Vzense TOF Camera

SDK User Guide

Linux

2020.06

Vzense Technology,Inc.

About This Document

This guide is mainly to introduce how to use Vzense TOF Camera and the Vzense SDK.

Document Structure

Chapter	Title	Contents
1	Overview	Introduce general information of Vzense SDK
2	Products	Introduce general information of Vzense products
3	Installation	Introduce how to install Vzense TOF Depth Camera and SDK
4	SDK Instruction	Introduce how to use Vzense SDK
5	API Introduction	Introduce APIs of Vzense SDK
6	Update Firmware	Introduce how to update firmware

Release Record

Date	Version	Instruction
2019/12/24	V3.0.0.7	Optimize SDK framework and code
2020/05/13	V3.0.0.8	Add DCAM800LITE
2020/06/08	V3.0.0.9	Add DCAM305 and DCAM500
2020/12/04	V3.2.2	Add DCAM550

Contents

1 Overview

Vzense TOF Camera is a series of 3D camera modules developed by Vzense which uses TOF (Time of Flight) technology. It has the advantages of high precision, strong environmental adaptability, small size and so on.

The Vzense SDK is a development kit based on Vzense TOF Camera, which is currently applicable to Windows/Linux/Android. It provides a series of friendly APIs and simple application examples for developers.

Developers can get high precision depth image data, gray image data and point cloud data through the SDK. It is convenient for users to develop gesture recognition, projection touch, face recognition, fatigue detection, 3D modeling, navigation, obstacle avoidance and so on.

2 Products

2.1 DCAM710

Figure 2.1 Vzense TOF RGBD Camera：DCAM710

DCAM710 is a 3D camera module developed by Vzense which uses TOF (Time of Flight) technology. The depth information it outputs can be applied to the next generation of UI which is based on gesture recognition, TV and Game motion-sensitivity interaction, face recognition, robot obstacle avoidance, advanced automotive vision system, industrial control and other frontier creative technologies.

2.2 DCAM800

Figure 2.2 Vzense TOF Camera：DCAM800

DCAM800 is a 3D camera based on TOF technology specially developed by Vzense for industrial application scenarios.It supports 100/1000M Ethernet. It has the features of easy installation, high reliability, IP56 level protection, etc. It can meet different industrial scenarios, and has the ability to detect at a longer distance.

2.3 DCAM800LITE

Figure 2.3 Vzense TOF Camera：DCAM800LITE

DCAM800LITE is a 3D camera based on TOF technology specially developed by Vzense for industrial application scenarios. It supports USB2.0, 100M Ethernet, inherits the ease of installation of DCAM800, and has higher cost performance.

2.4 DCAM305

Figure 2.4 Vzense TOF RGBD Camera：DCAM305

DCAM305 is a 3D camera based on TOF technology specially developed by Vzense for face recognition scenarios. It has the features of easy installation and high reliability.

2.5 DCAM500

Figure 2.5 Vzense TOF Camera：DCAM500

DCAM500 is a 3D camera based on TOF technology specially developed by Vzense for industrial application scenarios. It supports USB2.0, inherits the ease of installation of DCAM800LITE, and has higher cost performance.

2.6 DCAM550

Figure 2.6 Vzense TOF Camera：DCAM550

DCAM550 is a 3D camera based on TOF technology specially developed by Vzense for industrial application scenarios. It can be divided into DCAM550U and DCAM550P. DCAM550U supports USB2.0, DCAM550P supports 100M Ethernet. It has higher cost performance and better meet different application scenarios.

3 Installation

3.1 Recommended Development Environment

Item	Recommended Configuration
OS	Ubuntu 18.04 64 位 Ubuntu 16.04 64 位 ARM Linux(AArch64)
RAM	4G or above

3.2 Installation Instruction

3.2.1 Hardware Installation

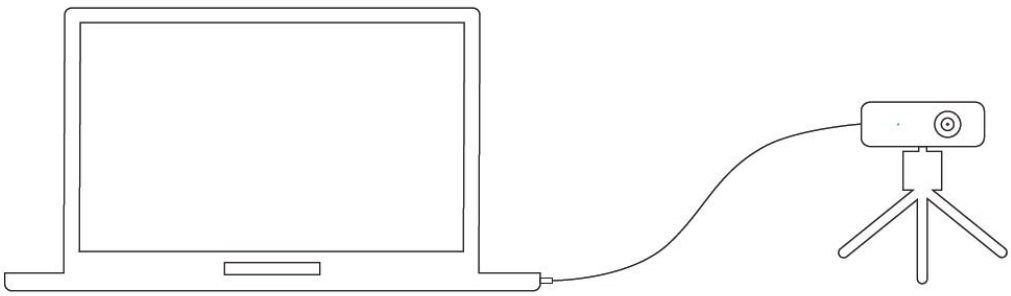


Figure 3.1 Hardware Installation

3.2.1.1USB

Connect the camera module to PC USB interface through USB cable.

In Ubuntu, run the command **sudo apt-get install v4l-utils** in terminal to install v4l related tools. When successfully installed, run **v4l2-ctl --list-devices**, you can see following

output information if device connected successfully.

Figure 3.2 Vzense RGBD Camera

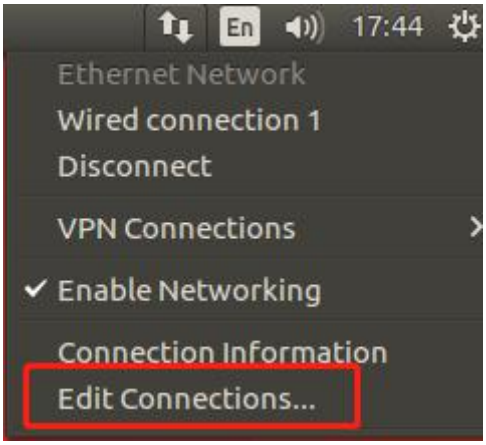
3.2.1.2Network

Network cable connection can be divided into fixed address direct connection and DHCP connection.

1. Fixed address

The fixed address connection can be directly connected to the camera and the computer, or it can be configured to be used in the switch of the same network segment.

Direct connection: one end is connected to the camera, and the other end is connected to the network cable interface of the PC host. The default IP of the camera is 192.168.1.101. On the PC side, set the subnet mask of "local connection" to 255.255.255.0, and the IP address to the same network segment (such as 192.168.1.100).



