FaceRecognition Android SDK User Guide

Model： DCAM305

Version： V1.0.8

About This Guide

This guide is mainly to introduce how to use Vzense DCAM305 TOF RGBD Camera and Vzense Android SDK.

Document Structure

Chapter	Title	Contents
1	Overview	Introduce general information of Vzense products and Android SDK
2	Installation	Introduce how to install Vzense TOF Depth Camera and Android SDK
3	SDK Instruction	Introduce how to use Vzense TOF Depth Camera and Android SDK
4	API Introduction	Introduce the APIs of Vzense Android SDK

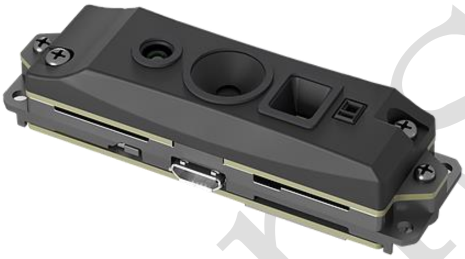
Release Records

Date	Version	Release Note
2019/07/24	V1.0.3	Release official version
2019/10/11	V1.0.4	Add upgrade and restart functions
2019/10/12	V1.0.5	Modify the upgrade API and SDK import method
2019/11/05	V1.0.6	Add 3.3.5; Add new API
2019/11/15	V1.0.7	Update related content about upgrade
2020/03/20	V1.0.8	Added upgrade instructions using SDK Sample APK

Contents

1. Overview

Vzense TOF RGBD Camera(DCAM305) is a 3D camera module 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 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.



⮙ Fig.1 Vzense TOF RGBD Camera: DCAM305

The Vzense android SDK is a development kit based on Vzense DCAM305 TOF RGBD Camera, which is currently applicable to single board or smart phone with android system. It provides a series of friendly APIs and simple application examples for developers.

Developers can get high precision depth image data, gray image 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. Installation

2.1. Recommended Development Environment

Item	Recommended Configuration
Development Environment	Android API 21 or above JDK1.7.0_01 or above
Running Environment	Android 5.0 or above ARMv7a/ARMv8a @ 1.4GHz+ 512M RAM USB 2.0(OTG capable)

2.2. Installation Instruction

Connect the camera module to Android development board or smartphone USB interface through USB cable, as Figure 2.

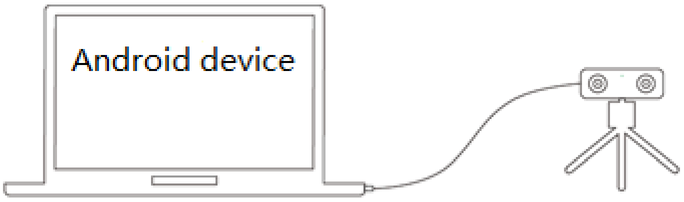
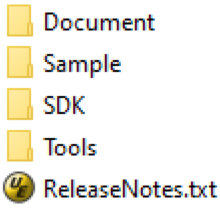


Fig. 2 Hardware Installation

3. SDK Instruction

3.1. SDK Structure

Vzense Android SDK contains SDK, Sample, APK installation package, user guide document, etc. The directory structure is as follows:



Document
Sample
SDK
Tools
ReleaseNotes.txt

Fig. 3 Android SDK directory

- **Document:** The document details the development and usage instructions of the SDK;

- **Samples:** contains sample project developed using Vzense Android SDK;

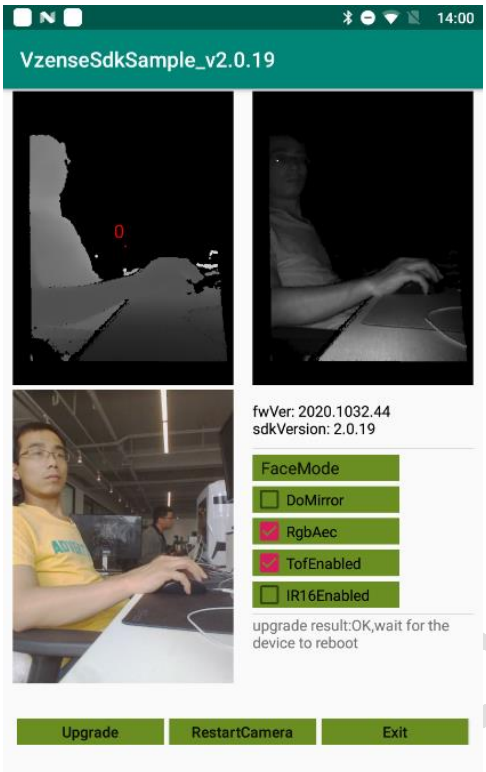
- **SDK:** contains Vzense Android SDK jar and so;

