ANG-SDK Development and Usage Specifications

THE-SDK-006

Revision History

Version number	time	Revision Record	Revision Department
v1.1.0	2023/04/20	First version released 1.	Software Department
v1.1.1	2023/05/19	Added interfaces and structures related to monitoring functions 2. Modify the AS_SDK_SetStreamParam interface and related structures 3. Added interface for obtaining camera model 4. Add network interface and structure 5. Added interface for setting timestamp 6. Added interface for obtaining resolution capability set	Software Department
v1.1.2	2023/06/08	Generalization of sample code	Software Department
v1.1.3	2023/06/21	Fix the problem of getting SN number and optimize demo logic	Software Department
v1.1.4	2023/06/28	Add timestamp and sequence number to point cloud messages in ROS/ROS2 demo	Software Department
v1.1.5	2023/07/03	Fix the problem that network device traversal may occur	Software Department
v1.1.6	2023/07/06	Added the option to select ordered or unordered point cloud in the hp60c/hp60cn/nuwa/vega configuration files 1.	Software Department
v1.1.7	2023/07/11	Added hp60v depth correction 2. Change the naming of some options in the nuwa configuration file	Software Department
v1.1.8	2023/07/12	1. Optimize the Kunlun algorithm 2. The sample code adds the function of judging whether it is a virtual machine and whether it matches the nuwa device	Software Department
v1.1.9	2023/07/14	1. Fixed the problem of NUWA hp60v (sh58) getting internal and external parameters incorrectly	Software Department
v1.1.10	2023/07/27	1. Added support for previewing color point clouds on ros/ros2 for hp60c/hp60cn modules 2. Fixed the problem of ros2 program crashing when exiting 3. Fix the firmware upgrade failure issue for kunlun-a / kunlun-c / hp60c / hp60cn 4. Fixed the crash issue when closing data stream in v4l2 module 5. Fixed the issue that the setting parameters of some nuwa series modules were not effective 6. Update and add nuwa configuration file options	Software Department
v1.1.11	2023/08/02	1. Fixed the issue of hp60cn depth map flickering in multi-module mode	Software Department
v1.2.0	2023/08/11	1. Added launch files for specific cameras, mono8 topic publishing, and ros1 tf tree topic release; 2. Added support for RGB and depth alignment status 3. Fix known issues and optimize internal functions.	Software Department
v1.2.1	2023/08/21	1. Added the release of ros2 TF tree topic; 2. Added X100 320x200 resolution; 3. Added instructions for using the upgrade function; 4. Optimize x100 internal logic	Software Department
v1.2.2	2023/08/22	1. Optimize x100 depth data processing algorithm 2. Added x100 frame rate configuration.	Software Department
v1.2.3	2023/08/29	1. Fixed the problem of kunlun series module switching between near and far view 2. Added sample code for using ros2 for Kunlun series modules	Software Department
v1.2.4	2023/08/31	1. Fixed the issue that nuwa camera may have errors when matching multiple nodes	Software Department

v1.2.5 2023/09/04 1. Relax the network camera hot plug detection time from 2s->6s			Software Department
v1.2.6	2023/09/07 Added XB40, HP60, and HP60V frame rate configurations.		Software Department
v1.2.7	2023/09/11	1. Fixed the bug that caused the deep denoising parameter error 2. Deleted the automatic restart function in the non-hot-swap demo 3. Fixed the bug in the network camera monitoring mechanism	Software Department
v1.2.8	2023/09/13 1. Optimize log printing and network broadcasting 1. Optimize NUWA		Software Department
v1.2.9	2023/10/23	camera parameter configuration sequence logic 2. Add HP60CN frame loss filter detection	Software Department
v1.2.10 2023/11/10		1. Added upgrades and capability set acquisition for each NUWA model 2. Optimize the release package structure and size 3. Add internal reference data, SDK call restriction description, and release package usage instructions to the development documentation 4. Delete the non-hot-plug reference example and remove the listener keyword from the original hot-plug example file name Character 5. Fixed the problem of occasional failure in creating handles for network cameras 6. Fix the abnormal topic publishing of Kunlun series ros/ros2 1. Modify the noise	Software Department
v1.2.11 2023/12/12		reduction parameters of HP60 module (maximum speckle size denoiseMaxSpeckleSize) Changed from 100 to 800 2. Modify HP60C 、 HP60CN camera internal parameter processing: cyRgb value from source data (rawData[12]+6)/2.25 to (rawData[12]+6)/2.25 and assign the value after parsing 3. Modify the configuration parameters of HP60C and HP60CN: Change the order from the original setting after opening the flow to opening the flow. After the device is installed, before the stream is opened 4. Modify the raw parsing order of KunlunA and KUNLUNC images, from the original upper left corner to the right, Parse line by line downwards, change to parse from the lower right corner to the left, and from bottom to top line by line	Software Department
v1.2.12 2023/12/28		1. KUNLUN-A/C implements the interface for obtaining resolution capability set 2. KUNLUN-A fixed the abnormal problem of obtaining firmware version number 3. KUNLUN-A/C changed the way of reading local calibration files to reading calibration data in the module 4. KUNLUN-A/C's width and height parameters for setting resolution are changed to need to be filled in the final output image Width and height 5. Update KUNLUN configuration file 6. Added the function of enumerating devices in WINDOWS monitoring function	Software Department
v1.2.13 2024/01/04 1. Fixed the problem that the depth and color images cannot be aligned due to the error in the output of internal and external parameters of HP60CN. Software 1. Update			
v1.2.14 2024/01/29		KUNLUN-A/C algorithm library V3_20240116 2. Fixed the problem that the frame sequence number returned by KUNLUN-A/C Windows version is abnormal 3. Added ROS/ROS2 topic publishing name with node name as prefix 4. Added the function of starting the camera stream and publishing the corresponding topic after subscribing to the ROS/ROS2 topic 5. Fixed the abnormal issue of KUNLUN-A/C ROS/ROS2 CameraInfo topic 6. Modify the ROS/ROS2 topic number to distinguish the topic name of multi-module or multi-node startup 7. Fix the HP60CN occasional failure to open the device 8. Fixed the occasional crash of HP60CN 9. Optimize KUNLUN-A/C image output time 10. Add KUNLUN-A default internal parameters	Software Department
v1.2.15 2024/02/21		1. Added support for hp60 320x240 resolution 2. Fixed the abnormal callback progress of KUNLUN-A firmware OTA upgrade 3. Fix the abnormal configuration of KUNLUN register	Software Department

		4. Modify KUNLUN default frame rate from 20 fps to 10 fps 5. Fixed the abnormal startup sequence number of ROS/ROS2 multi-module or multi-node 6. Added ROS1 nodelet function 7. Added HP60C/ HP60CN/KUNLUN-C mjpeg format data output to callback and displayed in linux-demo	
v1.2.16 2024/02/23		1.Fixed the issue that the register setting of KUNLUN-A/C does not take effect after opening the stream, causing the depth map to flicker 2. Update KUNLUN-A/C algorithm library V3_20240223 to fix the problem of automatic integration function	Software Department
v1.2.17 2024/03/25		1. Optimize the gain setting method of HP60C	Software Department
v1.2.18 2024/03/28		1. Mirror HP60CN depth map and color map	Software Department
v1.2.19 2024/04/08		1. Update KUNLUN algorithm library 2. Fixed the abnormal preview image issue of linux demo	Software Department
v1.2.20 2024/04/15		1. Update KUNLUN algorithm library	Software Department
v1.2.21 2024/04/25		1. Optimize the firmware upgrade order of NUWA series modules	Software Department
v1.2.22 2024/05/15		1. Added interface for obtaining the space size of camera private data 2. Add an interface for writing private data to the camera 3. Added an interface for reading private data from the camera 4. Update KUNLUN-A/C configuration files	Software Department
v1.2.23 2024/07/18		1. Add VEGA, CHANGJIANG-B, TAISHAN, TANGGULA-A model modules 2. Added the function of retaining decimals for VEGA model depth output, which is disabled by default and can be set in the configuration file Modify Decimal to true to enable 3. Fix ros2 component link error 4. Fixed the occasional crash of network camera when shutting down the stream 5. Fixed the bug that multiple network cameras interfered with each other and caused the failure of opening the device 6. Mirrored HP60CN depth and RGB direction	Software Department
v1.2.24 2024/08/08		1. Fixed the issue of multiple files in the OTA package failing to upgrade 2. Fixed the problem that the width and height of the infrared image returned by the changjiangB module are	Software Department
v1.2.25 2024/08/28		incorrect. 1. Fixed the problem that AS_SDK_DestoryCamHandle accesses the released memory 2. Fixed the issue of not releasing memory when exiting the demo of Linux/ros/ros2 3. TANGGULA-A adds the functions of reading image resolution from the module and depth shift from the configuration file. Bit number function 4. Added O2 camera (only supports WINDOWS) 5. Added TANGGULA-B module 6. Updated the depth stitching algorithm of changjiangB 7. Add OTA upgrade file type warp 8. Fixed the memory leak problem of the switch device of the changjiangB module 1.	Software Department
v1.2.26 2024/09/08		Update the TANGGULA-A configuration file 2. TANGGULA-B adds the function of simultaneous output of color image and IR (speckle) image	Software Department
v1.2.27 2024/09/12		1. TANGGULA-A and TANGGULA-B use the maximum frame rate by default 2. Fixed the point cloud distortion problem after reducing the resolution of hp60c/hp60cn 1.	Software Department
V1.2.28 2024/10/15		Updated TANGGULA-B configuration file, and enabled alignment by default 2. Added default internal and external parameters for TANGGULA-A/TANGGULA-B 3. Added support for XB40-V3 serial communication version firmware	Software Department
V1.2.28 2024/10/21		1. Corrected the incorrect description of ANG-LINUX-ROS User Manual 9.5	Software Department

Table of contents

1. Introduction

1.1 Product Overview

Ansjiang SDK API provides a one-stop service for third-party applications to integrate camera module functions. Based on SDK, customers can quickly Develop services to prevent other problems caused by irregular calls when using the Angstrong SDK API.

The SDK is used to drive various series of cameras of Ansijiang Technology, providing functions such as real-time acquisition of depth images, point cloud images, IR images, RGB images and PEAK images.

1.2 Environmental requirements

1.2.1 Operating Environment

ARM/AARCH64, X86_64 architectures.

