Vzense Nebula SDK

User Guide

Windows

2022.07

Vzense Technology,Inc.

About This Document

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

Document Structure

Chapter	Title	Contents
1	Overview	Introduce general information of Nebula SDK
2	Products	Introduce supported devices
3	Installation	Introduce how to setup VS environment and how to connect with Vzense device
4	SDK Instruction	Introduce how to use Nebula SDK
5	API Introduction	Introduce APIs of Nebula SDK
6	FAQs	

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.

Nebula SDK is a software development kit based on Vzense products, which is currently available for Windows, Linux, ARM Linux operating systems, provide friendly APIs and application examples for developer.

High precision depth image data, gray image data and point cloud data can be got using the SDK. It is convenient to develop gesture recognition, projection touch, face recognition, fatigue detection, 3D modeling, navigation, obstacle avoidance and so on.

The SDK download link:

https://github.com/Vzense/NebulaSDK

https://gitee.com/Vzense/NebulaSDK

2 Products

Nebula SDK currently supports products like:

- DS77 Lite/Pro

- DS77C Lite/Pro

2.1 DS77 Lite/Pro

Model	DS77 Lite	DS77 Pro
Sensor	SONY DepthSense ToF	
Laser	940nm VCSEL * 2	
TOF Resolution	640 * 480, Max. 25fps	
TOF FOV	70°(H) * 50°(V)	
Pixel Format	12bit Depth, 8bit IR	
Digital Interface	1000M Ethernet, RS485	
Power Supply	12V ~ 24V DC	12V ~ 24V DC or POE+
Accuracy	< 1% (4mm@1m)	
Detect Range	0.15m ~ 5m	
Operating Temperature	-20℃ ~ 50℃	
OS Support	Windows, Linux, Arm Linux	
Software Support	Nebula SDK, C++, C, Python	
Ingress Protection	IP42	IP67

2.2 DS77C Lite/Pro

Model	DS77C Lite	DS77C Pro
Sensor	SONY DepthSense ToF + RGB	
Laser	940nm VCSEL * 2	
TOF Resolution	640 * 480, Max. 25fps	
RGB Resolution	1600*1200, Max. 25fps	
TOF FOV	70°(H) * 50°(V)	
RGB FOV	77°(H) * 55°(V)	
Pixel Format	12bit Depth, 8bit IR, MJPEG RGB	
Digital Interface	1000M Ethernet, RS485	
Power Supply	12V ~ 24V DC	12V ~ 24V DC or POE+
Accuracy	< 1% (4mm@1m)	
Detect Range	0.15m ~ 5m	
Operating Temperature	-20℃ ~ 50℃	
OS Support	Windows, Linux, Arm Linux	
Software Support	Nebula SDK, C++, C, Python	
Ingress Protection	IP42	IP67

3 Installation

3.1 Recommended Operation System

Item	Recommended System
Operation System	Win7 32/64 bits Win10 64 bits Win11 64 bits

3.2 **Device Installation**
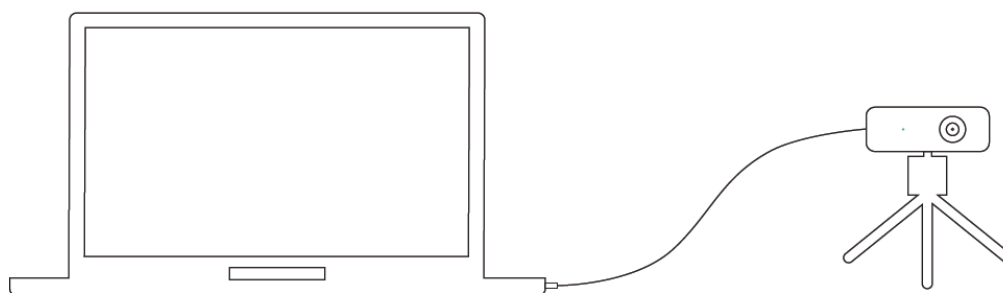
Figure 3.1 Hardware Installation

Vzense cameras support two connection method: static IP address and DHCP. Static IP address is the default mode of Vzense devices. The IP address, subnet mask and DHCP enable also can be changed with NebulaGUITool.

3.2.1 **Static Address**

Using static address connection method device can be connected to computer directly, or with one network switch repeat. But please ensure the camera and computer are resided on the same network segment.

For example, the direct connection: one end of the cable is connected to the camera, and the other end is plugged into RJ45 port of the PC. The default IP address of Vzense camera is 192.168.1.101, so the address of the PC can be set to 192.168.1.100. In Windows 10 system, the below picture can be referenced to set the network up.

Figure 3.2 Static IP Address Configuration

3.2.2 DHCP

Using DHCP connection mode the Vzense camera and PC need be set to 'DHCP' mode, then connect them to one same LAN that have router. For details on how to set the camera to DHCP connection mode, refer to the NebulaGUITool documentation. Setting the 'local connection' of the PC to obtain the IP address automatically is the recommended method.

Vzense Technology,Inc.

Figure 3.3 DHCP

Note:

1. The network card, router or switch all of these have to meet the requirements of 1000Mbps Ethernet.

2. When the application based on Nebula SDK first run, all options of the firewall should be chosen.
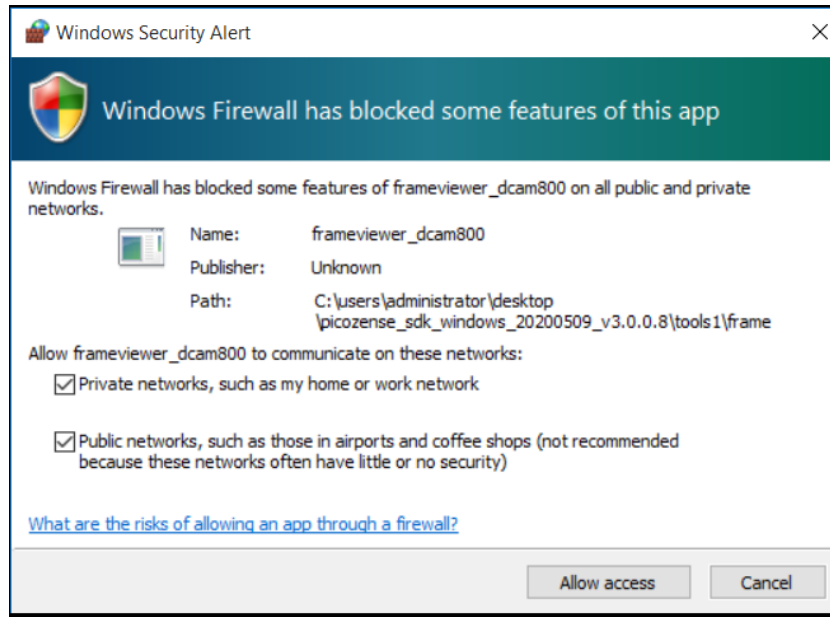
Figure 3.4 firewall setting

4 Nebula SDK Instruction

4.1 Nebula SDK Structure

In Vzense Nebula SDK, there are several directories: Bin, Document, Include, Lib, PrecompiledSamples, Samples.
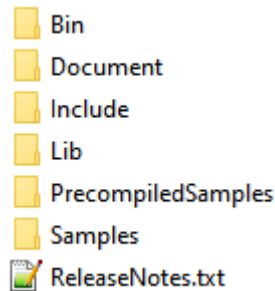
Figure 4.1 Windows Nebula SDK Directory

- Bin contains the dynamic library files of Nebula SDK.

