huawei.com>

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目录

1 文档导读	1
2 HiLens Framework_C++简介	2
3 准备工作	4
3.1 环境准备	4
3.2 SDK 下载	5
4 初始化	6
4.1 初始化 HiLens Framework	6
4.2 释放 HiLens Framework	6
5 视频输入模块	8
5.1 输入模块简介	8
5.2 视频采集器	8
5.3 读取摄像头视频帧	10
5.4 获取视频的宽度	10
5.5 获取视频的高度	10
6 音频输入模块	11
6.1 输入模块简介	11
6.2 音频采集器	11
6.3 读取音频数据	12
7 预处理	14
7.1 预处理模块简介	14
7.2 构造图像预处理器	14
7.3 改变图片尺寸	14
7.4 裁剪图片	15
7.5 转换图片颜色格式	16
8 模型管理	18
8.1 模型管理简介	18
8.2 创建模型	18
8.3 模型推理	19
9 输出模块	21
9.1 输出模块简介	21

9.2 构造用于输出的显示器.....	21
9.3 输出一帧图片.....	22
9.4 上传文件.....	22
9.5 上传缓冲区数据.....	23
9.6 发送 POST 请求.....	24
9.7 发送一条消息.....	25
9.8 播放音频文件.....	26
10 资源管理.....	28
10.1 获取模型路径.....	28
10.2 获得技能工作区目录.....	28
10.3 获得技能配置.....	29
10.4 下载 OBS 文件.....	29
10.5 计算文件的 md5 值.....	29
10.6 示例-资源管理.....	30
11 难例上传模块.....	32
11.1 难例上传介绍及说明.....	32
11.2 构造 HardSample 实例.....	33
11.3 初始化.....	33
11.4 难例图片判断.....	34
11.5 获取难例配置.....	35
11.6 更新难例配置.....	35
12 日志模块.....	36
12.1 设置打印日志的级别.....	36
12.2 打印 Trace 级别的日志.....	37
12.3 打印 Debug 级别的日志.....	37
12.4 打印 Info 级别的日志.....	38
12.5 打印 Warning 级别的日志.....	39
12.6 打印 Error 级别的日志.....	39
12.7 打印 Fatal 级别的日志.....	40
13 错误码.....	42

1 文档导读

文档指导您如何安装和配置开发环境、如何通过调用HiLens Framework SDK提供的接口函数进行二次开发。

表 1-1 文档导读

章节	说明
HiLens Framework_C++简介	快速了解HiLens Framework开发工具包。
环境准备	介绍使用HiLens Framework开发工具包的准备工作。
视频输入模块 音频输入模块 预处理 模型管理 输出模块 资源管理 日志模块	HiLens Framework封装的类和函数详细说明。
错误码	介绍使用HiLens Framework SDK时错误码以及建议解决方法。

2 HiLens Framework_C++简介

HiLens Framework开发工具包（HiLens Framework SDK，HiLens Framework Software Development Kit）是HiLens Kit上运行的HiLens Framework的c++开发包，使用户可以开发c++版本的技能并在HiLens Kit上运行。

HiLens Framework 简介

HiLens Framework通过封装底层接口、实现常用的管理功能，让开发者可以在HiLens管理控制台上方便地开发技能，培育AI生态。

HiLens Framework的分层结构如图2-1所示，HiLens Framework封装了底层的多媒体处理库（摄像头/麦克风驱动模块Media_mini），以及D芯片相关的图像处理库（DVPP）和模型管理库（ModelManager），另外开发者也可以使用熟悉的视觉处理库OpenCV。在此之上，HiLens Framework提供了以下6个模块供开发者使用，方便开发诸如人形检测、疲劳驾驶检测等技能，模块说明如表2-1所示。

图 2-1 HiLens Framework 框架

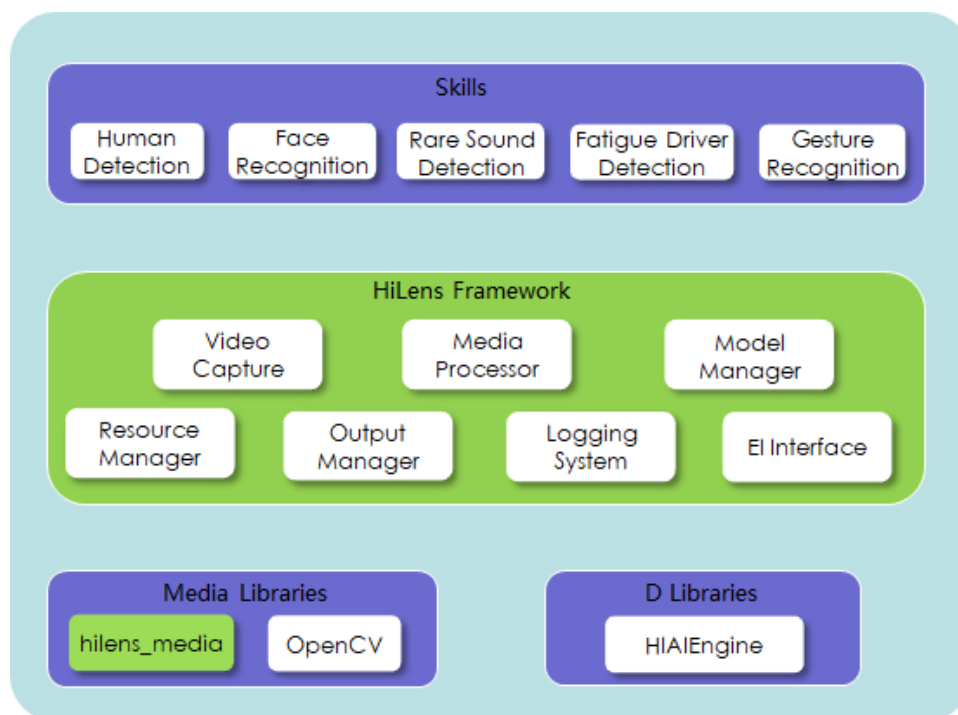


表 2-1 模块说明

序号	模块	功能
1	Input Manager	负责视频、音频等输入数据的接入管理。
2	Media Processor	负责视频、音频等媒体数据的处理。
3	Model Manager	负责模型的初始化与推理任务。
4	Output Manager	负责流、文件、消息通知等输出任务的管理。
5	Resource Manager	负责文件、图片、模型等资源的路径管理。
6	Logging System	负责日志系统管理。