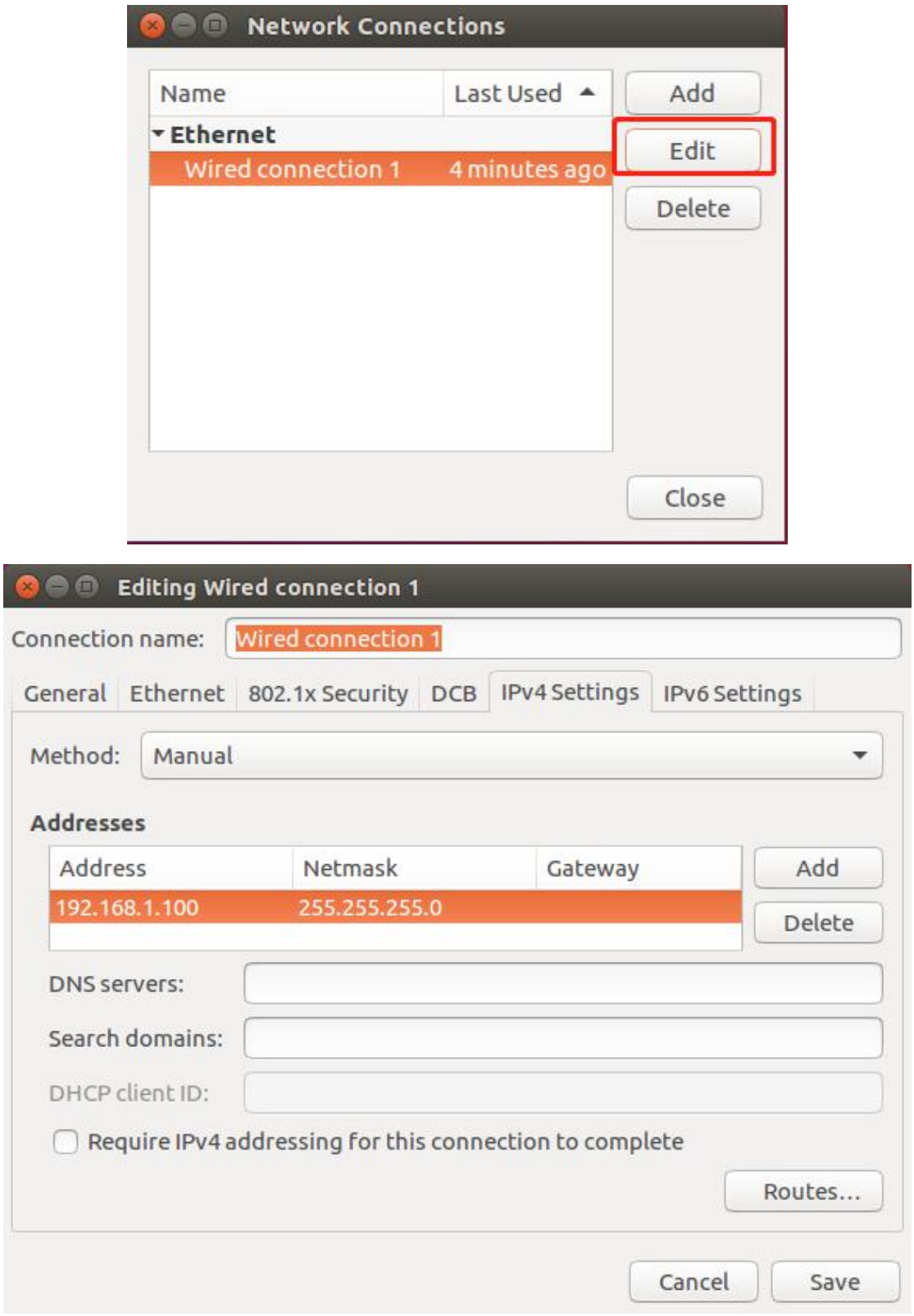


Figure 3.3 Direct connection

2. DHCP

For the DHCP connection mode, the camera needs to be connected to the router with DHCP enabled, and the PC in the same LAN is used for connection. It is recommended to set the "local connection" of the PC to obtain the IP address automatically.

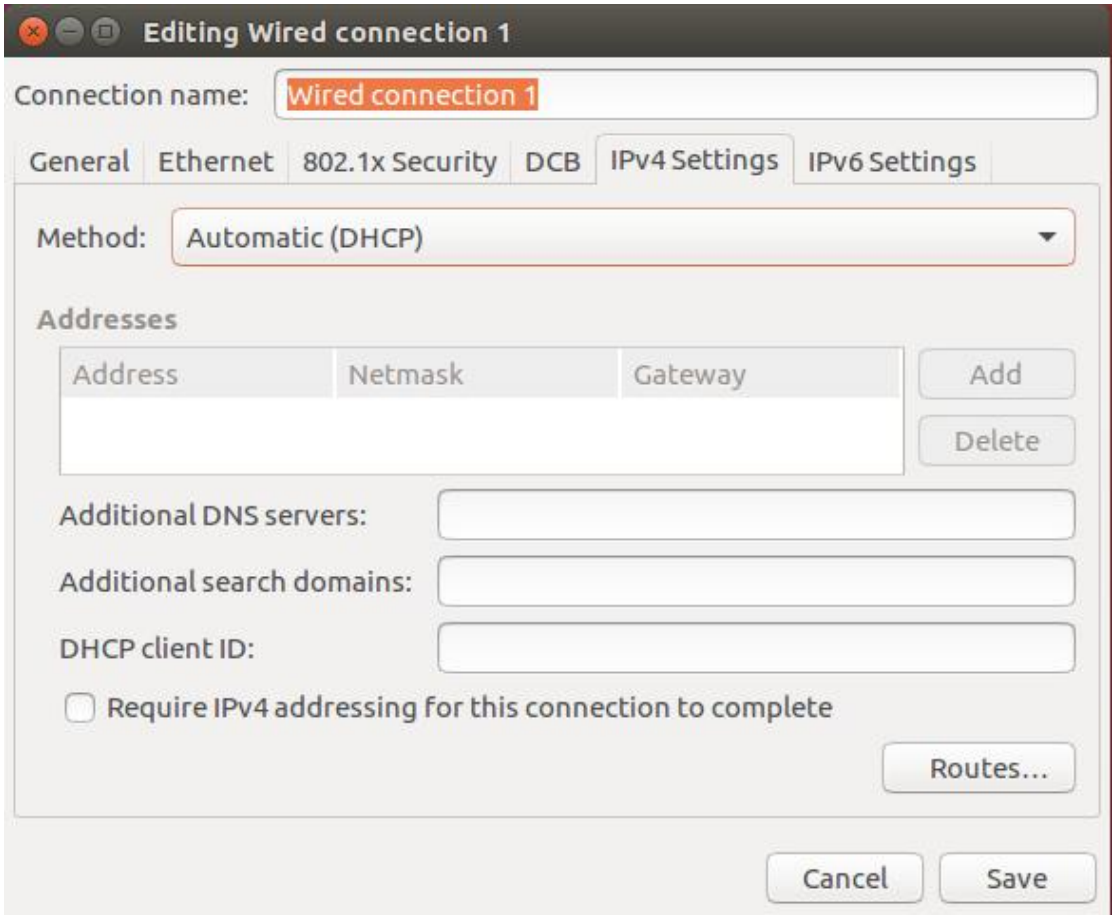


Figure 3.4 DHCP

Note: the network card, router and switch used at the PC end shall meet the requirements of Gigabit.

3.2.2 Software Environment Setup

In Linux, it need to setup the environment as following steps firstly.

NVIDIA graphic card

For the computer with NVIDIA graphic card , please update the graphic driver, as is showing in followed figure.

