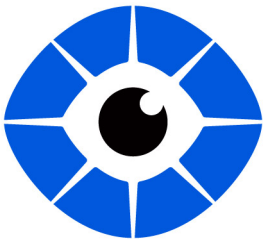

POINT SPREAD

Okulo Software Developer's Guide

Release 1.0

Point Spread Technology Co., Ltd.

Sep 28, 2022

TABLE OF CONTENTS

CHAPTER

ONE

INTRODUCTION

This document is to help developers interact with Okulo Camera using the provided Software Development Kit (SDK). The SDK provides cross-platform APIs to both the Linux and Windows operating systems. This document will first introduce the main concept of the SDK's framework and its exported classes and functions to developers, and then use several examples to illustrate the usage of the SDK. Furthermore, the SDK provides a Python wrapper based on the C/C++-based APIs for ease of programming. By reading this document, developers are expected to be able to control the camera and acquire the RGB-D data to advance their business goals.

The remaining sections of this document will be organized as follows:

- Section 2 guides you to download and install the SDK package.
- Section 3 elaborates on the details of the SDK's framework.
- Section 4 elaborates the classes as well as functions for software developing.
- Section 5 presents multiple projects as examples to interact with the camera and acquire its streaming data.
- Section 6 introduces the python wrappers of the SDK.
- Section 7 lists the License Agreement on the SDK.
- Section 8 introduces the Point Spread company that produces Okulo ™ camera series.

CHAPTER

TWO

SDK INSTALLATION

The Okulo ™ Software Development Kit manufactured by Point Spread must be installed on a computer to interact with an Okulo ™ camera.

2.1 Download SDK

Please download Okulo ™ SDK with the `git` tool:

- from GitHub:

```
>>> git clone https://github.com/point-spread/OkuloSdk.git
>>> cd OkuloSdk/
```

- from Gitee:

```
>>> git clone https://gitee.com/pointspread_0/OkuloSdk.git
>>> cd OkuloSdk/
```

The `OkuloSdk` repository contains two branches, `main` and `windows`, with the `main` branch for Linux operating system and the `windows` branch for Windows operating system. The default branch is `main`, and you can switch to `windows` or back to `main` as below:

```
# navigate into the OkuloSdk folder
>>> cd OkuloSdk
# switch to main/Linux
>>> git checkout main
# switch to Windows
>>> git checkout windows
```

2.2 Install SDK

2.2.1 Linux

- System Requirements:

  It is recommended to use the latest Ubuntu 20.04 LTS system or at least Ubuntu 18.04 system.

- Installation:

```
>>> git checkout master
>>> chmod +x ./install.sh
>>> sudo ./install.sh
>>> source ~/.bashrc
```

- The bash script `install.sh` mainly do the following things:
  1. Install the camera driver provided by FTDI
  2. Set the SDK environment variable $DYVCAM_GENTL64_PATH
  3. Install `libglfw3-dev`, `libglm-dev`
  4. Install `OpenCV` development library

2.2.2 Windows

- System Requirements:

  It is recommended to use the latest Microsoft Windows 10 Operating System.

- Installation:
  - git checkout windows branch
  - Extract `camdriver/FTD3XXDriver_WHQLCertified_v1.3.0.4_Installer.exe.zip` and then execute the program
  - Double click `install.bat`
  - Add `%DYVCAM_GENTL64_PATH%` to the environment variable path

2.3 Folder Structure

There are multiple folders within the Okulo ™ SDK:

- SDKbin:

  This folder contains the executable binary files including `getCameraParam`, `okulo_viewer` and `streamShow`.
  - `okulo_viewer` is a basic GUI interface for you to have a quick look at the captured 3D video stream;
  - `getCameraParam` is a tool that helps to retrieve the parameters located in the camera such as intrinsic and extrinsic parameters;
  - `streamShow` is a demo application to grab the RGB, ToF and PCL (PointCloud) stream data.

- SDKcode:

  This folder contains the header and source files for the Okulo ™ Cross Platform API. The developers can refer to the API classes, functions and macros to develop their software.

- SDKlib:

  This folder contains the Okulo dynamic libraries, third-party libraries from FTDI and GenICam, as well as some `.xml` files required by APIs.

- SDKpythonLib:

This folder contains the dynamic libraries that wrap the Okulo API classes and functions to a Python class and function, thus helping developers to program with Python rather than C/C++. The `okulo_test.py` is a quick python script for users to test connectivity to the camera.

- camdriver:

  This folder contains the driver software for the camera, which should be installed on the development computer.

- example:

  The OkuloSdk provides two sample codes for developers as references on how to call the API functions to get video data.

  - `getCameraParam`: this project helps developers to get the camera's intrinsics and extrinsics.
  - `streamShow`: this project helps developers on how to get the RGB stream, ToF stream as well as PCL stream data.

- license:

  Some third-party dependencies' license files are redistributed in our SDK, including GenICam, GLFW, OpenCV, and Pybind11.

CHAPTER

THREE

SDK FRAMEWORK

In this chapter, we will build up some concepts to help understand the SDK's framework.

3.1 System

We refer to a computer with an Okulo ™ camera as a system.

3.1.1 Host

A host or host computer is a physical computer with Linux or Windows operating system.

3.1.2 Device

A device is a physical Okulo ™ camera produced by Point Spread.

3.2 Physical and Logical Perspectives

3.2.1 Physical Device

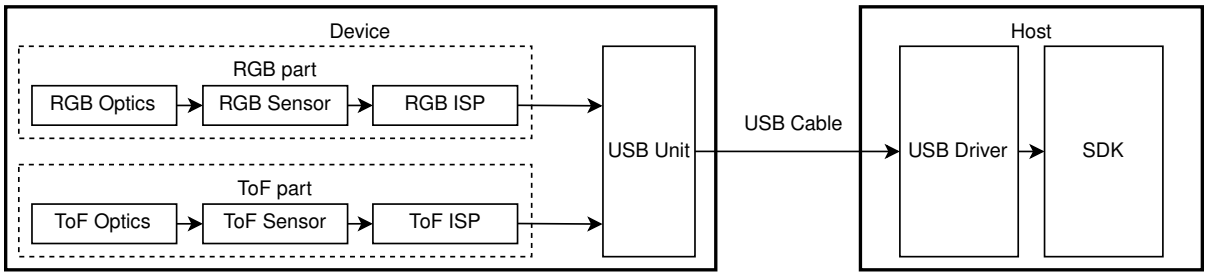

Fig. 3.1: Physical view of an Okulo ™ camera device

Fig. 3.1 illustrates the physical view of an Okulo ™ camera device. The device consists of an RGB part, a ToF part, as well as a USB unit. Specifically, the RGB part has its optics, sensor and image signal processing (ISP) units. In the ToF part, similar units also exist which are referred to as ToF optics, ToF sensor, and ToF ISP correspondingly. For both the RGB part and ToF part, when an image capturing operation is triggered, the light will be captured by optics and turned into a digital signal by sensors. The digital signal will be further processed by the ISPs. Finally, the RGB

and ToF data will be transmitted by the USB unit to the host computer and can be further accessed and processed by the SDK.

3.2.2 Logical Pipeline

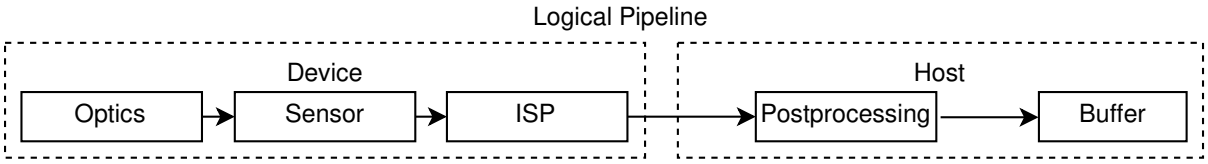

Fig. 3.2: The logical pipeline of the dataflow from device to host

We aim to avoid elaborating on the too-many hardware details and provide a rather simple, abstractive, and unified interface for software developers. Here, we use the term *logical pipeline* to summarize the data flow from the camera device to the host computer. A logical pipeline is composed of a set of optics, sensors, ISPs, postprocessing modules and data buffers. During video streaming, the pipeline will continuously fill up a buffer for the lateral modules to use. Moreover, developers can invoke unified functions to control the pipeline to obtain desired image data, without taking care of the hardware implementations of those functions. With the given Okulo ™ camera device, one can establish multiple kinds of logical pipelines to accomplish various tasks. We list the different use-cases below:

- **RGB Pipeline**

  One may only want to grab the RGB color image to do classical 2D image/video processing. In this case, the established pipeline will only be composed of the RGB stream and some RGB postprocessing modules, while the ToF parts are not utilized. See Fig. 3.3.

