

# **Users' Guide**

1. Company and product introduction.....	1-1
1.1 Introduction to Obi Zhongguang.....	1-1
1.1.1 3D Sensor .....	1-1
1.1.2 3D and 2D visual contrast... ..	1-2
1.1.3 Pros and cons of different 3D Sensing technical solutions.....	1-3
1.2 Orbbec product line and development resources.....	1-4
1.2.1 Orbbec product line.....	1-4
1.2.2 System requirements and development resources.....	1-5
1.3 Orbbec product overview.....	1-1
1.3.1 Product naming rules.....	1-1
1.3.2 Product PID Allocation Table... ..	1-1
1.3.3 Orbbec product parameters.....	1-2
1.4 Comparison of Sensor Types of Orbbec Products.....	1-3
1.5 Explanation of common terms in 3D Sensor.....	1-1
1.5.1 Depth .....	1-1
1.5.2 Depth Map.....	1-1
1.5.3 Commonly used terms.....	1-2
2. Product integration design.....	2-1
2.1 Hardware integration design.....	2-1
2.1.1 Product physical interface.....	2-1
2.2 Structural integration design.....	2-2
2.2.1 Reference for structural seal design.....	2-2
2.2.2 Absolute distance reference point.....	2-2
2.2.1 Reference for heat dissipation design.....	2-3
2.3 Driver installation and system adaptation.....	2-3
2.3.1 Windows system driver installation and diagnosis.....	2-3
1) System requirements.....	2-3
2) Driver import.....	2-3
2.3.2 Android system adaptation.....	2-4
1) System requirements.....	2-4
2) Test method.....	2-4
3) Exception handling.....	2-4
3. SDK introduction and use.....	3-1
3.1 Introduction to SDK ... ..	3-1
3.2 OpenNI2.3.0.50 SDK directory structure.....	3-2
3.2 SDK project environment and configuration.....	3-2

3.2.1 Windows engineering environment configuration.....	3-2
(1) Scope of application.....	3-2
(2) Development platform establishment.....	3-2
(3) SimpleCode description.....	3-4
3.2.2 Linux environment configuration.....	3-5
(1) Scope of application.....	3-5
(2) Linux environment configuration instructions.....	3-5
(3) udev description.....	3-6
(4) SimpleCode description.....	3-6
3.2.3 Android project environment configuration.....	3-6
(1) Scope of application.....	3-6
(2) Android environment configuration instructions.....	3-6
(3) SimpleCode description.....	3-8
3.2.4 Instructions for using .ini configuration file.....	3-11
(1) OpenNI.ini configuration instructions.....	3-11
(2) Orbbec.ini configuration instructions.....	3-11
4. Introduction to commonly used APIs.....	4-1
4.1 Introduction to Standard API.....	4-1
4.1.1 OpenNI class.....	4-2
1 Introduction.....	4-2
2) Basic access to equipment.....	4-2
3) Basic access to Video Streams.....	4-2
4) Event-driven access to the device.....	4-2
5) Error message.....	4-3
6) Version information.....	4-3
4.1.2 Device class.....	4-3
1 Introduction.....	4-3
2) Equipment connection conditions.....	4-3
3) Basic operation.....	4-3
4.1.3 VideoStream class.....	4-5
1 Introduction.....	4-5
2) Basic functions of video streaming.....	4-5
3) Get information about the video stream... ..	4-5
4) Configure the video stream.....	4-6
4.1.4 VideoFrameRef class.....	4-7
1 Introduction.....	4-7
2) Access frame data.....	4-7

3) Cropping data .....	4-7
4) Timestamp (Timestamp) .....	4-7
5) Frame Index (FrameIndex) .....	4-7
5) Video Mode .....	4-7
6) Array Stride .....	4-7
4.1.5 Other support categories.....	4-8
1 Introduction.....	4-8
2) Sensor configuration class.....	4-8
3) Sensor Info (SensorInfo) .....	4-8
4) Video Mode (VideoMode) .....	4-8
5) Camera Setting (CameraSetting) .....	4-8
4.2 Introduction to Extended API.....	4-8
4.2.1 Get the device model.....	4-8
4.2.2 Get the device serial number.....	4-8
4.2.3 Set LDP switch.....	4-9
4.2.4 Set LDM switch.....	4-9
4.2.5 Set and get IR exposure value.....	4-9
4.2.6 Setting and obtaining IR gain.....	4-9
4.2.7 Setting, saving and obtaining camera calibration parameters.....	4-9

## 1.0 Company and product introduction

### 1.1 Introduction to Obi Zhongguang

Aobi Zhongguang is a 3D vision sensor technology solution provider integrating R&D, production and sales. It is the fourth company in the world.

A company that can mass produce 3D Sensors. Mainly for face recognition, gesture recognition, skeleton

recognition, three-dimensional measurement, environmental perception and

Provide hardware layer (physical layer) support in technical fields such as 3D map reconstruction. For a complete set of solutions at the application layer, the company

Integrate the resources of partners to provide support to customers.

ORBEC®Astra series, P series, Deeye series and Canglong depth cameras use structured light 3D imaging technology to acquire objects

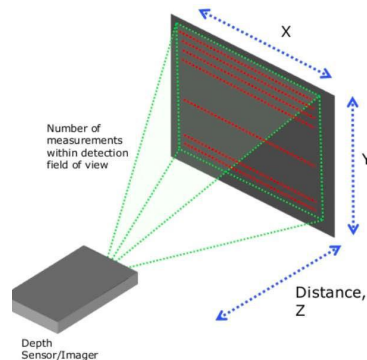
The depth image of the body, while using a color camera to collect color images of the object. Among them, Astra series, P series, Deeye series

The columns are based on monocular structured light 3D imaging technology, and the Canglong series are based on binocular structured light 3D imaging technology. Support cross-platform development

Toolkit SDK.

#### 1.1.1 3D Sensor

The 3D camera is characterized by not only being able to obtain a plane image, but also the depth information (Z axis) of the subject, that is, position and distance information. The 3D camera acquires the depth information, three-dimensional size and spatial information of environmental objects in real time, and provides basic technical support for "pain-type" application scenarios such as motion capture, three-dimensional modeling, VR/AR, indoor navigation and positioning.



What the 3D Sensor outputs is not a 3D model, but an RGB image plus a depth image.

#### 1.1.2 3D and 2D visual contrast

More feature information and higher recognition accuracy

The 3D camera technology collects the depth information of the face image when shooting, which can obtain more characteristic information and greatly improve the recognition accuracy on the basis of traditional face recognition technology. Compared with 2D face recognition systems, 3D face recognition can collect depth feature information such as the distance between the corners of the eyes, the tip of the nose, the alar point, the distance between the two temples, and the distance from the ear to the eye, and these parameters generally do not vary. As a person undergoes plastic surgery and changes his hairstyle, major changes occur, so 3D face recognition can continue to maintain a very high recognition accuracy rate when user characteristics change.

	3D	2D
FRA (错误接收率: 系统接受错误对象概率)	0.047%	0.120%
FRR (错误拒绝率: 系统拒绝真实对象概率)	0.103%	9.790%
姿态变化后识别率	100.00%	23.00%
头发遮挡后识别率	87.00%	50.00%
头部遮挡 (帽子、头盔) 后之识别率	95.00%	<5.00%
弱光环境识别率	100%	0.00%

资料来源: 新加坡出入境管理局、安信证券研究中心

### More accurate object segmentation and higher security

2D images cannot segment objects correctly due to lack of depth information in complex scenes;

2D images can also be used for face recognition, but the 2D camera has not yet been used for face-swiping payment because it is not safe enough. In fact, it means that the flat information collection has a relatively large security risk and may be compromised by photos.

### Gesture recognition: a new way of interaction

The key to gesture recognition lies in the 3D perception technology. The 3D camera acquires image depth information and recognizes user gestures through algorithm processing, so that users can control smart terminals in space, free users from touching the screen, and actively capture user gestures and actions. Recognition processing will become the next interactive pain point!

### Model reconstruction of 3D data

The 3D camera can be used to create a "point cloud" of the surrounding environment. The point cloud data combined with the RGB information of the environmental image can be used to restore the scene. After that, multiple applications such as distance measurement, virtual shopping, decoration, etc. can be derived on this basis, such as Virtual furniture placement, because the restored scene has in-depth information, the simulated furniture cannot continue to be pushed when it encounters an obstacle, and it has a strong sense of reality.

#### 1.1.3 Pros and cons of different 3D Sensing technical solutions

##### 3D structured light scheme

As the name suggests, structured light is light with a special structure, such as discrete light spots, fringe light, and coded structured light. Projecting such a one-dimensional or two-dimensional image onto the object to be measured, based on the size and distortion of the image, can determine the surface shape of the object to be measured, that is, the depth information. For example, take a flashlight to illuminate a wall, stand close or stand far, the light spots on the wall are of different sizes, and the light spots will appear different ellipses when illuminating the wall from different angles. This is the basis of structured light.

Because the energy density of the diffracted spot decreases at a certain distance, it is not suitable for long-distance depth information collection.

##### TOF program

What TOF emits is not speckle, but a surface light source, so there will not be a lot of attenuation within a

certain distance, and the depth accuracy will not decrease significantly as the distance increases. But the corresponding power consumption is higher, and the current depth image resolution is low.

### Binocular stereo vision

Binocular stereo vision uses two or more cameras to collect images at the same time. By comparing the differences in the images obtained by these different cameras at the same time, algorithms are used to calculate depth information, thereby multi-angle three-dimensional imaging.