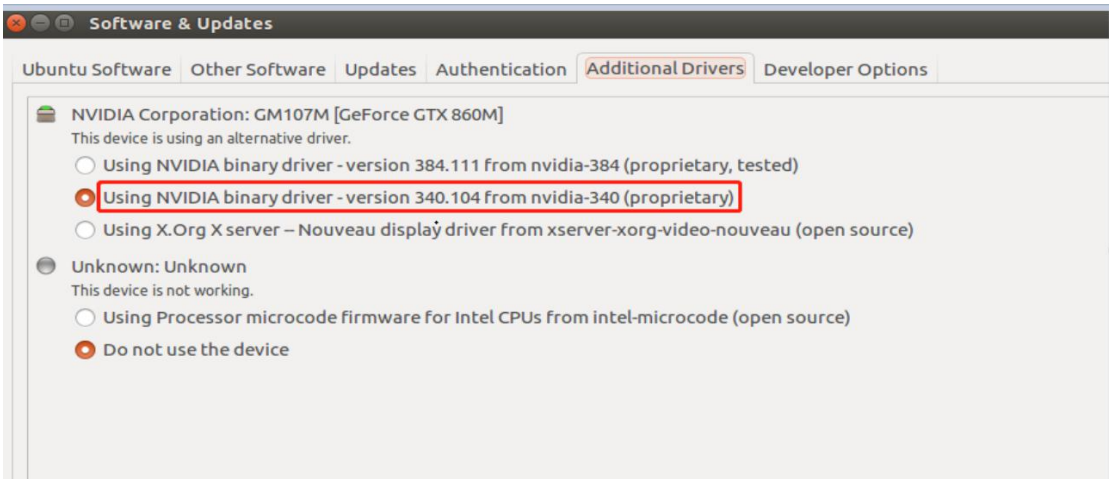


Figure 3.5 Graphics driver settings

Install VDPAU: libvdpau-dev ,vdpauinfo.

sudo apt-get install libvdpau-dev

sudo apt-get install vdpauinfo

Run vdpauinfo.

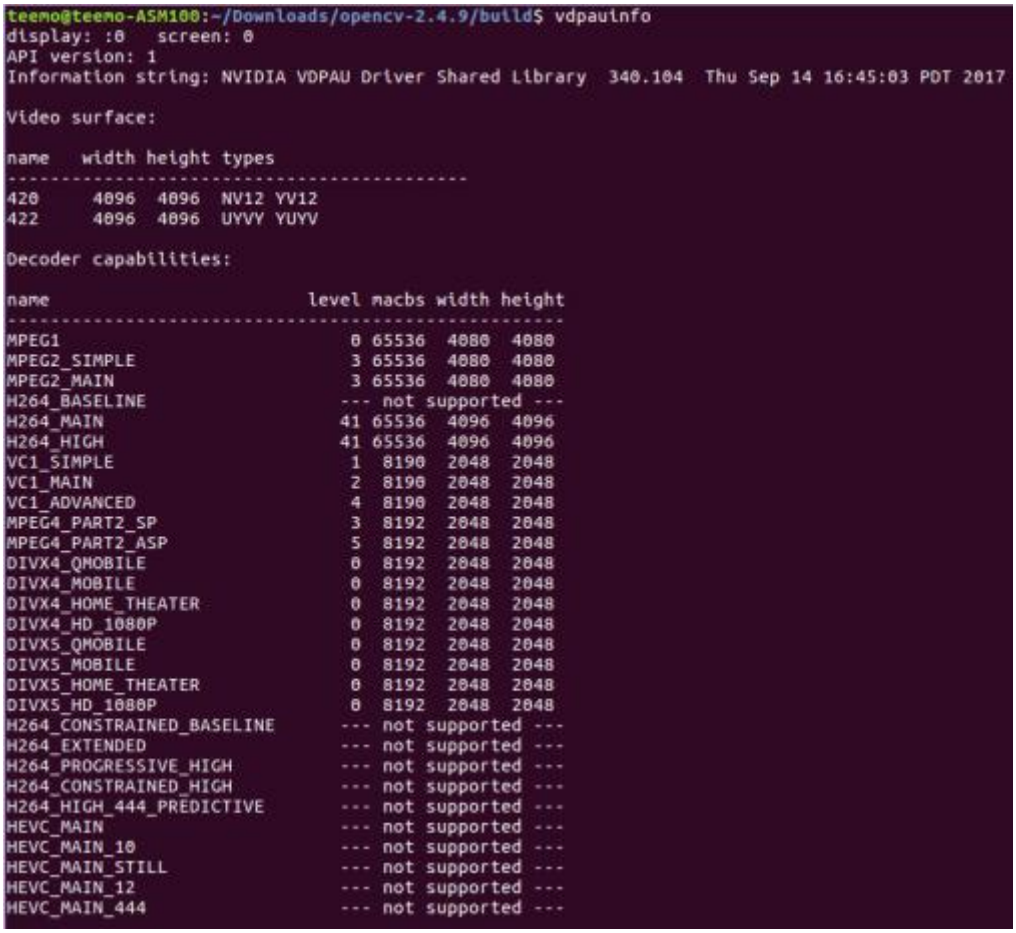


Figure 3.6 Install VDPAU

Intel graphic card

For the computer with Intel graphic card, execute the following instructions.

sudo add-apt-repository ppa:nilarimogard/webupd8

sudo apt-get update

sudo apt-get install libvdpau-va-gl1

sudo apt-get install i965-va-driver

sudo apt-get install vdpauinfo

run：vdpauinfo, sucess as Figure 3.6

if error：

display: :0 screen: 0

Failed to open VDPAU backend libvdpau_i965.so: cannot open shared object file: No such file or directory

Error creating VDPAU device: 1

Then make following steps：

cd /usr/lib/x86_64-linux-gnu/vdpau/

sudo ln -s libvdpau_va_gl.so libvdpau_i965.so

sudo ln -s libvdpau_va_gl.so.1 libvdpau_i965.so.1

4 SDK Instruction

4.1 SDK Structure

Vzense SDK contains several directories, including Bin, Document, Include, Lib, Samples, Tools.

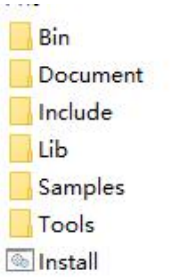
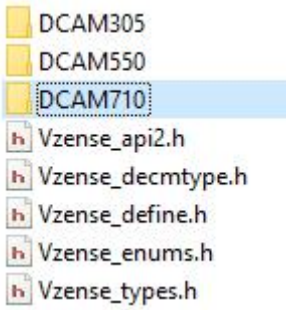


Figure 4.1 Linux SDK directory

The Bin directory has Vzense Windows SDK DLLs, such as vzense_api.dll, including both x64 and x86 versions. Before running the application developed by the Vzense SDK, it needs to copy the vzense_api.dll and whole Config directory to the directory in which executable application locates.

The Document directory contains English and Chinese version of SDK user guide. Include mainly includes the general header files of SDK(VZense_decmtype.h, VZense_api2.h, VZense_define.h, VZense_enums.h, VZense_type.h) and folders containing specific header files required by different models of products, such as DCAM710.



The Lib directory contains the lib files of the Windows SDK, such as libvzense_api.so.

The Samples directory mainly includes some code samples which are developed based

on the Vzense SDK.

The Tools directory includes the tool FrameViewer which can show depth and IR images of the Vzense camera.

Before run FrameViewer,run install.sh firstly,in SDK path with sudo ./install.sh

4.2 Tool Usage

Connect Vzense Camera to PC. Match with the camera and run FrameViewer_DCAMXXX.exe in the Tools directory. This application will show two windows to display IR image and depth image separately. As illustrated below, the RGB image displays normally without stutters, it indicates that the camera works normally.

Figure 4.2 Running the FrameViewer tool

4.3 Development Process

4.3.1 Project Configuration

To develop a new project using the Vzense Linux SDK under Linux, you need to add the include directory in the SDK to the include path in the makefile, and you need to add the Lib / x64 directory to the link search path, and link libvzense_api.so, such as - I... / include, - L... / Lib / x64, - lvzense_api. Please refer to the configuration of makefile in samples.

