

# Access AKDK Application Software with Femto Bolt

## Access AKDK Application Software with Femto Bolt

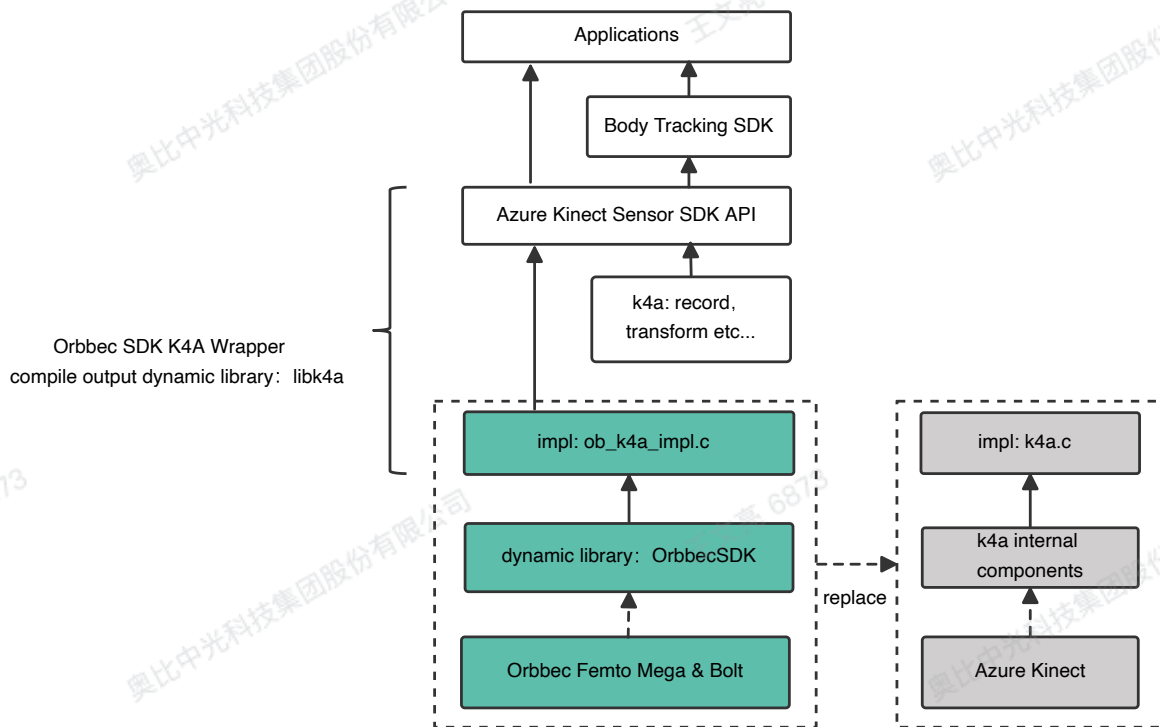
### 1. Overview

Orbbec SDK K4A Wrapper is designed and encapsulated based on Orbbec SDK, converting Orbbec SDK to Azure Kinect Sensor SDK interfaces. It mainly implements data stream reception, color parameter setting, D2C and point cloud functions, recording and playback, consistent APIs with Azure Kinect Sensor SDK, allowing users to quickly switch to Orbbec Femto Bolt and Orbbec Femto Mega cameras without modifying the application code.

Orbbec SDK K4A Wrapper open source link: <https://github.com/orbbec/OrbbecSDK-K4A-Wrapper>

The implementation principle of Orbbec SDK K4A Wrapper is as follows:

The characteristics of Orbbec SDK K4A Wrapper are as follows:



- Maintain the original interfaces of Azure Kinect Sensor SDK unchanged.
- Modify the implementation (impl) of Azure Kinect Sensor SDK C API, call Orbbec SDK internally to get video frames and control Femto Bolt and Femto Mega cameras.
- Coordinate transformation, D2C, C2D, point cloud reuse Azure Kinect Sensor SDK.

Orbbec SDK K4A Wrapper currently supports the following cameras:

Orbbec Femto Bolt:

Win10 x86/x64, Ubuntu18.04 x64, Ubuntu20.04 x64

Orbbec Femto Mega:

Win10 x86/x64, Ubuntu20.04 x64

## 2. How AKDK applications switch to Orbbec SDK K4A Wrapper

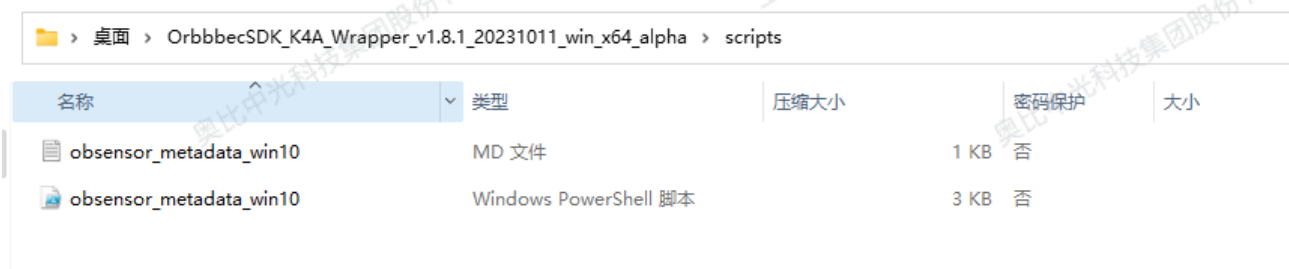
This chapter mainly shows how to implement code project switching and adaptation to Femto Bolt cameras by replacing Azure Kinect Sensor SDK library files and header files with Orbbec SDK K4A Wrapper under the user application code project.

For compiled application executables, you can also directly switch by replacing the libk4a library file linked by the application executable with Orbbec SDK K4A Wrapper.

## 2.1 Windows Platform Application

### 2.1.1 Environment Configuration: Modify Registry

Due to the Windows system mechanism, for UVC protocol devices, if you need to get timestamp and other metadata information, you need to register in the registry first. You can execute the `obsensor_metadata_win10.ps1` script to complete automatic registration according to the `obsensor_metadata_win10.md` document guide.