Program	Binocular	Structured light	TOF
Basic principles	Binocular matching Parallax algorithm	Laser speckle coding	Flight time (phase difference)
Laser light source	None (passive)	Speckle laser	Uniform surface laser
Power consumption	low	Medium	High
Image Resolution	Middle high	Medium	low
Precision	low	Medium	Middle high
Frame rate	low	Medium	High
Low light performance	weak	good	good
Strong light performance	good	weak	Medium
advantage	Mature technology, high resolution of plane information, Low power consumption	Good anti-interference, recognition distance far	Strong light has good anti-interference, power consumption high
Highlight shortcomings	The algorithm has high complexity and poor real-time performance; Not applicable in dim environments; The target feature texture change is not obvious Method of use.	Will be exposed to light under strong light conditions influences	Low depth image resolution;

To put it simply, the binocular has good outdoor performance, but it is not suitable for occasions where the texture changes are not obvious, such as white walls; TOF has better accuracy at long distances, but it is currently limited by low image resolution and high power consumption;

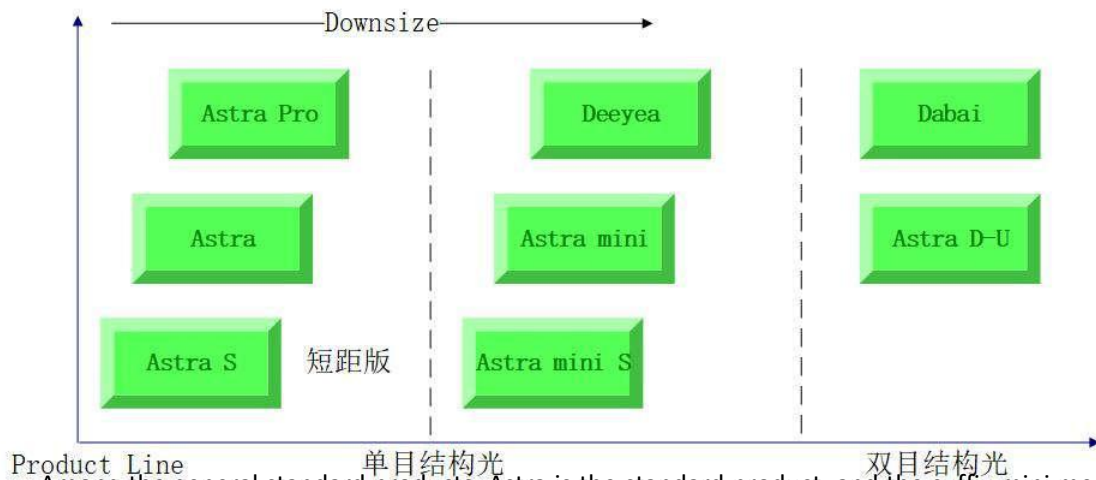
The comprehensive performance of structured light is currently the best among the three 3D sensing technologies, but the strong outdoor light interference is obvious and the accuracy at long distance

difference. Therefore, Aobi Zhongguang launched the second-generation product P1, Deeyea and binocular structured light products in the 940nm band.

Number, try to improve the inherent defects of structured light.

## 1.2 Orbbec product line and development resources

### 1.2.1 Orbbec Product Line



Among the general standard products, Astra is the standard product, and the suffix mini means that the size is miniaturized; the suffix S is short, which means that the depth detection range is shorter; the suffix pro means the RGB upgrade version, which supports higher resolution color images, and The color data stream is transmitted through the UVC protocol, which is different from other products.

Deeyea is our latest generation single target product

Astra D-U and Dabai are the latest binocular structured light products with the best outdoor performance

### 1.2.2 System requirements and development resources

#### 系统要求 Android

- Android OS 4.4/ 5.1/6.0/7.1
- USB 2.0 或 3.0 ( 支持 host 接口 )
- 建议 RK3288 ( 四核 Cortex-A17 , 主频 1.8GHz ) 或以上
- 建议 2GB RAM 或以上
- 支持 LibUSB+LibUVC
- 支持 UVC 设备
- 支持 SELinux 权限访问 UVC 设备

#### 系统要求 Windows

- Windows 7,10, 32-bit and 64-bit
- USB 2.0 或 3.0
- 双核 , 主频 2.2+GHz 或以上
- 建议 4GB RAM 或以上

Aobi Zhongguang provides customers with development code packages under Windows/Android/Linux/Unity/ROS systems and supports C++/Java language development. To obtain the SDK package, contact the staff.



**Note:**

The Astra SDK2.0 provided by Obi Zhongguang's Chinese and English official website, the SDK obtained through this channel is only for gesture and skeleton development and learning;

### 1.3 Orbbec product overview

#### 1.3.1 Product naming rules

For the five products of Astra, Astra S, Astra mini, Astra mini S and Astra Pro, they are named with letters + serial numbers.

The comparison of different letters and products is as follows.

product type	Logo letter
Astra	A
Astra S	B
Astra Pro	C
Astra mini	D
Astra mini S	E

By checking the device SN number, you can intuitively judge the product type, chip generation and production date, as follows.

E1804300018					
E	18	4	30	3	0018
mini s	年	月	日	二代芯片	序列号
备注:	2代表一代芯片				

### 1.3.2 Product PID allocation table

VID: 0x2BC5				
NO.	model	PID	UVC	Remarks
1	Astra	0x0401		Monocular structured light
2	Astra S	0x0402		Monocular structured light
3	Astra Pro	0x0403	0x0501	Monocular structured light
4	Astra mini	0x0404		Monocular structured light
5	Astra mini S	0x0407		Monocular structured light
6	Astra D-U	0x0608	0x0508	Binocular structured light
7	Deeyea	0x060b	0x050b	Monocular structured light
8	Astra Pro Plus	0x060d	0x050d	Monocular structured light
9	Dabai	0x060e	0x050e	Binocular structured light

Under Linux, you can check whether there is a corresponding device connected through lsusb.

### 1.3.3 Orbbec product parameters

Orbbec产品概览								
项目	Astra	Astra S	Astra Pro	Astra mini	Astra mini S	Deeyae	Dabai	Astra D-U
Orbbec ASIC	✓	✓	✓	✓	✓	✓	✓	✓
独立ISP	✗	✗	✓	✗	✗	✓	✓	✓
接近传感器	HW	HW	HW	✗	✗	SW	✗	✗
外壳	✓	✓	✓	✗	✗	✓	✗	✗
麦克风	✓	✓	✓	✗	✗	✗	✗	✗
深度范围 (米)	0.6-8	0.4-2	0.6-8	0.6-8	0.4-2	0.25-1.5	0.3-3	0.3-3
Baseline	75mm	75mm	75mm	55mm	55mm	40mm	40mm	40mm
平均功耗 (W)	<2.4	<2.4	<2.5	<2.4	<2.4	<2.5	1.6W	2.36W
彩色支持UVC	✗	✗	✓	✗	✗	✓	✓	✓
精度	±3mm@1m	±3mm@1m	±3mm@1m	±3mm@1m	±3mm@1m	±3mm@1m	TBD	±5mm@1m
深度分辨率	1280x1024@7fps 640x480@30fps 320x240@30fps 160x120@30fps	1280x1024@7fps 640x480@30fps 320x240@30fps 160x120@30fps	1280x1024@7fps 640x480@30fps 320x240@30fps 160x120@30fps	1280x1024@7fps 640x480@30fps 320x240@30fps 160x120@30fps	1280x1024@7fps 640x480@30fps 320x240@30fps 160x120@30fps	1280x800@30FPS 640x400@30FPS	640*400@30FPS 320*200@30FPS	640*400@30FPS 320*200@30FPS
深度FOV	H 58.4° V 45.5°	H 58.4° V 45.5°	H 58.4° V 45.5°	H 58.4° V 45.5°	H 58.4° V 45.5°	H 67.9° V 45.3°	H 67.9° V 45.3°	H 67.9° V 45.3°
彩色分辨率	1280x960@7fps 640x480@30fps	1280x960@7fps 640x480@30fps	1280x720@30fps 640x480@30fps 320x240@30fps	1280x960@7fps 640x480@30fps 320x240@30fps	1280x960@7fps 640x480@30fps 320x240@30fps	1920x1080@30FPS 1280x720@30FPS 640x480@30FPS	1920*1080@30FPS 1280*720@30FPS 640*480@30FPS	1920*1080@30FPS 1280*720@30FPS 640*480@30FPS

#### 1.4 Comparison of Sensor Types of Orbbec Products

Applicati on products	Astra/Astra S/Astra mini/Astra mini S	
Sensor	MT9M001	MT9M114
Spec		
制造商	ON semiconductor	ON semiconductor
Shutter	Rolling shutter	Rolling shutter
Pixels	1-Megapixel	1.26-Megapixel

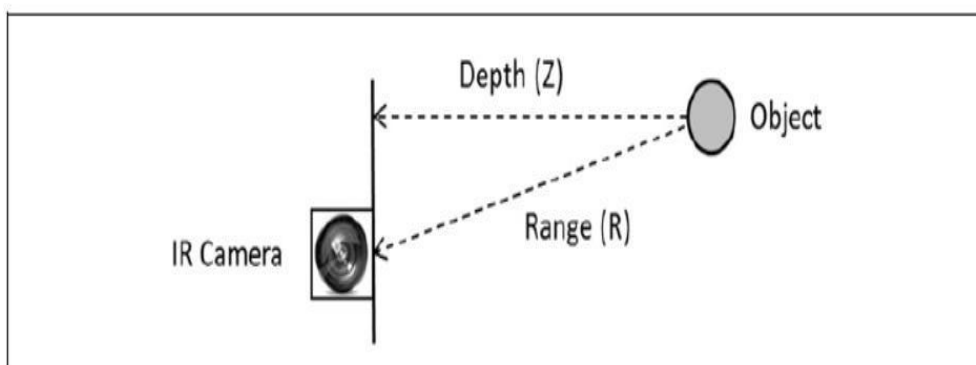
应用产品	Astra Pro	
Sensor	MT9M001	OV9712
Spec		
制造商	ON semiconductor	OmniVision
Shutter	Rolling shutter	Rolling shutter
Spec	1-MegaPixel	1-Megapixel

---

## 1.5 3D Sensor common term explanation

### 1.5.1 Depth

Compared with traditional 2D cameras, 3D cameras have added depth information. The so-called depth value Depth refers to the vertical distance from the measured object to the camera plane, that is, the Z value in the figure below, not R.