3 准备工作

3.1 环境准备

环境搭建

1. 查看您的编译机Linux系统环境。
执行**uname -a**可查看Linux系统信息,例如查看x86_64的ubuntu系统信息:
Linux ubuntu 4.4.0-144-generic #170-Ubuntu SMP Thu Mar 14 11:56:20 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
2. 查看端侧设备HiLens Kit的系统环境。
登录端侧设备的系统环境请参见用户指南>使用SSH连接到HiLens Kit。
例执**uname -a**，可得如下HiLens Kit系统环境：
Linux Euler 4.19.36-vhulk1907.1.0.h448.eulerosv2r8.aarch #1 SMP Mon Jul 22 00.00.00 UTC 2019 aarch64 aarch64 aarch64 GNU/Linux
3. 下载交叉编译工具，并解压下载的压缩包。
如果您的编译机是x86_64的Linux系统，可以直接下载[交叉编译工具](#)，里面已经包含了HiLens Kit上需要的库。
4. 进入交叉编译工具目录后，执行指令**pwd**获取路径位置，编译时设置编译器路径。
如果您的编译机是x86_64的Linux系统，路径位置是“.../aarch64-linux-gnu-gcc-7.3.0”，则编译时设置路径为：
export CC=".../aarch64-linux-gnu-gcc-7.3.0/bin/aarch64-linux-gnu-gcc"
export CXX=".../aarch64-linux-gnu-gcc-7.3.0/bin/aarch64-linux-gnu-g++"
详细的编译指导请参考步骤6中的示例代码。
5. 下载HiLens Framework SDK开发包：“cloud-c-sdk-HiLensFramework-1.0.4.tar.gz”，并解压，重命名文件名。
下载地址请参见[SDK下载](#)。
6. 参考示例程序和接口调用说明进行代码开发，编译指导和示例代码说明，详见示例代码中的“README.md”。
请单击[下载HiLens Framework示例代码](#)。

 说明

搭建环境时，下载SDK开发包和参考示例程序进行代码开发必须在编译机上进行安装、调试。

3.2 SDK 下载

请根据HiLens Framework固件版本号选择SDK软件包下载，如表3-1所示。HiLens Framework固件版本号请登录华为HiLens控制台，在“设备管理>设备列表”页面[查看固件版本信息](#)。

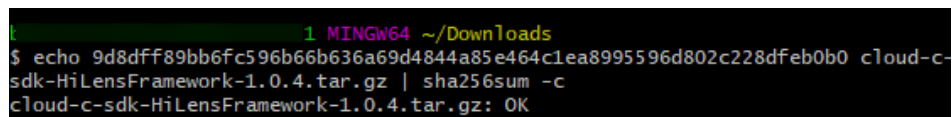
表 3-1 SDK 下载

固件版本	SDK下载
1.0.6及以下版本	请单击 下载HiLens Framework SDK (sha256: 9d8dff89bb6fc596b66b636a69d4844a85e464c1ea8995596d802c228dfef0b0)
1.0.7版本	请单击 下载HiLens Framework SDK (sha256: 690535e4682b8f008cf48cfb3ea698950316723a2bd8d2bdd20c9b92d66ed36e)
1.0.8版本	请单击 下载HiLens Framework SDK (sha256: 36a4476fe5dc0766c5a3e4d53792f54be55a69548eb43a9d63fa6cb5a312e1fb)
1.1.0及以上版本	请单击 下载HiLens Framework SDK (sha256: 57278a0d10d37031dd158094a3930dc9ba867b9d0362c833c9bf830a258fc94f)

SDK下载完成后，在文件存放目录完成签名校验（Windows系统推荐使用Git工具），验证成功将返回OK。命令如下：

```
echo [sha256值] [固件包] | sha256sum -c
```

图 3-1 签名校验示例



```
1 MINGW64 ~/Downloads
$ echo 9d8dff89bb6fc596b66b636a69d4844a85e464c1ea8995596d802c228dfef0b0 cloud-c-
sdk-HiLensFramework-1.0.4.tar.gz | sha256sum -c
cloud-c-sdk-HiLensFramework-1.0.4.tar.gz: OK
```

4 初始化

4.1 初始化 HiLens Framework

该接口用于初始化HiLens Framework。在调用HiLens Framework的其他接口之前，需要先做全局初始化。

接口调用

HiLensEC hilens::Init(const std::string & verify)

参数说明

表 4-1 参数说明

参数名	说明
verify	应与华为HiLens管理控制台上新建技能时，所填写的“基本信息”中的“检验值”一致。如果不一致，HiLens Framework会强制技能停止。

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

4.2 释放 HiLens Framework

调用下面的接口来释放相关资源。

接口调用

HiLensEC hilens::Terminate()