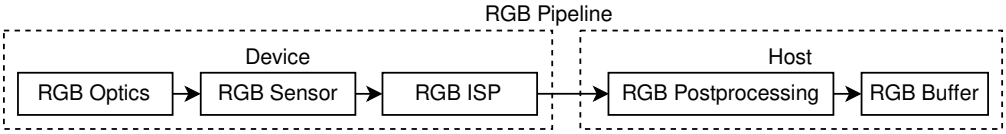

Fig. 3.3: RGB pipeline of the dataflow from device to host

- **ToF Pipeline**

  One only wants to grab the ToF data to do distance measuring tasks such as to know the closest object near the camera. A developer is expected to establish a pipeline that only invokes the ToF stream and ToF postprocessing modules. See Fig. 3.4.

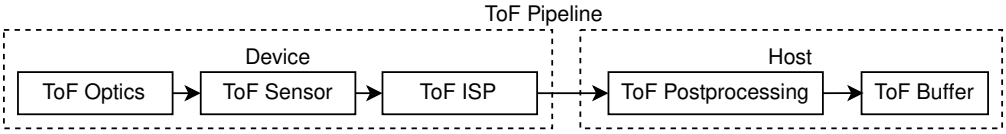

Fig. 3.4: ToF pipeline of the dataflow from device to host

- **Colorized-PCL Pipeline**

  A complex scenario is that one wants to simultaneously grab RGB and ToF to generate a point cloud, and draw the RGB colors as textures on the vertexes of the point cloud. The established pipeline should use both RGB stream and ToF stream and a more sophisticated postprocessing module. It includes both the above RGB pipeline and ToF pipeline but also needs to do some more work to combine both pipelines' results to make up a point cloud with colorized textures on it. See Fig. 3.5.

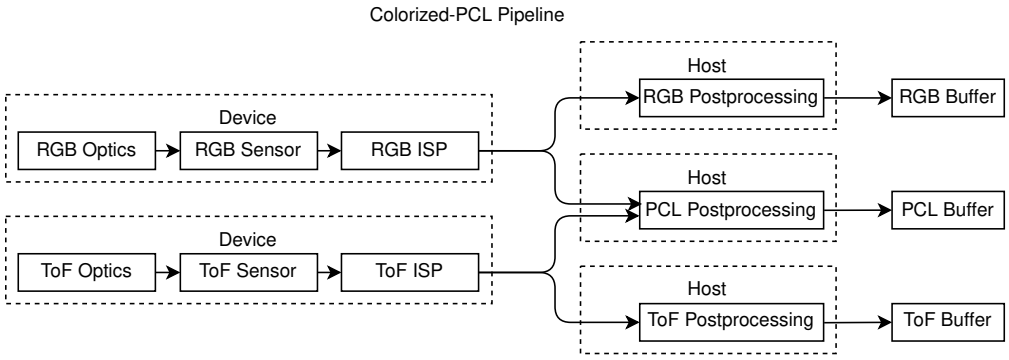

Fig. 3.5: Colorized-PCL pipeline of the dataflow from device to host

- **RGB + ToF Pipeline**

  This refers to a case where RGB and ToF data are required but they do not have to be combined in a way like in the Colorized-PCL pipeline. See Fig. 3.6.

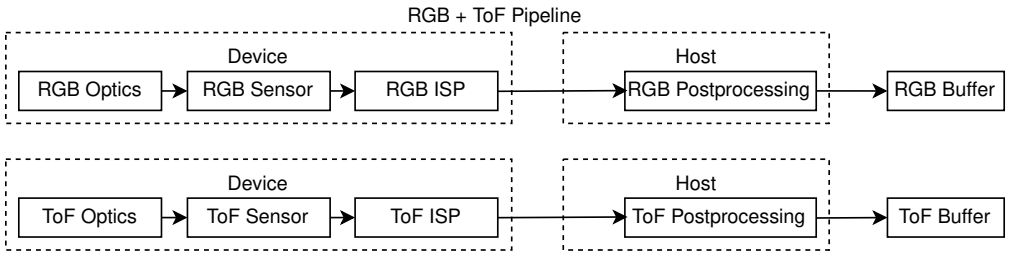

Fig. 3.6: Colorized-PCL pipeline of the dataflow from device to host

3.2.3 Pipeline Metadata

When a pipeline is established and started streaming, the pixel data from ISPs will be continuously transmitted through the USB unit. Along with each frame's pixel data, the camera additionally appends some other metadata to the frame to provide the context that generates the pixel data, such as time stamp, exposure time, board temperature, and accelerometer reading. Table 3.1, Table 3.2, and Table 3.3 list the metadata that can be retrieved by the SDK's programming interface according to the different pipelines. The specific function to call for reading the metadata is `GenTL::PDBufferGetMetaByName()`, see the explaination in Section 4.6.

Table 3.1: RGB Pipeline Metadata

Properties	Description	Type	R/W	Range	Default
Width	horizontal pixel number	uint16	Read	{1280, 1920}	1920
Height	vertical pixel number	uint16	Read	{960, 1080}	1080
FrameCount	counter per frame	uint32	Read	0~ $2^{32}-1$	~
TimeStamp	timing stamp per frame	uint32	Read	0~ $2^{32}-1$	~
CurFps	measured no. of frames per second	float32	Read	~	~
TransferFps	actual transferring no. of frames per second	float32	Read	~	~
PixelFormat	format of pixels in stream	uint32	Read	YUV422	YUV422
ExpoTime	exposure time (ms) for capturing light	float32	Read	~	~
Gain	sensor's ADC gain (in dB)	float32	Read	0.0~63.0	~
TempProcessor	temperature of processor	float32	Read	273.15f~	~

Table 3.2: ToF Pipeline Metadata

Properties	Description	Type	R/W	Range	Default
Width	horizontal pixel number	uint16	Read	640	640
Height	vertical pixel number	uint16	Read	480	480
FrameCount	counter per frame	uint32	Read	0~ $2^{32}-1$	~
TimeStamp	timing stamp per frame	uint32	Read	0~ $2^{32}-1$	~
CurFps	measured no. of frames per second	float32	Read	~	~
TransferFps	actual transferring no. of frames per second	float32	Read	~	~
PixelFormat	format of pixels in stream	uint32	Read	C16Y16	C16Y16
ExpoTime	exposure time (ms) for capturing light	float32	Read	~	~
Threshold	threshold of the confidence	uint32	Read	0~1000	100
Gain	sensor's ADC gain	float32	Read	0.0~20.0	10.0
TempSensor	temperature of ToF sensor	float32	Read	273.15f~	~
TempSenBrd	temperature of sensor board	float32	Read	273.15f~	~
TempProcessor	temperature of processor	float32	Read	273.15f~	~
TempVcsel	temperature of VCSE	float32	Read	273.15f~	~
TempImu	temperature of IMU	float32	Read	273.15f~	~
TempVcsel	temperature of VCSE	float32	Read	273.15f~	~
TempVcsel	temperature of VCSE	float32	Read	273.15f~	~
TempVcsel	temperature of VCSE	float32	Read	273.15f~	~
TempVcsel	temperature of VCSE	float32	Read	273.15f~	~
AccX	IMU accelerometer on x-axis	float32	Read	~	~
AccY	IMU accelerometer on y-axis	float32	Read	~	~
AccZ	IMU accelerometer on z-axis	float32	Read	~	~
GyroX	IMU's gyroscope on x-axis	float32	Read	~	~
GyroY	IMU's gyroscope on y-axis	float32	Read	~	~
GyroZ	IMU's gyroscope on z-axis	float32	Read	~	~
ModulateFreq	Modulation frequency (Hz) of light	float32	Read	{60, 70, 100}*10^6	60*10^6
Range	Detect range (m)	float32	Read	{1.5, 2.5, 7.5, 15}	2.5

Table 3.3: PCL Pipeline Metadata

Properties	Description	Type	R/W	Range	Default
Width	horizontal pixel number	uint16	Read	640	640
Height	vertical pixel number	uint16	Read	480	480
FrameCount	counter per frame	uint32	Read	0~ $2^{32}-1$	~
TimeStamp	timing stamp per frame	uint32	Read	0~ $2^{32}-1$	~
CurFps	measured no. of frames per second	float32	Read	~	~
DepthTimestamp	timing stamp per frame for depth map	uint32	Read	0~ $2^{32}-1$	~
TextureTimestamp	timing stamp per frame for texture map	float32	Read	~	~
TempProcessor	temperature (°C) of processor	float32	Read	273.15f~	~
Range	Detect range (m)	float32	Read	{1.5, 2.5, 7.5, 15}	~

3.2.4 Pipeline Controllable Properties

When a pipeline is established, a program may also have to change some properties such as the exposure, white balancing parameter, sensor's ADC gain etc. We list the properties that can be controlled by the SDK's programming interface according to the different pipelines. Table 3.4 , Table 3.5 and Table 3.6 list the properties for the RGB pipeline, ToF pipeline and PCL pipeline that can be modified by programming interfaces. For Colorized-PCL pipeline, since it incorporates both RGB and ToF pipeline, the properties in both Table 3.4 and Table 3.5 are modifiable. The specific function to call for controlling and obtaining these properties is `PDport::set()` and `PDport::get()`, see the explaination of them in *PDport Class*.