### 1.5.2 Depth Map

Unlike RGB images, each pixel in the depth map saves the depth value data of the object in the field of view from the camera plane. deep

The original data of the degree is usually 16-bit unsigned int type, the unit can be specified through the SDK, usually 1mm, that is, each

Pixels store 16-bit unsigned integer data, the unit is 1mm.

In order to display the depth data visually, it is usually converted to grayscale display, as shown in the following figure, different grayscales are represented

Different depth values.



### 1.5.3 Commonly used terms

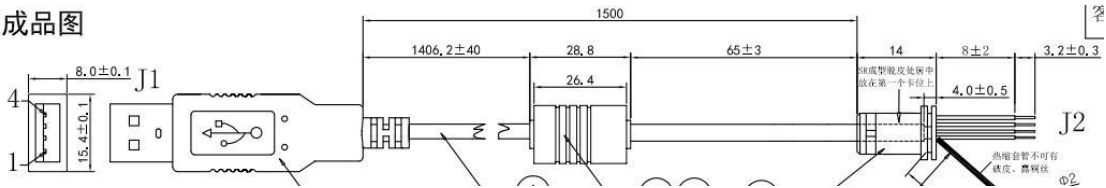
terminology	description
Baseline	1) Monocular structured light: the distance between the imaging center of the infrared camera and the projection center of the infrared projector 2) Binocular structured light: the distance between the imaging centers of the left and right infrared cameras
Depth	The depth video stream is just like the color video stream, except that each pixel has a value representing the distance from the camera  Instead of color information
FOV	Field of view, used to describe the angle range of the camera imaging a given scene, there are three main types: horizontal field of view (H FOV), vertical field of view (V FOV) and diagonal field of view (D FOV)
Depth processor	Depth calculation processor, dedicated ASIC chip used to implement depth calculation algorithm and output depth image, such as MX400, MX6000
IR camera	Infrared camera, or infrared camera
RGB camera	Color camera, or color camera
LDMP/LDM	IR projector, also called infrared laser projector, structured light projector, etc., used to emit structured light patterns
Depth camera	Depth camera, including depth imaging module and color imaging module. The depth imaging module is generally composed of an infrared projector, an infrared camera and a deep cloud computing processor. The color imaging module generally refers to a color camera

## 2.0 Product integrated design

### 2.1 Hardware integration design

#### 2.1.1 Product physical interface

All Obi Zhongguang products are USB interfaces, and most of them are standard Type-A interfaces.



Deeyea is a Type-C interface, and the detailed interface definition can be found in the specification.

P1 is a micro-USB interface, and the additional interface of 20PIN is customized, which can be open to ordinary users and does not provide wires.

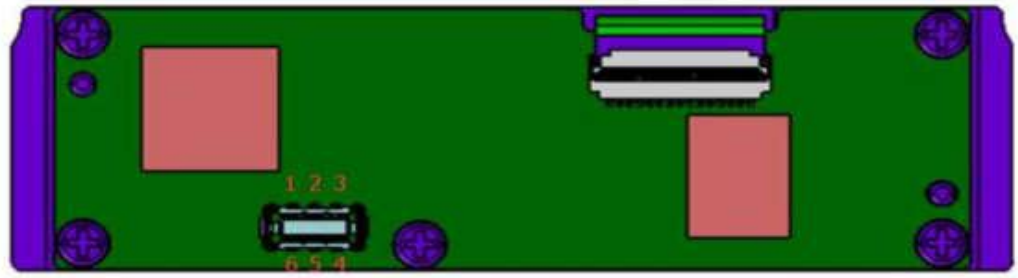
Pin	名称	线的颜色	描述
1	VBUS	Red (红)	电源正 5 V
2	D-	White (白)	数据线负
3	D+	Green (绿)	数据线正
4	ID	none (无)	分为 A 和 B 两种接口 A: 与地线相连 B: 不与地线相连
5	GND	Black (黑)	电源负

表 1 Micro USB 对应接口的功能

Pin	名称
1	VBUS
2	VBUS
3	VBUS
4	GND
5	GND
6	D+
7	D-
8	GND
9	GND
10	GND

表 2 20PIN 连接器对应接口的功能

Astra D-U physical interface is a 6PIN B2B connector, model CPB9506-0113E, which transmits data via USB protocol



POSITION	NAME
1	VBUS
2	D+
3	D-
4	GND
5	ID
6	GND

## 2.2 Structural integration design

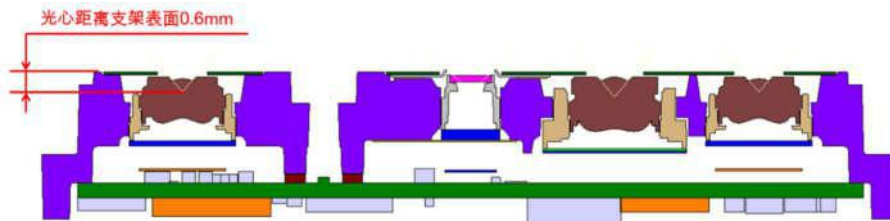
### 2.2.1 Structural seal design reference

For details, see the description of the structural seal design in the product datasheet.

### 2.2.2 Absolute distance reference point

#### 1) Astra D-U

The distance between the entrance pupil of the Astra D-U and the surface of the stent is 0.6mm. When testing the absolute distance, the outer surface of the stent is often used as a reference.



#### 2) Astra S

The distance from the entrance pupil of Astra S to the inner surface of the housing is 6.013mm. When testing the absolute distance, the housing surface is often used as a reference.

### 2.2.1 Thermal Design Reference

The company's miniaturized products (Astra mini, Deeyea, Astra D-U, etc.) need to consider heat dissipation when embedded in the design. For details, see the heat dissipation design reference document.

## 2.3 Driver installation and system adaptation

### 2.3.1 Windows system driver installation and diagnosis

#### 1) System requirements

Windows 7, 8 and 10 (x86, x86-64); x86-based processor @ 1.8+ GHz

## 2) Drive import

Before the Windows system test and use, you need to install the driver. The driver file is in the SDK/drivers directory. After the driver is installed correctly, you can

To view the device through the device manager. If you encounter an exception, please check the "Astra Driver Installation and Device Diagnosis Guide" in the SDK.

Row-driven exception handling.

### 2.3.2 Android system adaptation

#### 1) System requirements

The Android system does not need to install additional product drivers.

- Android OS 4.4/ 5.1/6.0/7.1
- USB 2.0 或 3.0 ( 支持 host 接口 )
- 建议 RK3288 ( 四核 Cortex-A17 , 主频 1.8GHz ) 或以上
- 建议 2GB RAM 或以上
- 支持 LibUSB+LibUVC
- 支持 UVC 设备
- 支持 SELinux 权限访问 UVC 设备

#### 2) Test method

Install the NiViewer.apk in the SDK and apply it to the corresponding device, you can view the depth map and color map of the device.

The following points need to be noted:

Orbbec equipment is divided into UVC and non-UVC equipment. For UVC equipment, such as Deeyea, Canglong, etc., it cannot pass

NiViewer\_for\_android.apk is used to view color images; UVC is a common protocol standard for USB video capture devices. You can use the Android system camera to directly open and view color images. You can use libuvc or

Android system camera.

Different devices of Orbbec have different resolutions. Use NiViewer\_for\_android.apk to pay attention to selecting the corresponding resolution. For example, Deeyea, P1, etc. need to choose 640\*400 resolution, while Orbbec Pro A needs to choose 640\*480, otherwise the depth map preview is abnormal.

#### 3) Exception handling

Android device cannot recognize Camera

Common problems that are not recognized by Android devices can be solved through the following steps:



a) Insert the device into the Windows side, install the driver SensorDriver, and check whether the device is loaded normally in the device manager. If it fails to load normally, install "Driver Installation and Diagnosis Guide" to troubleshoot the driver;

b) Use NiViewer.exe on the Windows side to test whether the device can produce pictures normally and troubleshoot the device

c) If it is normal, reconnect the Android device, connect adb, and use dumpsys USB to check whether it can be loaded normally

Orbbec device node. The Vendor ID of the Orbbec device is 2bc5.

d) If it cannot be loaded normally, it is usually judged to be a USB interface hardware circuit or power supply problem at the device side. Orbbec devices generally need 5V 500mA power supply, and the instantaneous current peak value can reach 700mA.

e) If the device node is found, it can basically be judged as a system compatibility problem, and confirm whether the system supports USB host. for

For systems above Android 6.0, you need to confirm to close the Selinux permission, and use getenforce to check whether the status is Permissive; if not, use setenforce 0 to close it under administrator permissions.

f) If you still cannot use the device normally, you can open the NiViewer application and grab the logfile for specific analysis.