4.3.2 API Invoke Process

The API of Vzense SDK invoking process is as below:

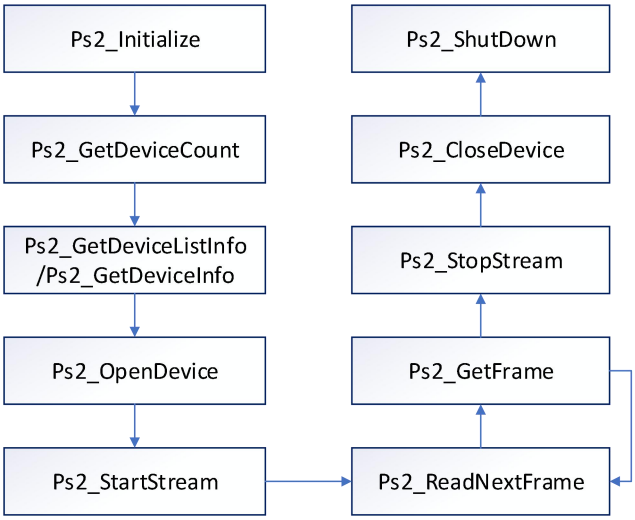


Figure 4.4 SDK API Invoke Process

1. Ps2_Initialize&Ps2_Shutdown

Call the Ps2_Initializeinterface and initialize the SDK. Call the PsShutdown interface finally to log out the SDK and release all the resources created by the SDK.

2. Ps2_GetDeviceCount &Ps2_GetDeviceListInfo/Ps2_GetDeviceInfo

Call the Ps2_GetDeviceCount interface to get the number of devices currently connected. Call the Ps2_GetDeviceListInfo/Ps_GetDeviceInfo interface to get the info of devices currently connected.

3. Ps2_OpenDevice&Ps2_CloseDevice

Call the Ps2_OpenDevice interface to open the specified depth camera device. Call the Ps2_CloseDevice interface to close the specified device.

4. Ps2_StartStream&Ps2_StopStream

Call the Ps2_StartStream interface to open the stream of the camera device. Call the Ps2_StopStream interface to close the stream of the camera device.

5. Ps2_ReadNextFrame&Ps2_GetFrame

In the main loop of image processing, each time Ps2_ReadNextFrame is called first to collect a frame image, and then call Ps2_GetFrame to obtain a frame image data of the specified image type, which is used for corresponding image processing.

6. Set&Get

The SDK provides a rich **Set** and **Get** type interface for setting and acquiring camera properties, parameters and data, as detailed in Section 4.3. If you need change the camera parameters before call the Ps2_ReadNextFrame, please invoking as below:

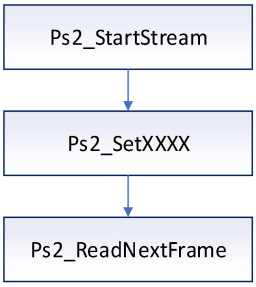


Figure 4.5 SDK API Invoke Process

4.4 SDK Sample

When using include, you need to modify the macro definition status in VZense_decmtype.h according to different models of devices. For example, if you use DCAM550, only the DCAM_550 macro definition is reserved, and other definitions are commented out.The DCAM800, DCAM800LITE, DCAM500 and DCAM550 use the same api.

```
#ifndef VZENSE_DECMTYPE_H
#define VZENSE_DECMTYPE_H

//#define DCAM_550
//#define DCAM_305
#define DCAM_710

#endif /* VZENSE_DECMTYPE_H */
```

When the Frameviewer project in the Samples directory is compiled, the Frameviewer.cpp has defined different modules according to the macro definition in VZense_decmtype.h. In the actual development process, please refer to the header file under the specific model folder in Include.

When run install.sh in 4.1, it will copy the opencv3.4.1 to system, if use another opencv version,you should modify the Makefile in the Sample.

5 SDK API Introduction

5.1 Enum Type

5.1.1 PsDepthRange

Description:

Depth Range mode

Enumerator:

PsNearRange: Near Range mode, Range0

PsMidRange: Middle Range mode, Range1

PsFarRange: Far Range mode, Range2

PsXNearRange: XNear range mode, Range3

PsXMidRange: XMid range mode, Range4

PsXFarRange: XFar range mode, Range5

PsXXNearRange: XXNear range mode, Range6

PsXXMidRange: XXMiddle range mode, Range7

PsXXFarRange: XXFar range mode, Range8

Note: Partial cameras may only support part of these nine modes.

5.1.2 PsDataMode

Description:

Data mode setting, determine which frame output from device and frame fps.

PS: the definition of enumeration values corresponding to different model products may be different. Please refer to the definition under the specific model folder in include.

Enumerator:

PsDepthAndRGB_30: Output both Depth and RGB frames in 30fps. Resolution of depth frame is 640*480. Resolution of RGB can be set by PsSetFrameMode, which support 1920*1080/1280*720/640*480/640*360;

PsDepth_30: Output only Depth frames in 30fps. Resolution of depth frame is 640*480.

PsIRAndRGB_30: Output both IR and RGB frames in 30fps. Resolution of ir frame is 640*480. Resolution of RGB can be set by PsSetFrameMode, which support 1920*1080/1280*720/640*480/640*360;

PsIR_30: Output only IR frames in 30fps. Resolution of IR frame is 640*480.

PsDepthAndIR_30: Output both Depth and IR frames in 30fps. Resolution of both Depth and IR frames is 640*480

PsDepthAndIR_15_RGB_30: Output Depth/IR frames alternatively in 15fps , resolution of both Depth and IR frames is 640*480. Resolution of RGB can be set by PsSetFrameMode, which support 1920*1080/1280*720/640*480/640*360.

PsDepthAndIR_15: Output only Depth/IR frames alternatively in 15fp, Resolution of both Depth and IR frames is 640*480.

PsWDR_Depth: WDR(Wide Dynamic Range) Depth mode, support multi range depth frame output alternatively, like Near/Far/Near/Far/Near..., and can be make fusion to one WDR frame.

5.1.3 PsPropertyType

Description:

Specific property type.

PS: the number of enumeration values corresponding to different models of products may be different. Please refer to the definition under the specific model folder in include.

Enumerator:

PsPropertySN_Str: Indicates the serial number of the device SN, the size does not exceed 64 bytes.

PsPropertyFWVer_Str: Indicates the device firmware version number, which does not exceed 64 bytes.

PsPropertyHWVer_Str: Indicates the device hardware version number, which does not exceed 64 bytes.

PsPropertyDataMode_UInt8: Set data mode, refer to PsDataMode, same with the api PsSetDataMode.

PsPropertyDataModeList: Get the supportive datamode list.

listPsPropertyDepthRangeList: Get the supportive depthrange list.

PsPropertyDeviceUpgradeFlag: Sets Gets the network device UpgradeFlag.

PsPropertyDeviceSN: Set/Get the network device SN.

PsPropertyDeviceMACAddr: Set/Get the network device MAC Addr.

PsPropertyDeviceSoftVer: Set/Get the network device SoftVer.

PsPropertyDeviceIPAddr: Set/Get the network device IPAddress.

PsPropertyDeviceSubnetMask: Set/Get the network device SubnetMask.

5.1.4 PsFrameType

Description:

Specific image frame type.