- Document contains documents of Nebula SDK.

- Include contains header files of Nebula SDK.

- Lib contains the import library files of Nebula SDK.

- PrecompiledSamples contain several prebuild samples that can be executed directly for depth, IR and color images previewing.

- Samples contains several example codes. The examples have two different parts: Base and OpenCV. Base part is the basic APIs using example. And the other part is used to demonstrate how to use third-party library like OpenCV. In the OpenCV sample depth image, IR image and color image can be previewed using imshow method.

4.2 Development Guidelines

4.2.1 Project Configuration

Create one new application project using Visual Studio 2017. Open the project property, insert the Include path of Nebula SDK to [Include Directories]. The Lib path also

need be added in [Library Directories]. Additionally, it needs to add the vzenseDS_api.lib

to [Additional Dependencies]. The project configuration in samples can be referenced.
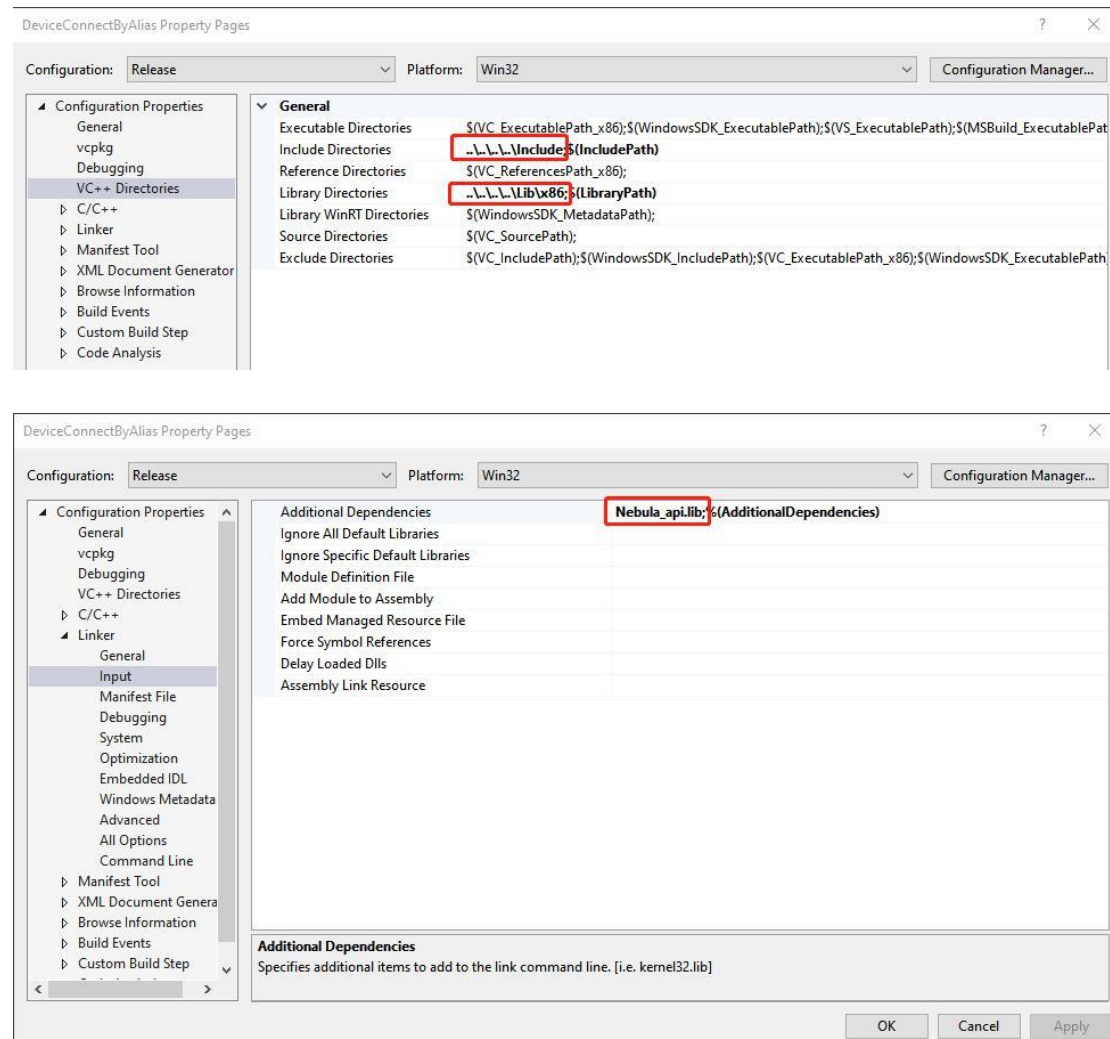
Figure 4.2 SDK Project Configuration

4.2.2 **API Invoke Flow**

The API of Nebula SDK invoking flow chat like below:

```
┌────────────────────┐                    ┌────────────────────┐
│   VZ_Initialize    │                    │   VZ_Shutdown      │
└────────────────────┘                    └────────────────────┘
          │                                          ▲
          ▼                                          │
┌────────────────────┐                    ┌────────────────────┐
│  VZ_GetDeviceCount │                    │   VZ_CloseDevice   │
└────────────────────┘                    └────────────────────┘
          │                                          ▲
          ▼                                          │
┌────────────────────┐                    ┌────────────────────┐
│  VZ_GetDeviceInfo  │                    │   VZ_StopStream    │
│ /VZ_GetDeviceInfoList │                 └────────────────────┘
└────────────────────┘                               ▲
          │                                          │
          ▼                                          │
┌────────────────────┐                    ┌────────────────────┐
│ VZ_OpenDeviceByXXX │                    │    VZ_GetFrame     │──┐
└────────────────────┘                    └────────────────────┘  │
          │                                          ▲            │
          ▼                                          │            │
┌────────────────────┐                    ┌────────────────────┐  │
│   VZ_StartStream   │─────────────────▶  │  VZ_GetFrameReady  │◀─┘
└────────────────────┘                    └────────────────────┘
```
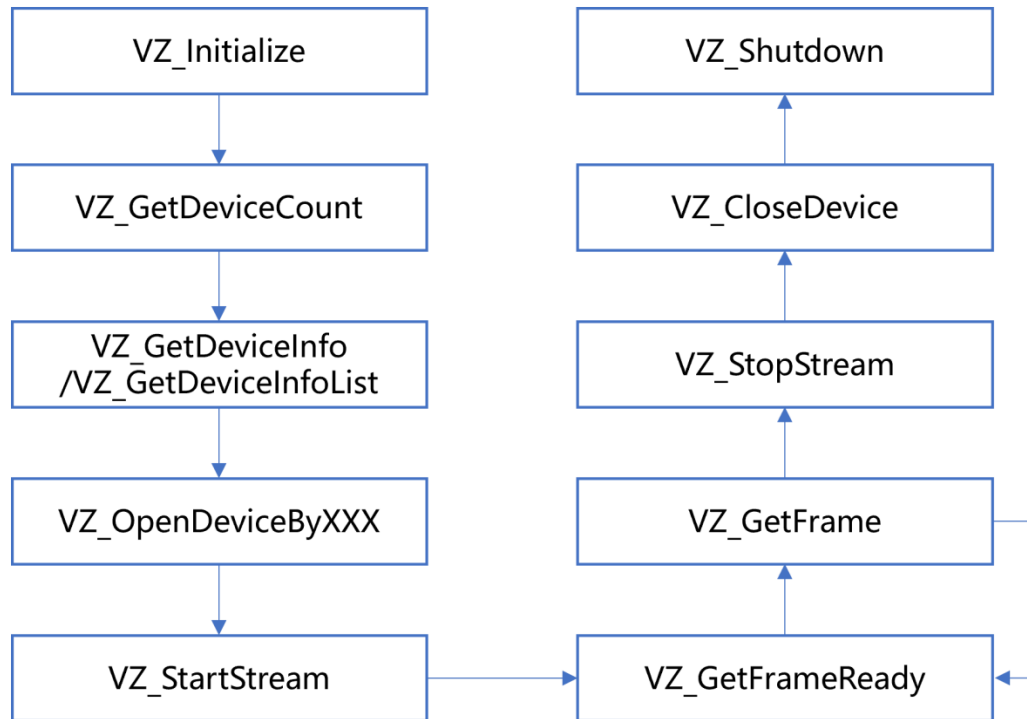
Figure 4.3 SDK API Invoke Flow Chat