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

5 视频输入模块

5.1 输入模块简介

hilens::VideoCapture 类

使用视频采集器来读取本地摄像头或IP摄像头的数据。

```
#include <video_capture.h>
```

构造及析构函数

```
~VideoCapture()  
virtual hilens::VideoCapture::~VideoCapture()
```

5.2 视频采集器

本地摄像头

构造视频采集器（本地摄像头），如果创建失败可以查看技能日志或输出来定位错误原因。

- **接口调用**
static std::shared_ptr<VideoCapture> hilens::VideoCapture::Create()
- **返回值**
成功则返回视频采集器实例, 失败则返回nullptr。

IP 摄像头

构造视频采集器（IP摄像头），如果创建失败可以查看技能日志或输出来定位错误原因。

- **接口调用**
static std::shared_ptr<VideoCapture> hilens::VideoCapture::Create(const std::string & name)

```
static std::shared_ptr<VideoCapture> hilens::VideoCapture::Create(const
std::string & name, const unsigned int width, const unsigned int height)
```

- **参数说明**

表 5-1 参数说明

参数名	说明
name	设备配置中的摄像头名（设备配置中的IPC）。优先读取设备配置中的摄像头名称，也可以直接传入形如rtsp://xxx的取流地址。1.0.7及以后版本支持直接读取本地MP4文件，且支持设置读取到的视频帧宽高。 配置摄像头的名称可在华为HiLens管理控制台上配置，详情请参见用户指南>配置摄像头。
width	设置读取到的视频帧图片宽度（要求为16的倍数，推荐为32的倍数，且最小为128），仅1.0.7及以后版本支持。
height	设置读取到的视频帧图片高度（要求为2的倍数，且最小为128），仅1.0.7及以后版本支持。

USB 摄像头

构造视频采集器（USB摄像头），如果创建失败则抛出一个CreateError，开发者可以查看技能日志或输出来定位错误原因。目前只支持插入一路UVC摄像头，摄像头ID为0。

- **接口调用**

```
static std::shared_ptr hilens::VideoCapture::Create(int dev)
```

- **参数说明**

表 5-2 参数说明

参数名	说明
dev	HiLens Kit系统中“/dev”中的UVC摄像头ID。 HiLens Kit系统可通过SSH登录，详情请参见用户指南>使用SSH连接到HiLens Kit。

- **返回值**

成功则返回视频采集器实例,失败则返回nullptr。

5.3 读取摄像头视频帧

读取一帧视频。如果摄像头读取发生错误，此接口将会抛出一个异常 (std::runtime_error)。

接口调用

```
virtual cv::Mat hilens::VideoCapture::Read()
```

返回值

如果是IPC或本地摄像头，则返回的是YUV_NV21的数据，如果是UVC摄像头，则返回BGR数据。

5.4 获取视频的宽度

返回视频宽度。

接口调用

```
virtual int hilens::VideoCapture::Width()
```

返回值

视频宽度。

5.5 获取视频的高度

返回视频高度。

接口调用

```
virtual int hilens::VideoCapture::Height()
```

返回值

视频高度。

6 音频输入模块

6.1 输入模块简介

hilens::AudioCapture 类

使用音频采集器来读取本地音频文件的数据，相关头文件已集成到“hilens.h”。

```
#include <hilens.h>
```

构造及析构函数

```
~AudioCapture()  
virtual hilens::AudioCapture::~AudioCapture()
```

6.2 音频采集器

本地音频文件

构造音频采集器，如果创建失败可以查看技能日志或输出来定位错误原因，本地麦克风使用默认参数采集数据，采样率44100，位宽16bit，双声道采集，每一帧采样点数1024。

- **接口调用**
 - 1.0.8及以上固件版本
static std::shared_ptr<AudioCapture> hilens::AudioCapture::Create(const std::string filePath)
 - 1.1.0及以上固件版本
static std::shared_ptr<AudioCapture> Create(const struct AudioProperties& property)
- **参数说明**

表 6-1 参数说明

参数名	说明
filePath	参数为音频文件在HiLens Kit设备上的绝对路径(不支持中文)时, 从该文件获取音频数据。
property	本地麦克风录音参数。结构体定义如下: <pre>struct AudioProperties{ unsigned int enSamplerate; unsigned int enBitwidth; unsigned int u32PtNumPerFrm; unsigned int soundMode; }</pre> 结构体各成员取值范围: <ul style="list-style-type: none"> ● enSamplerate (采样率): 可取值8000, 12000, 11025, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 96000。 ● enBitwidth (采样位宽): 取值1 (表示位宽16bit)。 ● u32PtNumPerFrm (每帧采样点数): 取值范围[80, 2048]。 ● soundMode (声道模式): 取值0 (单声道)和1 (双声道)。

说明

- 每帧的采样点个数和采样率enSamplerate的取值决定了硬件产生中断的频率, 频率过高会影响系统的性能, 跟其他业务也会相互影响, 建议这两个参数的取值满足算式: “(u32PtNumPerFrm * 1000)/enSamplerate >=10”, 比如在采样率为16000Hz时, 建议设置采样点个数大于或者等于 160。
- 本地麦克风只有一个, 不支持多个进程设置不同录音参数, 先设置的生效, 后设置的如果参数有不同, 会设置失败。
- 本接口与[播放音频文件](#)的接口不可同时调用。
- **返回值**
成功则返回音频采集器实例,失败则返回nullptr。

6.3 读取音频数据

读取一帧或者多帧音频。仅支持1.0.8及以上固件版本。

- **接口调用**
virtual int hilens::AudioCapture::Read(AudioFrame &frames, int n=1)
- **参数说明**
结构体**AudioFrame**定义如下, 参数如[表6-2](#)所示。

```
typedef struct AudioFrame_s{
    std::shared_ptr<void> data;
```



```
unsigned int size;
}AudioFrame;
```