1.2.2 System Requirements

Linux: Ubuntu16.04/Ubuntu18.04/Ubuntu20.04/Ubuntu22.04 (Ubuntu18.04 is recommended)

ROS1: Kinetic Kame/Melodic Morenia/Noetic Ninjemys (Melodic Morenia is recommended)

ROS2: Foxy Fitzroy/Galactic Geochelone/Humble Hawksbill (Humble Hawksbill is recommended)

Windows.

1.3 SDK Function Introduction

1.3.1 Obtaining Depth

Support driving Ansijiang camera and obtaining depth map in real time;

The spatial format of the depth data is as follows

Table 1 Depth map format

Bit width	16
unit	mm
type	unsigned short

1.3.2 Obtaining point cloud

Support driving Ansijiang camera and obtaining point cloud images in real time;

Get the point cloud image in the image data callback.

1.3.3 Get IR

Support driving Ansijiang camera and obtaining IR images in real time;

Get the IR image in the image data callback.

1.3.4 Get RGB

Support driving Ansijiang camera and obtaining RGB images in real time;

Get the RGB image in the image data callback.

1.3.5 Get PEAK

Support driving Ansijiang camera and obtaining PEAK images in real time;

Get the PEAK graph in the image data callback.

2. SDK Integration Guide

2.1 Package Structure Description

Version change description: Starting from v1.2.10, the provided linux/ros/ros2 demos are all references for the implementation of the hot-swap mechanism.

The example is equivalent to the reference example with xxx_listener in the previous version. The non-hot-swap example of the previous version is no longer provided. After switching to the new version, please note

Note: The name of the ROS system topic release has changed.

Starting from v1.2.10, the release package is in the format of xxx.tar.xz. In Linux environment, use tar -xJvf (note that it is capitalized

In Windows environment, you can use the decompression tool to decompress. In Linux environment, after decompression, perform the following operations to obtain the complete SDK package:

```
cd ${releasePackage}/demo/
./unpack_linux_ros.sh
```

The structure is described as follows:

```
AngstrongCameraSdk_vx.x.x.20xxxxxx
 ÿÿÿ CHANGELOG                                                   SDK Update History
 ÿÿÿ docs                                                        SDK/ROS development instructions and other information table
 ÿÿÿ demo
      ÿÿÿ linux_ros
      .      ÿÿÿ linux                                           Reference example of simple call under Linux system
      .      .      ÿÿÿ configurationfiles                       Configuration Files
      .      .      ÿÿÿ include                                  demo header file
      .      .      ÿÿÿ libs
      .      .      ÿ ÿÿÿ include                                 SDK Header Files
      .      .      ÿ ÿÿÿ lib                                     SDK library files
      .      .      .      ÿÿÿ aarch64-linux-gnu                  aarch64-linux-gnu-g++ version
      .      .      .      ÿÿÿ arm-linux-gnueabihf                arm-linux-gnueabihf-g++ version
      .      .      .      ÿÿÿ    x86_64-linux-gnu                x86_64-linux-gnu-g++ version
      .      .      ÿÿÿ    src                                   Demo source file
      .      ÿÿÿ ros                                             ROS1 reference example under ros system
      .      ÿÿÿ ros2                                            ROS2 reference example under ros system
      ÿÿÿ windows                                                Reference examples for Windows systems
```

2.2 Example Program Running Instructions

2.2.1 Camera Configuration File Description

The camera configuration file is in the directory configurationfiles . When calling the "Open Camera" interface, you need to pass in the configuration file path.

The default configuration files are all encrypted. If the user needs to modify the configuration file parameters, please contact FAE to obtain it.

2.2.2 Run the sample program on Ubuntu (Linux) system

Go to the demo/linux_ros/linux directory, where build.sh is the compilation script and run_ascamera.sh is the script to run the sample program.

Script. First, execute build.sh to compile, then execute run_ascamera.sh script to run the program. Executing the program requires root

Permissions, the script will be run as sudo

Compile:

```
./build.sh
```

run:

```
./run_ascamera.sh
```

2.2.3 Run ROS/ROS2 system sample program

Go to the demo/linux_ros/ros or ros2 directory, where build.sh is the compilation script and run_ascamera.sh is the startup node.

First, execute build.sh to compile, and then execute run_ascamera.sh script to run the program. Starting the node requires root

Permissions, the script will be run as sudo

Compile:

```
./build.sh
```

Start the node:

```
./run_ascamera.sh
```

2.2.4 How to cross compile

Cross-compilation is to generate executable code on one platform for another platform. For example, compiling on a Linux system with x86_64 architecture to generate executable code on another platform.

The executable file runs on the Linux system of arm/aarch64 architecture. In the demo code of Ansijiang NUWA Linux SDK,

The architecture automatically links the corresponding library files, so put the SDK into the arm development board that supports the compiler. The compilation method is the same as in the x86_64 architecture.

If you want to cross-compile in x86_64 Linux to generate arm/aarch64 executable files, just execute the build.sh script in the directory.

Please refer to the following chapters for more information. The dynamic link library for arm/aarch64 that has been compiled in the SDK can be found in the /libs/lib directory.

The corresponding directory name is the name of the compiler. If the required arm/aarch64 version of the dynamic link library is not available, you need to provide the Linux on x86_64

The cross-compilation tool chain recompiles and generates related dynamic link libraries.

2.2.5 Running the Windows System Sample Program

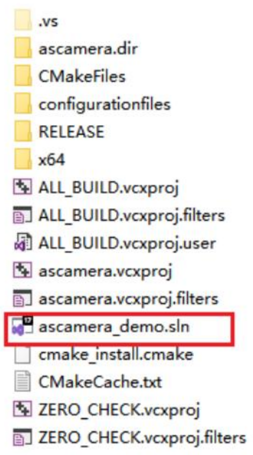
2.2.5.1 Run the compiled program directly

Enter the demo/windows/release directory, copy all the dll libraries in the demo/windows/libs/lib/x86_64-win-msvc/ directory to here, and double-click the executable file.

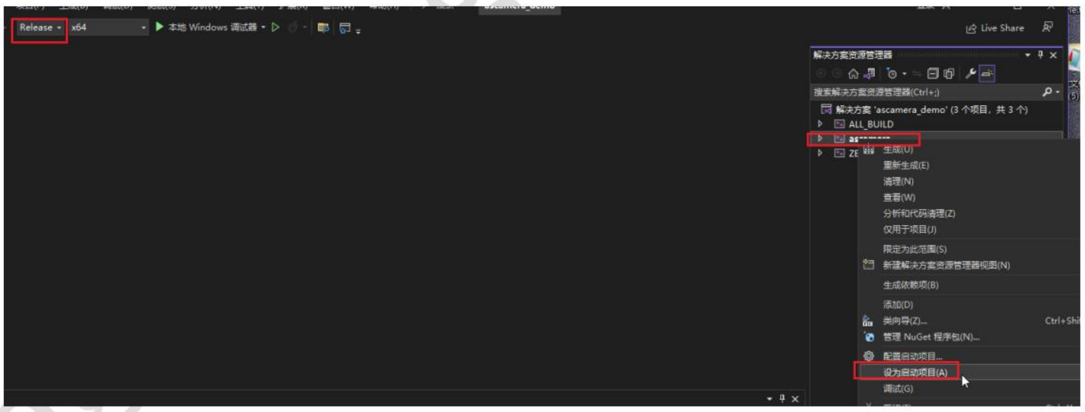
2.2.5.2 Compile and run the program

Step 1. Create a new build directory build in the demo/windows directory and use the command cmake .. to compile.

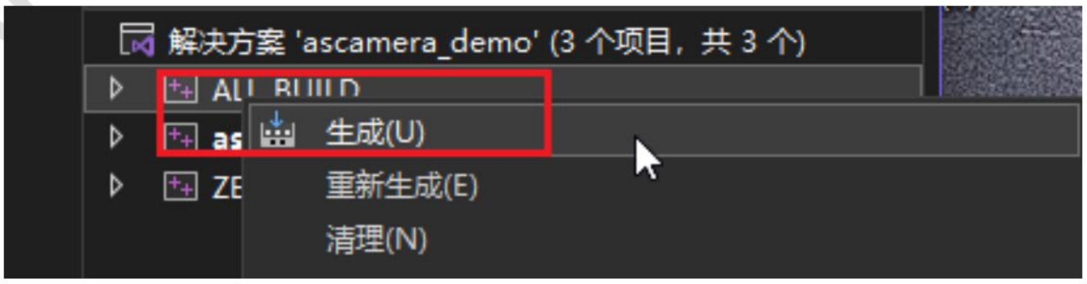
Step 2. Double-click the file with the suffix sln.


.vs
ascamera.dir
CMakeFiles
configurationfiles
RELEASE
x64
ALL_BUILD.vcxproj
ALL_BUILD.vcxproj.filters
ALL_BUILD.vcxproj.user
ascamera.vcxproj
ascamera.vcxproj.filters
ascamera_demo.sln
cmake_install.cmake
CMakeCache.txt
ZERO_CHECK.vcxproj
ZERO_CHECK.vcxproj.filters

Step 3. After selecting Release, right-click ascamera on the right and set it as the startup project.