1. VZ_Initialize and VZ_Shutdown

VZ_Initialize: initialize the SDK

VZShutdown: deinitialize the SDK, and destroy all of the resources.

2. VZ_GetDeviceCount and VZ_GetDeviceInfoList/VZ_GetDeviceInfo

VZ_GetDeviceCount: get the number of connected devices.

VZ_GetDeviceInfoList/VZ_GetDeviceInfo: get the information of connected devices.

3. VZ_OpenDeviceByXXX and VZ_CloseDevice

VZ_OpenDeviceByXXX: open the specified device, the interface can support using URI, IP Address and alias.

VZ_CloseDevice: close the specified device.

4. VZ_StartStream and VZ_StopStream

VZ_StatrStream: start streaming of the specified device.

VZ_StopStream: stop streaming of the specified device.

5. VZ_GetFrameReady and VZ_GetFrame

In the main loop of capture, VZ_GetFrameReady is called first to check the frames, then call VZ_GetFrame to obtain the frame image data of the specified image type.

6. Set and Get

The SDK provides Set and Get interface for writing and reading properties, parameters of device, as detailed in Section 5.3.

4.3 **SDK Sample**

Nebula software development kit provide several sample codes for API interface using demonstration in the Samples folder of the SDK. The following directories are currently available:

- Base: the basic APIs using example

- OpenCV: the example works with third-party library OpenCV

4.3.1 **Base Samples**

The base sample is used to demonstrate the single feature of basic APIs. In order to help developer familiar with SDK quickly, the examples are classified according to products, such as DS77, DS77C, etc. Open device, image acquisition, software/ hardware trigger, point cloud store are all included in the SDK example codes.

1. Download Nebula SDK from Gitee/GitHub

> git clone https://github.com/Vzense/NebulaSDK

Figure4.4 Download Nebula SDK

2. Select the corresponding example according to the device. For example, the DeviceConnectByAlias example of DS77C
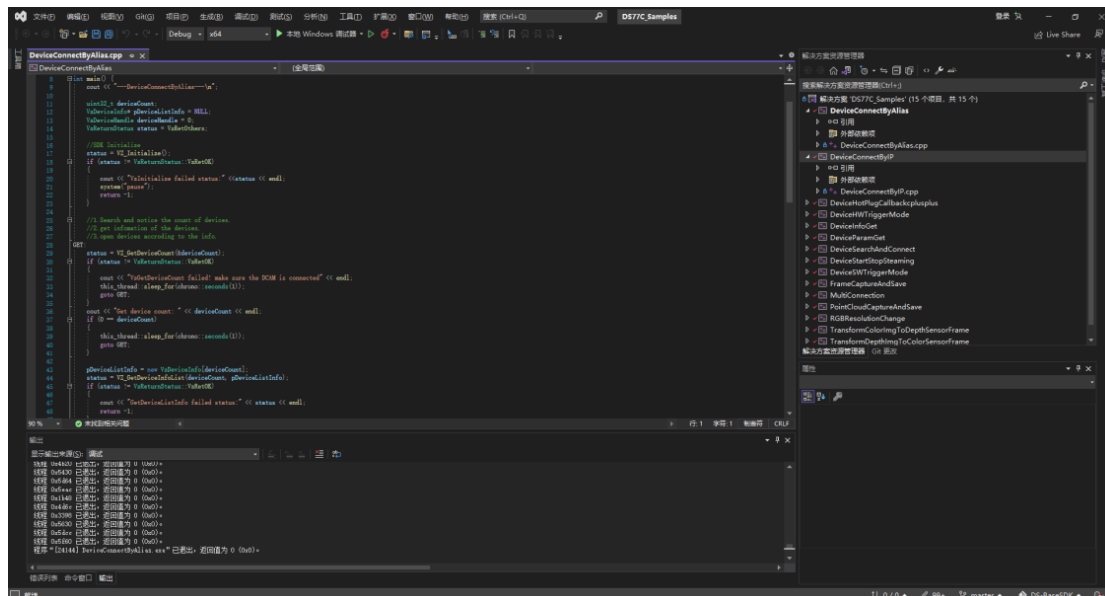
Figure 4.5 Compilation

3.After compilation, run it and check the result. The result is shown as below:

Figure 4.6 Execution

4.3.2 **OpenCV Samples**

The OpenCV samples show how to use Nebula SDK with third-party libraries. The example uses the image mapping function of OpenCV to display color depth image.

1. Download and install OpenCV3.0.0 from OpenCV official website. Any version could be used, it is just an example.
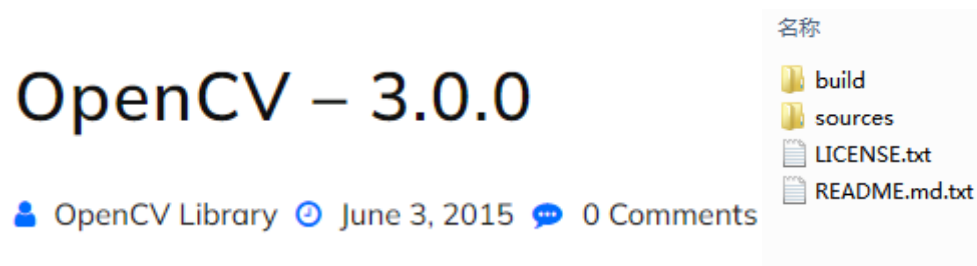
Figure 4.7 Download and Install OpenCV

2. Set the environment variable for OPENCV_DIR. The variable value should be the absolute path of the directory of OpenCV, for example:

D:\Program Files\opencv-3.0.0\build.
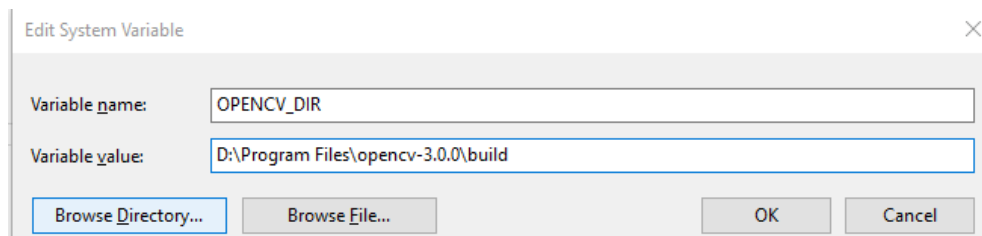
Figure 4.8 Set the Environment Variable OPENCV_DIR

3. Select the corresponding example according to the device, for example DS77.
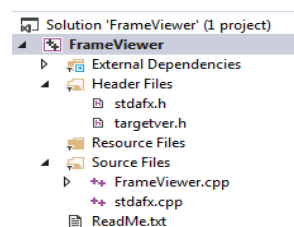
Figure 4.9 FrameViewer Project

4. Build the project, the execute file is in NebulaSDK\Windows\Bin\x86 or

NebulaSDK\Windows\Bin\x64.