PS: the number of enumeration values corresponding to different models of products may be different. Please refer to the definition under the specific model folder in include.

Enumerator:

PsDepthFrame: depth image frame

PsIRFrame: IR gray image frame.

PsGrayFrame: gray image frame.

PsRGBFrame: RGB image frame.

PsMappedRGBFrame: RGB image which is mapped to Depth space.

PsMappedDepthFrame: Depth image which is mapped to RGB space.

PsMappedIRFrame: IR image whichi is mapped to RGB space.

PsRawDepthFrame: Original depth image (no smoothing filter/undistortion) .

PsConfidenceFrame: Confidence frame with 16bits per pixel.

PsWDRDepthFrame: WDR depth frame with 16bits per pixel in mm, only take effect when data mode set to PsWDR_Depth.

5.1.5 PsSensorType

Description:

The camera sensor type.

PS: the number of enumeration values corresponding to different models of products may be different. Please refer to the definition under the specific model folder in include.

Enumerator:

PsDepthSensor: Depth camera

PsRgbSensor: RGB camera

5.1.6 PsPixelFormat

Description:

Specific image pixel type

PS: the number of enumeration values corresponding to different models of products may be different. Please refer to the definition under the specific model folder in include.

Enumerator:

PsPixelFormatDepthMM16: Data of each pixel is 16bit depth value

(in millimeter).

PsPixelFormatGray16: Data of each pixel is 16bit gray value

PsPixelFormatGray8: Data of each pixel is 8bit gray value

PsPixelFormatRGB888: Data of each pixel is 24bit RGB value

PsPixelFormatBGR888: Data of each pixel is 16bit BGR value

5.1.7 PsReturnStatus

Description:

Return status of API

Enumerator:

PsRetOK: Succeed

PsRetNoDeviceConnected: No depth camera connected or the camera connected abnormally. Please check HW connection or try to plug out and plug camera in again.

PsRetInvalidDeviceIndex: The input device index is invalid

PsRetDevicePointerIsNull: The device structure pointer is null

PsRetInvalidFrameType: The input frame type is invalid

PsRetFramePointerIsNull: The output frame is empty

PsRetNoPropertyValueGet: Cannot get the property value

PsRetNoPropertyValueSet: Cannot set the property value

PsRetPropertyPointerIsNull: The input property value buffer pointer is null

PsRetPropertySizeNotEnough: The input property value buffer size is not enough to store the returned property value

PsRetInvalidDepthRange: The input depth range mode is invalid

PsRetReadNextFrameError: Error when capturing the next image frame

PsRetCameraNotOpened: Camera is not opened

PsRetInvalidCameraType: The type of camera is invalid

PsRetInvalidParams: Parameter is invalid

PsRetOthers: Other error

5.1.8 PsWDRTotalRange

Description:

Count of ranges alternatively output in WDR mode.

Enumerator:

PsWDRTotalRange_Two: like Near/Far/Near/Far…

PsWDRTotalRange_Three: like Near/Mid/Far/Near/Mid/Far…

5.1.9 PsWDRStyle

Description:

WDR style setting used for API PsSetWDRStyle, which determine WDR image output is fusion from multi range (e.g. Near/Far) or output alternatively (e.g. Near/Far/Near/Far...)

Enumerator:

PsWDR_FUSION: WDR image output is fusion from multi range

PsWDR_ALTERNATION: WDR image output alternatively(e.g. Near/Far/Near/Far...)

5.1.10 PsResolution

Description:

Rgb frame resloution

PS: some model may not support RGB, such as DCAM550. Please refer to the definition under the specific model folder in include.

Enumerator:

PsRGB_Resolution_1920_1080: 1080P

PsRGB_Resolution_1280_720: 720P

PsRGB_Resolution_640_480: 480P

PsRGB_Resolution_640_360: 360P

5.2Struct Type

5.2.1 PsRGB888Pixel

Description:

Color image pixel type in 24-bit RGB format

PS: some model may not support RGB, such as DCAM550. Please refer to the definition under the specific model folder in include.

Members:

r: red

g: green

b: blue

5.2.2 PsBGR888Pixel

Description:

Color image pixel type in 24-bit BGR format

PS: some model may not support RGB, such as DCAM550. Please refer to the definition under the specific model folder in include.

Members:

b: blue

g: green

b: blue

5.2.3 PsVector3f

Description:

Vector for float data

Members:

float x, y, z

5.2.4 PsDepthVector3

Description:

Depth Image Coordination Vector

Members:

depthX: x in pixel

depthY: y in pixel

depthZ: z in mm

5.2.5 PsCameraParameters

Description:

Parameters of camera

Members:

fx: Focal length x (pixel)

fy: Focal length y (pixel)

cx: Principal point x (pixel)

cy: Principal point y (pixel)

k1: Radial distortion coefficient, 1st-order

k2: Radial distortion coefficient, 2nd-order

p1: Tangential distortion coefficient

p2: Tangential distortion coefficient

k3: Radial distortion coefficient, 3rd-order

k4: Radial distortion coefficient, 4st-order

k5: Radial distortion coefficient, 5nd-order

k6: Radial distortion coefficient, 6rd-order

5.2.6 PsCameraExtrinsicParameters

Description:

Camera extrinsic parameters

Members:

rotation[9]: 3x3 rotation matrix

translation[3]: 3-D translation vector

5.2.7 PsFrame

Description:

The image information

Members:

frameIndex: Frame index

frameType: type of frame

pixelFormat: Pixel type

imuFrameNo: Used to synchronize with IMU

pFrameData: frame data

dataLen: Length of data

exposureTime: exposure time(ms)

depthRange: Depth range of current frame, only for depth frame

width: width of the image

height: height of the image

5.2.8 PsWDROutputMode

Description:

Parameters of camera

Members:

totalRange: Currently only 2 or 3 ranges output setting supported

range1: First range

range1Count: Count of successive range1 frame

range2: Second range

range2Count: Count of successive range2 frame

range3: Third range, only take effect when totalRange is set to 3

range3Count: Count of successive range3 frame

5.2.9 PsMeasuringRange

Description:

Measuring range of camera

Members:

depthMode :0(near/mid/far) 1(xnear/xmid/xfar) 2 (xxnear/xxmid/xxfar)

depthMaxNear:the max depth value,in near range ,in “depthMode”

depthMaxMid:the max depth value,in mid range ,in “depthMode”

depthMaxFar:the max depth value,in far range ,in “depthMode”

effectDepthMaxNear:the effect max depth value,in near range ,in “depthMode”

effectDepthMaxMid:the effect max depth value,in mid range ,in “depthMode”

effectDepthMaxFar:the effect max depth value,in far range ,in “depthMode”

effectDepthMinNear:the effect min depth value,in near range ,in “depthMode”

effectDepthMinMid:the effect min depth value,in mid range ,in “depthMode”

effectDepthMinFar:the effect min depth value,in far range ,in “depthMode”

5.2.10PsDeviceInfo

Description:

The information of device

Members:

SessionCount:the count of session

devicetype:the type of device

uri:the identification of device

fw:the firmware version

status:the connect status

5.2.11PsDataModeList

Description:

The supportive datamode list of camera

Members:

index:fixed value 0x00

count: the count of datamode that supported

datamodelist:the list of datamode that supported

5.2.12PsDepthRangeList

Description:

The supportive depthrange list of camera

Members:

index:fixed value 0x01

count: the count of depthrange that supported

depthrangelist:the list of depthrange that supported

5.2.13PsFrameReady

Description:

The flg of the ready frame.1:available,0: unavailable

PS: the image types available for different models of products are different. Please refer to the definition under the specific model folder in the include.