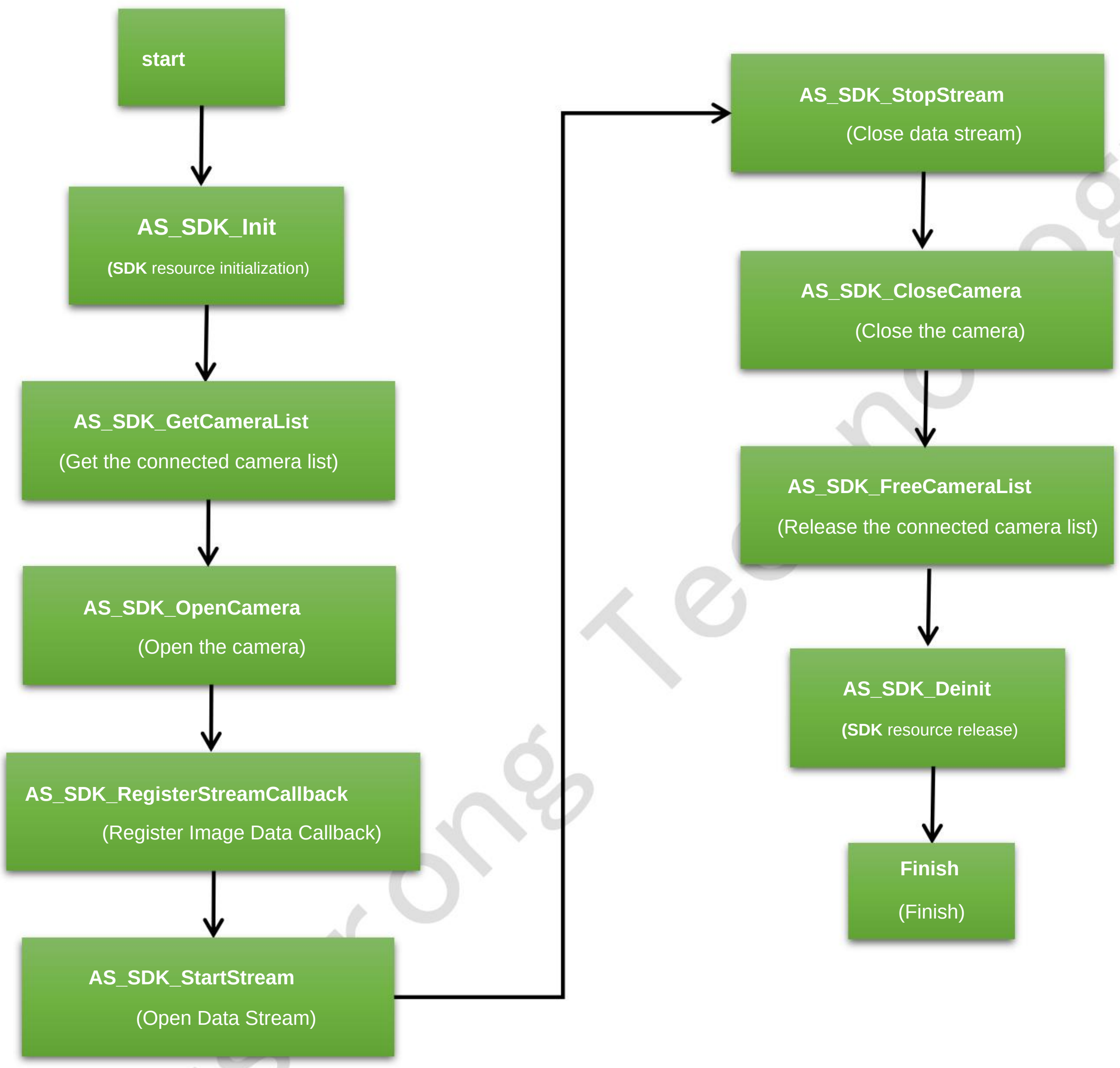
Step 4. Right-click ALL_BUILD on the right and click Generate.



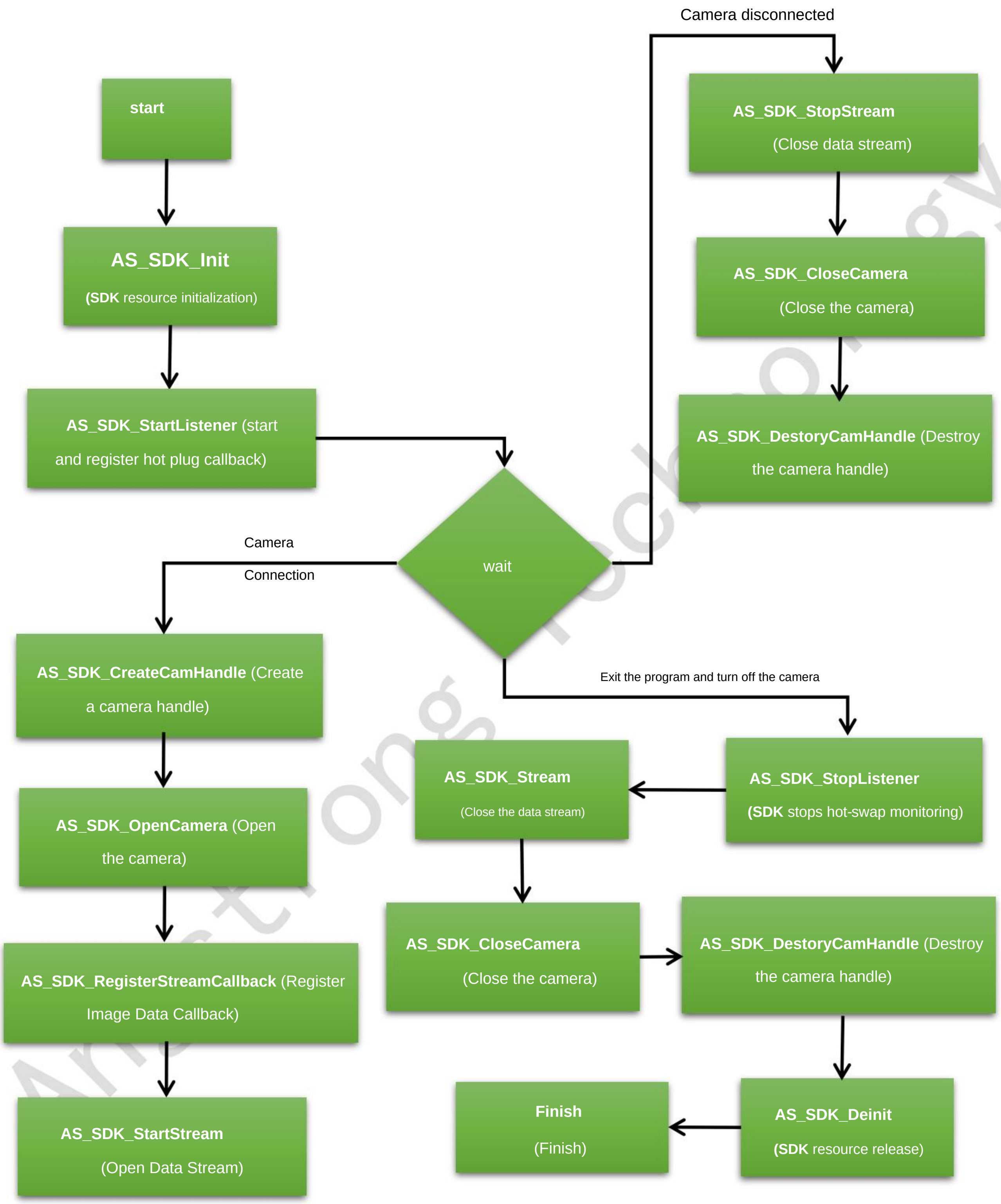
Step 5. Enter the demo/win/build/RELEASE directory and double-click ascamera.exe to run the program.

2.3 Call Flowchart

2.3.1 Actively obtain the camera list



2.3.2 Camera management based on hot swapping


Camera disconnected
start
AS_SDK_Init
(SDK resource initialization)
AS_SDK_StartListener (start and register hot plug callback)
AS_SDK_StopStream (Close data stream)
AS_SDK_CloseCamera (Close the camera)
AS_SDK_DestoryCamHandle (Destroy the camera handle)
Camera Connection
wait
Exit the program and turn off the camera
AS_SDK_CreateCamHandle (Create a camera handle)
AS_SDK_OpenCamera (Open the camera)
AS_SDK_RegisterStreamCallback (Register Image Data Callback)
AS_SDK_StartStream (Open Data Stream)
AS_SDK_Stream (Close the data stream)
AS_SDK_StopListener (SDK stops hot-swap monitoring)
AS_SDK_CloseCamera (Close the camera)
AS_SDK_DestoryCamHandle (Destroy the camera handle)
Finish (Finish)
AS_SDK_Deinit (SDK resource release)

3. Description of important data structures

3.1 Stream Interface Data Type

3.1.1 AS_CAM_PTR

ÿillustrateÿ

Camera handle

ÿdefinitionÿ

```
typedef void *          AS_CAM_PTRÿ
```

3.1.2 AS_FRAME_Type_e

ÿillustrateÿ

Data frame type

ÿdefinitionÿ

```
typedef enum FRAME_TYPE_E {
        AS_FRAME_TYPE_DEPTH = 0,
        AS_FRAME_TYPE_RGB,
        AS_FRAME_TYPE_IR,
        AS_FRMAE_TYPE_POINTCLOUD,
        AS_FRAME_TYPE_YUYV,
        AS_FRAME_TYPE_PEAK,
        AS_FRAME_TYPE_MJPEG,
        AS_FRAME_TYPE_BUTT
} AS_FRAME_Type_e;
```

ÿparameterÿ

Member	Description
AS_FRAME_TYPE_DEPTH	Depth data frame type
AS_FRAME_TYPE_RGB	Color image frame type
AS_FRAME_TYPE_IR	IR Image Frame Type
AS_FRMAE_TYPE_POINTCLOUD Point cloud data frame type	
AS_FRMAE_TYPE_YUYV	YUYV image frame type
AS_FRAME_TYPE_PEAK	PEAK Image Frame Type
AS_FRAME_TYPE_MJPEG	MJPEG image frame type
AS_FRAME_TYPE_BUTT	Invalid value

3.1.3 AS_Frame_s

ÿillustrateÿ

Frame structure

ÿdefinitionÿ

```
typedef struct CAM_FRAME {
        AS_FRAME_Type_e type;
        unsigned int width;
        unsigned int height;
        void *data;
        unsigned int size;
        unsigned int bufferSize;
        unsigned int frameId;
        unsigned long long ts;
    } AS_Frame_s;
```

ÿparameterÿ

Member	Description
type	Data frame type
width	Image width in pixels
height	Image height in pixels
data	Data cache address
size	The size of the valid data, in bytes.
bufferSize	The total size of the data cache space, in bytes.
frameId	The sequence number of the frame
ts	The timestamp of this frame is taken from the system time of the system (where the SDK is called) in milliseconds

3.1.4 AS_SDK_Data_s

ÿillustrateÿ

Image data structure. When the camera model used does not have corresponding image data or the corresponding data stream is not opened, the output image data The length of xxxImg.size is 0. Note that when the RGB camera opens the RGB image and the depth image, if the two images are aligned, the point cloud data The depth image corresponding to pointCloud is an aligned depth image, otherwise it is an unaligned depth image. Configure whether the RGB image and depth image are aligned. Alignment can be achieved by replacing the corresponding configuration files. Please contact FAE to obtain the method.