5. Run the FrameViewer.exe, and check the result.
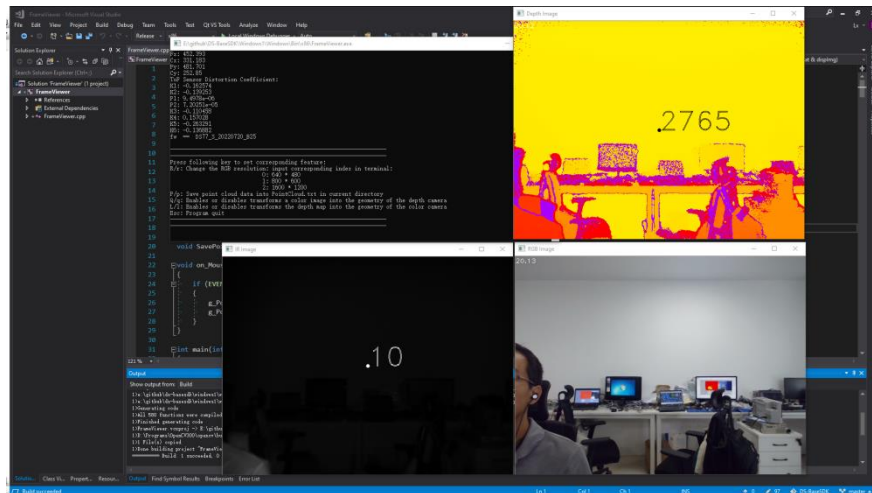
Figure 4.10 Execution Successful

5 Nebula SDK API Introduction

5.1 Enum Type

5.1.1 VzFrameType

Description：

Image type.

PS: The number of enumerated values may vary depending on the product model.

Enumerator：

VzDepthFrame: depth frame with 16 bits per pixel in millimeters

VzIRFrame: ir frame with 8 bits per pixel

VzColorFrame: color frame with 24 bits per pixel in RGB/BGR format

VzTransformColorImgToDepthSensorFrame: color frame with 24 bits per pixel in RGB/BGR format, that is transformed to depth sensor space where the resolution is the same as the depth frame's resolution. This frame type can be enabled VZ_SetTransformColorImgToDepthSensorEnabled.

VzTransformDepthImgToColorSensorFrame: depth frame with 16 bits per pixel, in millimeters, that is transformed to color sensor space where the resolution is same as the color frame's resolution. This frame type can be enabled using VZ_SetTransformDepthImgToColorSensorEnabled.

VzConfidenceFrame: Laser intensity image.

5.1.2 **VzPixelFormat**

Description：

Pixel type of image data.

PS: The number of enumerated values may vary depending on the product

model.

Enumerator：

VzPixelFormatDepthMM16: Represents the 16-bit depth value of each pixel data

in millimeters

VzPixelFormatGray8: Indicates the 8-bit grayscale value of each pixel

VzPixelFormatRGB888: Represents the 24-bit RGB value of each pixel data

VzPixelFormatBGR888: Represents the 24-bit BGR value of each pixel

5.1.3 **VzSensorType**

Description：

Sensor Type

PS: The number of enumerated values may vary depending on the product

model.

Enumerator：

VzToFSensor: indicates a depth sensor

VzColorSensor: indicates a color sensor.

5.1.4 **VzReturnStatus**

Description：

Returns of an interface function.

PS: The number of enumerated values may vary depending on the product

model. For details, see Include

Enumerator：

VzRetOK: indicates that the call is successful

VzRetNoDeviceConnected: indicates that no device is connected

VzRetInvalidDeviceIndex: indicates that the incoming device serial number is

invalid

VzRetDevicePointerIsNull: indicates that the passed device pointer is null

VzRetInvalidFrameType: indicates that the incoming image type is invalid

VzRetFramePointerIsNull: indicates that the passed image pointer is null

VzRetNoPropertyValueGet: indicates that the current property value cannot be

obtained

VzRetNoPropertyValueSet: indicates that the current property value cannot be

set

VzRetPropertyPointerIsNull: said the incoming pointer to storage property value

of the cache is empty

VzRetPropertySizeNotEnough: said the incoming cache space to store the value

of an attribute

VzRetInvalidDepthRange: indicates that the passed depth range is invalid

Timeout while VzRetGetFrameReadyTimeOut: access to images

VzRetInputPointerIsNull: indicates that the incoming pointer is null

VzRetCameraNotOpened: indicates that the camera is not opened

VzRetInvalidCameraType: indicates that the incoming camera type is invalid

VzRetInvalidParams: indicates that the passed parameters are invalid

VzRetCurrentVersionNotSupport said the current version is not supported

VzRetUpgradeImgError: indicates that the camera firmware upgrade fails

VzRetUpgradeImgPathTooLong: said the incoming camera firmware path length is too long

VzRetUpgradeCallbackNotSet: said not set the camera to upgrade the callback function

VzRetNoAdapterConnected: indicates that the power adapter is not connected

VzRetReInitialized: indicates repeated initialization

VzRetNoInitialized: indicates that the initialization is not performed

VzRetCameraOpened: indicates that the camera is opened

VzRetCmdError: The command fails to be delivered

VzRetCmdSyncTimeOut: indicates that the command is successfully sent, but synchronization fails

VzRetIPNotMatch: indicates that the camera IP address and host IP address are on different network segments

VzRetNotStopStream: indicates that the data stream is not opened

VzRetOthers: indicates another error

5.1.5 **VzResolution**

Description：

Image resolution type

PS: The number of enumerated values may vary depending on the product

model. For details, see Include

Enumerator：

VzColor_Resolution_640_480: indicates that the image resolution is 640x480 (w

x h)

VzColor_Resolution_800_600: indicates that the image resolution is 800x600

VzColor_Resolution_1600_1200: indicates that the image resolution is

1600x1200

5.1.6 **VzConnectStatus**

Description：

Device connection status.

PS: The number of enumerated values may vary depending on the product

model. For details, see Include.

Enumerator：

VzConnectUNKNOWN: indicates that the connection status is unknown

VzUnconnected: indicates that the device is not connected

VzConnected: indicates that the device is connected

VzOpened: indicates that the device is opened

VzUpgradeUnconnected: indicates that the device is in the upgrade state

VzUpgradeConnected: indicates that the device is upgraded and connected

5.1.7 **VzDeviceType**

Description：

Device type

Enumerator：

VzDS77Lite: indicates a DS77Lite camera that uses an RJ45 port and provides only ToF data.

