PicoZense TOF RGBD Camera

Android SDK User Guide

Android

2019.3

Pico Technology Co., Ltd.

About This Document

This guide is mainly to introduce how to use PicoZense TOF RGBD Camera and the PicoZense Android SDK.

Document Structure

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

Release Record

Date	Version	Instruction
2019.3.15	V1.3.6	First release

Contents

1 Overview

PicoZense TOF RGBD Camera（DCAM710） is a 3D camera module developed by Pico 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.

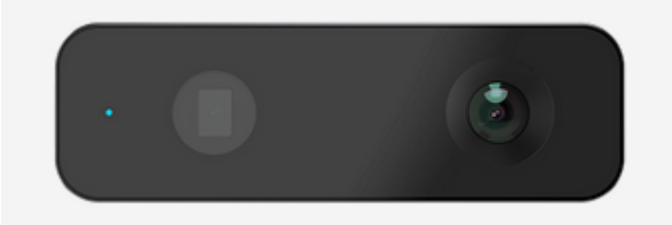

图 1.1 PicoZense TOF RGBD Camera：DCAM710

The PicoZense android SDK is a development kit based on PicoZense 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 Android-21 and above JDK: jdk1.7.0_01 and above
Running Environment	Android OS 5.0 and above ARMv7a/ARMv8a @ 1.8ghz+ 512M RAM USB 2.0 host support (OTG capable)

2.2 Installation Instruction

2.2.1 Hardware Installation

Connect the camera module to phone or board USB interface through USB cable, as Figure 2.1.

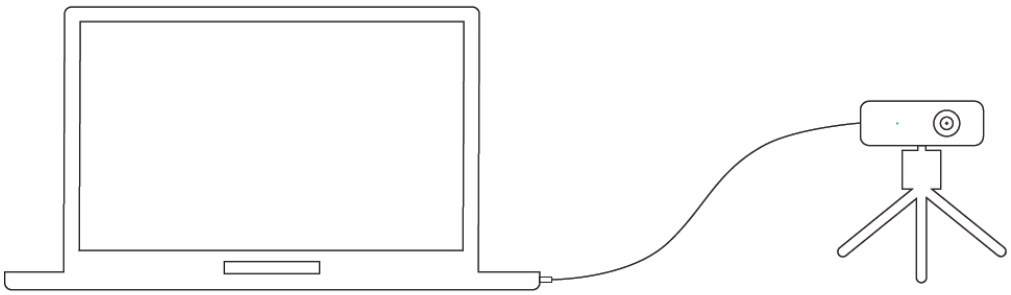

Figure 2.1 Hardware Installation

3 SDK Instruction

3.1 SDK Structure

PicoZense Android SDK contains several directories, including SDK, Sample, apk, ReleaseNotes.txt, user guide document. The directory structure is as follows：

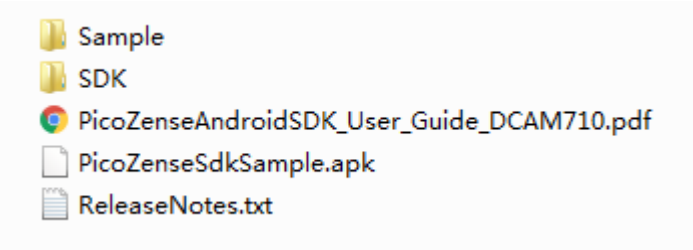

Figure 3.1 Android SDK directory

The SDK directory mainly contains PicoZense Android SDK jar and so.

Samples directory contains PicoZense Android SDK sample project.

APK can be installed directly into phone, and run it.

ReleaseNotes.txt introduces the main contents of this version update.

PDF describes development instructions on android system.

3.2 Application Installation and View Window

Connect the PicoZense 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. As shown in the following figure, the Depth image and the depth value of the center point will be displayed by default. You can click Spinner on

the interface to switch display data and set various modes.

Figure 3.2 Running PicoZenseSdkSample.apk

3.3 Development Process

3.3.1 Import Jar

Create a new Android Studio project, copy the Picozense.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 Picozense.jar.