ÿDefinitionÿ

```
    typedef struct CAM_DATA {
        AS_Frame_s depthImg;
        AS_Frame_s rgbImg;
        AS_Frame_s irImg;
        AS_Frame_s pointCloud;
        AS_Frame_s yuyvImg;
        AS_Frame_s peakImg;
        AS_Frame_s mjpegImg;
        std::vector<AS_POINTCLOUD_s> pointCloud2;
    } AS_SDK_Data_s;
```

ÿparameterÿ

Member	Description
depthImg	Depth data, unsigned short type, unit is millimeter
rgbImg	Color image data, BGR format, unsigned char type
irImg	Infrared image data, unsigned char type

Member	Description
pointCloud	Point cloud data corresponding to the depth image, float type, arranged in xyz order, z is the depth (distance) The value is in millimeters. Invalid points have been discarded in unordered point clouds, but not in ordered point clouds.
yuyvImg	YUYV data, unsigned char type
peakImg	PEAK image data, unsigned char type
pointCloud2	Polar coordinate point cloud data (only supported by KONDYOR, please ignore for other types of modules)
mjpegImg	MJPEG image frame type, only supported by RGB cameras, HP60C/HP60CN needs to do Flip along the x-axis (upside down)

ÿillustrateÿ

The pointCloud data of KUNLUN-A and KUNLUN-C cameras are ordered point clouds.

3.1.5 AS_SDK_MERGE_s

ÿillustrateÿ

Fusion image data structure

ÿdefinitionÿ

```
typedef struct CAM_MERGE {
    AS_Frame_s depthImg;
    AS_Frame_s pointCloud;
} AS_SDK_MERGE_s;
```

ÿparameterÿ

Member	Description
depthImg	Depth data, unsigned short type, unit is millimeter
pointCloud	The point cloud data corresponding to the depth image, float type, arranged in xyz order, z is the depth (distance) value, The unit is millimeter. Invalid points have been discarded in the unordered point cloud, but not in the ordered point cloud.

3.1.6 AS_CAM_StreamCallback

ÿillustrateÿ

Image data callback function

ÿÿÿÿ

```
typedef void(*AS_CAM_StreamCallback)(AS_CAM_PTR pCamera, const AS_SDK_Data_s *pstData,
void *privateData);
```

ÿparameterÿ

Member	Description
pCamera	Camera handle
pstData	Image data
privateData	User private data

3.1.7 AS_CAM_Stream_Cb_s

ÿillustrateÿ

Image data callback data structure

ÿdefinitionÿ

```
typedef struct STREAM_CALLBACK_S {
    AS_CAM_StreamCallback callback;
    void *privateData;
} AS_CAM_Stream_Cb_s;
```

ÿparameterÿ

Member	Description
callback	Data callback function
privateData	User private data

3.1.8 AS_CAM_MergeFrameCallback

ÿillustrateÿ

Fusion image data callback function

ÿÿÿÿ

```
typedef void(*AS_CAM_MergeFrameCallback)(AS_CAM_PTR pCamera, const AS_SDK_Merge_s *pstData,
void *privateData);
```

ÿparameterÿ

Member	Description
pCamera	Camera handle
pstData	Image data
privateData	User private data

3.1.9 AS_CAM_Merge_Cb_s

ÿillustrateÿ

Fusion image data callback data structure

ÿdefinitionÿ

```
typedef struct MERGE_CALLBACK_S {
    AS_CAM_MergeFrameCallback callback;
    void *privateData;
} AS_CAM_Merge_Cb_s;
```

ÿparameterÿ

Member	Description
callback	Data callback function
privateData	User private data

3.1.10 AS_Listener_Callback

ÿillustrateÿ

Listening function callback function

ÿdefinitionÿ

```
typedef void(*AS_Listener_Callback)(AS_CAM_ATTR_S *attr, void *privateData);
```

[Parameters]

Member	Description
attr	Camera Information
privateData	User private data

3.1.11 AS_LISTENER_CALLBACK_S

ÿillustrateÿ

Listening function callback data structure

ÿdefinitionÿ

```
typedef struct AS_LISTENER_CALLBACK {
        AS_Listener_Callback onAttached;
        AS_Listener_Callback onDetached;
        void *privateData;
} AS_LISTENER_CALLBACK_S;
```

ÿparameterÿ

Member	Description
onAttached	Camera connection callback
onDetached	Camera disconnect callback
privateData	User private data

3.1.12 AS_LISTENER_TYPE_E

ÿillustrateÿ

Monitor type, when using a USB camera, only the USB device monitor type needs to be registered, otherwise only the network monitor type needs to be registered. It is recommended to register only the required monitoring types.

ÿdefinitionÿ

```
typedef enum AS_LISTENER_TYPE {
        AS_LISTENNER_TYPE_USB = 0,
        AS_LISTENNER_TYPE_NET,
        AS_LISTENNER_TYPE_BUTT
} AS_LISTENER_TYPE_E;
```

ÿparameterÿ

Member	Description
AS_LISTENNER_TYPE_USB	USB Type
AS_LISTENNER_TYPE_NET Network type	
AS_LISTENNER_TYPE_BUTT invalid value	

3.1.13 IMG_TYPE_FLG

ÿillustrateÿ

Image type flag

ÿdefinitionÿ

```
#define DEFAULT_IMG_FLG          0x00000000
#define DEPTH_IMG_FLG            0x00000001
```

```
#define RGB_IMG_FLG #define          0x00000002
IR_IMG_FLG #define                   0x00000004
POINTCLOUD_IMG_FLG 0x00000008
#define YUYV_IMG_FLG 0x00000010
#define PEAK_IMG_FLG #define         0x00000020
SUB_DEPTH_IMG_FLG 0x00000040
#define MJPEG_IMG_FLG                0x00000080
```

ÿparameterÿ

Member	Description
DEFAULT_IMG_FLG	Default image type
DEPTH_IMG_FLG	Depth Image Type
RGB_IMG_FLG	Color image type
IR_IMG_FLG	IR Image Type
POINTCLOUD_IMG_FLG	Point cloud image type
YUYV_IMG_FLG	yuyv image type
PEAK_IMG_FLG	PEAK Image Type
SUB_DEPTH_IMG_FLG	Additional depth image type (only used by CHANGJIANG-B)
MJPEG_IMG_FLG	MJPEG image type

3.1.14 AS_POINTCLOUD_s

ÿillustrateÿ

Polar coordinate point cloud information

ÿdefinitionÿ

```
typedef struct AS_POINTCLOUD {
        double angle;
        double distance;
        double quality;
} AS_POINTCLOUD_s;
```

ÿparameterÿ

Member	Description
angle	Polar coordinate point cloud angle
distance	Polar coordinate point cloud distance
quality	Polar coordinate point cloud quality

ÿNoteÿThis structure is only supported by KONDYOR, please ignore it for other types of modules.

3.1.15 AS_CONFIG_INFO_s

ÿillustrateÿ

Color RGB camera configuration information

ÿdefinitionÿ

```
typedef struct AS_CONFIG_INFO {
        bool is_Registration;
        bool is_pclOrganized;
} AS_CONFIG_INFO_S;
```

ÿparameterÿ

Member	Description
is_Registration	Whether the depth data and RGB data are aligned
is_pclOrganized	and whether the output is an ordered point cloud

ÿNoteÿThis structure is used to query the RGBD camera configuration information. Please ignore it for other types of modules.

3.2 Communication interface data type

3.2.1 AS_CAM_Parameter_s

ÿDescriptionÿ

Camera internal and external parameters. Note that the depth camera and infrared (IR) camera use the same set of parameters. In the default configuration of the RGB camera, the depth image and the RGB image are generally aligned for output. If the output is data of the depth image and the RGB image alignment, the focal length and optical center coordinate parameters of the infrared/depth camera should use the focal length and optical center coordinates of the RGB camera, otherwise use the original

parameters. ÿDefinitionÿ

```
typedef struct CAM_Parameter { float
        fxir; float
        fyir; float
        cxir; float
        cyir; float
        fxrgb; float
        fyrgb; float
        cxrgb; float
        cyrgb;

        float R00;
        float R01;
        float R02;
        float R10;
        float R11;
        float R12;
        float R20;
        float R21;
        float R22;
        float T1;
        float T2;
        float T3;

        float K1ir;
        float K2ir;
        float K3ir;
        float P1ir;
        float P2ir;
        float K1rgb;

        float K2rgb;

        float K3rgb;

        float P1rgb;

        float P2rgb;

} AS_CAM_Parameter_s;
```

ÿparameterÿ

Member	Description
close	Infrared/depth camera horizontal focal length
for	Infrared/depth camera vertical focal length
cxir	Infrared/depth camera optical center horizontal coordinate
cyir	Infrared/depth camera optical center ordinate
fxrgb	Color camera horizontal focal length
fyrgb	Color camera vertical focal length
cxrgb	Color camera optical center horizontal coordinate
cyrgb	Color camera optical center ordinate
R00	Infrared/depth and color camera extrinsic rotation matrix elements
R01	Infrared/depth and color camera extrinsic rotation matrix elements
R02	Infrared/depth and color camera extrinsic rotation matrix elements
R11	Infrared/depth and color camera extrinsic rotation matrix elements
R12	Infrared/depth and color camera extrinsic rotation matrix elements
R20	Infrared/depth and color camera extrinsic rotation matrix elements
R21	Infrared/depth and color camera extrinsic rotation matrix elements
R22	Infrared/depth and color camera extrinsic rotation matrix elements
T1	Infrared/depth and color camera extrinsic translation matrix elements
T2	Infrared/depth and color camera extrinsic translation matrix elements
T3	Infrared/depth and color camera extrinsic translation matrix elements
K1ir	IR distortion parameters
K2ir	IR distortion parameters
K3ir	IR distortion parameters
P1ir	IR distortion parameters
P2ir	IR distortion parameters
K1rgb	RGB distortion parameters
K2rgb	RGB distortion parameters
K3rgb	RGB distortion parameters
P1rgb	RGB distortion parameters
P2rgb	RGB distortion parameters

3.2.2 AS_STREAM_Param_s

ÿillustrateÿ

Data stream parameters, including resolution and frame rate.