表 6-2 参数说明

参数名	说明
data	输出参数，存放读取到音频数据的智能指针。
size	输出参数，读取到音频数据的大小。
n	输入参数，一次读取音频帧数，最大不超过512。

- 返回值**
 成功返回0，失败则返回-1，失败时可通过日志查看原因。

7 预处理

7.1 预处理模块简介

hilens::Preprocessor 类

硬件加速的预处理器

```
#include <media_process.h>
```

析构函数

```
~Preprocessor()  
virtual hilens::Preprocessor::~Preprocessor()
```

7.2 构造图像预处理器

构造并初始化一个预处理器,用于进行Resize/Crop操作（3559硬件加速）。如果失败可以查看技能日志或输出来定位错误原因。

接口调用

```
static std::shared_ptr<Preprocessor> hilens::Preprocessor::Create()
```

返回值

成功则返回预处理器的指针，初始化失败则返回nullptr。

7.3 改变图片尺寸

对图片进行缩放。

接口调用

```
HiLensEC hilens::Preprocessor::Resize(const cv::Mat & src, cv::Mat & dst, unsigned  
int w, unsigned int h, int type = 0)
```

参数说明

表 7-1 参数说明

参数名	说明
src	源图，必须为NV21的格式。宽度范围[64, 1920], 2的倍数；高度范围[64, 1080], 2的倍数。 如果输入不是NV21格式，请把输入的源图片转换为NV21格式，详情请参见 转换图片颜色格式 。
dst	目的图片。
w	缩放宽度，范围[64, 1920], 2的倍数。
h	缩放高度，范围[64, 1080], 2的倍数。
type	目的图片的格式，0为NV21,1为NV12，默认为0。

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

7.4 裁剪图片

对图片进行裁剪。

接口调用

HiLensEC hilens::Preprocessor::Crop(const cv::Mat & src, cv::Mat & dst, unsigned int x, unsigned int y, unsigned int w, unsigned int h, int type = 0)

参数说明

表 7-2

参数名	说明
src	源图，必须为NV21的格式。宽度范围[64, 1920], 2的倍数；高度范围[64, 1080], 2的倍数。 如果输入不是NV21格式，请把输入的源图片转换为NV21格式，详情请参见 转换图片颜色格式 。
dst	目的图片。

参数名	说明
x	裁剪区域左上角x坐标，范围[0, 1920], 2的倍数。
y	裁剪区域左上角y坐标，范围[0, 1080], 2的倍数。
w	缩放宽度，范围[64, 1920], 2的倍数。
h	缩放高度，范围[64, 1080], 2的倍数。
type	目的图片的格式，0为NV21,1为NV12，默认为0。

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

7.5 转换图片颜色格式

转换图片的颜色格式。opencv原生未提供RGB/BGR到NV12/NV21的转换选项，故在这里做补充。

接口调用

HiLensEC hilens::CvtColor(const cv::Mat & src, cv::Mat & dst, CvtColor code)

参数说明

表 7-3 参数说明

参数名	说明
src	源图(BGR888或RGB888)。
dst	目的图片。
code	颜色转换码，指定何种转换类型，可选RGB2YUV_NV12、RGB2YUV_NV21、BGR2YUV_NV12、BGR2YUV_NV21。 enum hilens::CvtColor 具体枚举值详情请见 表7-4 。

表 7-4 颜色转换码

枚举值	说明
BGR2YUV_NV12	BGR转YUV_NV12。
RGB2YUV_NV12	RGB转YUV_NV12。

枚举值	说明
BGR2YUV_NV21	BGR转YUV_NV21。
RGB2YUV_NV21	RGB转YUV_NV21。

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

8 模型管理

8.1 模型管理简介

hilens::Model 类

模型管理器，使用模型管理器加载模型并进行推理。

```
#include <model.h>
```

析构函数

```
~Model()  
virtual hilens::Model::~Model( )
```

Model析构时会释放掉hiiai::Graph等资源。

8.2 创建模型

构造一个模型。HiLens kit可以使用昇腾310芯片支持的模型来进行推理，使用此方法来构造一个后续用于推理的模型。模型构造失败则会抛出一个CreateError，并在日志上打印出错误码（例如0x1013011为模型路径错误）。当返回的对象被析构时，对应的模型资源也被释放。

接口调用

```
static std::shared_ptr<Model> hilens::Model::Create(const std::string & filename)
```

参数说明

表 8-1 参数说明

参数名	说明
filename	模型文件路径。假设模型放在./mymodels/test.om，则filename为“./mymodels/test.om”。

返回值

成功则模型管理器实例的指针，失败则返回nullptr。

8.3 模型推理

将数据输入模型进行推理，推理结束后将推理结果返回。

接口调用

```
virtual HiLensEC hilens::Model::Infer(const InferDataVec & inputs, InferDataVec & outputs)
```

参数说明

表 8-2 参数说明

参数名	参数类型	说明
inputs	InferDataVec, 请参见 参数类型说明 。	推理输入数据。
outputs	InferDataVec, 请参见 参数类型说明 。	推理输出。

参数类型说明

- InferDataVec

模型推理输入输出。

```
typedef std::vector<InferData> hilens::InferDataVec
```

- InferData

```
struct InferData
{
    unsigned int size;           // 输出大小
    std::shared_ptr<unsigned char> data; // 数据指针
    /**
     * @brief 构造一个空的模型推理数据
     */
    InferData() : size(0), data(nullptr) {}

    /**
     * @brief 从一个cv::Mat构造一个InferData
     * @param img 输入图片
     */
    InferData(const cv::Mat &img);

    /**
     * @brief 从一组指针数据构造一个InferData
     * @param data 数据指针，此构造函数会拷贝这部分数据
     * @param size 数据大小（字节）
     */
    InferData(const unsigned char *data, unsigned int size);
};
```