VzDS77CLite: indicates a DS77CLite camera that uses an RJ45 interface and provides ToF+RGB data.

VzDS77Pro: indicates a DS77Pro camera, which uses an aviation plug and provides only ToF data.

VzDS77CPro: Represents a DS77CPro camera, using an aviation plug, while providing ToF+RGB data.

5.1.8 **VzWorkMode**

Description：

Equipment working condition.

Enumerator：

VzActiveMode: indicates that the device is actively working. At this point, after using API to open the camera, the device will actively upload image data.

VzHardwareTriggerMode: indicates that the device is in passive working state. At

this time, after the API is used to open the camera, the device will upload the

image data when the hardware triggers.

VzSoftwareTriggerMode: indicates that the device is in passive working state. At

this time, after using API to open the camera, the device will upload the image

data when triggered by the software.

5.1.9 **VzExposureControlMode**

Description：

Exposure mode of the sensor

Enumerator：

VzExposureControlMode_Auto: indicates that the sensor uses automatic

exposure mode

VzExposureControlMode_Manual: indicates that the sensor uses manual

exposure mode

5.2 **Struct Type**

5.2.1 **VzRGB888Pixel**

Function:

Color image pixel type RGB888.

PS: Different models may not support RGB, for example, DCAM550. Please

refer to the definition in the Include folder.

Members:

Uint8_t R: indicates the red channel

Uint8_t g: indicates the green channel

Uint8_t b: indicates the blue channel

5.2.2 **VzBGR888Pixel**

Function:

Color image pixel type BGR888.

PS: Different models may not support RGB, for example, DCAM550. Please refer to the definition in the Include folder.

Members:

Uint8_t b: indicates the blue channel

Uint8_t g: indicates the green channel

Uint8_t R: indicates the red channel

5.2.3 **VzVector3f**

Function:

Three dimensional point coordinates in millimeters.

Members:

Float x: Represents the coordinate value along the X-axis

Float y: Represents the coordinate value along the y axis

Float Z: Represents the coordinate value along the z axis

5.2.4 **VzVector2u16**

Function:

Two dimensional point coordinates.

Members:

Float x: Represents the coordinate value along the X-axis

Float y: Represents the coordinate value along the y axis

5.2.5 **VzDepthVector3**

Function:

A pixel representation of a depth image.

Members:

Int depthX: indicates the coordinate value along the X-axis in the image coordinate system

Int depthY: represents the coordinate value along the Y-axis in the image coordinate system

VzDepthPixel depthZ: Indicates the depth value at pixel coordinates (depthX, depthY), in millimeters

5.2.6 **VzSensorIntrinsicParameters**

Function:

Lens parameters and distortion parameters of the sensor. The internal parameter is usually used to calculate the point cloud and the distortion parameter is used in the image anti-distortion algorithm.

SDK has realized the function of converting depth image to point cloud and image anti-distortion, please refer to the routine to use the relevant interface.

Members:

Double FX: Focal length x (pixel)

Double FY: Focal length Y (pixel)

Double CX: Principal Point X (Pixel)

Double cy: Principal point Y (pixel)

Double K1: Radial distortion coefficient, 1st-order

Double K2: Radial distortion coefficient, 2nd-order

Double P1: Tangential distortion coefficient

Double P2: Tangential distortion coefficient

Double K3: Radial distortion coefficient, 3rd-order

Double K4: Radial distortion coefficient, 4ST-order

Double K5: Radial distortion coefficient, 5ND-order

Double K6: Radial distortion coefficient, 6RD-Order

5.2.7 **VzSensorExtrinsicParameters**

Function:

The external parameters R and T of the camera are used to align depth with RGB images. The reference formula is as follows:

$$\begin{bmatrix} X_{rgb} \\ Y_{rgb} \\ Z_{rgb} \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & r_2 \\ r_3 & r_4 & r_5 \\ r_6 & r_7 & r_8 \end{bmatrix} * \begin{bmatrix} X_{depth} \\ Y_{depth} \\ Z_{depth} \end{bmatrix} + \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}$$

Members:

Double rotation[9] : A 3×3 rotation matrix

Double translation[3] : 3×1 translation matrix

5.2.8 **VzFrame**

Function:

Image information

Members:

Uint32_t frameIndex: indicates the image frameIndex number

VzFrameType frameType: indicates the image data type

VzPixelFormat pixelFormat: indicates the pixel type

Uint8_t * pFrameData: a pointer to the image data cache

Uint32_t dataLen: Indicates the length of image data in bytes

Float exposureTime: exposureTime, in microseconds

Uint8_t depthRange: indicates the depthRange of the current frame, valid only
for depth images

Uint16_t Width: indicates the image width

Uint16_t height: indicates image height

Uint64_t deviceTimestamp: represents image timestamp

5.2.9 **VzFrameReady**

Function:

Whether the image data is ready (1 means ready, 0 means not ready)

Members:

Uint32_t Depth: 1: indicates whether the depth image data is ready

Vzense Technology,Inc.

Uint32_t IR: 1: Indicates whether grayscale image data is ready

Uint32_t Color: 1: indicates whether color image data is ready

Uint32_t transformedColor: 1: Indicates whether the color image aligned to the depth sensor space is ready

Uint32_t transformedDepth: 1: indicates whether the depth image is ready to align to the color sensor space

Uint32_t Confidence: 1: indicates whether the laser intensity image data is ready

Uint32_t reserved: 26: : Reserved bit

5.2.10 **VzDeviceInfo**

Function:

Equipment information

Members:

Int SessionCount: indicates the number of depth sensors in the device

VzDeviceType devicetype: indicates the devicetype

Char URI [256] : indicates the device identifier

Char alias[64] : indicates the device alias

Char serialNumber[64] : indicates the serialNumber of the device

Char IP [17] : indicates the IP address of the device

VzConnectStatus Status: indicates the connection status of the device

5.2.11 **VzConfidenceFilterParams**

Function:

Confidence filtering parameter

Members:

Bool ENABLE: indicates whether filtering is enabled. True indicates that filtering

is enabled. False indicates that filtering is disabled

Int threshold: indicates the filtering threshold

5.2.12 **VzFlyingPixelFilterParams**

Function:

To fly point filter parameters

Members:

Bool ENABLE: indicates whether filtering is enabled. True indicates that filtering

is enabled. False indicates that filtering is disabled

Int threshold: indicates the filtering threshold

5.2.13 **VzSpatialFilterParams**

Function:

Spatial filtering parameters.