ÿdefinitionÿ

```
typedef struct STREAM_PARAM_S {
        int width;
        int height;
        int fps;
} AS_STREAM_Param_s;
```

ÿparameterÿ

Member	Description
width	Image width
height fps	Image height
	Frame rate

3.2.3 AS_MEDIA_TYPE_E

ÿillustrateÿ

Stream Type

ÿDefinitionÿ

```
typedef enum MEDIA_TYPE_E {
        AS_MEDIA_TYPE_DEPTH,
        AS_MEDIA_TYPE_RGB,
        AS_MEDIA_TYPE_IR,
        AS_MEDIA_TYPE_PEAK,
        AS_MEDIA_TYPE_BUTT
} AS_MEDIA_TYPE_E;
```

ÿparameterÿ

Member	Description
AS_MEDIA_TYPE_DEPTH	Depth Type
AS_MEDIA_TYPE_RGB	RGB Type
AS_MEDIA_TYPE_IR	IR Type
AS_MEDIA_TYPE_PEAK	PEAK Type
AS_MEDIA_TYPE_BUTT	Invalid value

3.2.4 AS_CAM_UpGradeCallback

ÿillustrateÿ

Firmware upgrade status callback function

ÿÿÿÿ

```
typedef void(*AS_CAM_UpGradeCallback)(AS_CAM_PTR pCamera, void *privateData,float fProcess);
```

ÿparameterÿ

Member	Description
pCamera	Get the camera handle through the AS_SDK_GetCameraList interface or the AS_CreateCamHandle interface.

Member	Description
privateData User private data	
fProcess	Upgrade progress, output by the bottom layer, the value range is 0.00-100.00

3.2.5 AS_CAM_Upgrade_Dev_s

ÿillustrateÿ

Device firmware upgrade status callback data structure

ÿdefinitionÿ

```
typedef struct CAM_ UPGRADE_DEV_S {
    AS_CAM_UpGradeCallback callback;
    void *privateData;
} AS_CAM_Upgrade_Dev_s;
```

ÿparameterÿ

Member	Description
callback	Device firmware upgrade status callback function
privateData	User private data

3.2.6 AS_TOFMODE_E

ÿillustrateÿ

Firmware mode, please ignore for non-ToF cameras, only applicable to KUNLUN-A cameras

ÿDefinitionÿ

```
typedef enum TOFMode {
    odd_mode = 0,
    even_mode = 1,
    merge_mode = 2,
    mode_butt
} AS_TOFMODE_E;
```

ÿparameterÿ

Member	Description
odd_mode	Single odd frame mode
even_mode	Single even frame mode
merge_mode	Even-odd alternating frame mode
mode_butt	Invalid value

3.2.7 AS_SDK_CAM_MODEL_E

ÿillustrateÿ

ÿDefinitionÿ typedef enum CAMERA_MODEL_E {

```
        AS_SDK_CAM_MODEL_UNKNOWN,
        AS_SDK_CAM_MODEL_NUWA_XB40,
        AS_SDK_CAM_MODEL_NUWA_X100,
        AS_SDK_CAM_MODEL_NUWA_HP60,
        AS_SDK_CAM_MODEL_NUWA_HP60V,
        AS_SDK_CAM_MODEL_KUNLUN_A,
        AS_SDK_CAM_MODEL_KUNLUN_C,
        AS_SDK_CAM_MODEL_KONDYOR,
        AS_SDK_CAM_MODEL_KONDYOR_NET,
        AS_SDK_CAM_MODEL_HP60C,
        AS_SDK_CAM_MODEL_HP60CN,
        AS_SDK_CAM_MODEL_VEGA,
        AS_SDK_CAM_MODEL_CHANGA,
        AS_SDK_CAM_MODEL_CHANGJIANG_B,
        AS_SDK_CAM_MODEL_EMBANKMENT,
        AS_SDK_CAM_MODEL_EMBANKMENT_A,
        AS_SDK_CAM_MODEL_TAISHAN,
        AS_SDK_CAM_MODEL_O2,
        AS_SDK_CAM_MODEL_EMBANKMENT_B,
        AS_SDK_CAM_MODEL_BUTT
    } AS_SDK_CAM_MODEL_E;
```

ÿparameterÿ

Member	Description
AS_SDK_CAM_MODEL_UNKNOWN	Unknown Type
AS_SDK_CAM_MODEL_NUWA_XB40	XB40
AS_SDK_CAM_MODEL_NUWA_X100	X100
AS_SDK_CAM_MODEL_NUWA_HP60	HP60
AS_SDK_CAM_MODEL_NUWA_HP60V	HP60V
AS_SDK_CAM_MODEL_KUNLUN_A	KUNLUN-A
AS_SDK_CAM_MODEL_KUNLUN_C	KUNLUN-C
AS_SDK_CAM_MODEL_KONDYOR	KONDYOR(USB)
AS_SDK_CAM_MODEL_KONDYOR_NET	KONDYOR(NET)
AS_SDK_CAM_MODEL_HP60C	HP60C
AS_SDK_CAM_MODEL_HP60CN	HP60CN
AS_SDK_CAM_MODEL_VEGA	VEGA
AS_SDK_CAM_MODEL_CHANGA	CHANG-A
AS_SDK_CAM_MODEL_CHANGJIANG_B	CHANGJIANG-B
AS_SDK_CAM_MODEL_EMBANKMENT	TANGGUL
AS_SDK_CAM_MODEL_EMBANKMENT_A	TANGGULA-A
AS_SDK_CAM_MODEL_TAISHAN	TAISHAN
AS_SDK_CAM_MODEL_O2	O2

Member	Description
AS_SDK_CAM_MODEL_EMBANKMENT_B	TANGGULA-B

3.2.8 AS_CAM_ATTR_TYPE_E

ÿillustrateÿ

Camera Type

ÿDefinitionÿ typedef enum AS_CAM_ATTR_TYPE {

```
        AS_CAMERA_ATTR_LNX_USB,
        AS_CAMERA_ATTR_WIN_USB,
        AS_CAMERA_ATTR_NET,
        AS_CAMERA_ATTR_BUTT
    } AS_CAM_ATTR_TYPE_E;
```

ÿparameterÿ

Member	Description
AS_CAMERA_ATTR_LNX_USB	Linux USB Types
AS_CAMERA_ATTR_WIN_USB	Windows USB Type
AS_CAMERA_ATTR_NET	Network Type
AS_CAMERA_ATTR_BUTT	Invalid value

3.2.9 AS_USB_DEV_ATTR_S

ÿillustrateÿ

Linux USB type structure

ÿÿÿÿ

```
    typedef struct AS_USB_DEV_ATTR {
        int wide;
        int pid;
        char serial[64];
        int bnum;
        int dnum;
        int port_number;
        char port_numbers[64];
    } AS_USB_DEV_ATTR_S;
```

ÿparameterÿ

Member	Description
at	Supplier Identification Code
pid	Product Identification Code
serial	SN Serial Number
bnum	Bus number

Member	Description
bottom	Device Number
port_number	Port Number
port_numbers	Port Path

3.2.10 AS_NETWORK_DEV_ATTR_S

ÿillustrateÿ

Network type structure

ÿÿÿÿ

```
typedef struct AS_NETWORK_DEV_ATTR {
        char ip_addr[64];
        int port;
        AS_SDK_CAM_MODEL_E model;
} AS_NETWORK_DEV_ATTR_S;
```

ÿparameterÿ

Member	Description
ip_addr	IP address
port	Port Number
model	Camera Model

3.2.11 AS_USB_WIN_DEV_ATTR_S

ÿillustrateÿ

Windows USB Type Structure

ÿdefinitionÿ

```
#define WinBuffSize 2048

typedef struct AS_USB_WIN_DEV_ATTR {
        int wide;
        int pid;
        int deviceID;
        char symbol_link[WinBuffSize];
        char location_path[WinBuffSize];
} AS_USB_WIN_DEV_ATTR_S;
```

ÿparameterÿ

Member	Description
at	Supplier Identification Code
pid	Product Identification Code
deviceID	Device ID
symbol_link	Unique device identification code
location_path	Device Path

3.2.12 AS_CAM_ATTR_S

ÿillustrateÿ

Camera information structure

ÿDefinitionÿ

```
typedef struct AS_CAM_ATTR {
    AS_CAM_ATTR_TYPE_E type;
    union {
        AS_USB_DEV_ATTR_S usbAttrs;
        AS_NETWORK_DEV_ATTR_S netAttrs;
        AS_USB_WIN_DEV_ATTR_S winAttrs;
    } attr;
} AS_USB_WIN_DEV_ATTR_S;
```

ÿparameterÿ

Member	Description
type	Camera attribute type, which refers to the attribute type in the union
usbAttrs	Linux USB information structure
netImage	Network information structure
winAttrs	Windows USB information structure

3.2.13 AS_NET_DEV_Attrs_s

ÿillustrateÿ

Network device information. This information is supported for use by network devices.

ÿDefinitionÿ

```
typedef struct NET_DEV_ATTRS {
    char ip_addr[64];
    char mac_addr[64];
    int is_dhcp;
} AS_NET_DEV_Attrs_s;
```

ÿparameterÿ

Member	Description
ip_addr	IP address
mac_addr	MAC address
is_dhcp	Whether to enable DHCP. 0 disable, 1 enable, -1 not set

3.2.14 AS_TIME_STAMP_TYPE_E

ÿillustrateÿ

Timestamp Type

ÿDefinitionÿ typedef enum TIME_STAMP_TYPE {

```
        AS_TIME_STAMP_TYPE_STEADY_CLOCK,
        AS_TIME_STAMP_TYPE_SYSTEM_CLOCK,
        AS_TIME_STAMP_TYPE_BUTT
    } AS_TIME_STAMP_TYPE_E;
```

ÿparameterÿ

Member	Description
AS_TIME_STAMP_TYPE_STEADY_CLOCK	The system monotonic clock starts at the boot time 0
AS_TIME_STAMP_TYPE_SYSTEM_CLOCK	System time
AS_TIME_STAMP_TYPE_BUTT	Invalid value

4. ANG SDK Core API Description

4.1 Core interface definition

4.1.1 Initialize SDK

[Function]

SDK Initialization

[Format]

int AS_SDK_Init();

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

In the life cycle of using SDK, it only needs to be initialized once. Multiple initializations will return success, but a warning message will be printed.