### 2.1.2 Replace Header Files

Orbbec SDK K4A Wrapper is developed based on Azure Kinect Sensor SDK V1.4.1 version. The header files directly use the original K4A library header files. If the user project originally used Azure Kinect Sensor SDK V1.4.1 version, this step can be ignored.

- Find the header files in Orbbec SDK K4A Wrapper



- Replace the corresponding header files in Azure Kinect Sensor SDK

此电脑 > 本地磁盘 (C:) > Program Files > Azure Kinect SDK v1.4.1 > sdk > include

名称	修改日期	类型	大小
k4a	2023/10/11 20:41	文件夹	
k4arecord	2023/10/11 20:37	文件夹	

### 2.1.3 Replace Library Files

#### Compile

Copy the k4a.lib and k4arecord.lib library files from Orbbec SDK K4A Wrapper to the corresponding Azure Kinect Sensor SDK path.

桌面 > OrbbecSDK\_K4A\_Wrapper\_v1.8.1\_20231011\_win\_x64\_alpha > lib >

名称	类型	压缩大小	密码保护	大小
cmake	文件夹			
k4a.lib	LIB 文件	4 KB	否	
k4arecord.lib	LIB 文件	3 KB	否	

Replace the corresponding k4a.lib and k4arecord.lib library files in Azure Kinect Sensor SDK to compile the application program with Orbbec SDK K4A Wrapper.

此电脑 > 本地磁盘 (C:) > Program Files > Azure Kinect SDK v1.4.1 > sdk > windows-desktop > x86 > release > lib

名称	修改日期	类型	大小
k4a.lib	2020/6/15 22:45	LIB 文件	18 KB
k4arecord.lib	2020/6/15 22:45	LIB 文件	13 KB

#### Run

Find k4a.dll, k4arecord.dll, OrbbecSDK.dll, depthengine\_2\_0.dll library files in Orbbec SDK K4A Wrapper

桌面 > OrbbecSDK\_K4A\_Wrapper\_v1.8.1\_20231011\_win\_x64\_alpha > bin

名称	类型	压缩大小	密码保护	大小
k4arecorder	应用程序	28 KB	否	67 KB
k4aviewer	应用程序	622 KB	否	1,593 KB
depthengine_2_0.dll	应用程序扩展	92 KB	否	347 KB
k4a.dll	应用程序扩展	110 KB	否	253 KB
k4arecord.dll	应用程序扩展	272 KB	否	871 KB
OrbbecSDK.dll	应用程序扩展	2,404 KB	否	7,279 KB

Copy the above library files to the original executable application path of Azure Kinect Sensor SDK, and replace the original k4a.dll, k4arecord.dll, depthengine\_2\_0.dll library files to run the application program.

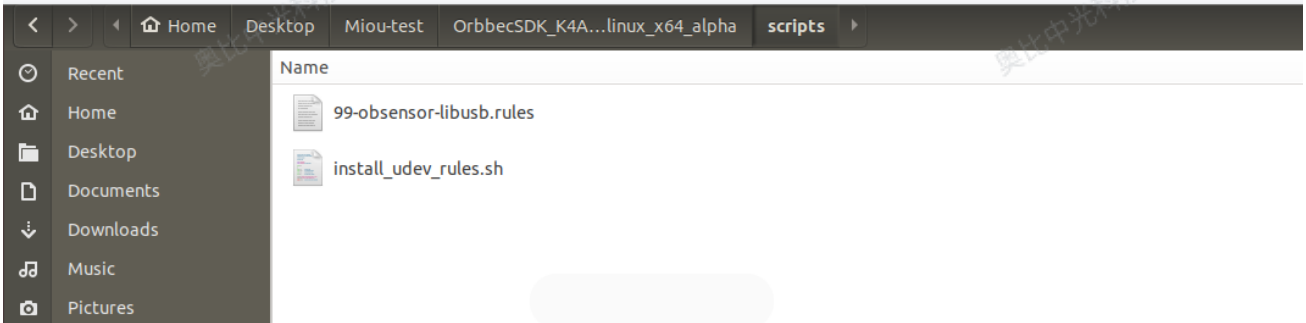
此电脑 > 本地磁盘 (C:) > Program Files > Azure Kinect SDK v1.4.1 > tools

名称	修改日期	类型	大小
k4arecord.dll	2023/10/11 15:51	应用程序扩展	871 KB
k4a.dll	2023/10/11 15:51	应用程序扩展	253 KB
OrbbecSDK.dll	2023/10/10 20:27	应用程序扩展	7,279 KB
depthengine_2_0.dll	2023/10/9 16:39	应用程序扩展	347 KB
k4arecorder	2020/6/15 23:03	应用程序	132 KB
AzureKinectFirmwareTool	2020/6/15 23:03	应用程序	439 KB
k4aviewer	2020/6/15 23:03	应用程序	2,348 KB
k4aviewer.pdb	2020/6/15 22:59	PDB 文件	14,692 KB
k4a.pdb	2020/6/15 22:59	PDB 文件	6,292 KB
k4arecord.pdb	2020/6/15 22:59	PDB 文件	12,996 KB
k4arecorder.pdb	2020/6/15 22:59	PDB 文件	2,044 KB
AzureKinectFirmwareTool.pdb	2020/6/15 22:59	PDB 文件	4,972 KB
firmware	2023/10/11 20:37	文件夹	

## 2.2 Linux Platform Application

### 2.2.1 Environment Configuration: Install udev rules Configuration

By default, Linux systems require root permissions for direct access to USB devices, which can be solved through the rules configuration file. Orbbec SDK K4A Wrapper provides a 99-obsensor-libusb.rules configuration file and install\_udev\_rules.sh installation script to complete the installation by executing the install\_udev\_rules.sh script.