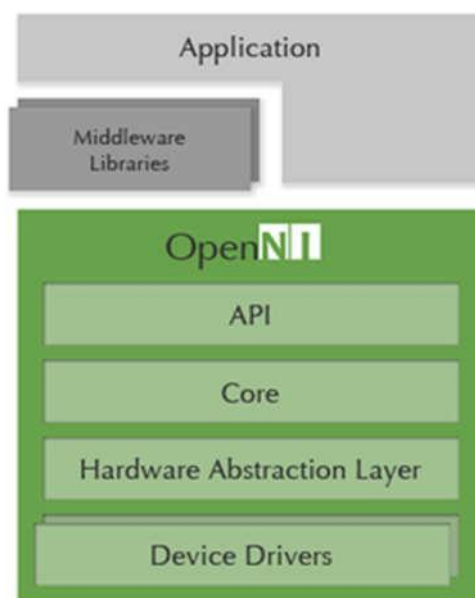
### 3.0 SDK introduction and use

#### 3.1 Introduction to SDK

The OpenNI2.3 series SDK provided by Obi Zhongguang is developed based on OpenNI2. OpenNI2 (Open Natural Interaction) is a multi-language, cross-platform framework that defines the interface between applications, middleware and 3D sensing devices.

Aobi Zhongguang's full range of products fully support the OpenNI protocol, which means that any application developed using OpenNI can be seamlessly connected to the company's products.

When using OpenNI2.3.0.50SDK, it is assumed that the user has a certain understanding of OpenNI. If you have not contacted OpenNI related development before, you can refer to the official OpenNI2 help document. The overall framework of OpenNI2 is shown below:



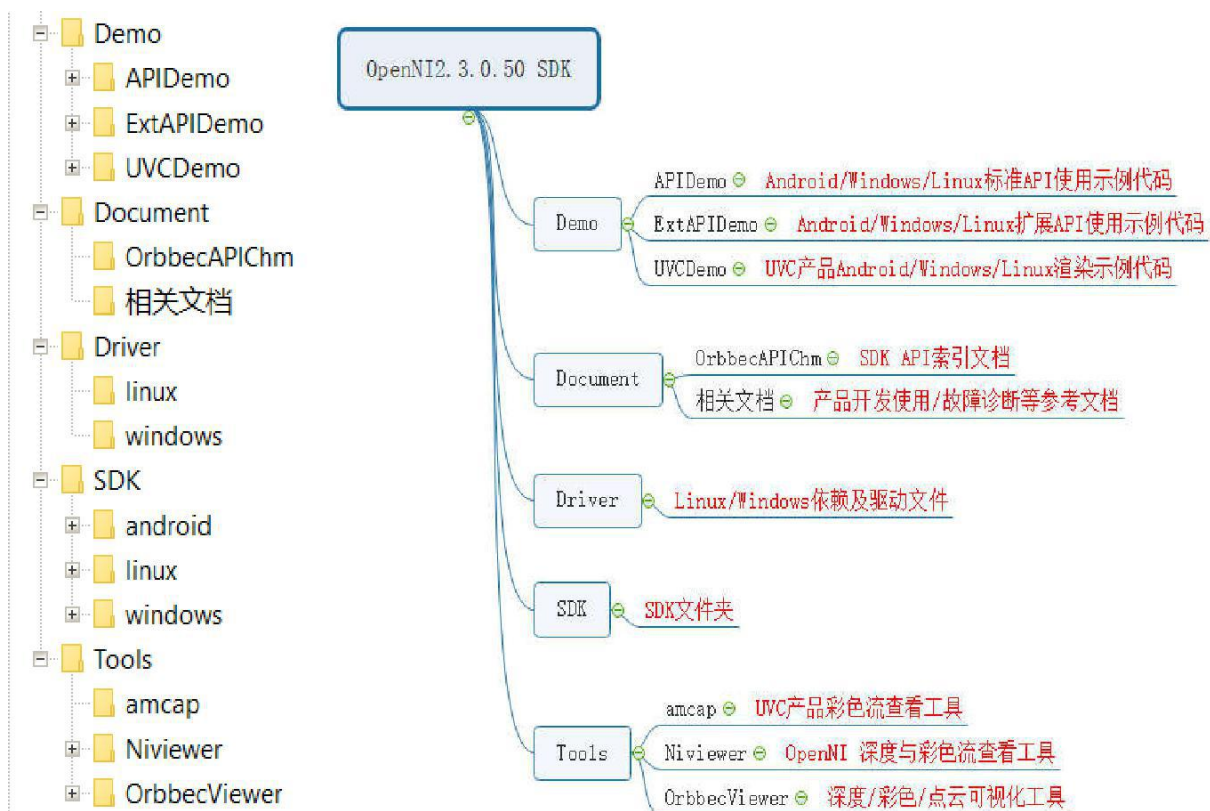
The top layer is about OpenNI2 applications such as NITE gesture recognition, body motion detection, etc.

Next is the unified interface provided by OpenNI2. The header file corresponding to these interfaces is OpenNI.h

OpenNI Core is the core part of OpenNI2, and the structure implementation in OpenNI.h is in this part; a unified API is provided for the Driver layer for the development and extension of the Driver. The header file corresponding to this part of the API is OniDriverAPI.h

The bottom layer is hardware drivers or third-party libraries

### 3.2 OpenNI2.3.0.50 SDK directory structure



## 3.2 SDK project environment and configuration

### 3.2.1 Windows engineering environment configuration

#### (1) Scope of application

The company's SDK is suitable for X86/X64 Windows7 and above platforms

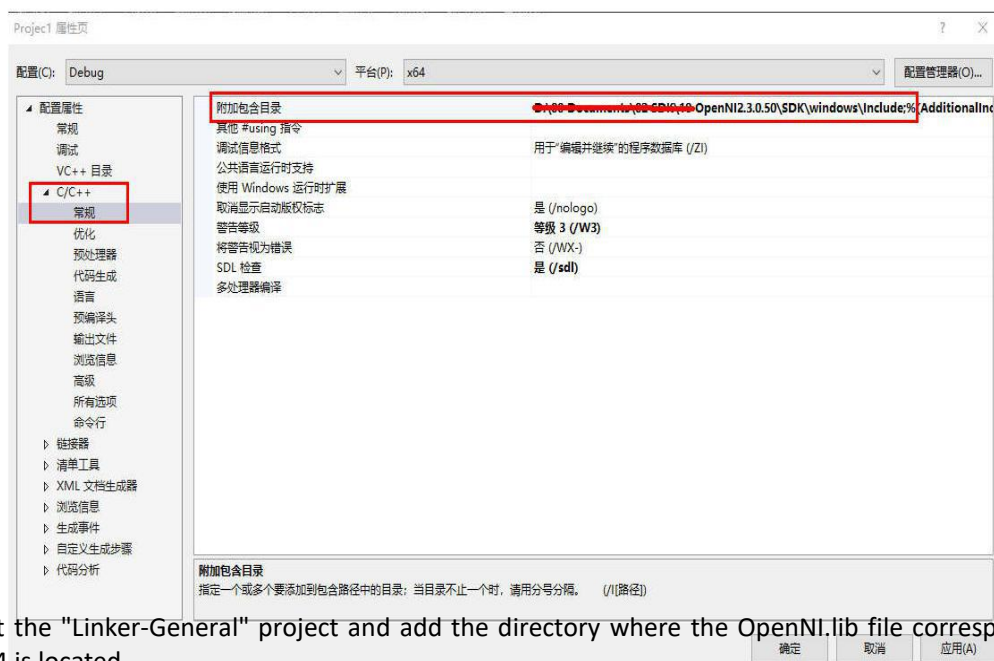
#### (2) Development platform construction

The recommended Windows development platform is Visual Studio 2013 or above. For the Visual Studio platform, you can use

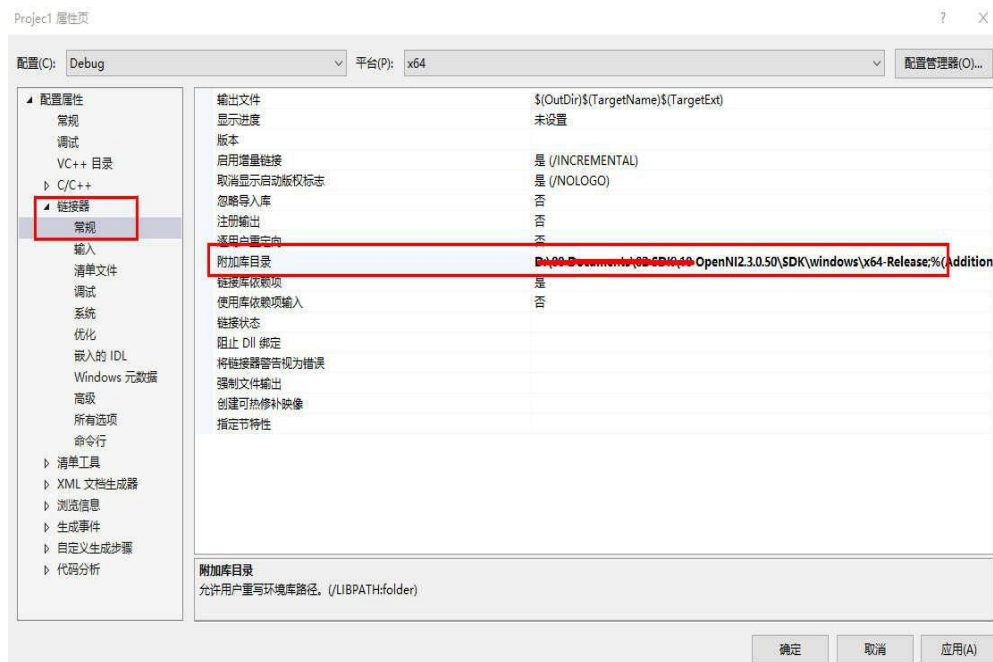
The next steps to build a development environment:

Create or open a C++ project

Select "Project-Properties", add the \SDK\windows\Include directory to the header file dependency path, pay attention to the property configuration (x86/x64) consistent with the runtime



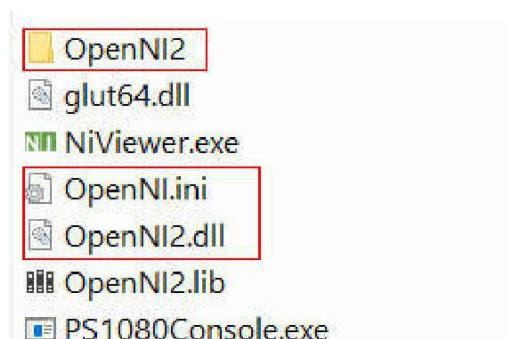
Select the "Linker-General" project and add the directory where the OpenNI.lib file corresponding to x86 or x64 is located,



Add the OpenNI.lib item in the "linker-input" directory, complete the environment configuration, and start editing your own code.