Table 3.4: RGB Pipeline Control Properties

Properties	Description	Type	R/W	Range	Default
StreamFps	set frame rate of the sensor	uint32	R/W	{25, 30, 50, 60, 80, 100}	60
Width	horizontal pixel number	uint16	Read	{1280, 1920}	1920
Height	vertical pixel number	uint16	Read	{960, 1080}	1080
PixelFormat	format of pixels in stream	uint32	Read	YUV422	YUV422
PayloadSize	no. of bytes per frame	uint32	Read	4147200	4147200
Exposure	exposure time (ms) for capturing light	float32	R/W	~	~
Gain	sensor's ADC gain (in dB)	float32	R/W	0.0~63.0	~
AutoExposure	enable or disable autoexposure	bool32	R/W	0~1	1
BlackLevelCompensation	enable or disable blc	bool32	R/W	0~1	1
RBlackLevel	R black level	uint32	R/W	0~100	16
GrBlackLevel	Gr black level	uint32	R/W	0~100	16
GbBlackLevel	Gb black level	uint32	R/W	0~100	16
BBlackLevel	B black level	uint32	R/W	0~100	16
AutoWhiteBalance	Choose the AWB function	uint32	R/W	0~2	1
AWBMinGain	min. gain for R/Gb/B if AWB==1 (automatic mode)	uint32	R/W	3096~24573	~
AWBMaxGain	max. gain for R/Gb/B if AWB==1 (automatic mode)	uint32	R/W	3096~24573	~
RGain	AWB gain for R if AWB==2 (mannual mode)	uint32	R/W	3096~24573	~
GbGain	AWB gain for Gb if AWB==2 (mannual mode)	uint32	R/W	3096~24573	~
BGain	AWB gain for B if AWB==2 (mannual mode)	uint32	R/W	3096~24573	~
AutoGammaCorrection	enable or disable auto gamma correction	bool32	R/W	0~1	1

Table 3.5: ToF Pipeline Control Properties

Properties	Description	Type	R/W	Range	Default
StreamFps	set frame rate of the sensor	uint32	R/W	10~100	60
Width	horizontal pixel number	uint16	R/W	640	640
Height	vertical pixel number	uint16	R/W	480	480
PixelFormat	format of pixels in stream	uint64	Read	C16Y16	C16Y16
Exposure	exposure time (ms) for capturing light	float32	R/W	0.008~1.2	~
Distance	max distance (m) to detetct	float32	R/W	{1.5, 2.5, 7.5, 15}	2.5
Medblur	remove the possible flying pixels	bool32	R/W	0~1	0
RemoveStrength	flying pixel removal strength	uint32	R/W	0~20	1
Threshold	threshold of the confidence	uint32	R/W	0~1000	100
Gain	sensor's ADC gain	float3	R/W	0~20.0	10
AutoExposure	enable or disable autoexposure	bool32	R/W	0~1	1
FilterStrength	filter strength of the spatial filter	uint32	R/W	0~3	1
DepthCompletion	enable or disable depth-completion	bool32	R/W	0~1	0
BorderRemove	enable or disable removing 20 border pixels	bool32	R/W	0~1	0

Table 3.6: PCL Pipeline Control Properties

Properties	Description	Type	R/W	Range	Default
FilterStrength	filter strength of flying pixels on PCL	float	R/W	0~20	10

CHAPTER

FOUR

CLASSES AND FUNCTIONS

4.1 Introduction to GenICam

The Okulo ™ P1 SDK is built upon the GenICam Standard to provide a generic programming interface for developers. The GenICam standard is widely used for machine vision or industrial cameras. The goal of the standard is to decouple industrial camera interface technology (such as GigE Vision, USB3 Vision, CoaXPress or Camera Link) from the user application programming interface (API).

GenICam consists of three modules to help solve the main tasks in the machine vision field in a generic way. These modules are:

- GenApi: Using an XML description file, is used to configure the camera and details how to access and control cameras;
- Standard Feature Naming Convention (SFNC): This is the recommended names and types for common features in cameras to promote interoperability;
- GenTL: This is the transport layer interface for enumerating cameras, grabbing images from the camera, and moving them to the user application.

4.2 PDport Class

This `PDport` class is a wrapper of the GenTL port handle. See Listing 4.1, `PDport` has a series of overloaded function `PDport::set()` and `PDport::get()`. Those functions wraps the `GenTL::GCWritePortByName` and `GenTL::GCReadPortByName` to write and read the properties of a pipeline.

Listing 4.1: Definition of class PDport

```
class PDport
{
private:
    PDHandle port = nullptr;
    bool inited = false;

protected:
    void setPort(PDHandle handle);
    bool isInited() const;
    virtual bool init();
    bool set(const char *name, const void *value, size_t size, int32_t dataType);
    bool get(const char *name, void *value, size_t size, int32_t dataType);

```

(continues on next page)

(continued from previous page)

```
public:
    virtual ~PDport() = default;

    virtual bool set(const char *name, int64_t value);
    virtual bool get(const char *name, int64_t &value);

    virtual bool set(const char *name, uint64_t value);
    virtual bool get(const char *name, uint64_t &value);

    virtual bool set(const char *name, int32_t value);
    virtual bool get(const char *name, int32_t &value);

    virtual bool set(const char *name, uint32_t value);
    virtual bool get(const char *name, uint32_t &value);

    virtual bool set(const char *name, int16_t value);
    virtual bool get(const char *name, int16_t &value);

    virtual bool set(const char *name, uint16_t value);
    virtual bool get(const char *name, uint16_t &value);

    virtual bool set(const char *name, float value);
    virtual bool get(const char *name, float &value);

    virtual bool set(const char *name, bool value);
    virtual bool get(const char *name, bool &value);

    operator PDHandle();
    operator bool();

    PDHandle getPort();
};
```

The available properties are listed in Table 3.4 and Table 3.5. For example, to write and read the `Exposure`, the code is:

Listing 4.2: Example of writing and reading Exposure value

```
PDdevice devInst;
auto stream = PDstream(devInst, "RGB");
stream.set("Exposure", 1.0f);
float exposure;
stream.get("Exposure", &exposure);
```

PDport is a base class for the PDdevice, PDstream and sPDstream. Therefore, in Listing 4.2, the PDstream object stream inherits the function set() and get() from PDport class. In general cases, developers do not have to instantiate a PDport object but use the PDstream object more often.

4.3 PDdevice Class

PDdevice class inherits from the PDport class. The definition of PDdevice is listed in Listing 4.3.

Listing 4.3: Definition of class PDdevice

```
class PDdevice: public PDport
{
    bool inited = false;
    uint32_t streamNum = 0;
    uint64_t deviceID = 0;
    std::unique_ptr<PDinterface> pIfhUsed;
    bool autoSelectDevice = false;

public:
    PDdevice(uint64_t _deviceID);
    PDdevice();
    ~PDdevice();

    bool init() override;
    uint32_t getStreamNum();
};
```

In a general case that there is only one Okulo ™ P1 Camera connected to a computer, by instantiating a PDdevice object with no specified deviceID, the property autoSelectDevice will be set to true. Then, by calling the member function PDdevice::init(), the program will try to automatically connect to a device and assign the deviceID, If a success (true) is returned, we can further acquire the stream number, namely the streamNum, in the device by calling PDdevice::getStreamNum(). See the code piece in Listing 4.4.