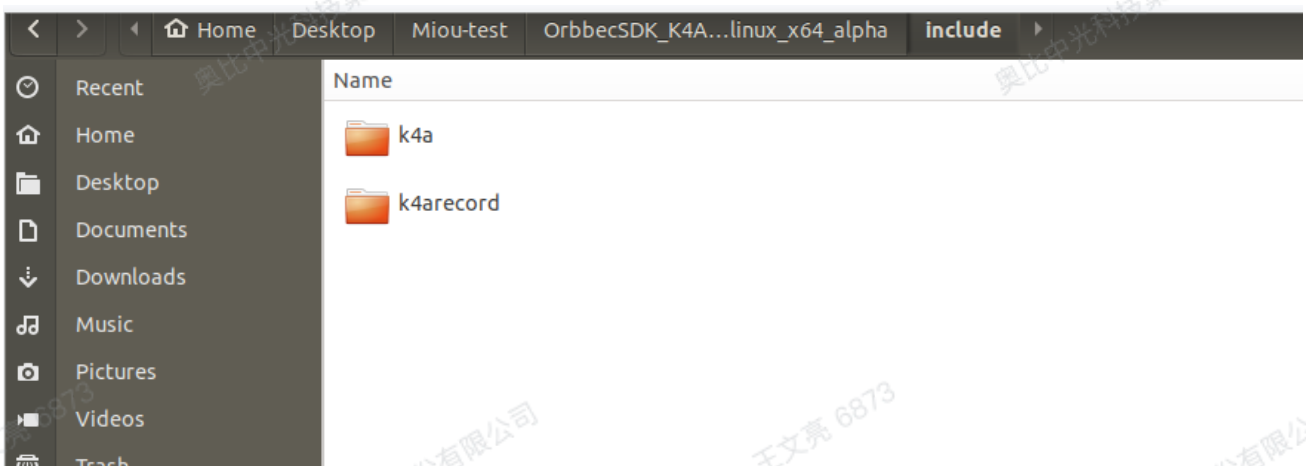
Execution method:

```
sudo chmod +x ./install.sh # Make sure the installation script is executable
sudo ./install.sh # Execute the script with sudo
```

### 2.2.2 Replace Header Files

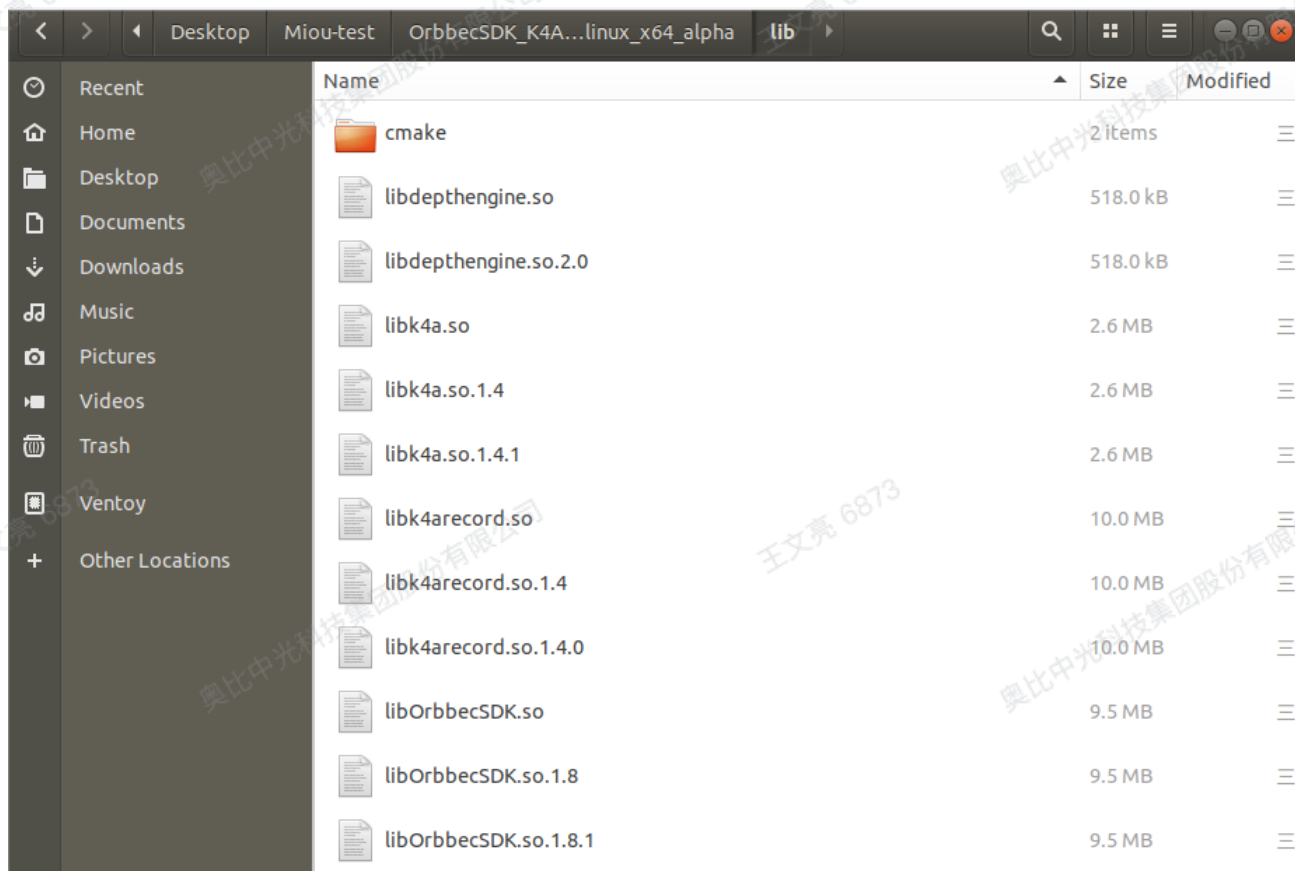
*Orbbec SDK K4A Wrapper is developed based on Azure Kinect Sensor SDK V1.4.1 version. The header files directly use the original K4A library header files. If the user project originally used Azure Kinect Sensor SDK V1.4.1 version, this step can be ignored.*

Find the original header files of Orbbec SDK K4A Wrapper and replace the corresponding header files under Azure Kinect Sensor SDK.