Note that when compiling and running the project, put the OpenNI folder, ini and dll in the x64-release or x86-release folder

Copy the file to the directory where the exe file is located, otherwise it will crash or report an error.



### (3) SimpleCode description

Under the Windows system, different sample codes are provided for applications such as the standard API

and extended API in the SDK. The specific operations and

For instructions, please check the readme file in the corresponding folder.

Steps to get the depth stream:

(1) Initialization of OpenNI, this method must be called, and then Videostream, Device, etc. can be used

```
Status rc = OpenNI::initialize();
```

(2) Open a device, where ANY\_DEVICE is the first device connected in the open system

```
Device device;
```

```
rc = device.open(ANY_DEVICE);
```

(3) After turning on the device, obtain the sensor information, and then create a video stream based on the sensor information

```
if (device.getSensorInfo(SENSOR_DEPTH) != NULL)
```

(4) Create a deep video stream

```
rc = depth.create(device, SENSOR_DEPTH);
```

(5) Call the start method to start collecting depth information

```
rc = depth.start();
```

(6) Use polling method to obtain code stream information

```
VideoFrameRef frame;
```

```
VideoStream* pStream = &depth;
```

```
rc = OpenNI::waitForAnyStream(&pStream, 1, &changedStreamDummy, SAMPLE_READ_WAIT_TIMEOUT);
```

(7) Read one frame of data

```
rc = depth.readFrame(&frame);
```

(8) Obtain the depth value of the center point of the depth image

```
DepthPixel* pDepth = (DepthPixel*)frame.getData();
```

```
float x,y,z;
```

```
CoordinateConverter coorvert;
```

```
int middleIndex = (frame.getHeight()+1)*frame.getWidth()/2;
```

```
coorvert.convertDepthToWorld(depth, frame.getWidth()/4, frame.getHeight()/4,
```

```
pDepth[middleIndex], &x, &y, &z);
```

(9) Turn off related equipment after stopping video capture

```
depth.stop();
```

```
depth.destroy();
```

```
device.close();
```

```
OpenNI::shutdown();
```

### 3.2.2 Linux environment configuration

#### (1) Scope of application

Under the Linux platform, our company provides SDK packages for Intel x86/x64 and ARM32/64. When using, please pay attention to platform consistency. For the ARM platform, depending on the processing capabilities, two SDK packages are provided, the normal version and the without-filter (no software filtering) version. If the normal version freezes, the SDK without the software filter can be used.

#### (2) Linux environment configuration instructions

Before using OpenNI2 SDK for related development on the Linux platform, please execute the following commands to install and configure the related environment. To

Take the x64 platform as an example, in order to run related programs, you first need to install the following programs:

```
$sudo apt-get install cmake
$sudo apt-get install libusb-1.0-0-dev
$sudo apt-get install libudev-dev
$sudo apt-get install libglfw3-dev
$sudo apt-get install build-essential freeglut3 freeglut3-dev
```

Check the udev version, Orbbec's driver needs to depend on libudev.so.1, you can execute the following command to find, if not found

Need to manually link libudev.so.x.x to libudev.so.1

```
$ldconfig -p | grep libudev.so.1
$cd /lib/x86_64-linux-gnu
$sudo ln -s libudev.so.x.x.x libudev.so.1
```

Copy the SDK package to a certain path, decompress it, and compile the routine.

```
$cd Demo/APIDemo/example/build      -----Common code routines
```

```
$cmake ..
```

```
$make
```

```
$cd Demo/ExtAPIDemo/linux-x64/ExtendedAPI      ---Extended Api code example
```

```
$make
```

```
$cd Demo/UVCDemo/Linux/OrbbecStreamSample-Linux      ---uvc code example
```

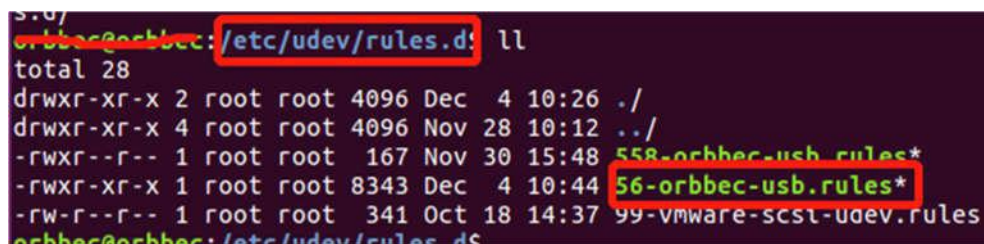
Plug in the camera to run the routine program.

```
$lsusb          ---Confirm whether the device is
                recognized
$./DepthRead    ---Run the corresponding routine
```

### (3) udev description

In Linux system, in order to ensure that the camera device can be recognized and opened normally, the corresponding udev should be added. The specific operations are as follows:

```
$cd Driver/linux
$ ./create_udev_rules
```



```
orbbec@orbbec: /etc/udev/rules.d$ ll
total 28
drwxr-xr-x 2 root root 4096 Dec  4 10:26 ./
drwxr-xr-x 4 root root 4096 Nov 28 10:12 ../
-rwxr--r-- 1 root root  167 Nov 30 15:48 558-orbbec-usb.rules*
-rwxr-xr-x 1 root root 8343 Dec  4 10:44 56-orbbec-usb.rules*
-rw-r--r-- 1 root root  341 Oct 18 14:37 99-vmware-scsi-udev.rules
orbbec@orbbec: /etc/udev/rules.d$
```

### (4) SimpleCode description

Under the Linux system, different sample codes are provided for applications such as standard API and extended API in the SDK. Specific operations and usage

For instructions, please refer to the routine compilation steps in the Linux environment configuration. For the steps to obtain the depth stream, please refer to Windows.

## 3.2.3 Android project environment configuration

### (1) Scope of application

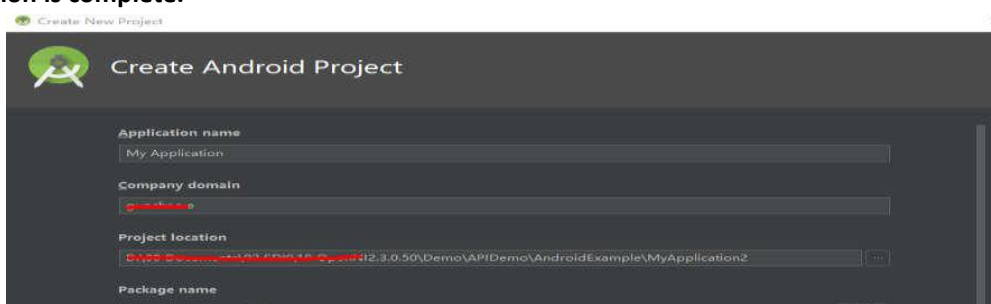
Under the Android platform, the company's SDK is suitable for Android 4.4.2 and above platforms

### (2) Android environment configuration instructions

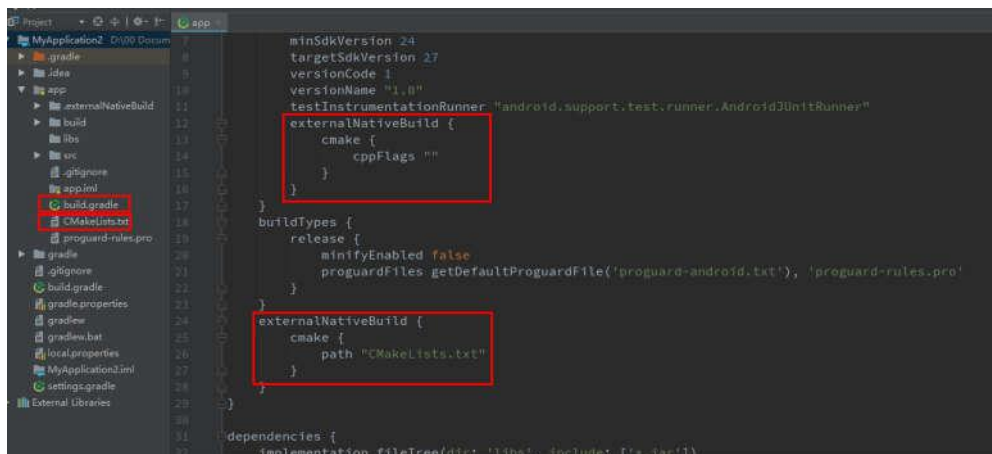
For Android SDK development, please build an Android Studio development environment. You can build a pure Java environment for development, or

Develop by building a Java project that supports C++, configure NDK, and call C++ API through JNI. Before Android2.2, JNI was implemented through NDK+Android.mk+Application.mk, and later versions were implemented using ndk+Cmake. The following describes the configuration process of the Android Studio NDK environment.