Listing 4.4: Instantiate PDdevice object

```
#include "inc/PDdevice.h"
int main(){
    ...
    PDdevice devInst;
    if (devInst.init())
    {
        size_t streamNum = devInst.getStreamNum();

```

(continues on next page)

(continued from previous page)

```
        // do stuff with multiple streams
        ...
    }
    ...
}
```

4.4 PDstream Related Classes

Before going to the class `PDstream`, we first have to introduce the `sPDstream` and `PCLstream`.

4.4.1 sPDstream Class

`sPDstream` class is a wrapper of the GenTL stream handle. See Listing 4.5, it inherits from `PDport` class. There are two functions to construct `sPDstream` objects, one is to specify `PDdevice` and `streamID`, and the other is to specify the `PDdevice` and `streamIDstr`. For an Okulo ™ P1 camera, you can create two independent `sPDstream` objects for RGB pipeline (see Fig. 3.3) and ToF pipeline (see Fig. 3.4) respectively. They share the same `PDdevice` but have different `streamID` and `streamIDstr` as listed below:

- RGB pipeline: `_streamID` = 0 , `streamIDstr` = "RGB"
- ToF pipeline: `_streamID` = 1, `streamIDstr` = "ToF"

Once an `sPDstream` object is constructed, one can get frames by `waitFrames()` and stop streaming by `stopStream()`. We can also retrieve its optical intrinsics and extrinsics by calling `getCamPara()` or get the `streamIDstr` value by `getStreamName()`.

Listing 4.5: Definition of class sPDstream

```
class sPDstream : public PDport
{
private:
    bool inited = false;
    PDHandle hEvent = nullptr;
    std::string streamIDstr;
    int BufferNum = 8;
    std::vector<std::function<void(void)>> destructFuncs;
    bool streamStarted = false;
    bool startStream();

    virtual PDHandle getMat(std::vector<cv::Mat> &Mats, uint64_t timeOut);
    const PDHandle hDev; // hStream intern will maintain the life of the device

protected:
    bool stopStream();

public:
    sPDstream(PDdevice &device, uint32_t _streamID = 0x0);
    sPDstream(PDdevice &device, const char *streamName);

    std::string getStreamName();

```

(continues on next page)

(continued from previous page)

```
24      virtual std::shared_ptr<PDbuffer> waitFrames(uint64_t timeOut = 1);
25
26      bool getCamPara(intrinsics &intr, extrinsics &extr);
27
28      bool init() override;
29
30      virtual ~sPDstream();
31  };
```

4.4.2 PCLstream Class

For the PCL pipeline (see Fig. 3.5), we define the `PCLstream` class. `PCLstream` inherits from `sPDstream` as shown in Listing 4.6. The `PCLstream` has a `streamVec` that wraps both RGB and ToF pipelines with a type of class `sPDstream`. The PCL pipeline has the following properties:

- PCL pipeline: `_streamID` = 2, `streamIDstr` = "PCL"

To read and write PCL pipeline's properties by calling `get()` and `set()` similar to RGB and ToF pipelines, our SDK override the two inherited function with a new version, namely `set() override` and `get() override`, see line 30 and 35 in Listing 4.6.

Listing 4.6: Definition of class PCLstream

```
1   class PCLstream : public sPDstream
2   {
3       std::vector<std::shared_ptr<sPDstream>> streamVec;
4
5   public:
6       PCLstream(PDdevice &device, uint32_t _streamID);
7       PCLstream(PDdevice &device, const char *streamName);
8       virtual std::shared_ptr<PDbuffer> waitFrames(uint64_t timeOut = 1) override;
9       bool init() override;
10      ~PCLstream();
11
12      sPDstream *getStream(const char *name, int &offset)
13      {
14          auto nameStr = std::string(name);
15          if (nameStr.size() > 5 && nameStr.compare(0, 5, "ToF::") == 0)
16          {
17              offset = 5;
18              return streamVec[0].get();
19          }
20          else if (nameStr.size() > 5 && nameStr.compare(0, 5, "RGB::") == 0)
21          {
22              offset = 5;
23              return streamVec[1].get();
24          }
25          offset = 0;
26          return this;
27      }
28
29  #define SET_GET_OVERRIDE_FUNC(type)                                    \
```

(continues on next page)

(continued from previous page)

```
30      bool set(const char *name, type value) override                 \
31      {                                                               \
32          int offset = 0;                                             \
33          return getStream(name, offset)->set(&name[offset], value);\
34      }                                                               \
35      bool get(const char *name, type &value) override                \
36      {                                                               \
37          int offset = 0;                                             \
38          return getStream(name, offset)->get(&name[offset], value);\
39      }
40
41      SET_GET_OVERRIDE_FUNC(int64_t)
42      SET_GET_OVERRIDE_FUNC(uint64_t)
43      SET_GET_OVERRIDE_FUNC(int32_t)
44      SET_GET_OVERRIDE_FUNC(uint32_t)
45      SET_GET_OVERRIDE_FUNC(int16_t)
46      SET_GET_OVERRIDE_FUNC(uint16_t)
47      SET_GET_OVERRIDE_FUNC(float)
48      SET_GET_OVERRIDE_FUNC(bool)
49  };
```

Since the RGB pipeline and ToF pipeline are part of PCL pipeline, if we need to modify or get the property of one child pipeline, we must decorate the property with the prefix `RGB::` or `ToF::`. During calling the member function `get()` or `set()` of `PCLstream`, the prefix will firstly be parsed by `getStream()` function (see line 12 in Listing 4.6) to get the corresponding `sPDstream` in `streamVec`, and then `sPDstream`'s `get()` or `set()` member function will be invoked. Listing 4.7 lists an example of how to set PCL pipeline's RGB Gain and get ToF exposure.

Listing 4.7: Example of writing and reading PCL's properties

```
1  PDdevice devInst;
2  auto pclstream = PDstream(devInst, "PCL");
3  pclstream.set("RGB::Gain", 10.0f);
4  float tof_exposure;
5  pclstream.get("ToF::Exposure", &tof_exposure);
```

4.4.3 PDstream Class

For now, we can establish the following three streams based on the previously defined class `sPDstream` and `PCLstream`.

- RGB pipeline: `_streamID` = 0 , `streamIDstr` = "RGB", instantiated as `sPDstream`
- ToF pipeline: `_streamID` = 1, `streamIDstr` = "ToF", instantiated as `sPDstream`
- PCL pipeline: `_streamID` = 2, `streamIDstr` = "PCL", instantiated as `PCLstream`

`PDstream` class inherits from `PDport`, see Listing 4.8. `PDstream` wraps up the `sPDstream` and `PCLstream` with member variable `pStream`. See Listing 4.9 for the construction function of `PDstream`, when the input argument `streamName` is "PCL" or `_streamID` is 2, the `pStream` will be instantiated with a pointer to `PCLstream` object; otherwise, for "RGB" and "ToF", the `pStream` will be instantiated with a pointer to `sPDstream` object.

Listing 4.8: Definition of class PDstream

```
class PDstream : public PDport // proxy class
{
    std::shared_ptr<sPDstream> pStream;

public:
    PDstream(PDdevice &device, uint32_t _streamID = 0x0);
    PDstream(PDdevice &device, const char *streamName);
    std::string getStreamName();
    bool getCamPara(intrinsics &intr, extrinsics &extr);
    std::shared_ptr<PDbuffer> waitFrames(uint64_t timeOut = 1);
    bool init() override;

    template <typename T> bool set(const char *name, T value)
    {
        return pStream->set(name, value);
    }

    template <typename T> bool get(const char *name, T &value)
    {
        return pStream->get(name, value);
    }
};
```

Listing 4.9: Construction function of class PDstream

```
PDstream::PDstream(PDdevice &device, const char *streamName)
{
    std::string streamIDstr = streamName;
    std::transform(streamIDstr.begin(), streamIDstr.end(), streamIDstr.begin(),
                [](unsigned char c) { return std::toupper(c); });
    if (streamIDstr == "PCL")
    {
        pStream = std::make_shared<PCLstream>(device, streamName);
    }
    else
    {
        pStream = std::make_shared<sPDstream>(device, streamName);
    }
}

PDstream::PDstream(PDdevice &device, uint32_t _streamID)
{
    if (device.getStreamNum() > 1 && _streamID == device.getStreamNum() - 1)
    {
        pStream = std::make_shared<PCLstream>(device, _streamID);
    }
    else
    {
        pStream = std::make_shared<sPDstream>(device, _streamID);
    }
}
```

In the code piece Listing 4.10, we instantiate all the 3 retrieved pipelines and initialize them sequentially. Note that we can get their stream name by member function `PDstream::getStreamName()`.