- **Tools:** contains Sample APK, functions testing tool, image quality testing tool;

- **ReleaseNotes.txt:** introduces the main contents of this version update;

3.2. Application Installation and View Window

Connect the Vzense camera to the USB interface of Android device, copying the APK file to the device, double-clicking the APK to install. After that run the application, an interface including image preview and menu buttons is launched as shown below. Sample and APK enter into face scan mode and display RGB image by default. You can click the setting menu on the interface to switch the display data and set different modes.



14:00

VzenseSdkSample_v2.0.19

fwVer: 2020.1032.44
sdkVersion: 2.0.19

□ DoMirror

☑ RgbAec

☑ TofEnabled

□ IR16Enabled

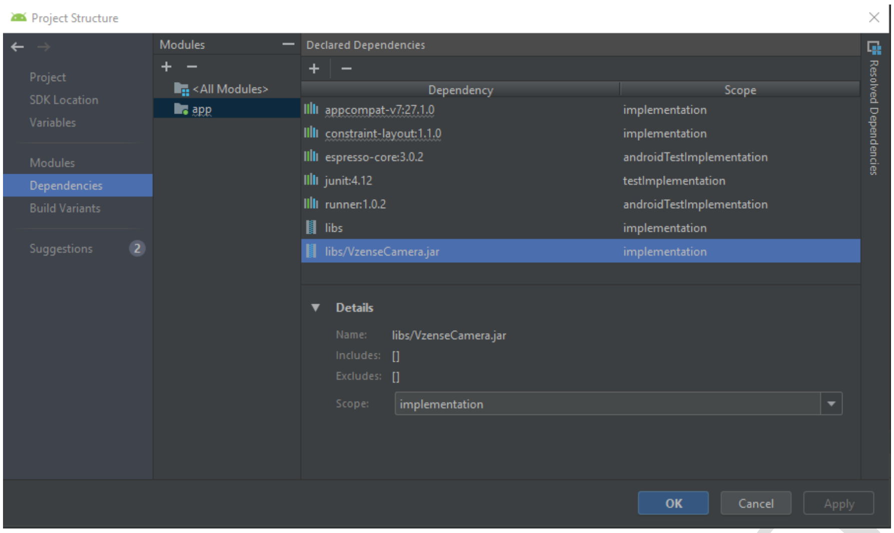
upgrade result:OK,wait for the device to reboot

⮝ Fig. 4 VzenseSdkSample.apk running interface

3.3. Development Process

3.3.1. Import Jar file

Create a new Android Studio project, copy the VzenseCamera.jar file to the app/libs directory, click File, select Project Structure in Lower Larry, and pop up the project component interface. As shown in the figure below, select the project under Modules, click the Dependencies menu on the right, and then click the '+' in the upper right corner. Select Jar Dependency in the pop-up menu, select VzenseCamera.jar.



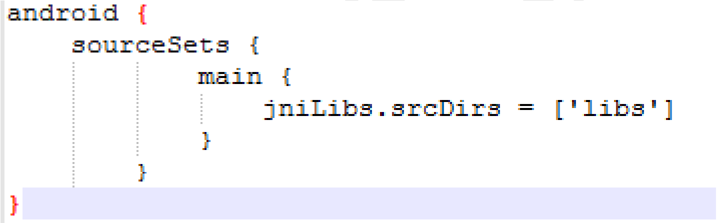
Dependency	Scope
appcompat-v7:27.1.0	implementation
constraint-layout:1.1.0	implementation
espresso-core:3.0.2	androidTestImplementation
junit:4.12	testImplementation
runner:1.0.2	androidTestImplementation
libs	implementation
libs/VzenseCamera.jar	implementation