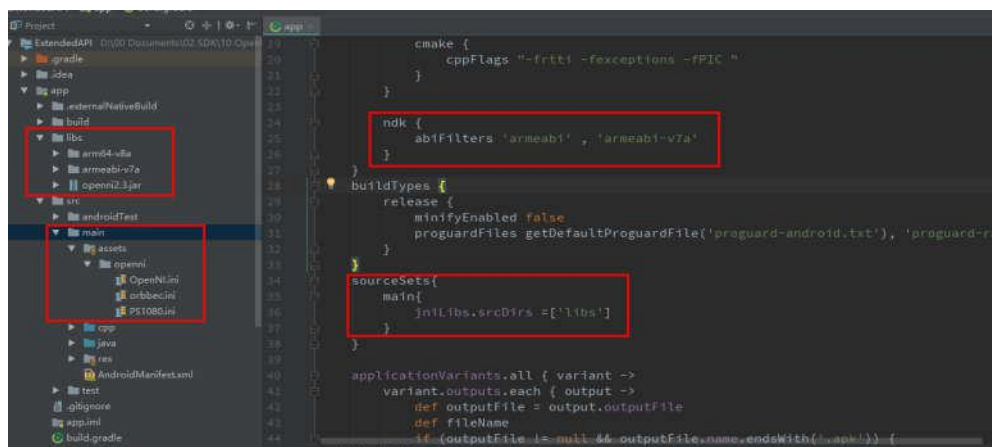
**Create a new Android Studio that supports C++, check Include C++ support, and click Next until the creation is complete.**



As can be seen from the directory structure, there are several differences compared to ordinary Android projects.



## Add library and resource files in SDK, configure gradle file



**Modify the cmake file and configure the created cpp file, then you can call the C++ API in the Java project, detailed project configuration and**

**API registration call can only test ExtAPIDemo project file.**



```

10 # You can define multiple libraries, and CMake builds it for you.
11 # Gradle automatically packages shared libraries with your APK.
12
13 add_library(OpenNI2
14             SHARED
15             IMPORTED
16 )
17
18 set_target_properties(OpenNI2
19                       PROPERTIES IMPORTED_LOCATION
20                                ../../../../libs/armeabi-v7a/libOpenNI2.so)
21
22 add_library( # Sets the name of the library.
23             IRUtils
24
25             # Sets the library as a shared library.
26             SHARED
27
28             # Provides a relative path to your source file(s).
29             # Associated headers in the same location as their source
30             # file are automatically included.
31             src/main/cpp/IRUtils.cpp )
32
33
34
35
36
37
38
39
40 find_library( # Sets the name of the path variable.
41             log-lib
42
43             # Specifies the name of the NDK library that
44             # you want CMake to locate.
45             log )
46
47 # Specifies libraries CMake should link to your target library. You
48 # can link multiple libraries, such as libraries you define in the
49 # build script, prebuilt third-party libraries, or system libraries.
50
51 target_link_libraries( # Specifies the target library.
52                       IRUtils
53                       OpenNI2
54                       # Links the target library to the log library
55                       # included in the NDK.
56                       ${log-lib} )

```

1. Create the OpenNIHelper class

```
private OpenNIHelper mOpenNIHelper;
```

```
mOpenNIHelper = new OpenNIHelper(this);
```

2. Register the device to open the listener

```
mOpenNIHelper.requestDeviceOpen(this);
```

3. Implement DeviceOpenListener interface abstract functions onDeviceOpened, onDeviceOpenFailed

@Override

```

public void onDeviceOpened(UsbDevice device) {
    init(device);
    mStream      =      VideoStream.create(mDevice,      SensorType.IR);
    mVideoModes = mStream.getSensorInfo().getSupportedVideoModes();
    for (VideoMode mode : mVideoModes) {
        int X = mode.getResolutionX();

```

```

        int Y = mode.getResolutionY();
        int fps = mode.getFps();
        Log.d(TAG, " support resolution: " + X + " x " + Y + " fps: " + fps + ", (" +
mode.getPixelFormat() + ")");
        if (X == mWidth && Y == mHeight && mode.getPixelFormat() == PixelFormat.RGB888 &&
fps == 30) {
            mStream.setVideoMode(mode);
            Log.v(TAG, " setmode");
        }
    }
    startThread();
}
@Override
public void onDeviceOpenFailed(String msg)
{ showAlertAndExit("Open Device failed: " +
msg);
}

```

4、init(device)中完成设备初始化，并打开 OpenNI 类设备

```

private void init(UsbDevice device) {
    OpenNI.setLogAndroidOutput(true);
    OpenNI.setLogMinSeverity(0);
    OpenNI.initialize();
    List<DeviceInfo> opennilist = OpenNI.enumerateDevices();
    if (opennilist.size() <= 0) {
        Toast.makeText(this, " openni enumerateDevices 0 devices", Toast.LENGTH_LONG).show();
        return;
    }
    mDevice = null;
}

```

```

//Find device ID
for (int i = 0; i < opennilist.size(); i++) {
    if (opennilist.get(i).getUsbProductId() == device.getProductId())
        { mDevice = Device.open();
          break;
        }
}
if (mDevice == null) {
    Toast.makeText(this, " openni open devices failed: " + device.getDeviceName(),
Toast.LENGTH_LONG).show();
    return;
}
}

```

#### 5、startThread（）开启线程读取视频流

```

void startThread() {
    mInit_Ok = true;
    m_thread = new Thread() {
        @Override
        public void run() {
            List<VideoStream> streams = new ArrayList<VideoStream>();
            streams.add(mStream);
            mStream.start();

            while (!mExit) {
                try {
                    OpenNI.waitForAnyStream(streams, 2000);
                } catch (TimeoutException e)
                { e.printStackTrace();
                  continue;
                }
                synchronized (m_sync) {
                    mGLView.update(mStream);
                }
            }
        }
    }
}

```

```

    }
}
};
m_thread.start();
}

```

### 3.2.4 Instructions for using .ini configuration file

#### (1) OpenNI.ini configuration instructions

OpenNI2.3.0.50 provides a wealth of debugging logs, open the corresponding level of log, you can view the current screen or file

Depth, color flow information. This function can be realized by modifying the OpenNI.ini file.

```

[Log]
; 0 - Verbose; 1 - Info; 2 - Warning; 3 - Error. Default - None
;Verbosity=0 // 去掉注释符 ;, 修改为 Verbosity=0 查看最详细调试信息
;LogToConsole=1 // 去掉注释符 ;,修改为 LogToConsole=1 将调试信息输出到控制台
;LogToFile=1 // 去掉注释符 ;,修改为 LogToFile=1 可以保持 log 到文件
;LogToAndroidLog=1 // 去掉注释符 ;,修改 LogToAndroidLog=1 输出 Log 到 logcat

[Device]
;Override=
;RecordTo=

[Drivers]
; Location of the drivers, relative to OpenNI shared library location. When not provided, "OpenNI2/Drivers"
will be used.
;Repository=OpenNI2/Drivers

; List of drivers to load, separated by commas. When not provided, OpenNI will try to load each shared
library in Repository.
;List=

```

#### (2) Orbbec.ini configuration instructions

OpenNI2.3.0.50 provides a flexible external configuration function. By modifying the orbbec.ini file, the software library analyzes the configuration file

Set the working status of the camera, the following are the key settings of the configuration file.

```

;----- Sensor Default Configuration -----
[Device]
; Mirroring. 0 - Off (default), 1 - On
;Mirror=1

; FrameSync. 0 - Off (default), 1 - On
;FrameSync=1

; Stream Data Timestamps. 0 - milliseconds, 1 - microseconds (default)
;HighResTimestamps=1

; Stream Data Timestamps Source. 0 - Firmware (default), 1 - Host
;HostTimestamps=0

; A filter for the firmware log. Default is determined by firmware.
;FirmwareLogFilter=0

; Automatic firmware log retrieval. 0 - Off (default), or the number of milliseconds between log retrievals operations.
;FirmwareLogInterval=1000

; Print firmware log to console when automatic firmware log retrieval is on. 0 - Off (default), 1 - On
;FirmwareLogPrint=1

; Is APC enabled. 0 - Off, 1 - On (default)
;APCEnabled=1

; USB interface to be used. 0 - FW Default, 1 - ISO endpoints (default on Windows), 2 - BULK endpoints (default on Linux/Mac/Android machines)
UsbInterface=2

```

The item in the symbol [] indicates the current configuration device, which can configure the device Device, depth camera Depth, color camera Image and infrared camera IR items, while retaining the configuration items Depth.Cropping and IR.Cropping for depth map and color map cropping . The configuration item in the above figure [Device] is taken as an example. If you want to modify a specific item, you need to remove the semicolon used in the comment, and then select the corresponding configuration value according to the description. The following describes common configuration items.

[Device]:

Mirror——device mirroring mode selection;

FrameSync —— Depth and color frame synchronization settings. It should be noted that only non-UVC devices support this configuration item.

[Depth]:

outputFormat——Select the depth raw data format of the output

Mirror——Set the depth map mirror mode

Resolution-depth image resolution setting

FPS-deep frame rate selection

Registration-Set RGBD hardware alignment

HoleFilter —— Set the filter mode. For models with inconsistent depth and color resolution, by setting HoleFilter=2,

The black hole on the depth map after RGBD alignment can be removed.

[Image]:

outputFormat——Select the output color raw data format

Mirror——Set the color image mirroring mode

The rest of the configuration items are similar to Depth, but it should be noted that the settings under the Image configuration item are only effective for color cameras that are not set by UVC. UVC cameras, such as Deeyea, Astra D-U, etc., cannot configure color cameras through this item.