Listing 4.10: Instantiate PDstream

```
#include "GenICam/GenTL.h"
#include "basic/inc/camParaDef.h"
#include "inc/PDdevice.h"
#include "inc/PDstream.h"
#include "stringFormat.h"

int main()
{
    ...
    PDdevice devInst;
    if (devInst.init())
    {
        size_t streamNum = devInst.getStreamNum();
        for (int i = 0; i < streamNum; i++)
        {
            PDstream streamInst(devInst, i);
            streamInst.init();
            PD_INFO("stream : %s\n", streamInst.getStreamName().c_str());
            // do stuff with streamInst
            ...
        }
    }
    ...
}
```

4.5 PDbuffer Class

After instantiating and starting a pipeline, we can obtain its frames by calling `PDstream::waitFrames(uint64_t timeOut)` with a `timeOut` setting. The returned content is an object of `PDbuffer`, see line 10 in Listing 4.8. The `PDbuffer` wraps up a vector of `cv::Mat` (see OpenCV). For RGB, ToF and PCL pipelines, the numbers of Mats are different. We can use `PDbuffer::getMatNum()` to obtain the Mat number, and use `PDbuffer::getMat()` to obtain the Mats.

- For RGB pipeline, there will be only 1 `cv::Mat` that contains `RGB` image of datatype CV_8UC3.
- For ToF pipeline, we can get 2 `cv::Mat`, one for the `phase` Mat of datatype CV_16UC1, another one for the `infrared` Mat of datatype CV_16UC1;
- For PCL pipeline, we can get 3 `cv::Mat` for `pointcloud` of datatype CV_32FC3, `infrared` of datatype CV_16UC1 and `RGB` (or say `color`) of datatype CV_8UC3 correspondingly.

Listing 4.11: Definition of class PDbuffer

```
class PDbuffer
{
    bool releaseMat(PDHandle pHandle);
    PDHandle hBuffer = nullptr;
    PDHandle hStream = nullptr; // hBuffer intern will maintain the life of stream
```

(continues on next page)

(continued from previous page)

```
    bool setHanle();
    friend class sPDstream;
    std::vector<cv::Mat> Mats;

public:
    PDbuffer(sPDstream *stream);

    operator PDHandle()
    {
        return hBuffer;
    }

    void *getPort()
    {
        return hBuffer;
    }

    ~PDbuffer();
    uint32_t getMatNum();
    const cv::Mat &getMat(uint32_t id = 0);
};
```

In Listing 4.12 we give an example of extracting the data from a ToF PDstream's output. We use `pha = frame->getMat(0);` and `infrared = frame->getMat(1);` to get phase and infrared data, and use `cv::imshow` to plot them in a window.

Listing 4.12: Extract ToF Mats

```
#include "inc/PDdevice.h"
#include "inc/PDstream.h"
#include "stringFormat.h"
#include "timeTest.h"
#include <chrono>
#include <thread>

int tofDemo()
{
    PDdevice devInst;
    if (devInst)
    {
        auto stream = PDstream(devInst, "ToF");
        if (stream)
        {
            bool isAutoExposure = false;
            stream.get("AutoExposure", isAutoExposure);
            if (isAutoExposure)
            {
                stream.set("AutoExposure", false);
            }
            stream.set("Distance", 7.5f);
            stream.set("StreamFps", 50.0f);
            stream.set("Threshold", 100);
```

(continues on next page)

(continued from previous page)

```
            stream.set("Exposure", 1.0f);

            float DistRange = 0.0f;
            while (1)
            {
                auto frame = stream.waitFrames();

                char key = cv::waitKey(1);
                if (frame)
                {
                    if (DistRange < 1e-5) // DistRange should be inited
                    {
                        size_t varSize = sizeof(varSize);
                        GenTL::PDBufferGetMetaByName(frame->getPort(),
                            "Range", nullptr, &DistRange, &varSize);
                        printf("max distance %f for distanceMap", DistRange);
                    }
                    const cv::Mat &pha = frame->getMat(0);
                    const cv::Mat &infrared = frame->getMat(1);
                    cv::imshow("pha", pha);
                    cv::imshow("infrared", infrared);
                    if (key == 'q')
                    {
                        break;
                    }
                    if (key == 'c')
                    {
                        GenTL::PDBufferSave(*frame, nullptr);
                        printf("saved  \n");
                    }
                }
                std::this_thread::sleep_for(std::chrono::milliseconds(1));
            }
            return true;
        }
    }
    return false;
}
```

4.6 GenTL Function

The metadata listed in the *RGB Pipeline Metadata* and *ToF Pipeline Metadata* has to be accesed by `GenTL::PDBufferGetMetaByName()`.

CHAPTER

FIVE

CODE SAMPLE

5.1 Generate executable samples

To run the code samples, use CMake Tool to generate executable samples of Okulo SDK

```
>>> #under OkuloSdk
>>> mkdir build && cd build
>>> cmake ../
>>> make -j
>>> ./example/streamShow
```

5.2 Get Camera Parameters

Listing 5.1: getCamPara

```
#include "GenICam/GenTL.h"
#include "basic/inc/camParaDef.h"
#include "debug.h"
#include "inc/PDdevice.h"
#include "inc/PDstream.h"
#include "stringFormat.h"

int main()
{
    PDdevice devInst;
    intrinsics intrin;
    extrinsics extrin;
    if (devInst.init())
    {
        size_t streamNum = devInst.getStreamNum();
        for (int i = 0; i < streamNum; i++)
        {
            PDstream streamInst(devInst, i);
            streamInst.init();
            PD_INFO("stream : %s\n", streamInst.getStreamName().c_str());
            if (streamInst.getCamPara(intrin, extrin))
            {
                print_intrinsics(&intrin);
```

(continues on next page)

(continued from previous page)

```
                print_extrinsics(&extrin);
            }
        }
        return system("pause");
    }
    return 0;
}
```

In Listing 5.1, we first initialize a `PDdevice` object named as `devInst`, and retrive its `streamNum` through member function `PDdevice::getStreamNum()`. Then we use the `devInst` and stream index starting from 0 to `streamNum - 1`, to instantiate the stream class `PDstream`'s object `streamInst`. Each stream logically corresponds to an RGB pipeline, a ToF pipeline or a PCL pipeline that has its own optics, sensors and ISPs. As a result, each stream has their intrinsics and extrinsics. We call the `PDstream` class's member function `PDstream::getCamPara` to retrieve the intrinsics and extrinsics. Finally, we print them with the wrapped function `print_intrinsics` and `print_extrinsics` correspondingly.

5.3 RGB Streaming Demo

Listing 5.2 lists the code to grab and show the RGB image captured by the Okulo ™ P1 camera.

Listing 5.2: RGB Streaming

```
#include "inc/PDdevice.h"
#include "inc/PDstream.h"
#include "stringFormat.h"
#include "timeTest.h"
#include <chrono>
#include <thread>

int rgbDemo()
{
    PDdevice devInst;

    if (devInst)
    {
        auto stream = PDstream(devInst, "RGB");
        if (stream)
        {
            bool isAutoExposure = false;
            stream.get("AutoExposure", isAutoExposure);
            if (isAutoExposure)
            {
                stream.set("AutoExposure", false);
            }
            stream.set("Exposure", 1.0f);
            stream.set("Gain", 1.0f);
            while (1)
            {
                auto frame = stream.waitFrames();
                char key = cv::waitKey(1);
                if (frame)
```

(continues on next page)

(continued from previous page)

```
                {
                    const cv::Mat &rgb = frame->getMat(0);
                    cv::imshow("rgb", rgb);

                    if (key == 'c')
                    {
                        GenTL::PDBufferSave(*frame, nullptr);
                        printf("saved  \n");
                    }
                }
                if (key == 'q')
                {
                    break;
                }

                std::this_thread::sleep_for(std::chrono::milliseconds(1));
            }
            return true;
        }
    }
    return false;
}
```

We instantiate the rgb stream with `PDdevice devInst` and the string name `RGB`. Once the `stream` object is successfully created (not a null pointer), we set the exposure mode, exposure time and ADC gain through `stream.set()` function. Then, we unterminatedly invoke `stream.waitFrames()` to refresh the current frame and use `cv::imshow()` to draw it in a window. Furthermore, with the "rgb" window selected, once clicking the keyboard's *c* key, we will save an RGB frame to disk.