Name: libs/VzenseCamera.jar
Includes: []
Excludes: []
Scope: implementation

➢ Fig.5 Import jar

3.3.2. Import so file

Copy the so file to the app / libs directory, open the build.gradle file of the project, add the following code, import so. After importing the library, the APIs can be invoked in the project for development, as shown in the following demo example:



```
android {
    sourceSets {
        main {
            jniLibs.srcDirs = ['libs']
        }
    }
}
```

➢ Fig.6 Import so

3.3.3. Interface Invoke

1． Import Interface class

import com.vzense.sdk.PsCamera;

2． Create PsCamera object and invoke *init* method

```
mVzenseCamera = new PsCamera();
if (mVzenseCamera != null) {
    mVzenseCamera.init(this, mOnVzenseCameraConnectLister);
}
```

3． Create image data callback

mFrameCallback = new FrameCallback();

4． Open camera, set working mode and frame callback

```
mVzenseCamera .setFrameCallback(mFrameCallback);
mVzenseCamera .setWorkMode(mWorkMode);// mWorkMode defaults to 1, face mode
mVzenseCamera .start(this);
```

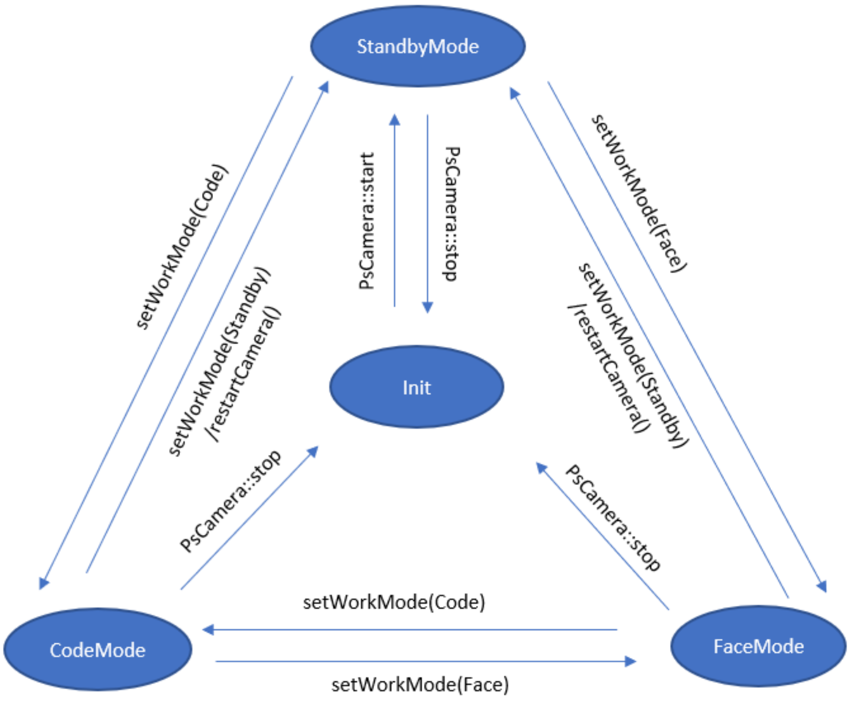
5． Get frame data in override callback function

```
public class FrameCallback implements IFrameCallback {
    @Override
    public void onFrame(PsFrame DepthFrame,PsFrame IrFrame,PsFrame RgbFrame) {
            //DataProcess
              }
}
```

6． After the device starts and connect normally, read Sn, fwVer and other operations in the device state callback

```
     @Override
     public void onConnect() {
     if (DEBUG) Log.i(TAG, "onConnect");
     if(mVzenseCamera != null) {
         String sn = mVzenseCamera .getSn();
         String fwVer = mVzenseCamera .getFWVerion();
         String hwVer = mVzenseCamera .getHWVerion();
         String sdkVersion = mVzenseCamera .getSDKVerion();
         String deviceName = mVzenseCamera .getDeviceName();
}
```

3.3.4. Description of working mode switching process



⮝ Fig.7 Working mode switching flowchart

- ➢ Launch the application and enter into FaceMode by default through *setWorkMode(Face)*, in which output Depth & IR & RGB and single RGB images alternately by default, Depth, IR and RGB images' resolution is 480x640, Depth and IR's frame rate is 15Hz and RGB images is 30hz. The Depth and IR obtained by the application from SDK are the mapped images with RGB. If the image is abnormal, *resetCamera()* can be called to reset the camera and the working mode can be changed to Standby mode.