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

如果推理的实际输入与模型输入大小不一致，推理将会失败。此时infer的返回值将是一个int的错误码，日志会报出错误信息，开发者可以通过错误信息来定位错误。

9 输出模块

9.1 输出模块简介

hilens::Display 类

使用Display类来将图片输出到显示器上。

```
#include <output.h>
```

构造及析构函数

```
~Display()  
virtual hilens::Display::~Display()
```

9.2 构造用于输出的显示器

构造显示器，用来将图片显示到显示器或是输出到视频流。如果创建失败则抛出一个CreateError，开发者可以查看技能日志或输出来定位错误原因。

如果是H264_FILE类型的，需要注意，生成的文件仅是h264编码的裸视频流，不含帧率等信息，而且HiLens Framework并未限制文件大小。所以此功能建议只作为调试使用。

接口调用

```
static std::shared_ptr hilens::Display::Create(Type type, const char * path = NULL)
```

参数说明

表 9-1 参数说明

参数名	说明
type	显示类型，可选HDMI、RTMP、H264_FILE。

参数名	说明
path	如果类型为HDMI则忽略此参数，如果是RTMP则path为RTMP服务器的URL（rtmp://xxx），为H264_FILE则path为输出文件的路径（如hilens::GetWorkspacePath()+"/out.h264"）。

返回值

成功则返回一个显示器实例，失败则返回nullptr。

9.3 输出一帧图片

输出一张图片。注意，在第一次调用该接口时，输出模块会根据输入的图片尺寸来设置视频尺寸，此后的调用中skill必须保证输入图片的尺寸与之前的一致。

接口调用

```
virtual HiLensEC hilens::Display::Show(const cv::Mat & frame)
```

参数说明

表 9-2 参数说明

参数名	说明
frame	被显示的图片，必须为NV21格式。

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

9.4 上传文件

UploadFile()

上传一个文件到OBS，此方法会阻塞线程，直至上传结束。目标OBS桶可在HiLens页面上进行配置，详情请参见用户指南>配置数据存储位置。

- **接口调用**
HiLensEC hilens::UploadFile(const std::string & key, const std::string & filepath, const std::string & mode)
- **参数说明**

表 9-3 参数说明

参数名	说明
key	上传到obs中的文件名。
filepath	待上传文件的绝对路径。
mode	上传模式。两种可选：“write”-覆盖方式，“append”-追加方式。

- **返回值**
返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

UploadFileAsync()

异步上传一个文件，会立即返回。

- **接口调用**
HiLensEC hilens::UploadFileAsync(const std::string & key, const std::string & filepath, const std::string & mode, void(*) (int) callback = NULL)
- **参数说明**

表 9-4 参数说明

参数名	说明
key	上传到obs中的文件名。
filepath	待上传文件的绝对路径。
mode	上传模式，“write” or “append”。
callback	回调函数。

- **返回值**
返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

9.5 上传缓冲区数据

UploadBuffer()

上传一个buffer到OBS，此方法会阻塞线程，直至上传结束。目标OBS桶可在华为HiLens控制台上进行配置，详情请参见[用户指南>配置数据存储位置](#)。

- **接口调用**
HiLensEC hilens::UploadBuffer(const std::string & key, const unsigned char * buffer, const unsigned char * buffer, size_t bufferSize, const std::string & mode)

- **参数说明**

表 9-5 参数说明

参数名	说明
key	上传到obs中的文件名。
buffer	待上传buffer的指针。
bufferSize	待上传buffer的大小。
mode	上传模式，” write” or “append”。

- **返回值**

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

UploadBufferAsync()

异步上传一个buffer，会立即返回。

- **接口调用**

```
HiLensEC hilens::UploadBufferAsync(const std::string & key,
std::shared_ptr<const unsigned char> buffer, size_t bufferSize, const std::string
& mode, void(*) (int) callback = NULL)
```

- **参数说明**

表 9-6 参数说明

参数名	说明
key	上传到obs中的文件名。
buffer	待上传buffer的指针。
bufferSize	待上传buffer的大小。
mode	上传模式，” write” or “append”。
callback	回调函数。

- **返回值**

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

9.6 发送 POST 请求

发送一个POST请求。此方法是同步的，请求发送过程中会阻塞直到发送完毕。支持 TLS1.2安全协定，超时设为20秒。

接口调用

```
int hilens::POST(const std::string & url, const Json::Value & body, long & httpcode,
std::string * response = NULL, POSTHeaders * headers = NULL)
```

参数说明

表 9-7 参数说明

参数名	说明
url	统一资源定位符。
body	表示消息体内容的Json对象。
httpcode	http请求返回值，如返回值为200则请求成功，返回值为404则不存在。
response	响应，不填则为空。
headers	请求头部，不填则忽略头部。 typedef std::vector<std::string> hilens::POSTHeaders POST请求的头部调用例如： headers.push_back(“Content-Type: application/json”);然后将其作为POST 的参数传入。

返回值

CURL返回值，0为成功。

9.7 发送一条消息

SendMessage()

发送一条消息（同步），阻塞直到发送完毕，需要先在控制台上配置好订阅，详情请见用户指南>配置订阅消息。仅支持1.0.7-1.2.2版本使用。

- **接口调用**
HiLensEC hilens::UploadBuffer(const std::string & subject, const std::string & message)
- **参数说明**

表 9-8 参数说明

参数名	说明
subject	邮件主题（仅配置为邮件时有效），长度不能超过170个字符。

参数名	说明
message	消息内容, 不超过85个字符。

- **返回值**
返回值为0即成功, 其他即失败, 失败响应参数如[错误码](#)所示。

SendMessageAsync()