5.4 Pointcloud Streaming Demo

In Listing 5.3, we show the code for the streaming of Point-Cloud. In this application, we need to use both RGB and ToF streams to support color mapping to 3D space. We directly instantiate the `pclstream` with `PDstream(devInst, "PCL")`, and the `PCL` string name will be recognized.

Listing 5.3: Point-Cloud Streaming

```
#include "inc/PDdevice.h"
#include "inc/PDstream.h"
#include "stringFormat.h"
#include "timeTest.h"
#include <chrono>
#include <thread>

int pclDemo()
{
    PDdevice devInst;
    if (devInst)
    {
        auto pclstream = PDstream(devInst, "PCL");
```

(continues on next page)

(continued from previous page)

```
        if (pclstream)
        {

            bool isTofAutoExposure = false;
            bool isRGBAutoExposure = false;
            pclstream.get("ToF::AutoExposure", isTofAutoExposure);
            pclstream.get("RGB::AutoExposure", isRGBAutoExposure);
            if (isTofAutoExposure)
            {
                pclstream.set("ToF::AutoExposure", false);
            }
            if (isRGBAutoExposure)
            {
                pclstream.set("RGB::AutoExposure", false);
            }

            pclstream.set("ToF::Distance", 7.5f);
            pclstream.set("ToF::StreamFps", 50.0f);
            pclstream.set("ToF::Threshold", 100);
            pclstream.set("ToF::Exposure", 1.0f);
            pclstream.set("RGB::Exposure", 10.0f);
            pclstream.set("RGB::Gain", 20.0f);

            bool saveReq = false;
            int count = 0;
            float DistRange = 0.0f;
            while (1)
            {
                // pcl frames work only when tof && rgb stream grab frames!!!
                auto pPclFrame = pclstream.waitFrames();
                char key = cv::waitKey(1);
                if (key == 'c')
                {
                    saveReq = true;
                }
                if (key == 'q')
                {
                    break;
                }
                if (pPclFrame)
                {
                    if (DistRange < 1e-5) // DistRange should be inited
                    {
                        size_t varSize = sizeof(varSize);
                        GenTL::PDBufferGetMetaByName(pPclFrame->getPort(),
                                                     "Range", &DistRange,
                                                     &varSize, nullptr);
                        printf("max distance %f for distanceMap\n", DistRange);
                    }
                    const cv::Mat &xyz = pPclFrame->getMat(0);
                    const cv::Mat &infrared = pPclFrame->getMat(1);
                    const cv::Mat &color = pPclFrame->getMat(2);
```

(continues on next page)

(continued from previous page)

```
                    std::vector<cv::Mat> channels(3);
                    cv::split(xyz, channels);
                    const cv::Mat &depth = channels[2];
                    cv::Mat u16Depth;
                    depth.convertTo(u16Depth, CV_16UC1, 65535.0 / DistRange);

                    cv::imshow("xyz", xyz);
                    cv::imshow("infrared", infrared);
                    cv::imshow("color", color);
                    cv::imshow("depth", depth);

                    if (saveReq)
                    {
                        saveReq = false;
                        GenTL::PDBufferSave(*pPclFrame, nullptr);
                        cv::imwrite(stringFormat("depth-%d.png", count), u16Depth);
                        printf("saved  \n");
                        count++;
                    }
                }
                pPclFrame.reset(); // release frame buffer immediately
                std::this_thread::sleep_for(std::chrono::milliseconds(1));
            }
            return true;
        }
    }
    return false;
}
```

CHAPTER

SIX

PYTHON WRAPPER

We use pybind11 tool to export Okulo ™ API classes and functions to python through a dynamically loadable module, e.g. the `pyokulo.cpython-38-x86_64-linux-gnu.so` in Linux or `pyokulo.cp38-win_amd64.pyd` for Windows. The class `PDdevice` and `PDstream` are well exposed to python. Thus, developers can use a Python-ish way to control the Okulo ™ camera by simply executing `import pyokulo`. Below we show a simple script that also retrieves the camera's intrinsic and extrinsic parameters as we have done in Listing 5.1.

Listing 6.1: Get Camera Param with Python Wrapper

```
import pyokulo as okulo
import cv2 as cv

intrinsic = okulo.intrinsics()
extrinsic = okulo.extrinsics()

device = okulo.PDdevice()

if not device.init():
    print("device init failed")
    exit(-1)
print("device init succeed")
stream_num = device.getStreamNum()
print("stream_num = {}\n".format(stream_num))

for i in range(stream_num):
    stream = okulo.PDstream(device, i)
    suc= stream.init()
    streamName = stream.getStreamName()
    if not suc:
        print("stream {} init failed\n".format(i))
    else:
        print("stream {} init success\n".format(i))
    while True:
        mats = stream.getPyMat()
        for i, mat in enumerate(mats):
            cv.imshow("%s: %i" %(streamName, i), mat)

        key = cv.waitKey(1)
        if key & 0xFF == ord('q'):
            break

    suc =  stream.getCamPara(intrinsic, extrinsic)
```

(continues on next page)

(continued from previous page)

```
    okulo.print_intrinsics(intrinsic)
    okulo.print_extrinsics(extrinsic)

    print("finish stream {}\n".format(i))
    cv.destroyAllWindows()


print("all set\n")
```

In the Listing 6.1, we import the `pyokulo` and `opencv` library, initialize empty `intrinsic`, `extrinsic` and `device` object through `okulo.intrinsics()`, `okulo.extrinsics()`, and `okulo.PDdevice()`. By `device.init()`, the program tries to connect to a camera. If there is a powered-up camera connected to the host computer, we will step forward to `device.getStreamNum()`. Then we initialize the `PDstream` object `stream` correspondingly to each stream and get their intrinsics and extrinsics by `stream.getCamPara(intrinsic, extrinsic)`. The whole procedure is the same as in its C++ version code in Listing 5.1.

CHAPTER

SEVEN

LICENSE AGREEMENT FOR SOFTWARE

Point Spread software license agreement to Okulo ™ Software Development Kit.

The Software License Agreement in Section 7.2 and the Supplement in Section 7.3 contain license terms and conditions that govern the use of Point Spread software. By accepting this agreement, you agree to comply with all the terms and conditions applicable to the product(s) included herein.

7.1 Description

The Okulo ™ Software Development Kit (SDK) provides command-line and graphical tools for building, debugging and optimizing the performance of applications accelerated by Point Spread Cameras, runtime and math libraries, and documentation including programming guides, user manuals, and API references.

Note: Important Notice—Read before downloading, installing, copying or using the licensed software:

This license agreement, including exhibits attached ("Agreement"), is a legal agreement between you and Point Spread Technology Co., Ltd. and governs your use of a Okulo ™ SDK.

Each SDK has its own set of software and materials, but here is a description of the types of items that may be included in each SDK: source code, header files, APIs, data sets and assets (examples include images, textures, models, scenes, videos, native API input/output files), binary software, sample code, libraries, utility programs, programming code and documentation.

This Agreement can be accepted only by an adult of legal age of majority in the country in which the SDK is used.

If you are entering into this Agreement on behalf of a company or other legal entity, you represent that you have the legal authority to bind the entity to this Agreement, in which case "you" will mean the entity you represent.

If you don't have the required age or authority to accept this Agreement, or if you don't accept all the terms and conditions of this Agreement, do not download, install or use the SDK.

You agree to use the SDK only for purposes that are permitted by (a) this Agreement, and (b) any applicable law, regulation or generally accepted practices or guidelines in the relevant jurisdictions.

7.2 License Agreement on Okulo ™ Software Development Kit

7.2.1 License Content

License Grant

Subject to the terms of this Agreement, Point Spread hereby grants you a non-exclusive, non-transferable license, without the right to sub-license (except as expressly provided in this Agreement) to:

- Install and use the SDK,
- Modify and create derivative works of sample source code delivered in the SDK, and
- Distribute those portions of the SDK that are identified in this Agreement as distributable, as incorporated in object code format into a software application that meets the distribution requirements indicated in this Agreement.