- ➢ Switch to CodeMode through *setWorkMode(Code)*. In this mode, TOF is turned off by default, and only RGB images are available. The resolution of RGB images is 1080x1920, which can be switched to other resolutions through setRgbResolution API, and TOF image data can also be turned on or off through *setTofFrameEnabled* API. If the image is abnormal, *resetCamera()* can be called to reset the camera and the working mode can be changed to Standby mode.

- Call *setWorkMode(Standby)* to switch into Standby mode. In this mode, TOF and RGB are turned off by default, and images can only be obtained by switching back to face or code mode through *setWorkMode* API.

- The above three working modes can be switched in real time through *setWorkMode* API. In non-standby mode, RGB image resolution and TOF image data can be switched in real time through corresponding APIs.

- In face mode and code mode, image mirroring and TOF/RGB auto exposure can be turned on or off. Image mirroring is turned off by default, and you can switch the mirroring feature on and off through the *setImageMirror* API. Automatic exposure for TOF and RGB is enabled by default and can be turned on or off via *setRgbAecEnabled* and *setTOFAecEnabled* APIs. When automatic exposure is turn off, the value of RGB exposure duration and gain can be manually set by *setRgbExposureTimeAndGain* API.

3.3.5. restartCamera API Use Process

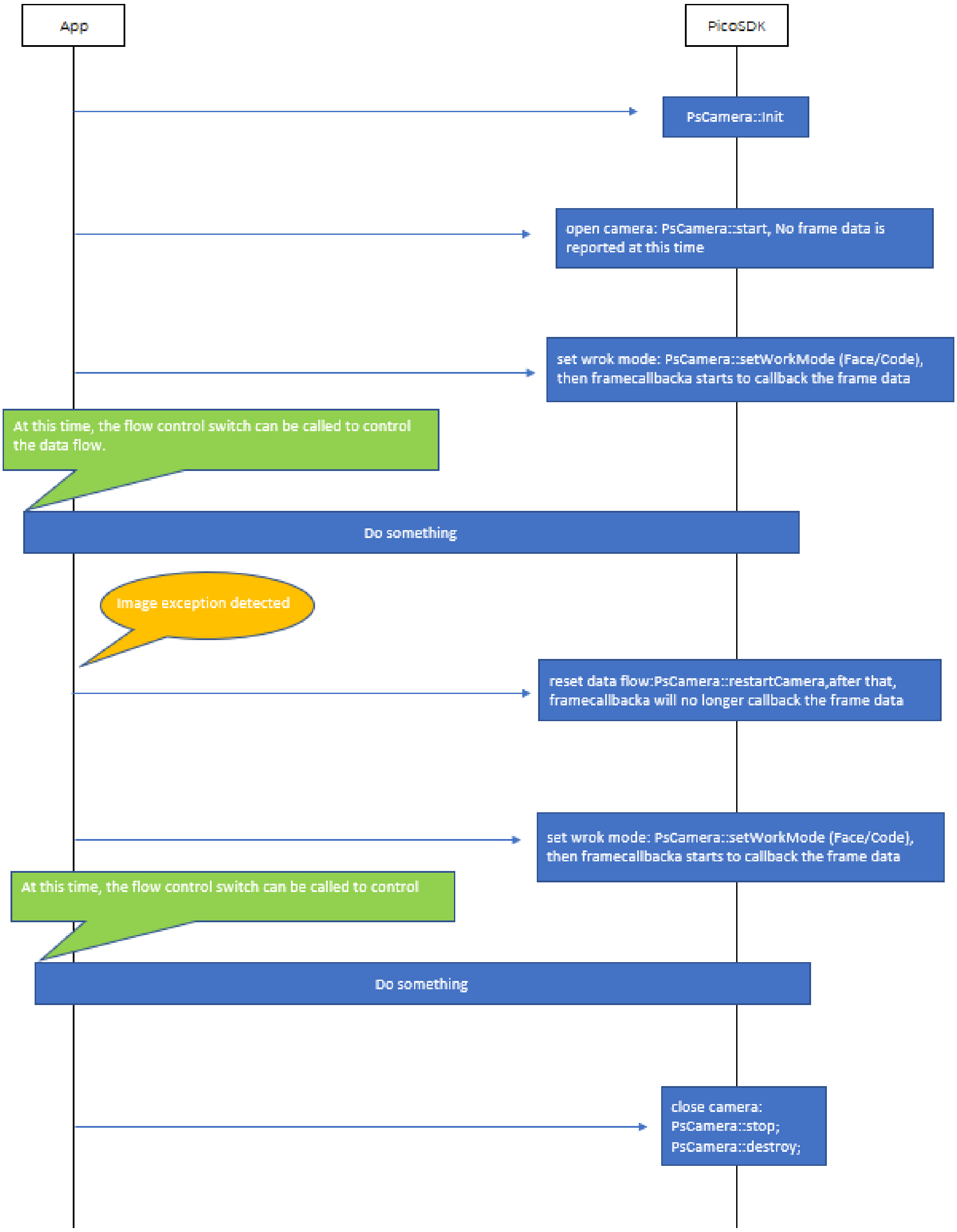
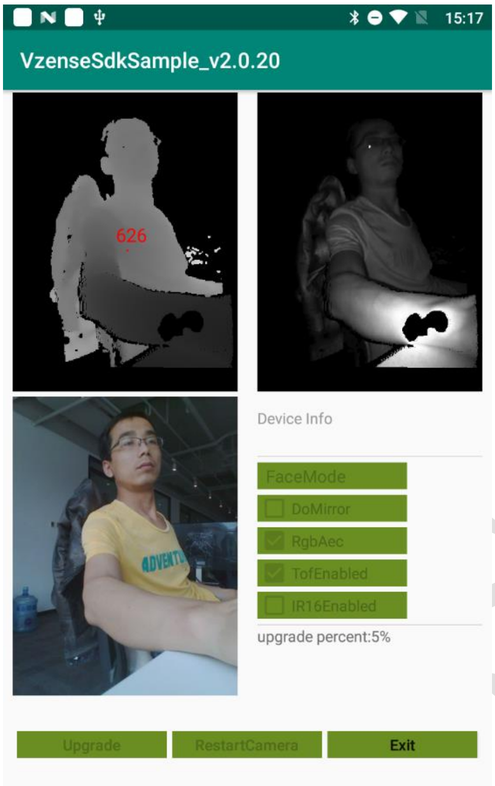