发送一条消息（异步），需要先在控制台上配置好订阅，详情请见用户指南>配置订阅消息。仅支持1.0.7-1.2.2版本使用。

- **接口调用**
HiLensEC hilens::UploadBuffer(const std::string & subject, const std::string & message, void (*callback)(int) = NULL)
- **参数说明**

表 9-9 参数说明

参数名	说明
subject	邮件主题（仅配置为邮件时有效），长度不能超过170个字符。
message	消息内容, 不超过85个字符。
callback	回调函数。

- **返回值**
返回值为0即成功, 其他即失败, 失败响应参数如[错误码](#)所示。

9.8 播放音频文件

播放本地AAC格式音频文件。在HiLens Kit设备的音频输出口接上耳机或者音箱，调用该接口时便可听到声音。

- **接口调用**
HiLensEC PlayAacFile(const std::string filePath, int vol)
- **参数说明**

表 9-10 参数说明

参数名	说明
filePath	本地音频文件绝对路径（不支持中文）。
vol	播放音频音量大小，取值范围[-121, 6]。

说明

本接口与[音频采集器](#)的接口不可同时调用。

- **返回值**
返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

10 资源管理

10.1 获取模型路径

返回技能模型目录的路径。

对于技能代码包和模型分离的情况，模型会下载到特定目录，使用此函数来获取该路径。如果HiLens Framework没有获取模型所在目录，则返回当前路径。

接口调用

```
std::string hilens::GetModelDirPath()
```

返回值

返回字符串，技能模型所在目录路径（末尾带“/”），如果获取失败则返回空字符串。

10.2 获得技能工作区目录

返回技能工作区目录的路径。

由于证书校验等问题，不允许在技能安装目录下写操作，故需要指定各技能可写的工作区位置。

接口调用

```
std::string hilens::GetWorkspacePath()
```

返回值

返回字符串，技能工作区路径（末尾带“/”），如果获取失败则返回空字符串。

10.3 获得技能配置

获取技能配置，即技能配置文件中的内容解析成的Json对象（jsoncpp）。注意此函数每次都会读取配置文件并解析成Json对象，所以如果需要读多个配置项，请将返回值存为一个变量，不要过于频繁的调用GetSkillConfig()。

接口调用

```
Json::Value hilens::GetSkillConfig()
```

返回值

返回技能配置Json对象。如果解析失败，则返回一个空的Json::Value（可用.empty()判断）。

10.4 下载 OBS 文件

从OBS下载一个文件。

接口调用

```
HiLensEC hilens::DownloadFileFromOBS(const std::string & url, const std::string & downloadTo)
```

参数说明

表 10-1 参数说明

参数名	说明
url	OBS资源的链接。资源链接获取详情请参见 OBS控制台指南>通过对象URL访问对象 。
download_to	指定下载到哪个目录。

返回值

返回值为0即成功，其他即失败，失败响应参数如[错误码](#)所示。

10.5 计算文件的 md5 值

计算一个文件的md5值。

接口调用

```
std::string hilens::MD5ofFile(const std::string & filepath)
```

参数说明

表 10-2 参数说明

参数名	说明
file	文件路径。

返回值

字符串，文件的MD5值。如果读取文件失败，则返回空字符串。

10.6 示例-资源管理

资源管理示例如下所示

```
#include <cstdio>
#include <hilens.h>
#include <string>

using namespace hilens;
using namespace cv;

void ResourceManage() {
    // 获得技能工作区目录的路径（末尾带"/"）
    auto skill_path = hilens::GetWorkspacePath();

    // 获得技能模型所在目录的路径（末尾带"/"）
    auto model_path = hilens::GetModelDirPath();

    // 获得技能配置。如果没有成功获取则返回None
    auto skill_config = hilens::GetSkillConfig();
    // 假设技能配置中有名为face_dataset的配置项，其值为obs中的人脸库文件face_dataset.zip的地址
    // 设置技能配置参数可参考《用户指南》相关操作
    auto face_dataset_url = skill_config["face_dataset"]["value"].asString();
    // 从OBS下载该文件到技能工作区目录，并通过返回值判断是否下载成功
    auto ret =
        hilens::DownloadFileFromOBS(face_dataset_url, hilens::GetWorkspacePath());
    if (ret != hilens::OK) {
        hilens::Error("Failed to download from obs");
    }

    // 在技能工作区目录新建文件夹并解压
    std::string cmd = "mkdir " + hilens::GetWorkspacePath() + "face_dataset";
    auto result = system(cmd.c_str());
    if (result != 0) {
        hilens::Error("Failed to mkdir");
    }

    cmd = "unzip " + hilens::GetWorkspacePath() + "face_dataset.zip -d " +
        hilens::GetWorkspacePath() + "face_dataset/";
    result = system(cmd.c_str());
    if (result != 0) {
        hilens::Error("Failed to unzip");
    }
}

int main() {
    auto ret = hilens::Init("hello");
    if (ret != hilens::OK) {
        hilens::Error("Failed to init");
    }
    return -1;
}
```

```
}  
ResourceManage();  
hilens::Terminate();  
return 0;  
}
```

11 难例上传模块

11.1 难例上传介绍及说明

1.1.2固件版本开始支持边缘AI难例发现算法，如果要使用难例上传相关接口，请先升级固件版本到1.1.2，详情请见[升级固件版本](#)。

当前主要支持的难例发现算法如下。

- **图片分类**

CrossEntropyFilter(threshold_cross_entropy)

原理：根据推理结果的交叉熵，判断熵是否小于交叉熵，小于则为难例。

输入：推理结果**prediction classes list**，例如[**class1-score, class2-score, class2-score,....**]，class-score表示类别得分，其范围为[0,1]。

输出：True or False，**True**是难例，**False**是非难例。

- **目标检测**

IBT (image-box-thresholds)