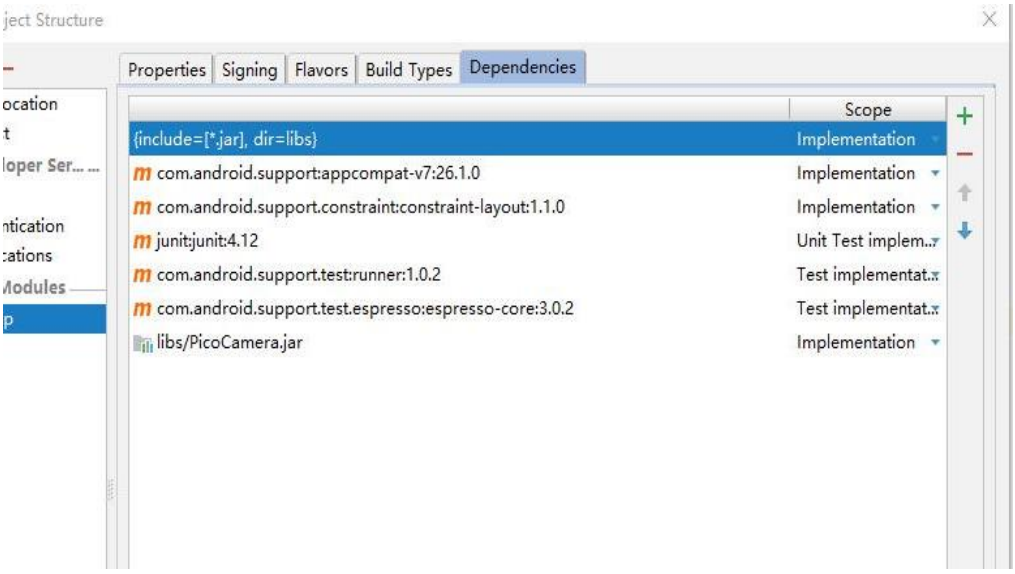

Figure 3.3 import jar

3.3.2 Import so

Copy the so to the app / libs directory, open the build.gradle file of the project, add the following code, import so.

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

Figure 3.4 import so

3.3.3 Interface Invoke

1. Import Interface class

```
import com.picozense.sdk.PsCamera;
```

2. Create *PsCamera* object and invoke *init* method

```
mPicoCamera = new PsCamera();
if (mPicoCamera != null) {
    mPicoCamera.init(this);
}
```

3. Create callback

```
mFrameCallback = new FrameCallback();
```

4. Set frame callback and other parameters. After that, invoke *start* method

```
mPicoCamera.setFrameCallback(mFrameCallback);
mPicoCamera.setBGThresdhold(20);
mPicoCamera.setDepthRange(0);
dataType = DataType.DATA_TYPE_DEPTH_RGB_30;
mPicoCamera.setDataType(dataType.ordinal());
mPicoCamera.setRgbResolution(2);
mPicoCamera.start(this);
```

5. Get Depth/Ir/Rgb frames in callback function *onFrame*

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

```
                //DataProcess
            }
        }
```

4 SDK API Introduction

4.1 Enum Type

4.1.1 DepthRange

Description：

Depth Range mode

Enumerator：

- **NearRange**: Near Range mode, Range0
- **MidRange**: Middle Range mode, Range1
- **FarRange**: Far Range mode, Range2
- **XNearRange**: XNear range mode, Range3
- **XMidRange**: XMid range mode, Range4
- **XFarRange**: XFar range mode, Range5
- **XXNearRange**: XXNear range mode, Range6
- **XXMidRange**: XXMiddle range mode, Range7
- **XXFarRange**: XXFar range mode, Range8

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

4.1.2 DateType

Description：

Data Type setting, determine which frame output from device and frame fps

Enumerator：

- **DATA_TYPE_DEPTH_30**：Output single Depth frames in 30fps

  Supported resolution:

  Depth ：640*480

- **DATA_TYPE_IR_30**：Output single IR frames in 30fps

  Supported resolution:

  IR：640*480

- **DATA_TYPE_RGB_30**：Output single RGB frames in 30fps

  Supported resolution:

  RGB ：1920*1080/1280*720/640*480/640*360

  Resolution of RGB can be set by *setRgbResolution* api

- **DATA_TYPE_DEPTH_RGB_30**: Output both Depth and RGB frames in 30fps.

  Supported resolution:

  Depth ：640*480

  RGB ：1920*1080/1280*720/640*480/640*360

  Resolution of RGB can be set by *setRgbResolution* api

- **DATA_TYPE_IR_RGB_30**: Output both IR and RGB frames in 30fps.

Supported resolution:

IR : 640*480

RGB : 1920*1080/1280*720/640*480/640*360

Resolution of RGB can be set by setRgbResolution api

- **DATA_TYPE_DEPTH_IR_30** : Output both Depth and IR frames in 30fps.

Supported resolution:

Depth : 640*480

IR : 640*480

- **DATA_TYPE_DEPTH_60** : Output single Depth frame in 60fps.

Supported resolution:

Depth : 640*360

- **DATA_TYPE_IR_60** : Output single IR frame in 60fps.