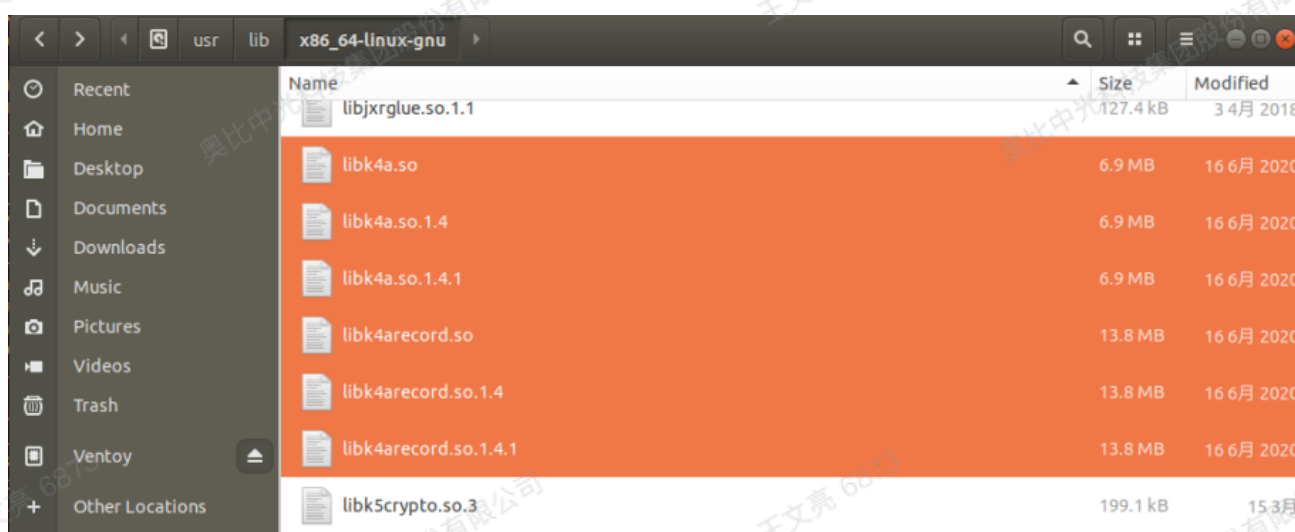


### 2.2.3 Replace Library Files

Find the library files in Orbbec SDK K4A Wrapper, including: libdepthengine, libk4a, libk4arecord, libOrbbecSDK.so.



Copy the above library files to the /usr/lib/x86\_64-linux-gnu directory to replace the Azure Kinect Sensor SDK libraries to compile and use the application program with Orbbec SDK K4A Wrapper.



**Note:**

On Linux platforms, the depth engine library requires OpenGL. If the application layer also uses OpenGL and makes cross-thread calls to OpenGL instances, context conflicts may occur, causing problems during depth engine initialization and inability to start the depth data stream properly.

If the application layer uses OpenGL rendering, the application layer needs to switch the Context under OpenGL:

([https://www.khronos.org/opengl/wiki/OpenGL\\_and\\_multithreading](https://www.khronos.org/opengl/wiki/OpenGL_and_multithreading))

The following is our solution using the glfw library:

- a. Call `glfwMakeContextCurrent(NULL)` before starting the stream.
- b. Then start the stream.
- c. After the stream is finished, `glfwMakeContextCurrent(currentContext)`.

### **3. AKDK User Skeleton Algorithm Adaptation**

The following uses the Azure Kinect Body Tracking SDK Sample as an example to describe how to obtain skeleton data by using Orbbec Femto Bolt camera data after replacing Azure Kinect Sensor SDK through Azure Kinect Body Tracking SDK (hereinafter referred to as K4ABT). (Users can switch by directly replacing the library files without recompiling)

#### **3.1 Windows Platform**

##### **3.1.1 Skeleton Installation Package Download and Installation**

Complete the installation according to Microsoft's installation documentation:

<https://learn.microsoft.com/en-us/azure/kinect-dk/body-sdk-setup>

Download Azure Kinect Body Tracking SDK 1.1.2.msi, then execute the file to complete the installation.

- by title
- Kinect DK documentation
- w
- arts
- p Azure Kinect DK
- rd sensor streams to a file
- your first application
- p **Body Tracking SDK**
- your first body tracking application
- ts
- guides
- es
- ces

Download and install the latest NVIDIA driver for your graphics card. Older drivers may not be compatible with the CUDA binaries redistributed with the body tracking SDK.

### Visual C++ Redistributable for Visual Studio 2015

Download and install Visual C++ Redistributable for Visual Studio 2015.

### Set up hardware

#### Set up Azure Kinect DK

Launch the [Azure Kinect Viewer](#) to check that your Azure Kinect DK is set up correctly.