原理：**box_threshold**框阈值用于计算图片难例系数，推理结果的置信度得分小于阈值的数量占总输出推理框的百分比；**img_threshold**图阈值用于判断该图片是否是难例。

输入：**prediction boxes list**，例如[**bbox1, bbox2, bbox3,....**]，其中**bbox = [xmin, ymin, xmax, ymax, score, label]**，x和y为框的坐标，**score**表示置信度得分，**label**表示类别标签，**score**的范围需要为[0,1]。

输出：True or False，**True**是难例，**False**是非难例。

CSF(confidence score filter)

原理：**box_threshold_low**和**box_threshold_up**框阈值用于判断该图片是否是难例，方法是只要有一个输出框置信度得分在区间[**box_threshold_low, box_threshold_up**]，就判断该图片是难例。

输入：**prediction boxes list**，例如[**bbox1, bbox2, bbox3,....**]，其中**bbox = [xmin, ymin, xmax, ymax, score, label]**，x和y为框的坐标，**score**表示置信度得分，**label**表示类别标签，**score**范围为[0,1]。

输出：True or False，**True**是难例，**False**是非难例。

11.2 构造 HardSample 实例

接口调用

```
HardSampleInterface &hilens::GetHardSampleInstance()
```

返回值

返回一个**HardSampleInterface**对象。

11.3 初始化

初始化**HardSampleInterface**对象。

接口调用

```
virtual bool hilens::HardSampleInterface::Init(const float thresholdOne, const float thresholdTwo, const DetectionFilterType filterType)
```

表 11-1 参数说明

参数名	说明
thresholdOne	阈值。
thresholdTwo	阈值。
filterType	<p>难例过滤器的类型，取值分别对应“CrossEntropyFilter”、“IBT”和“CSF”三种算法，详情请见难例上传介绍及说明。</p> <p>取“CrossEntropyFilter”算法的时候thresholdOne为算法“CrossEntropyFilter”的参数“threshold_cross_entropy”，thresholdTwo可为任意值。</p> <p>取“IBT”算法的时候thresholdOne和thresholdTwo分别对应“IBT”算法的“box_threshold”和“img_threshold”。</p> <p>取“CSF”的时候thresholdOne和thresholdTwo分别对应“CSF”算法的“box_threshold_low”和“box_threshold_up”。</p>

返回值

返回bool值，表示初始化成功或失败。

11.4 难例图片判断

根据结果判断输入图片是否是难例。

接口调用

virtual bool Filter(const float inferResult[], const int size);

virtual bool Filter(const std::vector<Bbox> &bboxList, DetectionFilterType type);

表 11-2 参数说明 1

参数名	说明
inferResult[]	float数组，分类算法得到的各类别置信度。
size	输入大小。

表 11-3 参数说明 2

参数名	说明
bboxList	std::vector<Bbox>，BBox结构体定义如下。 <pre>struct Bbox { float xmin; float ymin; float xmax; float ymax; float score; int label; Bbox(floatbboxXmin, floatbboxYmin, floatbboxXmax, floatbboxYmax, floatbboxScore, intbboxLabel): xmin(bboxXmin), ymin(bboxYmin), xmax(bboxXmax), ymax(bboxYmax), score(bboxScore), label(bboxLabel){} };</pre> 参数说明详见 表11-4 。
type	难例过滤器的类型，取值分别对应“CrossEntropyFilter”、“IBT”和“CSF”三种算法，详情请见 难例上传介绍及说明 。

表 11-4 Bbox 结构体说明

参数值	说明
xmin	检测框的坐标值。

参数值	说明
ymin	检测框的坐标值。
xmax	检测框的坐标值。
ymax	检测框的坐标值。
score	检测框的得分。
label	检测框的类别。

返回值

返回bool值，表示是否是难例图片。

11.5 获取难例配置

接口调用

Json::Value GetHardSampleConfig()

返回值

Json::Value对象，可以解析出各配置项的值。

11.6 更新难例配置

更新难例配置到难例配置文件，并根据输入更新云侧难例上传状态。

接口调用

HiLensEC SetHardSampleConfig(const std::string &confStr)

表 11-5 参数说明

参数名	说明
confStr	要更新的难例配置，需要是string格式的json值。

返回值

HiLensEC错误码，0成功，其他为失败。

12 日志模块

12.1 设置打印日志的级别

设置日志级别。

接口调用

```
void hilens::SetLogLevel(LogLevel level)
```

参数说明

表 12-1 参数说明

参数名	说明
level	日志级别。可选Trace、Debug、Info、Warning、Error、Fatal。 enum hilens::LogLevel 具体枚举值详情请见 表12-2 。

表 12-2 日志枚举值说明

枚举值	说明
TRACE	打印Trace及以上级别的日志。
DEBUG	打印Debug及以上级别的日志。
INFO	打印Info及以上级别的日志。
WARNING	打印Warning及以上级别的日志。
ERROR	打印Error及以上级别的日志。
FATAL	打印Fatal级别的日志。

返回值

无。

12.2 打印 Trace 级别的日志

打印一条Trace级别的日志。使用方式类似于printf。

接口调用

```
void hilens::Trace(const char * fmt, ... )
```

参数说明

表 12-3 参数说明

参数名	说明
fmt	字符串，可以包含嵌入的格式化标签，格式化标签可被随后的附加参数中指定的值替换，并按需求进行格式化。单条日志支持最大255个字符。
…（附加参数）	根据不同的fmt字符串，函数可能需要一系列的附加参数，每个参数包含了一个要被插入的值，替换了fmt参数中指定的每个 % 标签。参数的个数应与 % 标签的个数相同。使用方式类似printf，示例： <code>hilens::Trace("字符串为 %s \n", str);</code>