Supported resolution:

IR : 640*360

- **DATA_TYPE_DEPTH_IR_RGB_30** : Output Depth/IR/RGB frames in 30fps

Supported resolution:

Depth : 640*360

IR : 640*360

RGB : 1920*1080/1280*720/640*480/640*360

Resolution of RGB can be set by *setRgbResolution* api

- **DATA_TYPE_DEPTH_IR_15_RGB_30**：Output Depth/IR frames in 15fps and RGB frames in 30fps

  Supported resolution:

  Depth ：640*480

  IR ：640*480

  RGB ：1920*1080/1280*720/640*480/640*360

  Resolution of RGB can be set by *setRgbResolution* api

4.1.3 FrameType

Description：

Specific image frame type

Enumerator：

- **DepthFrame**: depth image frame
- **IRFrame**: IR gray image frame
- **RGBFrame**: RGB image frame

4.1.4 PixelFormat

Description：

Specific image pixel type

Enumerator：

- **PixelFormatDepthMM16**: Data of each pixel is 16bit depth value (in millimeter).
- **PixelFormatGray16**: Data of each pixel is 16bit gray value
- **PixelFormatGray8**: Data of each pixel is 8bit gray value
- **PixelFormatRGB888:** Data of each pixel is 24bit RGB value
- **PixelFormatBGR888:** Data of each pixel is 24bit BGR value

- **PixelFormatRGBA8888**: Data of each pixel is 32bit RGBA value

4.2 Main Class

4.2.1 CameraParameter

Description：

Parameters of camera

Members：

Parameters	Instruction
fx，fy，cx，cy	Camera intrinsic parameters
k1，k2，k3，p1，p2	Camera distortion parameters

4.2.2 CameraExtrinsicParameter

Description：

Camera extrinsic parameters

Members：

Parameters	Instruction
rotation[1-9]	Rotation matrix from TOF camera to RGB camera
translation[1-3]	Translation vector from TOF camera to

	RGB camera
e[1-9]	Output essential matrix
f[1-9]	Output fundamental matrix

4.2.3 PsFrame

Description：

The image information

Members：

Parameters	Instruction
frameIndex	frame index
frameType	type of frame
pixelFormat	pixel format
frameData	frame data
dataLength	length of data
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 PsCamera

Description：

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

Instruction:

API	void init(Context context)
Instruction	SDK init

API	void destroy()
Instruction	Release SDK resource

API	void start(Context context)
Instruction	Start image acquisition and transfer

API	void stop()
Instruction	Stop image acquisition and transfer

API	void setFrameCallback(final IFrameCallback callback)
Instruction	Set frame callback to get image data

API	void setDepthRange(int depthRange)
Instruction	Set value of depth range

API	int getDepthRange()

Instruction	Get current value of depth range

API	void setGmmGain(int gmmGain)
Instruction	Set value of GmmGain

API	int getGmmGain()
Instruction	Get current value of GmmGain

API	void setBGThresdhold(int threshold)
Instruction	Set value of filter threshold

API	int getBGThresdhold()
Instruction	Get current value of filter threshold

API	void setRgbResolution(int resolutionIndex)
Instruction	Set resolution of rgb

API	void setDataType(int dataType)
Instruction	Set value of data type

API	int getPulseCount()
Instruction	Get current value of pulse count

API	void getDepthCameraParameter(CameraParameter mDepthParameter)
Instruction	Get internal parameters of TOF

API	void getRgbCameraParameter(CameraParameter mRgbParameter)
Instruction	Get intenal parameters of RGB

API	void getCameraExtrinsicParameter(CameraExtrinsicParameter mExtrinsicParameter)

Instruction	Get external parameters of camera

API	void setDepthUndistortionEnabled(boolean bEnabled)
Instruction	Set do depth undistort or not

API	void setRgbUndistortionEnabled(boolean bEnabled)
Instruction	Set do rgb undistort or not

API	void setIrUndistortionEnabled(boolean bEnabled)
Instruction	Set do ir undistort or not

API	void setFilterEnabled(boolean bEnabled)
Instruction	Set do depth filter or not

API	void setTimeFilterEnabled(boolean bEnabled)
Instruction	Set do depth time filter or not

API	void setRemoveEdgeEnabled(boolean bEnabled)
Instruction	Set do depth remove edge or not

API	void setMapperEnabledDepthToRGB(boolean bEnabled)
Instruction	Set map depth to rgb or not

API	void setMapperEnabledRGBToDepth(boolean bEnabled)
Instruction	Set map rgb to depth or not

API	void setMapperEnabledRGBToIR(boolean bEnabled)
Instruction	Set map rgb to ir or not