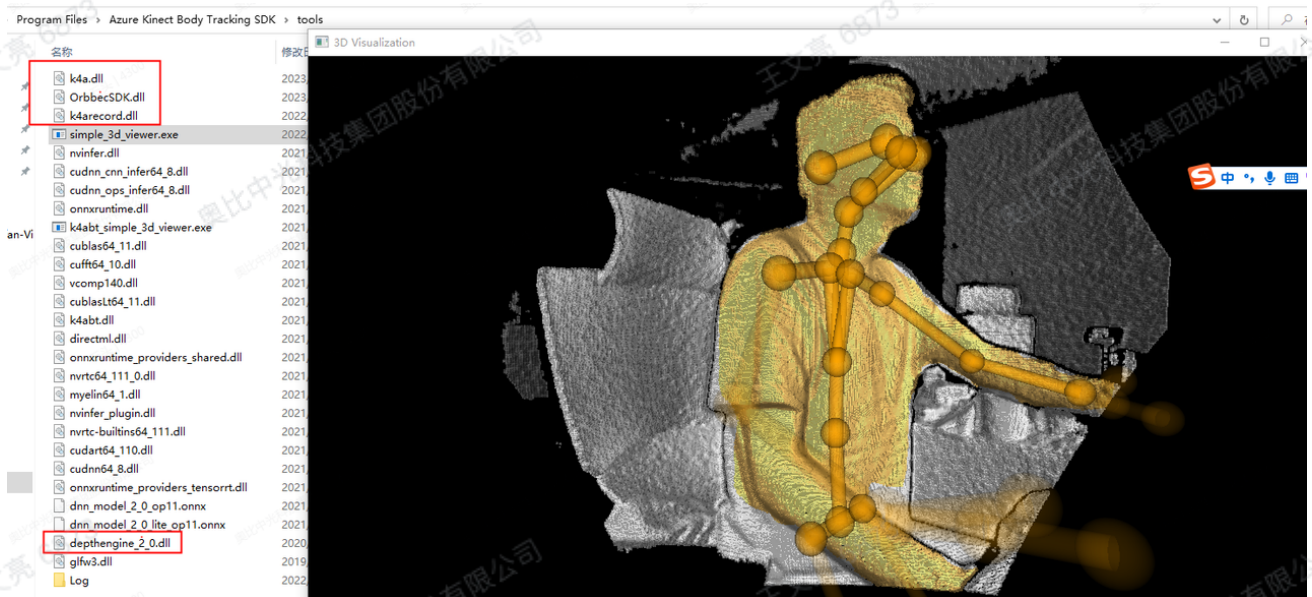
### Download the Body Tracking SDK

1. Select the link to [Download the Body Tracking SDK](#)
2. Install the Body Tracking SDK on your PC.

### 3.1.2 Demonstrate Skeleton Algorithm Effects

Complete the environment configuration (metadata registration) according to Chapter 2, then replace the following libraries of Azure Kinect Sensor SDK in the installation directory with Orbbec SDK K4A Wrapper libraries (k4a.dll, OrbbecSDK.dll, k4arecord.dll, depthengine\_2\_0.dll), then run simple\_3d\_viewer.exe with administrator rights.

The effect of the skeleton algorithm is as follows:



### 3.1.3 Skeleton Algorithm Secondary Development (sample compilation)

#### 1. Download Azure Kinect Samples

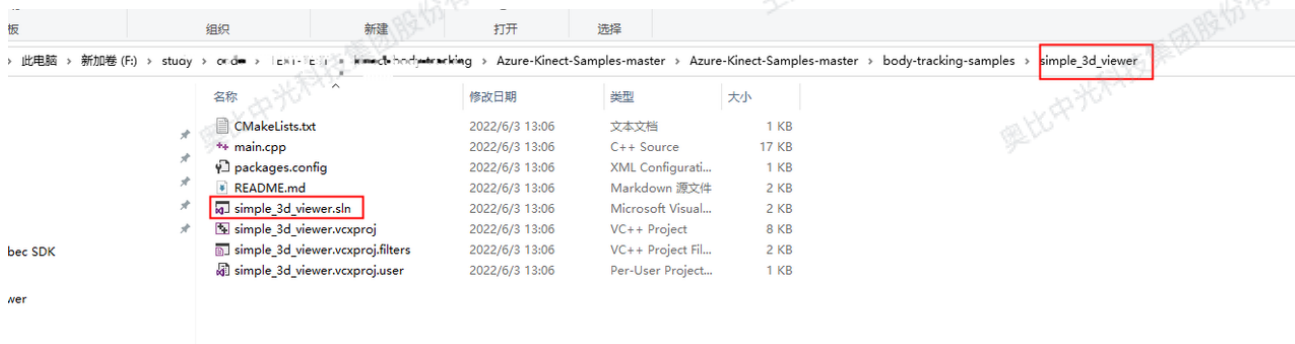
URL: <https://github.com/microsoft/Azure-Kinect-Samples>

Clone code:

```
git clone https://github.com/microsoft/Azure-Kinect-Samples.git
```

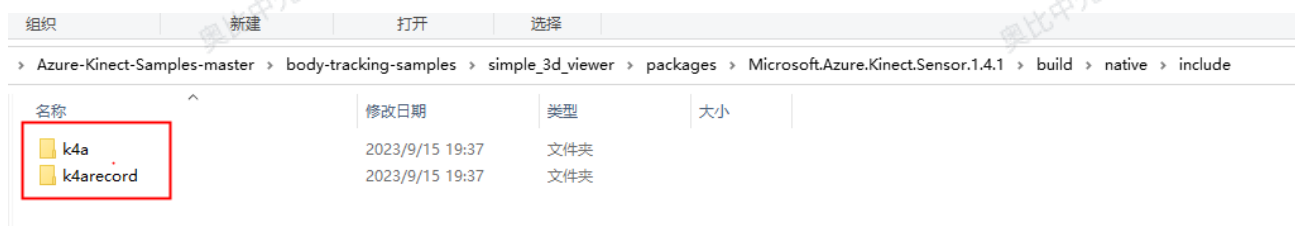
#### 2. Open Microsoft skeleton sample with Visual Studio

Microsoft skeleton sample only supports opening with visual studio. Use VS2019 to open the following projects.