App
PicoSDK
PsCamera::Init
open camera: PsCamera::start, No frame data is reported at this time
set wrok mode: PsCamera::setWorkMode (Face/Code), then framecallbacka starts to callback the frame data
At this time, the flow control switch can be called to control the data flow.
Do something
Image exception detected
reset data flow:PsCamera::restartCamera,after that, framecallbacka will no longer callback the frame data
set wrok mode: PsCamera::setWorkMode (Face/Code), then framecallbacka starts to callback the frame data
At this time, the flow control switch can be called to control
Do something
close camera:
PsCamera::stop;
PsCamera::destroy;

Fig.8 restartCamera API invoke process

3.4. Instructions for upgrading firmware using SDK Sample APK

Copy new firmware to Android device's sdcard root directory and Rename it to **Firmware.img**. Run SDK Sample, click the [Upgrade] button after the image is normally displayed, and then wait for the upgrade to complete. The camera may restart several times during the upgrade. Please do not manually unplug the camera. During the upgrade process, the progress of upgrade is displayed.



VzenseSdkSample_v2.0.20

Device Info

FaceMode

☐ DoMirror

☑ RgbAec

☑ TofEnabled

☐ IR16Enabled

upgrade percent:5%

⮝ Fig.9 Camera firmware upgrade

4. SDK API Introduction

4.1. Enum type

4.1.1. FrameType

Description:

Image data stream type

Enumerator:

- **DepthFrame**: 16bit depth image frame
- **IRFrame**: 16bit IR gray image frame
- **RGBFrame**: 24bit 3 channels RGB image frame

4.1.2. PixelFormat

Description:

Pixel type of image

Enumerator:

- **PixelFormatDepthMM16:** per pixel is a 16-bit depth value in millimeters
- **PixelFormatGray16:** per pixel is a 16-bit gray value
- **PixelFormatGray8:** per pixel is an 8-bit gray value
- **PixelFormatRGB888**: per pixel is a 24-bit RGB value
- **PixelFormatBGR888:** per pixel is a 24-bit BGR value
- **PixelFormatRGBA8888:** per pixel is a 32-bit RGBA value

4.2. Class

4.2.1. CameraParameter

Description:

Camera intrinsic and distortion parameters

Members:

Parameter	Description
fx, fy, cx, cy	Camera intrinsic parameters
k1, k2, k3, p1, p2	Camera distortion parameters

4.2.2. CameraExtrinsicParameter

Description:

Camera extrinsic parameters

Members:

Parameter	Description
rotation[1-9]	Rotation matrix from TOF camera to RGB camera
translation[1-3]	Translation matrix from TOF camera to RGB camera
e[1-9]	Essential matrix
f[1-9]	Fundamental matrix

4.2.3. PsFrame

Description:

Image information

Members:

Parameter	Description
frameIndex	Frame index
frameType	Type of frame
pixelFormat	Pixel format
frameData	Frame data
dataLength	Length of data(bytes)
timeStamp	Time stamp(ms)
fps	Frame rate
width	Image width in pixel
height	Image height in pixel
bytePerPixel	Bytes per pixel

4.2.4. IFrameCallback

Description:

Image callback interface

Instruction:

The application layer needs to create an interface object, which is set to native through the *setFrameCallback* interface. Native callbacks data to the application layer through the *OnFrame* method of the interface class.

4.2.5. IUpgradeStatusCallback

Description:

Upgrade status callback interface

Instruction:

The application layer needs to create an interface object, which is set to native through the *setUpgradeStatusCallback* interface. Native callbacks status to the application layer through the *OnUpgradeStatus* method of the interface class.

API	void onUpgradeStatus(int stage, int params)
Description	stage: upgrade status params: upgrade status result. -1 is failure ➢ DEVICE_PRE_UPGRADE_IMG_COPY: copy Firmware.img to Camera ➢ DEVICE_UPGRADE_IMG_CHECK_DOING: check Firmware.img ➢ DEVICE_UPGRADE_IMG_CHECK_DONE: check Firmware.img finished ➢ DEVICE_UPGRADE_UPGRAD_DOING: upgrading firmware. params now represents the percentage of progress (0~100) ➢ DEVICE_UPGRADE_RECHECK_DOING: recheck after upgrade ➢ DEVICE_UPGRADE_RECHECK_DONE: recheck finished ➢ DEVICE_UPGRADE_UPGRAD_DONE: upgrade finished

4.2.6. PsCamera

Description:

Interface class, through which users can open, close, set parameters, obtain data and other operations.

Instruction:

API	void init(Context context, final OnVzenseCameraConnectListener listener)
Description	SDK init, this interface must be called first at startup, and listen is the callback of device status. If you want to get the device connection status, you need to pass in this callback, or set it to null if you don't need to get.

API	void destroy()
Description	Release SDK resource, call it after *stop*().

API	void start(Context context)
Description	Start to capture image data, call it after *init*().

API	void stop()
Description	Stop capturing image data.

API	void setFrameCallback(final IFrameCallback callback)
Description	Set frame callback to get image data, call it before *start*().

API	int setGmmGain(int gmmGain)
Description	Set GmmGain value to adjust the brightness of IR image **Parameter:** gmmGain: gmmGain value, value range is 0-4095.

	Return: ➢ 0: Success ➢ Others: Failure

API	int getGmmGain()
Description	Get the current gmmgain value of IR image **Return:** Current gmmgain value of IR image

API	int setRgbResolution(int resolutionIndex)
Description	Set the RGB image resolution, and the parameter value range is 0-3 **Parameter:** resolutionIndex: ➢ 0: 1080x1920 ➢ 1: 720x1280 ➢ 2: 480x640 ➢ 3: 360x640 **Return:** ➢ 0: Success ➢ Others: Failure

API	void getDepthCameraParameter(CameraParameter mDepthParameter)
Description	Get the intrinsic parameters of TOF camera. For details refer to 4.2.1

API	void getRgbCameraParameter(CameraParameter mRgbParameter)