Distribution Requirements

These are the distribution requirements for you to exercise the distribution grant:

- Your application must have material additional functionality, beyond the included portions of the SDK.
- The distributable portions of the SDK shall only be accessed by your application.
- The following notice shall be included in modifications and derivative works of sample source code distributed: "This software contains source code provided by Point Spread Technology Co., Ltd."
- Unless a developer tool is identified in this Agreement as distributable, it is delivered for your internal use only.
- The terms under which you distribute your application must be consistent with the terms of this Agreement, including (without limitation) terms relating to the license grant and license restrictions and protection of Point Spread's intellectual property rights. Additionally, you agree that you will protect the privacy, security and legal rights of your application users.
- You agree to notify Point Spread in writing of any known or suspected distribution or use of the SDK not in compliance with the requirements of this Agreement, and to enforce the terms of your agreements with respect to distributed SDK.

Authorized Users

You may allow employees and contractors of your entity or your subsidiary(ies) to access and use the SDK from your secure network to perform work on your behalf.

If you are an academic institution you may allow users enrolled or employed by the academic institution to access and use the SDK from your secure network.

You are responsible for compliance with the terms of this Agreement by your authorized users. If you become aware that your authorized users didn't follow the terms of this Agreement, you agree to take reasonable steps to resolve the non-compliance and prevent new occurrences.

Pre-Release SDK

The SDK versions identified as alpha, beta, preview or otherwise as pre-release, may not be fully functional, may contain errors or design flaws, and may have reduced or different security, privacy, accessibility, availability, and reliability standards relative to commercial versions of Point Spread software and materials. Use of a pre-release SDK may result in unexpected results, loss of data, project delays or other unpredictable damage or loss.

You may use a pre-release SDK at your own risk, understanding that pre-release SDKs are not intended for use in production or business-critical systems.

Point Spread may choose not to make available a commercial version of any pre-release SDK. Point Spread may also choose to abandon development and terminate the availability of a pre-release SDK at any time without liability.

Updates

Point Spread may, at its option, make available patches, workarounds or other updates to this SDK. Unless the updates are provided with their separate governing terms, they are deemed part of the SDK licensed to you as provided in this Agreement. You agree that the form and content of the SDK that Point Spread provides may change without prior notice to you. While Point Spread generally maintains compatibility between versions, Point Spread may in some cases make changes that introduce incompatibilities in future versions of the SDK.

Components Under Other Licenses

The SDK may come bundled with, or otherwise include or be distributed with, Point Spread or third-party components with separate legal notices or terms as may be described in proprietary notices accompanying the SDK. If and to the extent there is a conflict between the terms in this Agreement and the license terms associated with the component, the license terms associated with the components control only to the extent necessary to resolve the conflict.

Subject to the other terms of this Agreement, you may use the SDK to develop and test applications released under Open Source Initiative (OSI) approved open source software licenses.

Reservation of Rights

Point Spread reserves all rights, title, and interest in and to the SDK, not expressly granted to you under this Agreement.

7.2.2 Limitations

The following license limitations apply to your use of the SDK:

- You may not reverse engineer, decompile or disassemble, or remove the copyright or other proprietary notices from any portion of the SDK or copies of the SDK.
- Except as expressly provided in this Agreement, you may not copy, sell, rent, sub-license, transfer, distribute, modify, or create derivative works of any portion of the SDK. For clarity, you may not distribute or sublicense the SDK as a stand-alone product.
- Unless you have an agreement with Point Spread for this purpose, you may not indicate that an application created with the SDK is sponsored or endorsed by Point Spread.
- You may not bypass, disable, or circumvent any encryption, security, digital rights management or authentication mechanism in the SDK.
- You may not use the SDK in any manner that would cause it to become subject to an open-source software license. As examples, licenses that require as a condition of use, modification, and/or distribution that the SDK be:

  - Disclosed or distributed in source code form;
  - Licensed for the purpose of making derivative works; or
  - Redistributable at no charge.

- You acknowledge that the SDK as delivered is not tested or certified by Point Spread for use in connection with the design, construction, maintenance, and/or operation of any system where the use or failure of such system could result in a situation that threatens the safety of human life or results in catastrophic damages (each, a "Critical Application"). Examples of Critical Applications include use in avionics, navigation, autonomous vehicle applications, AI solutions for automotive products, military, medical, life support or other life-critical applications. Point Spread shall not be liable to you or any third party, in whole or in part, for any claims or damages arising from such uses. You are solely responsible for ensuring that any product or service developed with the SDK as a whole includes sufficient features to comply with all applicable legal and regulatory standards and requirements.
- You agree to defend, indemnify and hold harmless Point Spread and its affiliates, and their respective employees, contractors, agents, officers and directors, from and against any and all claims, damages, obligations, losses, liabilities, costs or debt, fines, restitutions and expenses (including but not limited to attorney's fees and costs incident to establishing the right of indemnification) arising out of or related to products or services that use the SDK in or for Critical Applications, and for use of the SDK outside the scope of this Agreement or not in compliance with its terms.
- You may not reverse engineer, decompile or disassemble any portion of the output generated using SDK elements for the purpose of translating such output artifacts to target a non-Point Spread platform.

7.2.3 Ownership

Point Spread's rights

- Point Spread or its licensors hold all rights, title and interest in and to the SDK and its modifications and derivative works, including their respective intellectual property rights, subject to Title *Your rights*. This SDK may include software and materials from Point Spread's licensors, and these licensors are intended third-party beneficiaries that may enforce this Agreement with respect to their intellectual property rights.

Your rights

- You hold all rights, title and interest in and to your applications and your derivative works of the sample source code delivered in the SDK, including their respective intellectual property rights, subject to Title *Point Spread's rights*.
- You may, but don't have to, provide to Point Spread suggestions, feature requests or other feedback regarding the SDK, including possible enhancements or modifications to the SDK. For any feedback that you voluntarily provide, you hereby grant Point Spread and its affiliates a perpetual, non-exclusive, worldwide, irrevocable license to use, reproduce, modify, license, sub-license (through multiple tiers of sub-licenses), and distribute (through multiple tiers of distributors) it without the payment of any royalties or fees to you. Point Spread will use the feedback of its choice. Point Spread is constantly looking for ways to improve its products, so you may send feedback to Point Spread by submitting new issues at https://github.com/point-spread/OkuloSdk/issues/new?

7.2.4 No Warranties

THE SDK IS PROVIDED BY POINTSPREAD "AS IS" AND "WITH ALL FAULTS." TO THE MAXIMUM EXTENT PERMITTED BY LAW, POINTSPREAD AND ITS AFFILIATES EXPRESSLY DISCLAIM ALL WARRANTIES OF ANY KIND OR NATURE, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT, OR THE ABSENCE OF ANY DEFECTS THEREIN, WHETHER LATENT OR PATENT. NO WARRANTY IS MADE ON THE BASIS OF TRADE USAGE, COURSE OF DEALING OR COURSE OF TRADE.

7.2.5 Limitation of Liability

TO THE MAXIMUM EXTENT PERMITTED BY LAW, POINTSPREAD AND ITS AFFILIATES SHALL NOT BE LIABLE FOR ANY SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, OR ANY LOST PROFITS, LOSS OF USE, LOSS OF DATA OR LOSS OF GOODWILL, OR THE COSTS OF PROCURING SUBSTITUTE PRODUCTS, ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT OR THE USE OR PERFORMANCE OF THE SDK, WHETHER SUCH LIABILITY ARISES FROM ANY CLAIM BASED UPON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR ANY OTHER CAUSE OF ACTION OR THEORY OF LIABILITY. IN NO EVENT WILL POINTSPREAD'S AND ITS AFFILIATES TOTAL CUMULATIVE LIABILITY UNDER OR ARISING OUT OF THIS AGREEMENT EXCEED US$10.00. THE NATURE OF THE LIABILITY OR THE NUMBER OF CLAIMS OR SUITS SHALL NOT ENLARGE OR EXTEND THIS LIMIT.

These exclusions and limitations of liability shall apply regardless if Point Spread or its affiliates have been advised of the possibility of such damages, and regardless of whether a remedy fails its essential purpose. These exclusions and limitations of liability form an essential basis of the bargain between the parties, and, absent any of these exclusions or limitations of liability, the provisions of this Agreement, including, without limitation, the economic terms, would be substantially different.