返回值

无。

12.3 打印 Debug 级别的日志

打印一条Debug级别的日志。使用方式类似于printf。

接口调用

```
void hilens::Debug(const char * fmt, ... )
```

参数说明

表 12-4 参数说明

参数名	说明
fmt	字符串，可以包含嵌入的格式化标签，格式化标签可被随后的附加参数中指定的值替换，并按需求进行格式化。单条日志支持最大255个字符。
…（附加参数）	根据不同的fmt字符串，函数可能需要一系列的附加参数，每个参数包含了一个要被插入的值，替换了fmt参数中指定的每个 % 标签。参数的个数应与 % 标签的个数相同。示例：hilens::Debug("字符串为 %s \n", str);

返回值

无。

12.4 打印 Info 级别的日志

打印一条Info级别的日志。使用方式类似于printf。

接口调用

```
void hilens::Info(const char * fmt, ...)
```

参数说明

表 12-5 参数说明

参数名	说明
fmt	字符串，可以包含嵌入的格式化标签，格式化标签可被随后的附加参数中指定的值替换，并按需求进行格式化。单条日志支持最大255个字符。
…（附加参数）	根据不同的fmt字符串，函数可能需要一系列的附加参数，每个参数包含了一个要被插入的值，替换了fmt参数中指定的每个 % 标签。参数的个数应与 % 标签的个数相同。示例：hilens::Info("字符串为 %s \n", str);

返回值

无。

12.5 打印 Warning 级别的日志

打印一条Warning级别的日志。使用方式类似于printf。

接口调用

```
void hilens::Warning(const char * fmt, ... )
```

参数说明

表 12-6 参数说明

参数名	说明
fmt	字符串，可以包含嵌入的格式化标签，格式化标签可被随后的附加参数中指定的值替换，并按需求进行格式化。单条日志支持最大255个字符。
…（附加参数）	根据不同的fmt字符串，函数可能需要一系列的附加参数，每个参数包含了一个要被插入的值，替换了fmt参数中指定的每个 % 标签。参数的个数应与 % 标签的个数相同。示例：hilens::Warning("字符串为 %s \n", str);

返回值

无。

12.6 打印 Error 级别的日志

打印一条Error级别的日志。使用方式类似于printf。

接口调用

```
void hilens::Error(const char * fmt, ... )
```

参数说明

表 12-7 参数说明

参数名	说明
fmt	字符串，可以包含嵌入的格式化标签，格式化标签可被随后的附加参数中指定的值替换，并按需求进行格式化。单条日志支持最大255个字符。
…（附加参数）	根据不同的fmt字符串，函数可能需要一系列的附加参数，每个参数包含了一个要被插入的值，替换了fmt参数中指定的每个 % 标签。参数的个数应与 % 标签的个数相同。示例：hilens::Error("字符串为 %s \n" , str);

返回值

无。

12.7 打印 Fatal 级别的日志

打印一条Fatal级别的日志。使用方式类似于printf。

接口调用

```
void hilens::Fatal(const char * fmt, ... )
```

参数说明

表 12-8 参数说明

参数名	说明
fmt	字符串，可以包含嵌入的格式化标签，格式化标签可被随后的附加参数中指定的值替换，并按需求进行格式化。单条日志支持最大255个字符。
…（附加参数）	根据不同的fmt字符串，函数可能需要一系列的附加参数，每个参数包含了一个要被插入的值，替换了fmt参数中指定的每个 % 标签。参数的个数应与 % 标签的个数相同。示例：hilens::Fatal("字符串为 %s \n" , str);

返回值

无。

13 错误码

HiLens Framework返回的错误码属于HiLens EC类型，错误码（HiLens EC枚举值）如表13-1所示。

表 13-1 错误码

错误码	说明
OK=0	没有错误。
UNKNOWN_ERROR	未知错误。
INIT_CURL_ERROR	初始化CURL错误。
CREATE_DIR_FAILED	创建文件夹失败。
OPENFILE_FAILED	打开文件失败。
RENAME_FAILED	重命名失败。
ACCESS_FILE_FAILED	文件不存在或无文件访问权限。
INVALID_BUF	无效的BUF。
COULDNT_RESOLVE_HOST	无法解析，查看网络是否通畅。
WRITE_ERROR	写错误，检查下载目录是否有写权限，空间是否足够。
TIMEOUT	请求超时。
AUTH_FAILED	认证信息错误，检查ak, sk, token是否有效。
NOT_FOUND	没有这个对象。
SERVER_ERROR	服务端内部错误。
OBJECT_CONFLICT	对象冲突。
APPEND_FAILED	追加失败（比如追加到不可追加的对象上）。

错误码	说明
HIAI_SEND_DATA_FAILED	hiai engine发送数据失败，请根据日志来分析具体情况。
HIAI_INFER_ERROR	hiai engine推理错误，请根据日志来分析具体情况（可能是实际输入大小与模型的输入大小不匹配）。
INVALID_SRC_SIZE	图片处理，src尺寸不符合约束条件。
INVALID_DST_SIZE	图片处理，dst尺寸不符合约束条件。
MPP_PROCESS_FAILED	mpp处理图片失败。
WEBSOCKET_ERROR	WebSocket错误。
CONFIG_FILE_ERROR	配置文件错误。
INVALID_PARAM	参数有误。
INIT_LOG_ERROR	日志初始化失败。
INIT_MIC_ERROR	麦克风初始化失败。
INIT_AENC_ERROR	音频编码初始化失败。
INIT_ADEC_ERROR	音频解码初始化失败。
INIT_AO_ERROR	音频输出初始化失败。
AUDIO_CHECK_ERROR	音频文件不支持。
AUDIO_SYSTEM_INIT_FAILED	音频系统初始化失败。