4.1.2 Deinitialize SDK

[Function]

SDK deinitialization

[Format]

int AS_SDK_Deinit();

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Used together with the initialization interface AS_SDK_Init(), the SDK life cycle is completed from initialization to deinitialization.

4.1.3 Get camera list

[Function]

Get a list of connected cameras

[Format]

int AS_SDK_GetCameraList(std::list<AS_CAM_PTR> &devList);

[parameter]

Type	Name	Description	IN/OUT
std::list<AS_CAM_PTR>	devList camera list		OUT

[Return value]

Type	Description
int	Greater than 0, the number of cameras; Other abnormal values

[Notes]

The camera list returned by this interface will be recorded in the SDK . If you need to call it repeatedly, please close the stream of the camera that has been opened,

Close the opened camera, then pass the devList obtained by AS_SDK_GetCameraList to the interface for releasing the camera list to release it.

Get the camera list again. Otherwise, an unknown error will occur.

4.1.4 Release camera list

[Function]

Release camera list resources

[Format]

int AS_SDK_FreeCameraList(std::list<AS_CAM_PTR> &devList);

[parameter]

Type	Name	Description	IN/OUT
std::list<AS_CAM_PTR>	devList	Camera List	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

After calling this function, devList will be cleared and the camera pointer in it will be released. Other operations may cause unknown errors.

To prevent unknown errors such as memory leaks, please call this interface at an appropriate location before exiting the SDK to release the camera list resources.

4.1.5 Open the camera

[Function]

Open the specified camera

[Format]

int AS_SDK_OpenCamera(AS_CAM_PTR pCamera, const char *pParaFilePath);

[Parameters]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
char*	pParaFilePath	Camera configuration file path	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Camera profiles are described above in 2.2.1 of this document.

You need to close the camera before calling it repeatedly, otherwise it returns a negative value.

4.1.6 Turn off the camera

[Function]

Close the specified camera

[Format]

int AS_SDK_CloseCamera(AS_CAM_PTR pCamera);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

4.1.7 Registering image data callback function

[Function]

Register image data callback function

[Format]

int AS_SDK_RegisterStreamCallback(AS_CAM_PTR pCamera, AS_CAM_Stream_Cb_s *pstCallback);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_CAM_Stream_Cb_s	pstCallback	Callback Function	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Please note that calling SDK interfaces in the callback function pstCallback is not allowed, as this may cause deadlock.

Perform blocking/delay/time-consuming operations in the callback function, otherwise unknown problems will occur. It is recommended to copy the data from the callback before use to avoid

The problem of data anomaly occurs because the previous frame of data is overwritten by the next frame of data.

4.1.8 Registering image fusion data callback function

[Function]

Register image fusion data callback function

[Format]

int AS_SDK_RegisterMergeFrameCallback(AS_CAM_PTR pCamera, AS_CAM_Merge_Cb_s *pstCallback);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_CAM_Merge_Cb_s	pstCallback	Callback Function	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Only kunlunA is supported. Please note that calling SDK interfaces in the callback function pstCallback is not allowed, as this may cause deadlock problems.

At the same time, please do not perform blocking/delay/time-consuming operations in the callback function, otherwise unknown problems will occur. It is recommended to copy the callback data

Use it later to avoid data anomalies caused by the previous frame data being overwritten by the next frame data.

4.1.9 Open the camera data stream

[Function]

Open an image from the specified camera

[Format]

int AS_SDK_StartStream(AS_CAM_PTR pCamera, int type);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
int	type	Image type IMG_TYPE_FLG, depth, point cloud, etc. One or a combination of Valid parameters. When it is 0, the SDK will follow the default Value output. Combination is done by | operation, such as DEPTH_IMG_FLG|POINGT_CLOUD_FLG	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

4.1.10 Close the camera data stream

[Function]

Close the image of the specified camera

[Format]

int AS_SDK_StopStream(AS_CAM_PTR pCamera, int type);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
int	type	Image type IMG_TYPE_FLG, depth, point cloud, etc. One or a combination of Valid parameter. When it is 0, all images are closed.	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

This interface will be blocked until the image data callback function is executed.

4.1.11 Start device monitoring function

[Function]

Enable the device monitoring function to monitor the status of the camera made by Ansjiang when it is plugged in or out

[Format]

int AS_SDK_StartListener(const AS_LISTENER_CALLBACK_S &callback, AS_LISTENER_TYPE_E type,

bool enumerate = true);

[parameter]

Type	Name	DescriptionDevice	IN/OUT
AS_LISTENER_CALLBACK_S callback		monitoring callback	IN
AS_LISTENER_TYPE_E	type	functionMonitoring	IN
bool	enumerate	typeWhether to enumerate devices	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

This interface only needs to be called once and cannot be called repeatedly, otherwise unknown problems will occur. If only a USB camera is used, only the USB device monitor needs to be registered. Type, otherwise only the network listening type needs to be registered. In order to reduce CPU usage, it is recommended to only register the required listening type.

4.1.12 Stop device monitoring function

[Function]

Stop device monitoring function

[Format]

int AS_SDK_StopListener(void);

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

4.1.13 Create a camera handle

[Function]

Create a camera handle

[Format]

int AS_SDK_CreateCamHandle(AS_CAM_PTR &pCamera,AS_CAM_ATTR_S *attr);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	OUT
AS_CAM_ATTR_S attr		Camera properties.	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

If you are using a Linux-USB camera, the camera attribute attr you enter must contain bnum and dnum information. If you are using a Win-USB camera,

If you are using a network camera, the input camera attribute attr must contain the symbol_link information; if you are using a network camera, the input camera attribute attr must contain the ip_addr

For detailed camera attributes, refer to the AS_CAM_ATTR_S structure.

4.1.14 Destroy the camera handle

[Function]

Destroy the camera handle

[Format]

int AS_SDK_DestoryCamHandle(AS_CAM_PTR pCamera);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

4.1.15 Get camera properties

[Function]

Get camera properties.

[Format]

int AS_SDK_GetCameraAttrs(AS_CAM_PTR pCamera, AS_CAM_ATTR_S &attr);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_CAM_ATTR_S attr		Camera Properties	OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

In the device attributes of Linux-USB, the SN number is not available for acquisition. The SN number can be obtained through AS_SDK_GetSerialNumber().

For non-hot-plug mechanism, this interface can be called to obtain the camera's

Properties; for the hot-plug mechanism, this interface can be called to obtain the properties of the camera during the life cycle from obtaining the camera handle to destroying the handle.

4.1.16 Get SDK version number

[Function]

Get the SDK version number

[Format]

int AS_SDK_GetSwVersion(char *pszVersion, unsigned int size);

[parameter]

Type	Name	Description	IN/OUT
char	pszVersion	sdk version	OUT
unsigned int	size	number pszVersion space maximum value, unit byte IN	

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

none

4.2 Communication interface definition

The communication interface is to communicate with the camera firmware, so all APIs of this type should be used before calling AS_SDK_OpenCamera() to open the camera. Otherwise, an error will be returned.

4.2.1 Get the camera firmware version number

[Function]

Get the camera firmware version number

[Format]

int AS_SDK_GetFwVersion(AS_CAM_PTR pCamera, char *pszVersion, int size);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR pCamera		Camera handle, by AS_SDK_GetCameraList interface or AS_CreateCamHandle interface gets the camera handle Pattern.	IN
char	pszVersion	The maximum value of	OUT
int	size	the camera firmware version number pszVersion space, in bytes IN	

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

none

4.2.2 Get the stream type

[Function]

Get the image type of the stream

[Format]

int AS_SDK_GetStreamType(AS_CAM_PTR pCamera, int *type);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR pCamera		Camera handle, by AS_SDK_GetCameraList interface or AS_CreateCamHandle interface gets the camera handle Pattern.	IN
int*	type	Get the stream image type (IMG_TYPE_FLG)	OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Not currently supported.

4.2.3 Get the camera SN serial number

[Function]

Get the serial number

[Format]

int AS_SDK_GetSerialNumber(AS_CAM_PTR pCamera, char *pszSerialNo, int size);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR pCamera pszSerialNo		The maximum	IN
char*		value of the space pszSerialNo	OUT
int	size	used to store the serial number of the camera handle, in bytes IN	

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

After opening the specified camera through AS_SDK_OpenCamera(), the entire life cycle before closing the camera through AS_SDK_CloseCamera()

This interface can be called at any time.

4.2.4 Upgrading the camera firmware

[Function]

Upgrading the firmware

[Format]

int AS_SDK_UpgradeDev(AS_CAM_PTR pCamera, char *pszFilePath, AS_CAM_Upgrade_Dev_s

*pstCallback);

[Firmware package format]

ÿ—xxx.zip	Total compressed package
ÿÿxx_a.bin	Module Application
ÿÿxx_k.bin	Module kernel
ÿÿxx_b.bin	Background Image
ÿÿxx_h.bin	Himax Files
ÿÿxx_c.bin	Reference drawing/calibration drawing
ÿÿxx_cb.bin	Alternative reference map/calibration map
ÿÿxx_p.bin	Internal and external reference documents
ÿÿxx_x.bin	MAPX File
ÿÿxx_y.bin	MAPY File
ÿÿxx_r.bin	ISP Files
ÿÿxx_wa.bin	Wrap File
ÿÿxx_wb.bin	Alternative Wrap File

ÿÿxx_waxy.bin	Wrap_xy File
ÿÿxx_wbxy.bin	Alternative Wrap_xy file

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera Camera handle,	AS_SDK_GetCameraList interface or AS_CreateCamHandle interface gets the camera handle Pattern.	IN
char*	pszFilePath The firmware path to be upgraded		IN
AS_CAM_Upgrade_Dev_s pstCallback The upgrade status callback function			IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

The firmware upgrade is not supported when streaming is on. If streaming is already on, please turn off streaming before upgrading. The upgrade tool can be obtained from FAE.

4.2.5 Obtaining camera internal and external parameters

[Function]

Get the internal and external parameters of the specified camera

[Format]

int AS_SDK_GetCamParameter(AS_CAM_PTR pCamera, AS_CAM_Parameter_s *pstParam);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_CAM_Parameter_s pstParam		Camera internal and external parameters	OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

For HP60C/HP60CN/VAGE modules, you need to call AS_SDK_StartStream to start the stream and obtain the image data before

The correct internal and external parameters can be obtained. The other models of modules can be obtained after the camera is turned on.