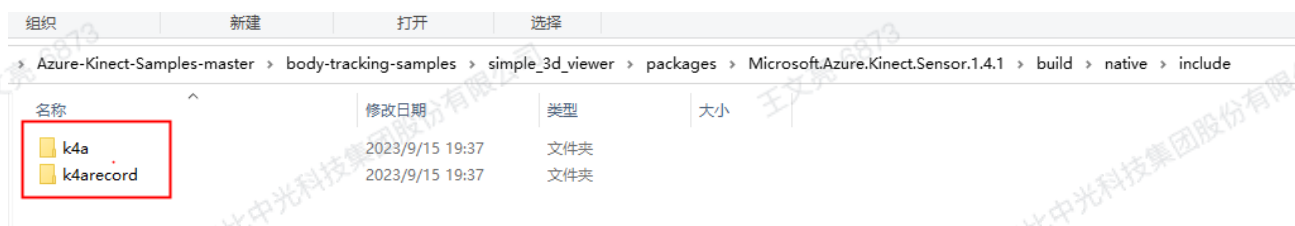


#### 3. Use Orbbec SDK K4A Wrapper header files and library files to replace Azure Kinect Sensor SDK header files and library files. (k4a.dll, OrbbecSDK.dll, k4arecord.dll, depthengine\_2\_0.dll)。

##### a. Replace header files



##### b. Replace library files



#### 4. Compile & Run

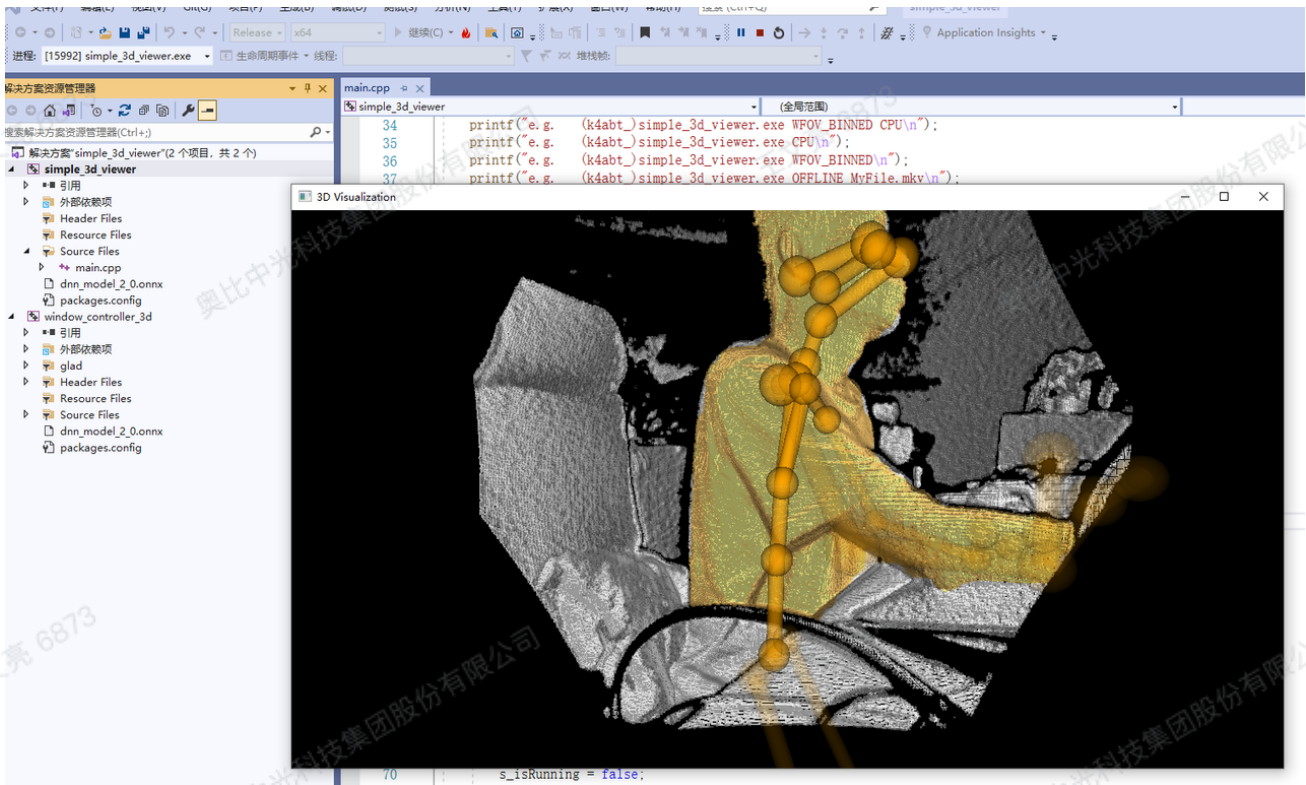
a. The simple\_3d\_viewer.exe example relies on the following 2 libraries, which are copied from the installation package to the bin directory where the compiled application is generated.

组织	新建	打开	选择
本地磁盘 (C:) > Program Files > Azure Kinect Body Tracking SDK > tools			
名称	修改日期	类型	大小
cudnn_cnn_infer64_8.dll	2022/5/12 19:36	应用程序扩展	614,886 KB
cudnn_ops_infer64_8.dll	2022/5/12 19:35	应用程序扩展	354,471 KB
nvinfer.dll	2022/5/12 19:35	应用程序扩展	353,204 KB
cufft64_10.dll	2022/5/12 19:35	应用程序扩展	353,061 KB
cublasLt64_11.dll	2022/5/12 19:35	应用程序扩展	275,435 KB
onnxruntime_providers_cuda.dll	2022/5/12 19:36	应用程序扩展	248,619 KB
dnn_model_2_0_op11.onnx	2022/5/12 18:32	ONNX 文件	162,870 KB
cublas64_11.dll	2022/5/12 19:35	应用程序扩展	139,197 KB
nvrvc64_112_0.dll	2022/5/12 19:35	应用程序扩展	31,848 KB
nvinfer_plugin.dll	2022/5/12 19:35	应用程序扩展	25,441 KB
onnxruntime.dll	2022/5/12 19:34	应用程序扩展	11,714 KB
directml.dll	2022/5/12 19:34	应用程序扩展	9,265 KB
nvrvc-builtins64_114.dll	2022/5/12 19:34	应用程序扩展	6,907 KB
OrbbecSDK.dll	2023/3/11 19:03	应用程序扩展	4,578 KB

b. Copy the dependent libraries to the running directory:

组织	新建	打开	选择
fy > code > TEST-TEST > kinect-body-tracking > Azure-Kinect-Samples-master > Azure-Kinect-Samples-master > body-tracking-samples > simple_3d_viewer > build > bin > Release			
名称	修改日期	类型	大小
dnn_model_2_0_op11.onnx	2022/5/13 2:32	ONNX 文件	162,870 KB
dnn_model_2_0_lite_op11.onnx	2022/5/13 2:32	ONNX 文件	44,070 KB
k4arecord.pdb	2020/6/16 6:46	Program Debug...	12,996 KB
onnxruntime.dll	2022/5/12 19:34	应用程序扩展	11,714 KB
directml.dll	2022/5/12 19:34	应用程序扩展	9,265 KB
window_controller_3d.lib	2023/9/15 21:56	Object File Library	6,441 KB
k4a.pdb	2020/6/16 6:46	Program Debug...	6,292 KB
OrbbecSDK.dll	2023/3/11 19:03	应用程序扩展	4,578 KB
k4abt.dll	2022/5/13 3:36	应用程序扩展	4,414 KB
simple_3d_viewer.pdb	2023/9/15 21:56	Program Debug...	1,716 KB
k4arecord.dll	2023/3/11 19:03	应用程序扩展	868 KB
window_controller_3d.pdb	2023/9/15 21:56	Program Debug...	788 KB
depthengine_2_0.dll	2020/6/16 7:03	应用程序扩展	414 KB
k4a.dll	2023/3/11 19:03	应用程序扩展	246 KB
glfw3.dll	2019/4/16 8:23	应用程序扩展	221 KB
simple_3d_viewer.exe	2023/9/15 21:56	应用程序	132 KB

c. Compilation running interface:



## 3.2 Running Kinect Skeleton Algorithm on Linux

Since Microsoft Azure Kinect Sensor SDK libraries only provide installation on Ubuntu 18.04, it is recommended to complete the following on Ubuntu 18.04 system.

### 3.2.1 Install Azure Kinect Sensor SDK

Refer to the Linux installation instructions section of Microsoft's installation documentation to complete the installation. Document link:

[Azure Kinect Sensor SDK download | Microsoft Learn](#)

Installation instructions:



```
curl -sSL -O https://packages.microsoft.com/config/ubuntu/18.04/packages-
microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
sudo apt-get update
sudo apt-get install libk4a1.4-dev
sudo apt-get install k4a-tools
```

### 3.2.2 Install Azure Kinect Body Tracking SDK

Refer to the Linux installation instructions section of Microsoft's installation documentation to complete the installation. Document link:

[Azure Kinect Body Tracking SDK download | Microsoft Learn](#)

The following instructions were executed during installation of Azure Kinect Sensor SDK and do not need to be repeated:

Install libk4abt:

```
curl -sSL -O https://packages.microsoft.com/config/ubuntu/18.04/packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
sudo apt-get update
sudo apt install libk4abt1.1-dev
```

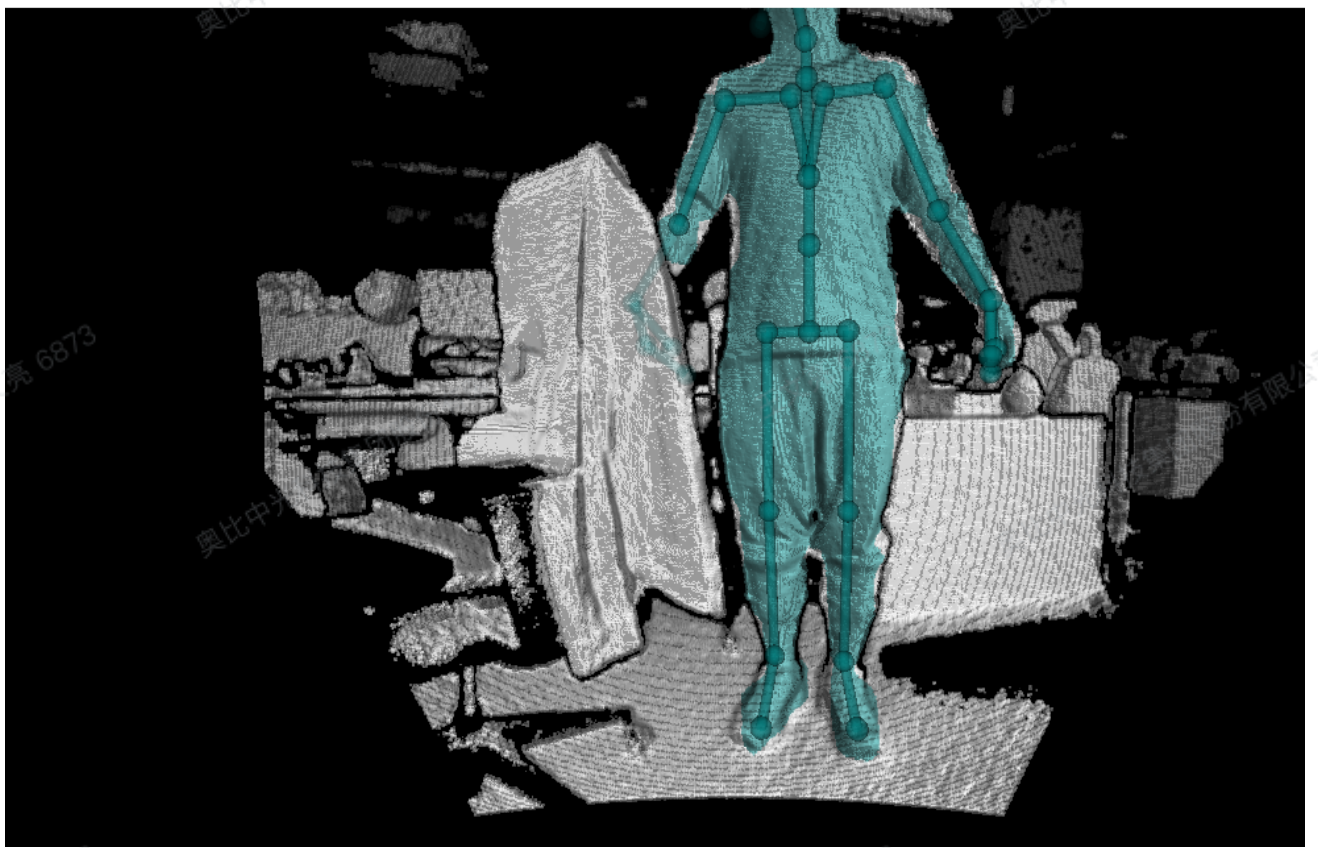
### 3.2.3 Use Orbbec SDK K4A Wrapper to Replace Azure Kinect Sensor SDK Library