Description	Get the intrinsic parameters of RGB camera. For details refer to 4.2.1

API	void getCameraExtrinsicParameter(CameraExtrinsicParameter mExtrinsicParameter)
Description	Get the extrinsic parameters of camera. For details refer to 4.2.2

API	String getSn()
Description	Get SN of device, such as PD3051AGD5130013M

API	String getFWVersion()
Description	Get the firmware version number of device, such as 2019.0518.02

API	String getHWVersion()
Description	Get hardware version number of device, such as R2

API	String getSDKVersion()
Description	Get SDK version number, such as 2.0.20

API	String getDeviceName()
Description	Get device name, such as Vzense RGBD DCAM305

API	int setWorkMode(int workMode)
Description	Set the working mode. There are three working modes that can be set to 1-3. **Parameters:** workMode:

	➢ 1: Face mode. In this mode, TOF data is 15fps, RGB data is 30fps, TOF and RGB resolutions are 480*640. TOF data can be turned on or off by *setTofFrameEnabled*. TOF images are synchronized and aligned with RGB images. ➢ 2: Code mode. In this mode, there is only RGB data by default, and the resolution of RGB image is 480*640. Users can turn on or off TOF data through *setTofFrameEnabled*. If TOF is turned on, the TOF image is synchronized and aligned with the RGB image. ➢ 3: Standy mode. In this mode, both TOF and RGB are turned off by default, and images can only be obtained by switching back to face or code mode through *setWorkMode.* **Return:** ➢ 0: Success ➢ Others: Failure

API	int setRgbAecEnabled(boolean bEnabled)
Description	Set whether to turn on automatic exposure for RGB cameras, which defaults to true. **Parameter:** bEnabled: ➢ True: Turn on automatic exposure for RGB camera ➢ False: Turn off automatic exposure for RGB camera **Return:** ➢ 0: Success ➢ Others: Failure

API	int setTofFrameEnabled(boolean bEnabled)
Description	Set whether to obtain TOF image data. The TOF is turned off by default in code mode. If you want to obtain TOF data, you need to call this API to set TOF status to true. **Parameter:** bEnabled: ➢ True: Open TOF data ➢ False: Close TOF data **Return:** ➢ 0: Success ➢ Others: Failure

API	int setImageMirror(int mirrorValue)
Description	Set image mirroring **Parameter:** mirrorValue: ➢ 0: No mirroring ➢ 1: Left and right mirroring ➢ 2: Up and down mirroring ➢ 3: Up, down, left and right mirroring(Rotate 180°) **Return:** ➢ 0: Success ➢ Others: Failure

API	int setRgbExposureTimeAndGain(float exposureTime, float gain)
Description	Set the exposure duration and Gain value of RGB image. *setRgbAecEnabled* needs to be called to turn off automatic exposure before calling this API.

	Parameter: ➢ exposureTime : RGB Image exposure duration, value range is [0.0015-0.03], unit is second. ➢ gain： Gain value of RGB image, value range is [1.0-15.5] **Return:** ➢ 0: Success ➢ Others: Failure

API	int restartCamera()
Description	Restart Camera **Parameter:** Null **Return:** ➢ 0: Success ➢ Others: Failure

API	int StartUpgradeFirmWare(String imagePath)
Description	Start Camera upgrade **Parameter:** imagePath： The storage path of image file to be upgraded. **Return:** ➢ 0: Success ➢ 1: build version is the same and can be upgraded ➢ -1: Do not repeat call during upgrade process ➢ -2: Firmware check failed ➢ -3: Firmware version is too low to support upgrades

API	int setRgbFrameEnabled(boolean bEnabled)
Description	Sets whether to get RGB image data. The Setting in Standby mode is invalid. **Parameter:** bEnabled: ➢ True: Open RGB data ➢ False: Close RGB data **Return:** ➢ 0: Success ➢ Others: Failure

API	int *setFramePixelFormat*(PsFrame.FrameType type, PsFrame.PixelFormat format)
Description	Set pixel format of image frame **Parameter:** type: image frame type. Now only support IRFrame format: pixel format. Now only support PixelFormatGray16, PixelFormatGray8 **Return:** ➢ 0: Success ➢ Others: Failure