[IR]:

outputFormat——Select the output NIR image data format Mirror——Set IR image mirroring

Resolution——Select IR image resolution

It should be noted that since the Depth image is calculated based on the IR image, the resolution and frame rate of the Depth image

When the configuration is modified, the corresponding options in the IR configuration items need to be modified simultaneously.

## 4.0 Introduction to Common APIs

### 4. Introduction to commonly used APIs

In the Document folder of the SDK package, the OrbbecSDK\_v2.3.chm file is provided, which summarizes all the APIs supported in the SDK, and provides interface introduction and usage instructions. You can also select c++.chm or openni2java.chm to view the C++ or Java API description according to the development language. The document directory structure is as follows:



### 4.1 Introduction to Standard API

The standard API is a native API provided by the OpenNI open source software framework, including commonly used classes such as Device, VideoStream, and VideoMode.

There are 4 main classes required to obtain a deep video stream.

1.openni::OpenNI provides a static API entry point. It provides access to the device, device related events, version and error information. Of course, you must first make sure you connect the device.

2.openni::Device provides an interface for sensor device connection system (personal understanding is to access and control sensors through the Device class). Before it is created, the OpenNI class needs to be initialized. Device can access streams (Streams).

3. `openni::VideoStream` extracts a video stream from a device (Device), you need to obtain a video frame reference

(`VideoFrameRefs`).

4. `openni::VideoFrameRef` extracts a video frame from the related source data. This is obtained from a specific stream. In addition to these main classes, there are many classes and structures used to hold some special types of data. The `Recorder` class is used to store OpenNI video streams to files. There is also the `Listener` class to monitor events generated by the `OpenNI` and `Stream` classes.

The video stream can obtain data in two ways: polling and events. The following specifically introduces each commonly used class:

#### 4.1.1 OpenNI Class

##### 1 Introduction

The `OpenNI` class provides a static entry for library functions. Each OpenNI2.0 application needs to use the `OpenNI` class to initialize the SDK and device

drive. It is also possible to generate many device connection and disconnection events, just like providing the function of accessing data streams in a polling manner.

##### 2) Basic access to equipment

The `OpenNI` class provides the `OpenNI::initialize()` method to initialize library functions and scan all available sensor devices in the system. So

Any application that uses `OpenNI` should call this method before using other APIs.

Once the initialization method is completed, it is possible to create device (Device) objects and use these objects to communicate with the real sensor hardware.

Line interaction. The `OpenNI::enumerateDevices()` method returns a list of available sensor devices connected to the system.

When the application is ready to exit, it must call the `OpenNI::shutdown()` method to shut down all drivers and exit correctly.

##### 3) Basic access to Video Streams

The polling access interface of the stream is implemented by the `OpenNI::waitForStream()` method. One of the parameters of this method is the list of streams. When called, it will be locked until the stream in the list has new data available or it times out. Then return a status code and point to the specific data stream.

##### 4) Event-driven access to equipment

The `OpenNI` class provides a framework for accessing devices in an event driven manner. `OpenNI` defines three events: device connection event (`onDeviceConnected`), device disconnection event (`onDeviceDisconnected`), and device state change event (`onDeviceStateChanged`). The device connection event is generated when a new device is connected and available through `OpenNI`, and the device disconnection event is generated when a device is removed from the system. The device status change event is generated when the device settings are changed.

You can use the following methods to add or remove `Listener` classes from the event processing list:

OpenNI::addDeviceConnectedListener()	//Add	a	device	connection	event	listener
OpenNI::addDeviceDisconnectedListener()	//Add	a	device	disconnect	event	listener
OpenNI::addDeviceStateChangedListener()	//Add	device	state	change	event	listener
OpenNI::removeDeviceConnectedListener()	//Remove	device	connection	event	listener	
OpenNI::removeDeviceDisconnectedListener()	//Remove	device	disconnect	event	listener	
OpenNI::removeDeviceStateChangedListener()	//Remove the device state change event listener					

All three events provide a pointer to the `OpenNI::DeviceInfo` object. This object is used to obtain the details and identification of the device submitted by the event. In addition, the device state change event also provides a pointer to the `DeviceState` object, which is used to view the new state information of the device. Event-driven access to the real video stream through the `VideoStream` class

## 5) Error message

There are many methods in the SDK that return a value of type "Status". When an error occurs, Status will contain a record or code displayed to the user. The `OpenNI::getExtendedError()` method returns more readable information about the error.

## 6) Version information

The version information of the API is obtained by the `OpenNI::getVersion()` method. This method returns the version of the API currently used by the application

information

### 4.1.2 Device class

#### 1 Introduction

The `openni::Device` class provides an abstract interface for a physical hardware device, and it can also pass an ONI record from a physical device

The recording file provides an interface for simulating hardware devices.

The `Device` object is used to connect and configure the underlying file or hardware device, and create a stream from the device. The `Device` object provides

The interface for configuration query and modification can be used to enable RGBD alignment or frame synchronization.

Before creating and initializing the video stream, the `Device` object must be constructed and pointed to the physical device.

#### 2) Equipment connection conditions

Before the device class can be connected to the physical hardware device, the device must be physically and correctly connected to the host, and the driver must be installed

complete. If the connection is an ONI file, the ONI record file must be available when the system is running the application, and the application

The sequence has sufficient permissions to access.

Of course, the `openni::OpenNI::initialize()` method needs to be called before contacting the device. This will initialize the driver so that the API knows



Road equipment is connected.

### 3) Basic operation

#### Constructor

The constructor of the Device class has no parameters, nor does it contact the physical hardware device. Just simply create the object.

`Device::open()`

This method is used to connect to physical hardware devices. The `open()` method has one parameter, the URI (Uniform Resource Identifier) of the device, and the method

A status code is returned to indicate success.

The simplest usage is to use the constant `openni::ANY_DEVICE` as the URI of the device. Using this constant will make the system connect to all hardware

Pieces of equipment, but only suitable for occasions where only one device is connected.

If multiple sensors are connected, you must first call `OpenNI::enumerateDevices()` to get the list of available devices. then

Find the device you are looking for, get the URI by calling `DeviceInfo::getUri()`, and use the output of this method as `Device::open()`

Then you can open the corresponding device.

If you open the file, the parameter is the path of the ONI file.

`Device::close()`

The `close()` method is used to close the hardware device. By convention, all open devices must be closed. This will separate the hardware device and the driver so that subsequent applications can open the device again.

`Device::isValid()`

The `isValid()` method is used to determine whether the device is correctly connected to the device object

`Device::setImageRegistrationMode()`

Orbbec products will generate depth stream and color image stream at the same time, which are generated by different physical cameras (RGB camera and IR camera). The actual position of the physical camera is different, resulting in a difference in the depth and field of view of the color picture, which makes the images obtained from different streams of the same device object different.

The geometric relationship and distance between the two cameras are known to the device object, and the two images can be consistent and superimposed on each other through mathematical transformations, for example, each pixel of the depth map is superimposed on the color image on. This process is alignment (each pixel is superimposed on another image).

Orbbec equipment supports hardware calculations and can calibrate data in the module ISP. At the same

time, there is a flag on the hardware to turn it on or off.

The device object provides the `isImageRegistrationSupported()` method to test whether the connected device supports the alignment function. If it is supported, then `getImageRegistrationMode()` can be used to query the status of this function, and `setImageRegistrationMode()` can set it. The `openni::ImageRegistrationMode` enumeration provides the following values for set or get:

`IMAGE_REGISTRATION_OFF` - Hardware registration features are disabled

`IMAGE_REGISTRATION_DEPTH_TO_IMAGE` - The depth image is transformed to have the same apparent vantage point as the RGB image. The depth image is transformed and superimposed on the color image

It should be noted that the FOV of the two sensors does not overlap in part. This causes some depth maps to not be displayed in the results.

`Device::setDepthColorSyncEnabled()`

When both depth and color image streams are available, slight difference in frame rate or frame arrival time may cause frame synchronization problems.

Orbbec provides frame synchronization function for non-UVC devices. In order to obtain two frames respectively within a certain maximum time range, usually

This maximum value is less than the two-frame interval.

To enable or disable this function, use `setDepthColorSyncEnable()`.

However, it should be noted that all UVC devices cannot perform frame synchronization settings through this API.

#### 4.1.3 VideoStream class

##### 1 Introduction

The video stream class created by the device class encapsulates all data streams. The user can start, stop and configure the data stream, and can also configure the parameters of the stream level (as opposed to the device level).

##### 2) Basic functions of video streaming

###### Create and initialize video stream

Calling the default constructor of the video stream will create an empty uninitialized video stream object. Before use, this object must be initialized by calling `VideoStream::create()`. The `create()` method requires an initialized device object. Once created, you can call the `VideoStream::start()` method to generate a data stream. The `VideoStream::stop()` method will stop the data stream.

###### Data reading based on polling

Once the video stream is created, you can read the data through the `VideoStream::readFrame()` method. If there is new data available, this method will return a reference (`VideoFrameRef`) that can access the latest video frame generated by the video stream.

If there is no new frame available, it will lock until a new frame is available.

It should be noted that if you read from the record acyclically, the program will always be stuck here after the last frame is read.

law.

#### Event-based data reading

Data can be read from the video stream in an event driven manner.

First, you need to create an inherited class of `VideoStream::Listener`, which should implement the method `onNewFrame()`. Once you have created this class and instantiated it, you can add listeners via the `VideoStream::addListener()` method. When a new frame arrives, the `onNewFrame()` method of the custom listener class is called. Then you can call `readFrame()` to read.

### 3) Get information about the video stream

#### Sensor Info (`SensorInfo`) and video mode (`VideoMode`)

Sensor information and video mode can always track the information of the video stream. The video mode encapsulates the frame rate of the video stream (frame

rate), resolution (resolution) and pixel format (pixel format). The sensor information contains the type of sensor that generates the video stream and the list of video mode objects for each stream. By traversing this list, all possible patterns of the flow generated by the sensor can be determined.