4.2.6 Get the camera firmware mode

[Function]

Get the camera firmware mode

[Format]

int AS_SDK_GetTofMode(AS_CAM_PTR pCamera, AS_TOFMODE_E &mode);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_TOFMODE_E mode		Firmware Mode	OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Only supports KUNLUN series cameras.

4.2.7 Obtaining the resolution capability set

[Function]

Get the resolution capability set

[Format]

int AS_SDK_GetCapability(AS_CAM_PTR pCamera, AS_MEDIA_TYPE_E type,std::vector<AS_STREAM_Param_s> &capability);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_MEDIA_TYPE_ AND	type	Image Type	IN
std::vector<AS_STREAM_Param_s>	capability	Camera image parameters	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

After turning on the camera, the user can obtain the capability set through this interface and set it through the image parameter setting interface according to the needs.

Specifications HP60C/HP60CN/VAGE modules require the frame rates of RGB/DEPTH/IR (if IR is used) to be consistent.

4.2.8 Setting image parameters

[Function]

Setting image parameters

[Format]

int AS_SDK_SetStreamParam(AS_CAM_PTR pCamera, AS_MEDIA_TYPE_E type,AS_STREAM_Param_s *pstStreamParam);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
type AS_MEDIA_TYPE_E type		Stream	IN
AS_STREAM_Param_s pstStreamParam Camera image parameters			IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

You can use the AS_SDK_GetCapability() interface to obtain the supported resolution capability set. For HP60C/HP60CN/VAGE modules, The frame rates of RGB and DEPTH/IR (if IR is used) must be consistent. For all models, the parameters can only be set from the Get Resolution Capability Set. The frame rate and resolution obtained by the port. Note that the frame rate and resolution must be set before the stream is opened to take effect.

4.2.9 Get image parameters

[Function]

Get image parameters

[Format]

int AS_SDK_GetStreamParam(AS_CAM_PTR pCamera, AS_STREAM_Param_s *pstStreamParam);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_STREAM_Param_s pstStreamParam Camera image parameters			OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

4.2.10 Get camera model

[Function]

Get the camera model

[Format]

int AS_SDK_GetCameraModel(AS_CAM_PTR pCamera, AS_SDK_CAM_MODEL_E &model);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	Camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_SDK_CAM_MODEL_E	model	Camera model	OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Can be called after getting the camera list.

4.2.11 Set camera network information

[Function]

Set the camera network information

[Format]

int AS_SDK_SetDevNetAttrs(AS_CAM_PTR pCamera, AS_NET_DEV_Attrs_s attrs);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	Camera handle, received by AS_SDK_GetCameraList Use the AS_CreateCamHandle interface to get the camera Handle.	IN
AS_NET_DEV_Attrs_s	attrs	Including IP address, MAC address, whether DHCP is turned on	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Pay attention to the network information attribute input requirements, see AS_NET_DEV_Attrs_s for details.

4.2.12 Get camera network information

[Function]

Get camera network information

[Format]

int AS_SDK_GetDevNetAttrs(AS_CAM_PTR pCamera, AS_NET_DEV_Attrs_s &attrs);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	Camera handle, obtained by AS_SDK_GetCameraList interface Or use AS_CreateCamHandle interface to get the camera handle Pattern.	IN
AS_NET_DEV_Attrs_s attrs		Including IP address, MAC address, whether DHCP OUT is turned on	

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

none.

4.2.13 Setting the timestamp type

[Function]

Set the timestamp type

[Format]

int AS_SDK_SetTimeStampType(AS_CAM_PTR pCamera, AS_TIME_STAMP_TYPE_E type);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_TIME_STAMP_TYPE_E	type	Timestamp Type	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Only supports Linux version hp60c.

4.2.14 Get camera configuration information

[Function]

Get camera configuration information

[Format]

int AS_SDK_GetConfigInfo(AS_CAM_PTR pCamera, AS_CONFIG_INFO_S &config_info);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList Interface or AS_CreateCamHandle interface to get The camera handle.	IN
AS_CONFIG_INFO_S config_info Configuration information			OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Only supports RGBD cameras. This interface is used to obtain whether RGB and depth are aligned, and the ordered/disordered state of the point cloud.

4.2.15 Get the size of the camera's private data space

[Function]

Get the size of the camera's private data space

[Format]

int AS_SDK_GetPrivateMaxSize(AS_CAM_PTR pCamera, int &size);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList interface or AS_CreateCamHandle interface to obtain The camera handle.	IN
int	size	The size of the private data space obtained	OUT

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Only supports KUNLUN-A and KUNLUN-C cameras

4.2.16 Writing private data to the camera

[Function]

Write private data to the camera

[Format]

int AS_SDK_WritePrivateData(AS_CAM_PTR pCamera, void *data, int size, int offset);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList interface or AS_CreateCamHandle interface to obtain The camera handle.	IN
void	data	Private	IN
int	size	dataPrivate data	IN
int	offset	sizeThe offset of the private data space in the camera	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Only supports KUNLUN-A and KUNLUN-C cameras

4.2.17 Reading Private Data from the Camera

[Function]

Read private data from the camera

[Format]

int AS_SDK_ReadPrivateData(AS_CAM_PTR pCamera, void *data, int size, int offset);

[parameter]

Type	Name	Description	IN/OUT
AS_CAM_PTR	pCamera	camera handle, obtained by AS_SDK_GetCameraList interface or AS_CreateCamHandle interface to obtain The camera handle.	IN
void	data	Private	OUT
int	size	dataPrivate data	IN
int	offset	sizeThe offset of the private data space in the camera	IN

[Return value]

Type	Description
int	0 Success Other abnormal values

[Notes]

Only supports KUNLUN-A and KUNLUN-C cameras

5. Help

5.1 Notes

The ROS package provided by this SDK does not need to be moved to the ROS workspace. You can directly compile and run it according to the method provided in Section 2.2.

5.2 FAQ

5.2.1 How to run Linux programs without root privileges

In Linux systems, root privileges are required to access devices. For programs that operate devices by operating device nodes in the /dev directory,

The first method is to modify the permissions of the device node through the chmod command, but this is only temporary and will not work for devices that are not operated through the /dev device node.

The device's program will not be able to change permissions through chmod. In this case, you can use Linux's udev and rules to manage device permissions.

For Linux systems using the Ansjiang camera module, you can execute the script in the demo/linux_ros/linux/scripts directory of our SDK package.

Create_udev_rules.sh script, you can run Linux programs with normal permissions to access the Ansijiang camera.

5.2.2 How to run a ROS node without root privileges

In Linux systems, root privileges are required to access devices. For programs that operate devices by operating device nodes in the /dev directory,

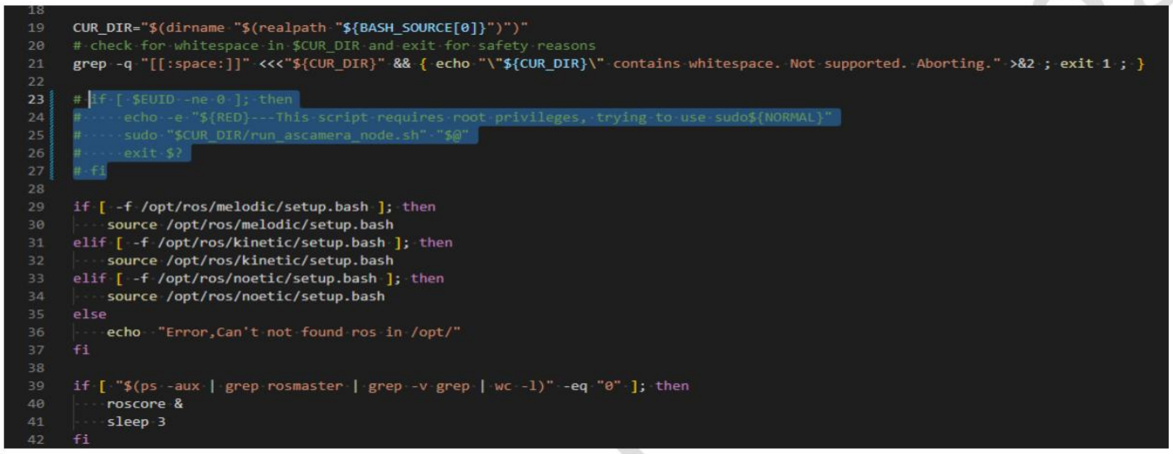
The first method is to modify the permissions of the device node through the chmod command, but this is only temporary and will not work for devices that are not operated through the /dev device node.

The device's program will not be able to change permissions through chmod. In this case, you can use Linux's udev and rules to manage device permissions.

For Linux systems using the Ansjiang camera module, you can execute the script in the demo/linux_ros/ros/src/ascamera/scripts directory.