Complete the environment configuration (udev rules script installation) according to Chapter 2, then replace the Azure Kinect Sensor SDK library files with the library files in the Orbbec SDK K4A Wrapper package (libk4a.so, libOrbbecSDK.so, depthengine2). After replacing and connecting the Orbbec camera, enter the simple\_3d\_viewer command in the terminal to see the following running effect:

```

[I20230919 09:17:42.261677 21304 FrameBufferManager.cpp:54] Frame buffer released=0.351989MB, total usage: {al
.3681MB, max limit=2048MB}
[I20230919 09:17:42.275681 21304 FrameBufferManager.hpp:58] FrameBufferManager created! @class libobsensor::Fr
anager<class libobsensor::ColorFrame>, obj addr:1961987970448, frame obj total size:0.351989MB
[I20230919 09:17:42.275681 21304 FrameBufferManager.cpp:123] ColorFrame bufferManager created!
[I20230919 09:17:42.275681 21304 FrameBufferManager.cpp:33] New frame buffer allocated=0.351989MB, total usage
ed=13.7201MB, max limit=2048MB}
[I20230919 09:17:42.288677 19828 FrameTimestampAdjuster.cpp:51] updateBaseTimeStamp:
[I20230919 09:17:42.288677 19828 FrameTimestampAdjuster.cpp:52] srcTimeStamp=0, prevSrcTsp_=0, tspDecr
[I20230919 09:17:42.288677 19828 FrameTimestampAdjuster.cpp:53] srcTspDiffMs=0, bestTspDiffMs=24, tsp

```



#### 4. Differences between Orbbec SDK K4A Wrapper and Azure Kinect Sensor SDK

##### 1. Functional points with differences

No.	Function Point	Orbbec SDK K4A Wrapper	Azure Kinect Sensor SDK	Impact on Application
-----	----------------	------------------------	-------------------------	-----------------------

1	Recording	<pre> c++ typedef struct _k4a_re cord_configura tion_t { /** * The timestam p offset of th e start of th e recording. A ll recorded ti mestamps are o ffset by this value such tha t * the record ing starts at timestamp 0. T his value can be used to syn chronize times tamps between 2 recording fi les. */ uint64 _t start_times tamp_offset_us ec; } k4a_reco rd_configurati on_t; </pre>	<pre> c++ typedef struct _k4a_re cord_configura tion_t { /** * The timestam p offset of th e start of th e recording. A ll recorded ti mestamps are o ffset by this value such tha t * the record ing starts at timestamp 0. T his value can be used to syn chronize times tamps between 2 recording fi les. */ uint32 _t start_times tamp_offset_us ec; } k4a_reco rd_configurati on_t; </pre>	Need to replace and recompile the header file of Orbbec SDK K4a Wrapper
---	-----------	---	---	---

**2. Unimplemented interfaces in Orbbec SDK K4A Wrapper (return empty value or exception state)**

No.	Azure Kinect Sensor SDK Interface Meaning	Impact of Differences
-----	---	-----------------------

1	<pre>c++ k4a_result_t k4a_session_t_allocator(k4a_memory_allocator_cb_t allocate, k4a_memory_destroy_cb_t free)</pre> <p>Pass in external user-defined memory manager for SDK internal memory application</p>	<p>User cannot use their custom memory manager for SDK internal use. Basic functioning of SDK itself is barely affected.</p>
2	<pre>c++ void k4a_capture_session_t_temperature_c(k4a_capture_handle_t capture_handle, float temperature_c)</pre> <p>Set temperature information for capture</p>	<p>User cannot modify this value, impacting storage of custom values</p>
3	<pre>c++ float k4a_capture_session_t_temperature_c(k4a_capture_handle_t capture_handle)</pre> <p>Get temperature information for capture</p>	<p>User cannot get this value, impacting algorithms or applications that rely on this value</p>
4	<pre>c++ void k4a_image_session_t_image_handle, uint64_t exposure_usec)</pre> <p>Set exposure value for image</p>	<p>User cannot modify this value, impacting storage of custom values</p>

5	<pre>c++ void k4a_image_set_ white_balance(k4a_image_ t image_handle, uint32_t white_balance)</pre> <p><b>Set white balance value for image</b></p>	
6	<pre>c++ void k4a_image_set_ iso_speed(k4a_image_t ima ge_handle, uint32_t iso_s peed)</pre> <p><b>Set ISO speed for image</b></p>	
7	<pre>c++ uint64_t k4a_image_ get_exposure_usec(k4a_ima ge_t image_handle)</pre> <p><b>Get exposure value for image</b></p>	
8	<pre>c++ uint32_t k4a_image_ get_white_balance(k4a_ima ge_t image_handle)</pre> <p><b>Get white balance value for image</b></p>	User cannot get this value, impacting algorithms or applications that rely on this value
9	<pre>c++ uint32_t k4a_image_ get_iso_speed(k4a_image_ t image_handle)</pre> <p><b>Read ISO speed for image</b></p>	

<p>10</p>	<pre> c++ k4a_result_t k4a_device_get_sync_jack(k4a_device_t device_handle, bool *sync_in_jack_connected, bool *sync_out_jack_connected) </pre> <p>Get sync cable connection status for device</p>	<p>User application cannot rely on this interface to determine multi-camera sync cable linkage status</p>
-----------	--	---