7.2.6 Termination

This Agreement will continue to apply until terminated by either you or Point Spread as described below:

- If you want to terminate this Agreement, you may do so by stopping to use the SDK.
- Point Spread may, at any time, terminate this Agreement if:
  - you fail to comply with any term of this Agreement and the non-compliance is not fixed within thirty (30) days following notice from Point Spread (or immediately if you violate Point Spread's intellectual property rights); or
  - you commence or participate in any legal proceeding against Point Spread with respect to the SDK; or
  - Point Spread decides to no longer provide the SDK in a country or, in Point Spread's sole discretion, the continued use of it is no longer commercially viable.
- Upon any termination of this Agreement, you agree to promptly discontinue use of the SDK and destroy all copies in your possession or control. Your prior distributions in accordance with this Agreement are not affected by the termination of this Agreement. Upon written request, you will certify in writing that you have complied with your commitments under this section. Upon any termination of this Agreement all provisions survive except for the license grant provisions.

7.2.7 General

If you wish to assign this Agreement or your rights and obligations, including by merger, consolidation, dissolution or operation of law, contact Point Spread to ask for permission. Any attempted assignment not approved by Point Spread in writing shall be void and of no effect. Point Spread may assign, delegate or transfer this Agreement and its rights and obligations, and if to a non-affiliate you will be notified.

You agree to cooperate with Point Spread and provide reasonably requested information to verify your compliance with this Agreement.

This Agreement will be governed in all respects by the laws of the People's Republic of China and as those laws are applied to contracts entered into and performed entirely, without regard to the conflicts of laws principles. The United Nations Convention on Contracts for the International Sale of Goods is specifically disclaimed. You agree to all terms of this Agreement in the English language.

The courts residing in Nantong, Jiangsu, P.R.C such as Nantong Intermediate People's Court shall have exclusive jurisdiction over any dispute or claim arising out of this Agreement. Notwithstanding this, you agree that Point Spread shall still be allowed to apply for injunctive remedies or an equivalent type of urgent legal relief in any jurisdiction.

If any court of competent jurisdiction determines that any provision of this Agreement is illegal, invalid or unenforceable, such provision will be construed as limited to the extent necessary to be consistent with and fully enforceable under the law and the remaining provisions will remain in full force and effect. Unless otherwise specified, remedies are cumulative.

Each party acknowledges and agrees that the other is an independent contractor in the performance of this Agreement.

The SDK has been developed entirely at private expense and is a "commercial item" consisting of "commercial computer software" and "commercial computer software documentation" provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the P.R.C. Government or a P.R.C. Government subcontractor is subject to the restrictions in this Agreement or as outlined in subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable. Contractor/manufacturer is Point Spread, Room 212, Building #10A, Zilang Park, No. 60 Chongchuan Rd., Nantong Central Innovation District, Nantong, Jiangsu, 226000.

The SDK is subject to the People's Republic of China export laws and regulations. You agree that you will not ship, transfer or export the SDK into any country, or use the SDK in any manner, prohibited by the People's Republic of China Bureau of Industry and Security or economic sanctions regulations administered by the P.R.C. Department of Treasury's Office of Foreign Assets Control (OFAC), or any applicable export laws, restrictions or regulations. These laws include restrictions on destinations, end users and end use. By accepting this Agreement, you confirm that you are not a resident or citizen of any country currently embargoed by the P.R.C. and that you are not otherwise prohibited from receiving the SDK.

Any notice delivered by Point Spread to you under this Agreement will be delivered via mail, email or fax. You agree that any notices that Point Spread sends you electronically will satisfy any legal communication requirements. Please direct your legal notices or other correspondence to Point Spread Technology Co., Ltd., Room 212, Building #10A, Zilang Park, No. 60 Chongchuan Rd., Nantong Central Innovation District, Nantong, Jiangsu, 226000, People's Republic of China, Attention: Legal Department.

This Agreement and any exhibits incorporated into this Agreement constitute the entire agreement of the parties with respect to the subject matter of this Agreement and supersede all prior negotiations or documentation exchanged between the parties relating to this SDK license. Any additional and/or conflicting terms on documents issued by you are null, void, and invalid. Any amendment or waiver under this Agreement shall be in writing and signed by representatives of both parties.

7.3 Supplement to License Agreement on Okulo ™ Software Development Kit

The terms in this supplement govern your use of the Point Spread Okulo ™ SDK under the terms of your license agreement ("Agreement") as modified by this supplement. Capitalized terms used but not defined below have the meaning assigned to them in the Agreement.

This supplement is an exhibit to the Agreement and is incorporated as an integral part of the Agreement. In the event of conflict between the terms in this supplement and the terms in the Agreement, the terms in this supplement govern.

7.3.1 License Scope

The SDK is licensed for you to develop applications only for use in systems with Point Spread Okulo ™ devices.

7.3.2 Distribution

The portions of the SDK that are distributable under the Agreement are listed in Attachment A.

7.3.3 Operating Systems

Those portions of the SDK designed exclusively for use on the Windows, or Linux based operating systems, or other operating systems derived from the source code to these operating systems, may be copied and redistributed for use in accordance with this Agreement, provided that the object code files are not modified in any way (except for unzipping of compressed files).

7.3.4 Questions

If the distribution terms in this Agreement are not suitable for your organization, or for any questions regarding this Agreement, please contact Point Spread at sales@pointspread.cn.

7.3.5 Attachment A

The following Okulo ™ SDK files may be distributed with License Applications developed by you.

OS	Component
Linux	– libDYVGenTL.cti – libDYVGenTL.so – pyokulo.cpython-36m-x86_64-linux-gnu.so – pyokulo.cpython-37m-x86_64-linux-gnu.so – pyokulo.cpython-38-x86_64-linux-gnu.so – pyokulo.cpython-39-x86_64-linux-gnu.so
Windows	– DYVGenTL.cti – DYVGenTL.lib – pyokulo.cp36-win_amd64.pyd – pyokulo.cp37-win_amd64.pyd – pyokulo.cp38-win_amd64.pyd – pyokulo.cp39-win_amd64.pyd
Linux/ Windows	– DYVInterface.xml – DYVRemoteDevice.xml – DYVTL.xml – normConf.xml – pclConf.xml – tofConf.xml

7.3.6 Attachment B

Additional Licensing Obligations The following third-party components included in the SOFTWARE are licensed to Licensee pursuant to the following terms and conditions:

- Licensee's use of the GDB third party component is subject to the terms and conditions of GNU GPL v3:

  This product includes copyrighted third-party software licensed under the terms of the GNU General Public License v3 ("GPL v3"). All third-party software packages are copyright by their respective authors. GPL v3 terms and conditions are hereby incorporated into the Agreement by this reference: http://www.gnu.org/licenses/gpl.txt

- Some of the library routines uses code from imgui and Open3D, which is subject to the following license:

  License URL https://github.com/isl-org/Open3D/blob/master/LICENSE

  License Text The MIT License (MIT)

  Open3D: www.open3d.org Copyright (c) 2018-2021 www.open3d.org

  Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- The GenICam runtime library is subject to the following license:

  Redistribution and use in source and binary forms, without modification, are permitted provided that the following conditions are met:

  – Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

  – Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

  – Neither the name of the GenICam standard group nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- The glfw is subject to the following license:

  Copyright (c) 2002-2006 Marcus Geelnard Copyright (c) 2006-2016 Camilla Löwy <elmindreda@glfw.org>

  This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

  2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

  3. This notice may not be removed or altered from any source distribution.

- The rc_genicam_api package is subject to the following license:

  License URL: https://github.com/roboception/rc_genicam_api/blob/master/LICENSE.md

The files in the sub-directories 'rc_genicam_api' and 'tools' are provided by Roboception GmbH under the 3-clause BSD license. See the license text in the header of all source files.

The sub-directory 'baumer' contains a GenTL producer for GigE Vision from Baumer. See baumer/license.txt for license information on this lib.

The sub-directory 'genicam' provides the full GenICam reference implementation. See genicam/License_ReadMe.txt for license information of all files in this sub-directory.

- The stb_image package is subject to the following license:

  This software is in the public domain. Where that dedication is not recognized, you are granted a perpetual, irrevocable license to copy and modify this file however you want.

- The tinyfiledialogs package is subject to the following license:

  This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

  2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

  3. This notice may not be removed or altered from any source distribution.

CHAPTER

EIGHT

ABOUT US

Point Spread Technology Co., Ltd. is committed to revolutionize computational photography, computational optics with its world-leading computational imaging technology. We vow to push forward imaging in the industrial fieds including consumer electronics, vehicle-mounted, and more to initiate the automotive optimization era in optics design and the joint-optimization for optics and image signal processing.

Point Spread Technology Co., Ltd. is located in China and have multiple branches in Shenzhen and Nantong. Please feel free to get help by contacting support@pointspread.cn.

- genindex
- modindex
- search