Create_udev_rules.sh script, you can run ROS node with normal permissions to access the Ansijiang camera. Modify the demo/ros/ directory

In the run_ascamera_node.sh script, comment out the statements that check and apply for root permissions.



```
CUR_DIR="$(dirname "$(realpath "${BASH_SOURCE[0]}")")"
# check for whitespace in $CUR_DIR and exit for safety reasons
grep -q "[[:space:]]" <<<"${CUR_DIR}" && { echo "\"${CUR_DIR}\" contains whitespace. Not supported. Aborting." >&2 ; exit 1 ; }

# if [ $EUID -ne 0 ]; then
#     echo -e "${RED}---This script requires root privileges, trying to use sudo${NORMAL}"
#     sudo "$CUR_DIR/run_ascamera_node.sh" "$@"
#     exit $?
# fi

if [ -f /opt/ros/melodic/setup.bash ]; then
    source /opt/ros/melodic/setup.bash
elif [ -f /opt/ros/kinetic/setup.bash ]; then
    source /opt/ros/kinetic/setup.bash
elif [ -f /opt/ros/noetic/setup.bash ]; then
    source /opt/ros/noetic/setup.bash
else
    echo  "Error,Can't not found ros in /opt/"
fi

if [ "$(ps -aux | grep rosmaster | grep -v grep | wc -l)" -eq "0" ]; then
    roscore &
    sleep 3
fi
```

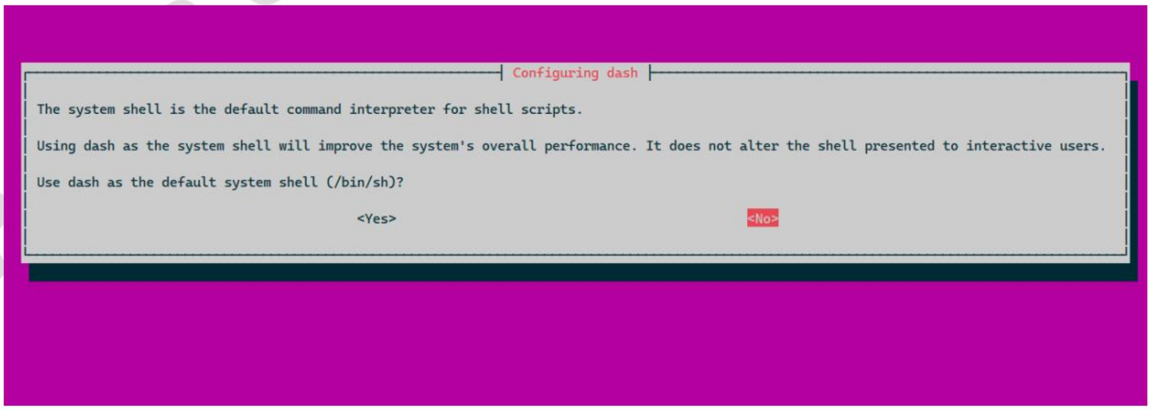
5.2.3 Executing the script provided by SDK and reporting an error

When executing the script provided by the SDK, errors such as Bad substitution or Syntax error: redirection unexpected are reported. This is because the SDK

The shell script is bash shell, and the default shell parser of Ubuntu system is dash. Bash is more powerful than Dash, and some Bash syntax may not be compatible with Dash.

Parsed by dash. You can change the default shell of ubuntu to bash by following the steps below.

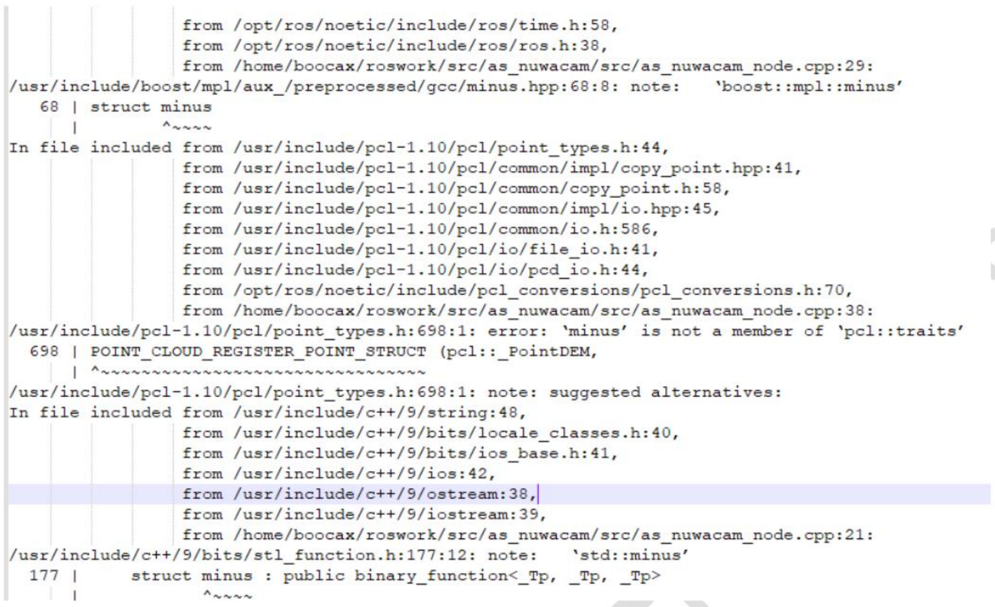
Execute sudo dpkg-reconfigure dash in the terminal and select NO in the pop-up window.



```
Configuring dash

The system shell is the default command interpreter for shell scripts.

Using dash as the system shell will improve the system's overall performance. It does not alter the shell presented to interactive users.

Use dash as the default system shell (/bin/sh)?

                    <Yes>                         <No>
```

5.2.4 Compile error on ubuntu20.04/ubuntu22.04 ROS

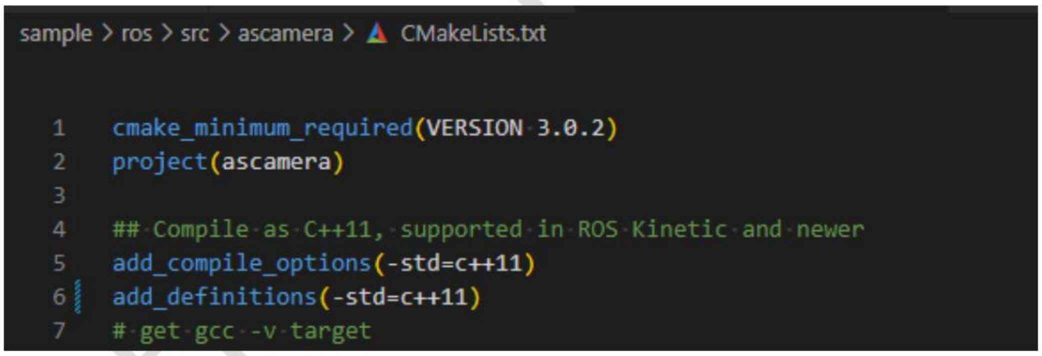
The following error is reported when compiling ROS (Noetic or higher) in ubuntu20.04/ubuntu22.04.



```
                 from /opt/ros/noetic/include/ros/time.h:58,
                 from /opt/ros/noetic/include/ros/ros.h:38,
                 from /home/boocax/roswork/src/as_nuwacam/src/as_nuwacam_node.cpp:29:
/usr/include/boost/mpl/aux_/preprocessed/gcc/minus.hpp:68:8: note:   'boost::mpl::minus'
   68 | struct minus
      |        ^~~~~
In file included from /usr/include/pcl-1.10/pcl/point_types.h:44,
                 from /usr/include/pcl-1.10/pcl/common/impl/copy_point.hpp:41,
                 from /usr/include/pcl-1.10/pcl/common/copy_point.h:58,
                 from /usr/include/pcl-1.10/pcl/common/impl/io.hpp:45,
                 from /usr/include/pcl-1.10/pcl/common/io.h:586,
                 from /usr/include/pcl-1.10/pcl/io/file_io.h:41,
                 from /usr/include/pcl-1.10/pcl/io/pcd_io.h:44,
                 from /opt/ros/noetic/include/pcl_conversions/pcl_conversions.h:70,
                 from /home/boocax/roswork/src/as_nuwacam/src/as_nuwacam_node.cpp:38:
/usr/include/pcl-1.10/pcl/point_types.h:698:1: error: 'minus' is not a member of 'pcl::traits'
  698 | POINT_CLOUD_REGISTER_POINT_STRUCT (pcl::_PointDEM,
      | ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/usr/include/pcl-1.10/pcl/point_types.h:698:1: note: suggested alternatives:
In file included from /usr/include/c++/9/string:48,
                 from /usr/include/c++/9/bits/locale_classes.h:40,
                 from /usr/include/c++/9/bits/ios_base.h:41,
                 from /usr/include/c++/9/ios:42,
                 from /usr/include/c++/9/ostream:38,
                 from /usr/include/c++/9/iostream:39,
                 from /home/boocax/roswork/src/as_nuwacam/src/as_nuwacam_node.cpp:21:
/usr/include/c++/9/bits/stl_function.h:177:12: note:   'std::minus'
  177 |     struct minus : public binary_function<_Tp, _Tp, _Tp>
      |            ^~~~~
```

Since our ROS node sample program is developed based on the melodic version of ROS on Ubuntu 18.04, when ported to ROS-

When using Noetic, you can change -std=c++11 in the ros/src/ascamera/CMakeLists.txt file to -std=c++14 or -std=c++17



```
cmake_minimum_required(VERSION 3.0.2)
project(ascamera)

## Compile as C++11, supported in ROS Kinetic and newer
add_compile_options(-std=c++11)
add_definitions(-std=c++11)
# get gcc -v target
```

5.2.5 NUWA series cameras match abnormally in virtual machines

Since the NUWA series camera is composed of 2 USB devices combined into 1 camera (1 USB communication device + 1 UVC device),

When running on a virtual machine, the USB device may not be paired correctly because the device topology of the virtual machine may be incorrect.

Therefore, when using a virtual machine to run the program, only one NUWA series camera can be connected. If more than one camera is connected, it may cause the above reasons.

Match failed.

5.2.6 KUNLUN-A camera runs abnormally in the virtual machine

KUNLUN-A modules currently only support physical machines and cannot be used on virtual machines.

5.2.7 Abnormal streaming when multiple cameras are running

One possible cause of this problem is that the system kernel parameter usbfs_memory_mb is set too small. usbfs_memory_mb is a kernel parameter. Used to specify the memory size used by the USB file system (usbfs). usbfs is a virtual file system used to transfer data between user space and the kernel. USB device data. This parameter specifies the memory size allocated by usbfs to the USB device communication buffer in MB. By default, This value is 16 MB. By increasing or decreasing this value, you can adjust the memory size of the USB device communication buffer.

Temporary modification method (will become invalid after system restart): Change this kernel parameter to 64/128/512 or larger.

ÿÿÿecho "64" | sudo tee -a /sys/module/usbcore/parameters/usbfs_memory_mb

Customers can set the permanent modification method according to the system they are using.

5.2.8 Failed to open USB camera in Windows 10/Windows 11

If the camera fails to be opened on Windows, and the SDK log prints "SETUP – COM already setpup – threaded VI might not be possible. The reason for this problem is that the video stream capture SDK on Windows requires the use of SDK components, and the SDK is designed to use COM multi-threaded However, some software, such as QT components, also use com components, but in single-instance mode. Different modes may cause sdk Failed to open the camera.

Because most of them need to disable the use of COM components, or use CoUninitializ(void) to deinitialize the COM components before using the SDK, and then Always start the SDK and let the SDK determine the mode of the com component.

Service and After-Sales Support

Email: info@angstrong.com

sales@angstrong.com

Website: www.angstrong.com

Tel: 0755-86568667