Use `VideoStream::getSensorInfo` to get the sensor information object of the current stream

#### Field of View

This function determines the field of view of the sensor. Use `getHorizonFieldOfView()` and `getVerticalFieldOfView()` methods to determine the field of view, and the returned value is radians.

#### Min and Max PixelValues

In depth streams, it is often useful to know the maximum and minimum possible values for a pixel. Use `getMinPixelValue()` and

The `getMaxPixelValue()` method can get this information.

### 4) Configure the video stream

#### Video Mode (`Video Mode`)

You can set the frame rate, resolution, and pixel type of a given stream. To set these, use the `setVideoMode()` method. Before that, you first need to obtain the sensor information (`SensorInfo`) of the configured video stream, and then you can select an available video mode.

#### Cropping

If the sensor supports cropping, the video stream provides a way to control it. Use `VideoStream::isCroppingSupported()` method to determine whether it is supported.

If supported, use `setCropping()` to enable cropping and set the specific configuration of cropping. The `ResetCropping()` method is used to turn off cropping again. The `getCropping()` method is used to obtain the current cropping settings.

## Mirroring

Mirroring, as the name suggests, is to make what the video stream shows looks like in a mirror. To enable or disable mirroring, use the `VideoStream::setMirroringEnable()` method. Set true to enable, set false to disable. Use `getMirroringEnable()` to get the current setting.

## General Properties

At the firmware level, most sensor settings are stored as address/value pairs (address/value pairs).

So it can be directly operated by the `setProperty` and `getProperty` methods. These methods are used internally by the SDK to implement cropping, mirroring, and so on. And they are usually not used frequently by applications, because most useful properties are encapsulated in a more friendly way.

### 4.1.4 VideoFrameRef Class

#### 1 Introduction

The video frame reference class encapsulates all the data of a single frame read from the video stream. It provides access to a basic array containing frame data (metadata, frames required for work).

The video frame reference object is obtained from the `VideoStream::readFrame()` method.

The video frame reference data can be obtained from an infrared camera, an RGB camera or a depth camera. The `getSensorType()` method is used to determine the sensor type that generated this frame. It will return the sensor type.

#### 2) Access frame data

The `VideoFrameRef::getDate()` method returns a pointer directly to the frame data. The type is void, so that the data type of each pixel can be indexed correctly.

#### 3) Cropping data

The data frame reference knows the crop settings of the video stream, so it can be used to determine the origin of the crop box, the size of the crop box and whether the frame is enabled for cropping. The implementation method is as follows: `getCropOriginX()`, `getCropOriginY()`, `getCroppingEnable()`. If the cropping function is enabled, the cropping frame size is equal to the frame size.

#### 4) Timestamp (Timestamp)

Each frame of data has a time stamp. This value is based on the number of microseconds since any value of 0. Different from the time difference between two frames, all streams of the same device use the same 0 value, so the difference in timestamp can be used to compare frames of different streams.

In OpenNI 2.0, the 0 value of the time stamp is the arrival time of the first frame of data. However, there is no guarantee that it will be the same every time, so the program code should use timestamp increments. The time stamp value itself should not be used as an absolute time point.

#### 5) Frame Index (FrameIndex)

In addition to the time stamp, the frame also provides consecutive frame index numbers. This is useful in determining how many frames are between two known frames. If the frame synchronization is enabled, the frame numbers of the corresponding frames should be consistent.

If there is no synchronization, the frame numbers will not necessarily match. In this case, it is more effective to use the time stamp to determine the position of the relevant frame.

#### 5) Video Mode

`VideoFrameRef::getVideoMode()` is used to determine the video mode of the sensor that generated the current frame. The information includes the pixel format,

Resolution, frame rate.

#### 6) Array Stride

The span of the array containing the frames can be obtained with `getStrideBytes()`. It will return the size of each row of the array, in bytes.

Mainly used to index two-dimensional image data.

### 4.1.5 Other support categories

#### 1 Introduction

In addition to the main classes mentioned above, OpenNI also has a series of supporting classes. These classes mainly serve to encapsulate data and are mentioned in the chapters of other main classes.

#### 2) Sensor configuration class

Device Information (`DeviceInfo`)

This class records all the configuration of the device, including device name, URI, USB VID/PID descriptor and vendor.

#### 3) Sensor Info (`SensorInfo`)

This class stores all the configurations of the sensor, and the sensor here is only one of the three sensors. A device has multiple sensors.

#### 4) Video mode (`VideoMode`)

This class stores the resolution, frame rate and pixel format. Used for video streaming settings and viewing settings, referenced by the video frame to view these settings, and sensor information provides a list of video modes.

#### 5) Camera setting (`CameraSetting`)

The settings of the RGB camera of non-UVC devices are stored, and auto white balance and auto exposure can be enabled or disabled.

### 4.2 Introduction to Extension API

Orbbec is extended on the basis of the general OpenNI2 function. It is an API interface that can realize functions such as reading device serial number, obtaining device type, using flash to save data, reading camera calibration parameters, IR camera parameter adjustment, and LDP control.

The extension API is defined in the `Device` class. If users need to use these extension functions, they need to

make sure before calling the extension API

OpenNI2 initialize is successful, and the device is successfully opened. The specific API instructions are as follows:

#### 4.2.1 Get device model

```
char devType[32];
int dataSize = sizeof(devType);
memset(devType, 0, dataSize);
g_Device.getProperty(openni::OBEXTENSION_ID_DEVICETYPE, (uint8_t*)&devType, &dataSize);
```

#### 4.2.2 Get device serial number

```
char serNumber[12];
int dataSize = sizeof(serNumber);
memset(serNumber, 0, dataSize);
g_Device.getProperty(openni::OBEXTENSION_ID_SERIALNUMBER, (uint8_t*)&serNumber,
&dataSize);
```

#### 4.2.3 Set LDP switch

```
int dataSize = 4;
int ldp_en = enable;
g_Device.setProperty(openni::OBEXTENSION_ID_LDP_EN, (uint8_t*)&ldp_en, dataSize);
```

#### 4.2.4 Set LDM switch

```
int dataSize = 4;
int laser_en = enable;
g_Device.setProperty(openni::OBEXTENSION_ID_LASER_EN, (uint8_t*)&laser_en, dataSize);
```

#### 4.2.5 Set and get IR exposure value

```
int exposure = 0;
int dataSize = 4;
g_device.getProperty(openni::OBEXTENSION_ID_IR_EXP, (uint8_t*)&exposure, &dataSize);
printf("ir exposure value : 0x%x\n", exposure);
exposure += 256;
g_device.setProperty(openni::OBEXTENSION_ID_IR_EXP, (uint8_t*)&exposure, dataSize);
```

#### 4.2.6 Set and get IR gain

```
int gain = 0;
```

```
int dataSize = 4;
g_device.getProperty(openni::OBEXTENSION_ID_IR_GAIN, (uint8_t *)&gain, &dataSize);
printf("ir gain value : 0x%x\n", gain);
gain++;
g_device.setProperty(openni::OBEXTENSION_ID_IR_GAIN, (uint8_t *)&gain, dataSize);
```

#### 4.2.7 Setting, saving and obtaining camera calibration parameters

```
typedef struct OBCameraParams
{
float l_intr_p[4];           //[fx,fy,cx,cy]
float r_intr_p[4];          //[fx,fy,cx,cy]
```

---

```

float r2l_r[9];           //[r00,r01,r02;r10,r11,r12;r20,r21,r22]
float r2l_t[3];           //[t1,t2,t3]
float k[5];               //[k1,k2,p1,p2,k3]
int is_mirror;
} OBCameraParams;
OBCameraParams m_CamParams = { 0 };
int dataSize = sizeof(OBCameraParams);
m_CamParams.l_intr_p[0] = 577.318970;
m_CamParams.l_intr_p[1] = 577.318970;
m_CamParams.l_intr_p[2] = 308.729004;
m_CamParams.l_intr_p[3] = 269.143005;
m_CamParams.r_intr_p[0] = 517.447998;
m_CamParams.r_intr_p[1] = 517.447998;
m_CamParams.r_intr_p[2] = 305.432007;
m_CamParams.r_intr_p[3] = 250.410995;
m_CamParams.r2l_r[0] = 0.999972;
m_CamParams.r2l_r[1] = -0.005735;
m_CamParams.r2l_r[2] = 0.004735;
m_CamParams.r2l_r[3] = 0.005736;
m_CamParams.r2l_r[4] = 0.999983;
m_CamParams.r2l_r[5] = -0.000298;
m_CamParams.r2l_r[6] = -0.004733;
m_CamParams.r2l_r[7] = 0.000325;
m_CamParams.r2l_r[8] = 0.999989;
m_CamParams.r2l_t[0] = -25.147900;
m_CamParams.r2l_t[1] = 0.015202;
m_CamParams.r2l_t[2] = -0.648167;
m_CamParams.k[0] = -0.077348;
m_CamParams.k[1] = 0.208761;
m_CamParams.k[2] = -0.196780;
m_CamParams.k[3] = 0.000617;
m_CamParams.k[4] = 0.001059;
m_CamParams.is_mirror = 0;

```



---

```
g_Device.setProperty(openni::OBEXTENSION_ID_CAM_PARAMS, (uint8_t
```

```
*)&m_CamParams,
```

```
dataSize);
```

```
g_Device.getProperty(openni::OBEXTENSION_ID_CAM_PARAMS, (uint8_t
```

```
*)&m_CamParams,
```

```
&dataSize);
```

---