Members:

depth:flg of the ready Depth frame

ir:flg of the ready IR frame

rgb:flg of the ready RGB frame

mappedRGB:flg of the ready mappedRGB frame

mappedDepth:flg of the ready mappedDepth frame

mappedIR:flg of the ready mappedIR frame

confidence:flg of the ready confidence frame

wdrDepth:flg of the ready wdrdepth frame

reserved:not used

5.3 API

5.3.1 Ps2_Initialize

Prototype:

PsReturnStatus Ps2_Initialize()

Description:

Initialize Vzense SDK. It should be called first before calling any other SDK API

Parameters:

None

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.2 Ps2_Shutdown

Prototype:

PsReturnStatus Ps2_Shutdown()

Description:

Shutdown the Vzense SDK. It is forbidden to call any other SDK API after the

PsShutdown is called.

Parameters:

None

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.3 Ps2_GetDeviceCount

Prototype:

PsReturnStatus Ps2_GetDeviceCount(int32_t* pDeviceCount)

Description:

Get the connected device count.

Parameters:

pDeviceCount **[out]**:The pointer to the variable that need to store the returned device count. It needs to create an int variable first and then pass its pointer to this function.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.4 Ps2_GetDeviceListInfo

Prototype:

PsReturnStatus Ps2_GetDeviceListInfo(PsDeviceInfo* pDevicesList,

uint32_t deviceCount)

Description:

Get the info list of devices currently connected

Parameters:

deviceCount**[in]**: the count of devices

pDevicesList**[out]**: The pointer to the variable that need to store the returned

devices info.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.5 Ps2_GetDeviceInfo

Prototype:

PsReturnStatus Ps2_GetDeviceInfo(PsDeviceInfo* pDevices,

uint32_t deviceIndex)

Description:

Get the info of the device which index is deviceIndex

Parameters:

deviceIndex**[in]**: the index of device

pDevices**[out]**: The pointer to the variable that need to store the returned

device info.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.6 Ps2_OpenDevice

Prototype:

PsReturnStatus Ps2_OpenDevice(const char* uri, PsDeviceHandle *pDevice)

Description:

Open the specific device indicated by uri and return the device handle.

Parameters:

uri**[in]**: the Identifier of device

pDevice**[out]**: Thehandle of the device

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.7 Ps2_CloseDevice

Prototype:

PsReturnStatus Ps2_CloseDevice(PsDeviceHandle device)

Description:

Close the specific device indicated by pDevice.

Parameters:

device**[in]**: The device handle.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.8 Ps2_StartStream

Prototype:

PsReturnStatus Ps2_StartStream(PsDeviceHandle device,

uint32_t sessionIndex)

Description:

Start to capture the specific session stream indicated by device and sessionIndex.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: The index of the session that include N TOF sensors and maximum N RGB sensors.range from 0 to ::SessionCount - 1. See ::PsDeviceInfo for more information. For example, the camera has 2 TOF sensor and 1 RGB sensor,

the ::SessionCount is 2.If the sessionIndex is 0 mean that start 1 TOF stream and the RGB stream, and if the sessionIndex is 1 mean that start only 1 TOF stream.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.9 Ps2_StopStream

Prototype:

PsReturnStatus Ps2_StopStream(int32_t deviceIndex,

PsFrameType frameType)

Description:

Stop to capture the specific session stream indicated by device and sessionIndex.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: The index of the session that include N TOF sensors and maximum N RGB sensors.range from 0 to ::SessionCount - 1. See ::PsDeviceInfo for more information. For example, the camera has 2 TOF sensor and 1 RGB sensor, the ::SessionCount is 2.If the sessionIndex is 0 mean that stop 1 TOF stream and the RGB stream, and if the sessionIndex is 1 mean that stop only 1 TOF stream.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.10Ps2_ReadNextFrame

Prototype:

PsReturnStatus Ps2_ReadNextFrame(PsDeviceHandle device,

uint32_t sessionIndex, PsFrameReady* pFrameReady)

Description:

Capture the next image frame of the specific device. This API should be called

first before getting the frame data using Ps2_GetFrame.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index. see Ps2_StartStream Ps2_StopStream for more info.

pFrameReady**[out]**: the flg of ready frame, see Ps2_FrameReady for more info.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.11Ps2_GetFrame

Prototype:

PsReturnStatus Ps2_GetFrame(PsDeviceHandle device, uint32_t sessionIndex,

PsFrameType frameType, PsFrame* pPsFrame)

Description:

Get the image data of current frame indicated by frame type. It needs to call

Ps2_ReadNextFrame to capture one frame of image first before calling this API.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

frameType**[in]**: the frame type.see PsFrameType for more info.

pPsFrame**[out]**: The pointer of buffer to store the returned image data.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.12Ps2_SetDataMode

Prototype:

PsReturnStatus Ps2_SetDataMode(PsDeviceHandle device,

uint32_t sessionIndex, PsDataMode dataMode)

Description:

Set the output data mode

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

dataMode[**in**]: output data mode, refer to PsDataMode

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.13Ps2_GetDataMode

Prototype:

PsReturnStatus Ps2_GetDataMode(PsDeviceHandle device,

uint32_t sessionIndex, PsDataMode* dataMode)

Description:

Set the output data mode

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

dataMode**[out]**: output data mode, refer to PsDataMode

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.14Ps2_GetDepthRange

Prototype:

PsReturnStatus Ps2_GetDepthRange(PsDeviceHandle device,

uint32_t sessionIndex, PsDepthRange* pDepthRange)

Description:

Get the depth range mode of the specific device.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pDepthRange **[out]**: The pointer of variable to store the returned depth range mode. Refer to PsDepthRange.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.15Ps2_SetDepthRange

Prototype:

PsReturnStatus Ps2_SetDepthRange(PsDeviceHandle device,

uint32_t sessionIndex, PsDepthRange depthRange)

Description:

Set the depth range mode of the specific device.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

depthRange **[in]**: The depth range that needs to set. Refer to PsDepthRange.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.16Ps2_GetThreshold

Prototype:

PsReturnStatus Ps2_GetThreshold(PsDeviceHandle device,

uint32_t sessionIndex, uint16_t * pThreshold)

Description:

Get the threshold value

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pThreshold **[out]**: the threshold value

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.17Ps2_SetThreshold

Prototype:

PsReturnStatus Ps2_SetThreshold(PsDeviceHandle device,

uint32_t sessionIndex, uint16_t threshold)

Description:

Set the threshold value

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pThreshold **[in]**: the threshold value

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.18Ps2_GetPulseCount

Prototype:

PsReturnStatus Ps2_GetPulseCount(PsDeviceHandle device,

uint32_t sessionIndex, uint16_t pulseCount)

Description:

Set the pulse count

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pPulseCount **[out]**: pointer to the variable that used to store returned pulse count

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.19Ps2_SetPulseCount

Prototype:

PsReturnStatus Ps2_SetPulseCount(PsDeviceHandle device,

uint32_t sessionIndex, uint16_t pulseCount)

Description:

Set the pulse count

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pPulseCount **[in]**: the pulse count value

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.20Ps2_GetGMMGain

Prototype:

PsReturnStatus Ps2_GetGMMGain(PsDeviceHandle device,

uint32_t sessionIndex, uint16_t* gmmgain)

Description:

Getting Gamma Gain of Device

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

gmmgain**[out]**: To store the returned Gamma value variable pointer, you need

to first create an unsigned short type variable and pass its pointer to the function

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.21Ps2_SetGMMGain

Prototype:

PsReturnStatus Ps2_SetGMMGain(PsDeviceHandle device,

uint32_t sessionIndex, uint16_t* gmmgain)

Description:

Setting Device Gamma Gain

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

gmmgain **[in]**: Gamma gain to be set

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.22Ps2_GetProperty

Prototype:

PsReturnStatus Ps2_GetProperty(PsDeviceHandle device,

uint32_t sessionIndex, PsPropertyType propertyType,

void* pData, int32_t* pDataSize)

Description:

Get the property value of the specific device indicated by deviceIndex.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

propertyType **[in]**: The property type. Refer to PsPropertyType.

pData **[out]**: The pointer of buffer to store the returned property value.

pDataSize **[in/out]**: Pass the buffer size of pData. Also return the actual size of returned property value in byte.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.23Ps2_SetProperty

Prototype:

PsReturnStatus Ps2_SetProperty(PsDeviceHandle device,

uint32_t sessionIndex, PsPropertyType

propertyType, const void* pData, int32_t dataSize)

Description:

Set the property value of the specific device.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

propertyType **[in]**: The property type. Refer to PsPropertyType.

pData **[in]**: The pointer of buffer which stores the property value to set.

pDataSize **[in]**: The property value size.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.24Ps2_GetCameraParameters

Prototype:

PsReturnStatus Ps2_GetCameraParameters(PsDeviceHandle device,

uint32_t sessionIndex, PsSensorType sensorType,

PsCameraParameters* pCameraParameters)

Description:

Get the camera internal parameters

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

sensorType **[in]** : Type of sensor，0 indicates the depth camera ，1 indicates the RGB camera

pCameraParameters**[out]**: Output the camera internal parameters, refer to PsCameraParameters

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.25Ps2_GetCameraExtrinsicParameters

Prototype:

PsReturnStatus Ps2_GetCameraExtrinsicParameters(PsDeviceHandle device,

uint32_t sessionIndex, PsCameraExtrinsicParameters*

pCameraExtrinsicParameters)

Description:

Get camera rotation and transmission coefficient parameters

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pCameraExtrinsicParameters [**out**]: Pointer to the structural variable used

to store the returned camera parameters

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.26Ps2_SetColorPixelFormat

Prototype:

PsReturnStatus Ps2_SetColorPixelFormat(PsDeviceHandle device,

uint32_t sessionIndex, const PsPixelFormat pixelFormat);

Description:

Set the format of pixel

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pixelFormat **[in]**: format of pixel,

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.27Ps2_SetRGBResolution

Prototype:

PsReturnStatus Ps2_SetRGBResolution(PsDeviceHandle device,

uint32_t sessionIndex,PsResolution resolution);

Description:

Set RGB resolution

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

resolution**[in]**: RGB resolution,See PsResolution for more info.

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.28Ps2_GetRGBResolution

Prototype:

PsReturnStatus Ps2_GetRGBResolution(PsDeviceHandle device,

uint32_t sessionIndex,uint16_t* resolution);

Description:

Get RGB resolution

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

resolution**[out]**: RGB resolution,See PsResolution for more info.

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.29Ps2_SetWDROutputMode

Prototype:

PsReturnStatus Ps2_SetWDROutputMode(PsDeviceHandle device,

uint32_t sessionIndex, PsWDROutputMode* pWDRMode)

Description:

Set WDR output mode, refer to PsWDROutputMode

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pWDRMode**[In]**: the WDR output mode, refer to PsWDROutputMode

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.30Ps2_GetWDROutputMode

Prototype:

PsReturnStatus Ps2_GetWDROutputMode(PsDeviceHandle device,

uint32_t sessionIndex, PsWDROutputMode* pWDRMode)

Description:

Get WDR mode.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pWDRMode[**Out**]: the WDR mode, refer to PsWDROutputMode

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.31Ps2_SetWDRStyle

Prototype:

PsReturnStatus Ps2_SetWDRStyle(PsDeviceHandle device,

uint32_t sessionIndex, PsWDRStyle wdrStyle)

Description:

Set output style of WDR mode

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

wdrStyle[**in**]: the output style, in fusion or alternation, refer to PsWDRStyle

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.32Ps2_GetMeasuringRange

Prototype:

PsReturnStatus Ps2_GetMeasuringRange(PsDeviceHandle device,

uint32_t sessionIndex, PsDepthRange depthRange,

PsMeasuringRange* pMeasuringRange)

Description:

Get Measuring Range

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

depthRange[**in**]: the depth range.

pMeasuringRange[**Out**]: the measuring range, refer to PsMeasuringRange

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.33Ps2_ConvertWorldToDepth

Prototype:

PsReturnStatus Ps_ConvertWorldToDepth(PsDeviceHandle device,

uint32_t sessionIndex, PsVector3f*pWorldVector,

PsDepthVector3* pDepthVector, int32_t pointCount)

Description:

Convert the input points from the World coordinate system to the Depth coordinate system.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pWorldVector **[in]**: The pointer to the buffer which stored the x,y,z value of world coordinate of the input points to be converted, measured in millimeters.

pDepthVect **[out]**: The pointer to the buffer to store the output x,y,z value of depth coordinate. (x,y) is measured in pixels with (0,0) at the top left of the image. z is measured in millimeters, it is the depth value of the point to be converted.

pointCount **[in]**: The point count to be converted.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.34Ps2_ConvertDepthToWorld

Prototype:

PsReturnStatus Ps_2ConvertDepthToWorld(PsDeviceHandle device,

uint32_t sessionIndex, PsDepthVector3* pDepthVector,

PsVector3f* pWorldVector, int32_t pointCount)

Description:

Convert the input points from the Depth coordinate system to the World coordinate system.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

pDepthVect **[in]**: The pointer to the buffer to store the output x,y,z value of depth coordinate. (x,y) is measured in pixels with (0,0) at the top left of the image. z is

measured in millimeters, it is the depth value of the point to be converted.

pWorldVector **[out]**: The pointer to the buffer which stored the x,y,z value of world coordinate of the input points to be converted, measured in millimeters.

pointCount **[in]**: The point count to be converted.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.35Ps2_ConvertDepthFrameToWorldVector

Prototype:

PsReturnStatus Ps_2ConvertDepthFrameToWorldVector(PsDeviceHandle device, uint32_t sessionIndex, const PsFrame& depthFrame, PsVector3f* pWorldVector)

Description:

Convert all points in depthframe from the Depth coordinate system to the World coordinate system.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

depthFrame**[in]**: The depth frame.

pWorldVector**[out]**: The pointer to the buffer which stored the x,y,z value of world coordinate of the input points to be converted, measured in millimeters.