Members:

Depth indicates the reachable state of the Depth graph

Ir indicates the available state of the IR graph

RGB Indicates the available status of an RGB image

MappedRGB indicates the available status of aligned RGB images

MappedDepth indicates the available status of the Depth map after the alignment

MappedIR indicates the availability status of the IR graph after alignment

Confidence represents the accessible state of the confidence graph

WdrDepth Indicates the availability of the Depth map in WDR mode

Reserved indicates the reserved status, which is not used

5.2.14 **VzFillHoleFilterParams**

Function:

Parameter of hole filling filter

Members:

Bool ENABLE: indicates whether filtering is enabled. True indicates that filtering is enabled. False indicates that filtering is disabled

Int validCount: indicates the number of reference points used in filtering calculation

Int threshold: indicates the filtering threshold

Int doCount: indicates how many times the filtering is performed

5.2.15 **VzExposureTimeParams**

Function:

Sensor exposure parameter

Members:

VzExposureControlMode Mode: Sensor exposure type

Int exposureTime: sensor exposureTime in microseconds

5.3 **API**

5.3.1 **VZ_Initialize**

Prototype:

VzReturnStatus VZ_Initialize()

Description:

To complete initialization of the SDK, you need to call it before calling other apis

Parameters:

There is no

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.2 **VZ_Shutdown**

Prototype:

VzReturnStatus VZ_Shutdown()

Description:

The SDK is deregistered and all resources created during the SDK use are released. After this interface is called, no other interfaces other than VZ_Initialize should be called

Parameters:

There is no

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.3 **VZ_GetSDKVersion**

Prototype:

const char* VZ_GetSDKVersion()

Description:

Obtain the SDK version: X.X.X

Parameters:

There is no

Returns:

The SDK version number

5.3.4 **VZ_GetDeviceCount**

Prototype:

VzReturnStatus VZ_GetDeviceCount(uint32_t* pDeviceCount)

Description:

Get the number of connected devices

Parameters:

Uint32_t * pDeviceCount: Returns the number of connected devices

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.5 **VZ_GetDeviceInfo**

Prototype:

VzReturnStatus VZ_GetDeviceInfo(uint32_t deviceIndex, VzDeviceInfo*

pDevicesInfo)

Description:

Gets information about the device with the specified index number

Parameters:

Uint32_t deviceIndex: deviceIndex number

VzDeviceInfo* pDevicesInfo: Returns device information

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.6 **VZ_GetDeviceInfoList**

Prototype:

VzReturnStatus VZ_GetDeviceInfoList(uint32_t deviceCount, VzDeviceInfo*

pDevicesInfoList)

Description:

Obtain the device list of the number of Devicecounts

Parameters:

Uint32_t deviceCount: Number of devices that need to get the information list

VzDeviceInfo* pDevicesInfo: Returns a list of device information that should

point to a cache of size sizeof(VzDeviceInfo)*deviceCount

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.7 **VZ_OpenDeviceByUri**

Prototype:

VzReturnStatus VZ_OpenDeviceByUri(const char* pURI, VzDeviceHandle*

pDevice)

Description:

Open the device using the device identifier

Parameters:

Const char* pURI: indicates the device identifier

VzDeviceHandle* pDevice: The device handle returned after successfully

opening the device

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.8 **VZ_OpenDeviceByAlias**

Prototype:

VzReturnStatus VZ_OpenDeviceByAlias(const char* pAlias, VzDeviceHandle*

pDevice)

Description:

Open the device using the device alias

Parameters:

Const char* pAlias: indicates the device alias

VzDeviceHandle* pDevice: The device handle returned after successfully
opening the device

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.9 **VZ_OpenDeviceByIP**

Prototype:

VzReturnStatus VZ_OpenDeviceByIP(const char* pIP, VzDeviceHandle*
pDevice)

Description:

Use the device IP address to open the device

Parameters:

Const char* pIP: indicates the IP address of the device

VzDeviceHandle* pDevice: The device handle returned after successfully
opening the device

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.10 **VZ_CloseDevice**

Prototype:

VzReturnStatus VZ_CloseDevice(VzDeviceHandle* pDevice)

Description:

Close the equipment

Parameters:

VzDeviceHandle* pDevice: Handle to the device to close

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.11 **VZ_StartStream**

Prototype:

VzReturnStatus VZ_StartStream(VzDeviceHandle device)

Description:

Open data Stream

Parameters:

VzDeviceHandle Device: Handle to the device on which the data stream is to be

closed

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.12 **VZ_StopStream**

Prototype:

VzReturnStatus VZ_StopStream(VzDeviceHandle device)

Description:

Close the data stream

Parameters:

VzDeviceHandle Device: Handle to the device on which the data stream is to be

closed

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.13 **VZ_GetFrameReady**

Prototype:

VzReturnStatus VZ_GetFrameReady(VzDeviceHandle device, uint16_t

waitTime, VzFrameReady* pFrameReady)

Description:

Gets the image ready state. This function must be called before calling

VZ_GetFrame, otherwise the image cannot be retrieved.

Parameters:

VzDeviceHandle Device: Device handle

Uint16_t waitTime: Allows timeout duration (ms) for waiting for image to be ready.

This value depends on the frame rate of the image. The recommended value is

2*1000/ FPS. For example, if the current frame rate is 20, you are advised to set

waitTime to 2 * 1000/20 = 100. If set waitTime of 40, call the function might return

VzRetGetFrameReadyTimeOut.

VzFrameReady* pFrameReady: Returns the ready state of the image

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.14 **VZ_GetFrame**

Prototype:

VzReturnStatus VZ_GetFrame(VzDeviceHandle device, VzFrameType

frameType, VzFrame* pVzFrame)

Description:

Gets image data of the specified image type. VZ_GetFrameReady must be

called before calling this function.

Parameters:

VzDeviceHandle Device: Device handle

VzFrameType frameType: specifies the type of the image to be obtained

VzFrame* pVzFrame: The returned image data

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.15 **VZ_SetWorkMode**

Prototype:

VzReturnStatus VZ_SetWorkMode(VzDeviceHandle device, VzWorkMode

mode)

Description:

Set the camera working mode

Parameters:

VzDeviceHandle Device: Device handle

VzWorkMode mode: indicates the working mode to be set

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.16 **VZ_GetWorkMode**

Prototype:

VzReturnStatus VZ_GetWorkMode(VzDeviceHandle device, VzWorkMode*

pMode)

Description:

Gets the working mode of the camera

Parameters:

VzDeviceHandle Device: Device handle

VzWorkMode* pMode: indicates the working mode of the obtained device

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.17 **VZ_SetSoftwareSlaveTrigger**

Prototype:

VzReturnStatus VZ_SetSoftwareSlaveTrigger(VzDeviceHandle device)

Description:

Performs a software trigger, valid only when the camera is in

VzSoftwareTriggerMode

Parameters:

VzDeviceHandle Device: Device handle

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.18 **VZ_GetSensorIntrinsicParameters**

Prototype:

VzReturnStatus VZ_GetSensorIntrinsicParameters(VzDeviceHandle device,

VzSensorType sensorType,   VzSensorIntrinsicParameters*

pSensorIntrinsicParameters)

Description:

Gets an internal parameter for the sensor lens

Parameters:

VzDeviceHandle Device: Device handle

VzSensorType sensorType: indicates the sensorType

VzSensorIntrinsicParameters * pSensorIntrinsicParameters: return sensor inside the lens

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.19 **VZ_GetSensorExtrinsicParameters**

Prototype:

VzReturnStatus VZ_GetSensorExtrinsicParameters(VzDeviceHandle device, VzSensorExtrinsicParameters* pSensorExtrinsicParameters)

Description:

Obtain the foreign parameter of the device

Parameters:

VzDeviceHandle Device: Device handle

VzSensorExtrinsicParameters * pSensorExtrinsicParameters: return outside the cords of the equipment

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.20 **VZ_GetFirmwareVersion**

Prototype:

VzReturnStatus VZ_GetFirmwareVersion(VzDeviceHandle device, char*

pFirmwareVersion, int length)

Description:

Obtain the firmware version of the device

Parameters:

VzDeviceHandle Device: Device handle

Char * pFirmwareVersion: indicates the firmware version of the device

Int length: indicates the byte length of the cache that pFirmwareVersion points to

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.21 **VZ_GetDeviceMACAddress**

Prototype:

VzReturnStatus VZ_GetDeviceMACAddress(VzDeviceHandle device, char*

pMACAddress)

Description:

Obtain the MAC address of the device

Parameters:

VzDeviceHandle Device: Device handle

Char * pMACAddress: Returns the MAC address of the device. By default, it is a string of 18 bytes ending in '\0'

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.22 **VZ_SetIRGMMGain**

Prototype:

VzReturnStatus VZ_SetIRGMMGain(VzDeviceHandle device, uint8_t gmmgain)

Description:

Set the digital gain of IR image

Parameters:

VzDeviceHandle Device: Device handle

Uint8_t GMmgain: IR gain value to set to the device

Return:

VzRetOK: The call is successful

Other values: Call failed

5.3.23 **VZ_GetIRGMMGain**

Prototype:

VzReturnStatus VZ_GetIRGMMGain(VzDeviceHandle device, uint8_t* pGmmgain)

Description:

The digital gain of IR image is obtained

Parameters:

VzDeviceHandle Device: Device handle

Uint8_t * pGmmgain: Returns the IR gain value of the device

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.24 **VZ_SetColorPixelFormat**

Prototype:

VzReturnStatus VZ_SetColorPixelFormat(VzDeviceHandle device,

VzPixelFormat pixelFormat)

Description:

Sets the pixel format of the color image

Parameters:

VzDeviceHandle Device: Device handle

VzPixelFormat pixelFormat: pixelFormat for the color image to set

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.25 **VZ_GetColorResolution**

Prototype:

VzReturnStatus VZ_GetColorResolution(VzDeviceHandle device, VzResolution* pResolution)

Description:

Gets the pixel format of a color image

Parameters:

VzDeviceHandle Device: Device handle

VzResolution* pResolution: Returns the pixel format of the color image

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.26 **VZ_SetFrameRate**

Prototype:

VzReturnStatus VZ_SetFrameRate(VzDeviceHandle device, int value)

Description:

Sets the device's image frame rate for both depth and color images. This interface is a synchronization interface, which takes about 500ms

Parameters:

VzDeviceHandle Device: Device handle

Int value: target frame rate to be set

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.27 **VZ_GetFrameRate**

Prototype:

VzReturnStatus VZ_GetFrameRate(VzDeviceHandle device, int* pValue)

Description:

Gets the image frame rate of the device

Parameters:

VzDeviceHandle Device: Device handle

Int * pValue: Returns the frame rate of the device image

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.28 **VZ_SetExposureControlMode**

Prototype:

VzReturnStatus VZ_SetExposureControlMode(VzDeviceHandle device,

VzSensorType sensorType, VzExposureControlMode controlMode)

Description:

Set the exposure mode of the sensor

Parameters:

VzDeviceHandle Device: Device handle

VzSensorType sensorType: sensorType for which the exposure mode is to be

set

VzExposureControlMode controlMode: Exposure mode to set

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.29 **VZ_GetExposureControlMode**

Prototype:

VzReturnStatus VZ_GetExposureControlMode(VzDeviceHandle device,

VzSensorType sensorType,   VzExposureControlMode* pControlMode)

Description:

Gets the exposure mode of the sensor

Parameters:

VzDeviceHandle Device: Device handle

VzSensorType sensorType: sensorType to obtain the exposure mode

VzExposureControlMode controlMode: Returns the sensor's exposure mode

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.30 **VZ_SetExposureTime**

Prototype:

VzReturnStatus VZ_SetExposureTime(VzDeviceHandle device, VzSensorType

sensorType, VzExposureTimeParams exposureTime)

Description:

Set the exposure time of the sensor

Depth sensor, only support manual exposure mode, set exposure time

Color sensor, support in automatic exposure mode, set the maximum exposure time; The exposure time can be set in manual exposure mode

Parameters:

VzDeviceHandle Device: Device handle

VzSensorType sensorType: sensorType to obtain the exposure time

VzExposureTimeParams exposureTime: exposureTime parameter to set

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.31 **VZ_GetExposureTime**

Prototype:

VzReturnStatus VZ_GetExposureTime(VzDeviceHandle device, VzSensorType sensorType, VzExposureTimeParams* pExposureTime)

Description:

Obtain the exposure time of the sensor

Depth sensor, support in manual exposure mode, access to exposure time;

The maximum exposure time can be obtained

Color sensor, support in manual exposure mode, access to exposure time;

The maximum exposure time can be obtained

Parameters:

VzDeviceHandle Device: Device handle

VzSensorType sensorType: sensorType to obtain the exposure time

VzExposureTimeParams* pExposureTime: Returns the acquired exposure time

parameter

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.32 **VZ_SetTimeFilterEnabled**

Prototype:

VzReturnStatus VZ_SetTimeFilterEnabled(VzDeviceHandle device, bool

bEnabled)

Description:

Set the time domain filter switch of depth image

Parameters:

VzDeviceHandle Device: Device handle

Bool bEnabled: True indicates that filtering is enabled. False indicates that

filtering is disabled

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.33 **VZ_GetTimeFilterEnabled**

Prototype:

VzReturnStatus VZ_GetTimeFilterEnabled(VzDeviceHandle device, bool

*pEnabled)

Description:

Obtain the time domain filter switch state of depth image

Parameters:

VzDeviceHandle Device: Device handle

Bool *pEnabled: Returns filtering switch status

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.34 **VZ_SetConfidenceFilterParams**

Prototype:

VzReturnStatus VZ_SetConfidenceFilterParams(VzDeviceHandle device,

VzConfidenceFilterParams params)

Description:

Set the confidence filtering parameter of depth image

VzDeviceHandle Device: Device handle

Bool *pEnabled: Returns filtering switch status

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.35 **VZ_GetConfidenceFilterParams**

Prototype:

VzReturnStatus VZ_GetConfidenceFilterParams(VzDeviceHandle device,

VzConfidenceFilterParams *pParams)

Description:

The confidence filter parameters of depth image are obtained

Parameters:

VzDeviceHandle Device: Device handle

Bool *pEnabled: Returns filtering switch status

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.36 **VZ_SetFlyingPixelFilterParams**

Prototype:

VzReturnStatus VZ_SetFlyingPixelFilterParams(VzDeviceHandle device, const

VzFlyingPixelFilterParams params)

Description:

Set the fly - point filter parameter of depth image

Parameters:

VzDeviceHandle Device: Device handle

Const VzFlyingPixelFilterParams params: filter parameters

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.37 **VZ_GetFlyingPixelFilterParams**

Prototype:

VzReturnStatus VZ_GetFlyingPixelFilterParams(VzDeviceHandle device,

VzFlyingPixelFilterParams* params)

Description:

To obtain the flying point filter parameters of depth image

Parameters:

VzDeviceHandle Device: Device handle

VzFlyingPixelFilterParams * params: filter parameters

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.38 **VZ_SetFillHoleFilterParams**

Prototype:

VzReturnStatus VZ_SetFillHoleFilterParams(VzDeviceHandle device, const

VzFillHoleFilterParams params)

Description:

Set the hole filling filter parameters of depth image

Parameters:

VzDeviceHandle Device: Device handle

Const VzFillHoleFilterParams Params: indicates the filtering parameter

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.39 **VZ_GetFillHoleFilterParams**

Prototype:

VzReturnStatus VZ_GetFillHoleFilterParams(VzDeviceHandle device,

VzFillHoleFilterParams* params)

Description:

The hole filling filter parameters of depth image are obtained

Parameters:

VzDeviceHandle Device: Device handle

VzFillHoleFilterParams* Params: indicates the obtained filtering parameter

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.40 **VZ_SetSpatialFilterParams**

Prototype:

VzReturnStatus VZ_SetSpatialFilterParams(VzDeviceHandle device, const

VzSpatialFilterParams params)

Description:

Set the spatial filtering parameters of the depth image

Parameters:

VzDeviceHandle Device: Device handle

Const VzSpatialFilterParams Params: indicates the filtering parameter

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.41 **VZ_GetSpatialFilterParams**

Prototype:

VzReturnStatus VZ_GetSpatialFilterParams(VzDeviceHandle device,

VzSpatialFilterParams* params)

Description:

Set the spatial filtering parameters of the depth image

Parameters:

VzDeviceHandle Device: Device handle

VzSpatialFilterParams* PARams: indicates the obtained filtering parameter

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.42 **VZ_SetTransformColorImgToDepthSensorEnabled**

Prototype:

VzReturnStatus

VZ_SetTransformColorImgToDepthSensorEnabled(VzDeviceHandle device,

bool bEnabled)

Description:

Switch to set color image alignment to depth camera space. Only devices with

color sensors support this operation. If turn on the switch, then call

VZ_GetFrameReady, VzFrameReady. TransformedColor value is 1, Then call

VZ_GetFrame can get VzTransformColorImgToDepthSensorFrame types of

color image, the image size is the same size and depth.

Parameters:

VzDeviceHandle Device: Device handle

Bool bEnabled: True turns alignment on, false turns alignment off

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.43 **VZ_GetTransformColorImgToDepthSensorEnabled**

Prototype:

VzReturnStatus

VZ_GetTransformColorImgToDepthSensorEnabled(VzDeviceHandle device,

bool *bEnabled)

Description:

Gets the on/off state for color image alignment to depth camera space

Parameters:

VzDeviceHandle Device: Device handle

Bool *bEnabled: Return switch status

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.44 **VZ_SetTransformDepthImgToColorSensorEnabled**

Prototype:

VzReturnStatus

VZ_SetTransformDepthImgToColorSensorEnabled(VzDeviceHandle device,

bool bEnabled)

Description:

Switch that sets the alignment of depth images to color camera space. Only

devices with color sensors support this operation. If turn on the switch, then call

VZ_GetFrameReady, VzFrameReady. TransformedDepth value is 1, Then call

VZ_GetFrame can get VzTransformDepthImgToColorSensorFrame type of

depth image, its size and color image size is the same.

Parameters:

VzDeviceHandle Device: Device handle

Bool bEnabled: True turns alignment on, false turns alignment off

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.45 **VZ_GetTransformDepthImgToColorSensorEnabled**

Prototype:

VzReturnStatus

VZ_GetTransformDepthImgToColorSensorEnabled(VzDeviceHandle device,

bool *bEnabled)

Description:

Gets the on/off state of the depth image alignment to color camera space

Parameters:

VzDeviceHandle Device: Device handle

Bool *bEnabled: Return switch status

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.46 **VZ_TransformedDepthPointToColorPoint**

Prototype:

VzReturnStatus VZ_TransformedDepthPointToColorPoint(const

VzDeviceHandle device, const VzDepthVector3 depthPoint,   const

VzVector2u16 colorSize, VzVector2u16* pPointInColor)

Description:

Align the points on the depth image to the color image space to obtain the

coordinates of the points corresponding to the passing depth image coordinate

points on the color image

Parameters:

VzDeviceHandle Device: Device handle

Const VzDepthVector3 depthPoint: Coordinate point of the depth image

Const VzVector2u16 colorSize: Color image size

VzVector2u16* pPointInColor: obtained color image coordinate points

corresponding to the depth image coordinate points

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.47 **VZ_ConvertDepthToPointCloud**

Prototype:

VzReturnStatus VZ_ConvertDepthToPointCloud(VzDeviceHandle device,

VzDepthVector3* pDepthVector,   VzVector3f* pWorldVector, int32_t pointCount,

VzSensorIntrinsicParameters* pSensorParam)

Description:

The incoming depth image coordinate point set is converted to the world

coordinate point set. The origin of world coordinates is in the center of the depth

sensor lens, and the Z-axis is perpendicular to the front cover of the device, and its positive direction is pointing away from the device. The X-axis points to the laser from the depth lens, and its positive direction points to the distance from the device; The Y-axis is perpendicular to the device pointing to the ground, and its positive direction is pointing away from the device.

Parameters:

VzDeviceHandle Device: Device handle

VzDepthVector3* pDepthVector: A collection of coordinate points of the depth image

VzVector3f* pWorldVector: The set of coordinate points of the transformed point cloud

Int32_t pointCount: indicates the number of coordinate points

VzSensorIntrinsicParameters * pSensorParam: sensor inside

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.48 **VZ_ConvertDepthFrameToPointCloudVector**

Prototype:

VzReturnStatus VZ_ConvertDepthFrameToPointCloudVector(VzDeviceHandle device, const VzFrame* pDepthFrame,   VzVector3f* pWorldVector)

Description:

Convert the depth image to a set of world frame points. The size of the converted world frame points is vzframe.width * vzframe.height, Support VzDepthFrame

and VzTransformDepthImgToColorSensorFrame images

Parameters:

VzDeviceHandle Device: Device handle

Const VzFrame* pDepthFrame: Depth image

VzVector3f* pWorldVector: The set of coordinate points of the transformed point cloud

Returns:

VzRetOK: The call is successful

Other values: Call failed

5.3.49 **VZ_SetHotPlugStatusCallback**

Prototype:

VzReturnStatus VZ_SetHotPlugStatusCallback(PtrHotPlugStatusCallback pCallback, const void* pUserData)

Description:

Example Set the callback function for hot swap status

Parameters:

PtrHotPlugStatusCallback pCallback: indicates the callback function

Const void* pUserData: user data; may be empty

Returns:

VzRetOK: The call is successful

Other values: Call failed

6 FAQ

6.1 Where is SDK log stored?

For Windows, default location: C:\Users\<user name>\AppData\Local\Vzense\Log

6.2 Nebula SDK cannot search the camera

Following lists need to be checked if Ethernet access cameras can't be searched.

1. Make sure good connection between the DUT and the host, and confirm if the network adapter of the host works well.

2. Make sure the DUT is under the same LAN as the host. If the DUT is set in non-DHCP mode, make sure the DUT's fixed IP is in the same internet segment as the host, like 192.168.1.X; If the DUT is set in DHCP mode, make sure the DUT is in the same LAN as the host, and the router/switch has DHCP server function.

3. Make sure the software's internet permission is not prohibited.

4. Make sure Internet UDP function is not prohibited by LAN safety policy.

5. Make sure LAN 9007, 9008, 9009 port is not prohibited

6. Make sure power supply is correct. If not POE approach, make sure power adapter is plugged.

If the above check items are OK, but the camera still cannot be opened, please contact

FAE。

Email: info@vzense.com