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.36Ps2_SetSynchronizeEnable

Prototype:

PsReturnStatus Ps2_SetSynchronizeEnabled (PsDeviceHandle device,

uint32_t sessionIndex,, bool bEnabled)

Description:

Set whether the output RGB, Depth, IR and other images are synchronized in time

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: True is set to synchronize and false is set to asynchronize

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section 5.1.7

5.3.37Ps2_GetSynchronizeEnable

Prototype:

PsReturnStatus Ps2_GetSynchronizeEnabled (PsDeviceHandle device,

uint32_t sessionIndex,, bool* bEnabled)

Description:

Get whether the output RGB, Depth, IR and other images are synchronized in time

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: True is set to synchronize and false is set to asynchronize

Returns:

PsRetOK: Succeed

Others: Failed, refer to PsReturnStatus. Reference to section5.1.7

5.3.38Ps2_SetDepthDistortionCorrectionEnabled

Prototype:

PsReturnStatus Ps2_SetDepthDistortionCorrectionEnabled(PsDeviceHandle device, uint32_t sessionIndex, bool bEnabled)

Description:

Set to enable or disable the Depth distortion correction feature

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.39Ps2_GetDepthDistortionCorrectionEnabled

Prototype:

PsReturnStatus Ps2_GetDepthDistortionCorrectionEnabled(PsDeviceHandle device, uint32_t sessionIndex, bool* bEnabled)

Description:

Get the Depth distortion correction feature, enable or disable

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.40Ps2_SetIrDistortionCorrectionEnabled

Prototype:

PsReturnStatus Ps2_SetIrDistortionCorrectionEnabled(PsDeviceHandle device, uint32_t sessionIndex,bool bEnabled)

Description:

Set to enable or disable the IR distortion correction feature

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.41Ps2_GetIrDistortionCorrectionEnabled

Prototype:

PsReturnStatus Ps2_GetIrDistortionCorrectionEnabled(PsDeviceHandle device, uint32_t sessionIndex,bool* bEnabled)

Description:

Get the IR distortion correction feature, enable or disable

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.42Ps2_SetRGBDistortionCorrectionEnabled

Prototype:

PsReturnStatus Ps_SetRGBDistortionCorrectionEnabled(PsDeviceHandle device, uint32_t sessionIndex, bool bEnabled)

Description:

Set to enable or disable the RGB distortion correction feature

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.43Ps2_GetRGBDistortionCorrectionEnabled

Prototype:

PsReturnStatus Ps_GetRGBDistortionCorrectionEnabled(PsDeviceHandle device, uint32_t sessionIndex, bool* bEnabled)

Description:

Get the RGB distortion correction feature, enable or disable

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.44Ps2_SetComputeRealDepthCorrectionEnabled

Prototype:

PsReturnStatus Ps2_SetComputeRealDepthCorrectionEnabled

(PsDeviceHandle device, uint32_t sessionIndex, bool bEnabled)

Description:

Set to enable or disable the computer real depth correction feature

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for

more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.45Ps2_GetComputeRealDepthCorrectionEnabled

Prototype:

PsReturnStatus Ps2_GetComputeRealDepthCorrectionEnabled

(PsDeviceHandle device, uint32_t sessionIndex, bool* bEnabled)

Description:

Set the computer real depth correction feature,enable or disable.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.46Ps2_SetSpatialFilterEnabled

Prototype:

PsReturnStatus Ps_SetSpatialFilterEnabled(PsDeviceHandle

device, uint32_t sessionIndex, bool bEnabled)

Description:

Set to enable or disable the Spatial Filter feature

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.47Ps2_GetSpatialFilterEnabled

Prototype:

PsReturnStatus Ps_GetSpatialFilterEnabled(PsDeviceHandle

device, uint32_t sessionIndex, bool bEnabled)

Description:

Get the Spatial Filter feature,enable or disable

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for

more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.48Ps2_SetTimeFilterEnabled

Prototype:

PsReturnStatus Ps_SetTimeFilterEnabled(PsDeviceHandle

device, uint32_t sessionIndex, bool bEnabled)

Description:

Set to enable or disable the Time Filter feature

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.49Ps2_GetTimeFilterEnabled

Prototype:

PsReturnStatus Ps_GetTimeFilterEnabled(PsDeviceHandle

device, uint32_t sessionIndex, bool bEnabled)

Description:

Get the Time Filter feature,enable or disable

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.50Ps2_SetMapperEnabledDepthToRGB

Prototype:

PsReturnStatus Ps2_SetMappedEnabledDepthToRGB(PsDeviceHandle

device, uint32_t sessionIndex, bool bEnabled)

Description:

Set to enable or disable the feature of mapping RGB image to depth camera space,if this feature is enabled, the mapped RGB image can be get through PsGetFrame with input frame type "PsMappedRGBFrame"

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.51Ps2_GetMapperEnabledDepthToRGB

Prototype:

PsReturnStatus Ps2_GetMappedEnabledDepthToRGB(PsDeviceHandle

device, uint32_t sessionIndex, bool* bEnabled)

Description:

Get the feature of mapping RGB image to depth camera space,if this feature is enabled, the mapped RGB image can be get through PsGetFrame with input frame type "PsMappedRGBFrame"

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.52Ps2_SetMapperEnabledRGBToDepth

Prototype:

PsReturnStatus Ps2_SetMappedEnabledRGBToDepth(int32_t deviceIndex, bool bEnabled)

Description:

Set to enable or disable the feature of mapping depth image to RGB camera space,if this feature is enabled, the mapped depth image can be get through PsGetFrame with input frame type "PsMappedDepthFrame"

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[in]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

5.3.53Ps2_GetMapperEnabledRGBToDepth

Prototype:

PsReturnStatus Ps2_GetMappedEnabledRGBToDepth(int32_t deviceIndex, bool* bEnabled)

Description:

Get the feature of mapping depth image to RGB camera space,if this feature is enabled, the mapped depth image can be get through PsGetFrame with input frame type "PsMappedDepthFrame"

PS: some model may not support the API, such as DCAM550. Please refer to the definition under the specific model folder in include.

Parameters:

device**[in]**: The device handle.

sessionIndex**[in]**: the session index.see Ps2_StartStream Ps2_StopStream for more info.

bEnabled **[out]**: true to enable the feature, false to disable the feature

Returns:

PsRetOK: Succeed

Others: Failed. Reference to section 5.1.7

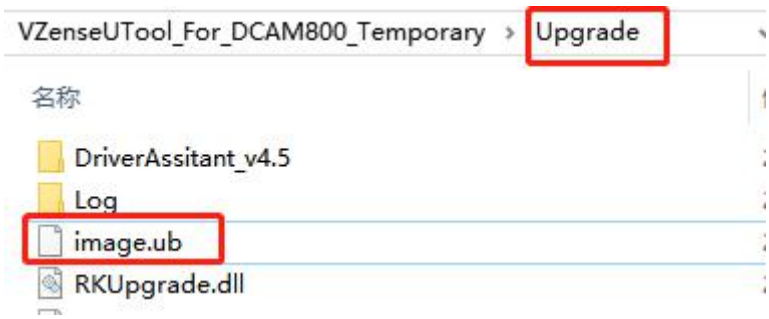
6 Update Firmware

6.1 DCAM800

To upgrade firmware you need the upgrade tool UTool, which can be downloaded from our

official website or obtained by contacting our colleagues. The specific steps are as follows:

1. Place the upgrade file (image.ub) in the upgrade directory of UTool:



2. Connect the device to PC (direct connection and DHCP are available) stage

3. Open UTool, click "start" when the "start" button is displayed in green, select DeviceInfo to "upgrade flag", and click "write". After that, the device will automatically enter the upgrade state, in which the red indicator light of the device is always on.

4. After the upgrade is completed, the red indicator light of the device will flash all the time. At this time, the device can be used normally after power failure and restart.