LIPSense™ 3D Body Pose SDK User's Guide

V3.5.1

Although every effort has been made to ensure the accuracy of this manual, errors and inconsistencies may remain. The manufacturer assumes no liability resulting from errors or omission in this manual, even if damages arise from the use of the information. All contents are subject to constant revision to improve its reliability and may be changed without prior notice.

September 2023

Contents

Notes for Programmers

I. System Requirements

For optimal performance, it is recommended to use the following operating system configuration.

Minimum requirement	
CPU	6th Generation Intel® Core™ processor
RAM	2 GB RAM or above
GPU (Tensor RT 8.5)	NVIDIA GTX 750 Ti
Recommended requirement	
CPU	8th generation Intel Core™ processor
RAM	4 GB RAM or above
GPU	Intel® HD Graphics 630 (with Intel® OpenVINO™ compatibility)
GPU (Tensor RT 8.5)	GTX1050 Ti
Interfaces	Gigabit Ethernet Interface USB-C 3.0 or higher
Operating systems	Windows 10 64-bit System Ubuntu 18.04 / 20.04 / 22.04 LTS System

Note: LIPSense™ 3D Body Pose do not support **ARM processor**.

II. Hardware Compatibility

The LIPSense™ 3D Body Pose SDK is compatible with the following 3D cameras:

Series	Models
LIPSedge™	AE400
	AE450
	DL (Linux only)
Intel® RealSense™	Depth Camera D415
	Depth Camera D435
	Depth Camera D435i
	Depth Camera D455

III. Intel® RealSense™ Models Image Registration Issues

The performance of image registration in the Intel® RealSense™ D400 series varies across different models due to differences in optical properties, such as the camera's field of view (FoV). During the image registration process, the 3D depth image is typically transformed to align with the RGB image. Consequently, there will always be some differences between the transformed 3D depth image and the original depth image. Based on experiments conducted by LIPS Corp., no significant issues were observed in the skeletal detection results with all supported models of the Intel® RealSense™ D400 series.

IV. External Dependency Issues

The LIPSense™ 3D Body Pose SDK utilizes third-party libraries, including **Intel® RealSense™ SDK**, **NVIDIA® TensorRT**, **Intel® OpenVino**, and **OpenCV**. Consequently, the functionality of the LIPSense™ 3D Body Pose SDK depends on the conditions and performance of these external libraries. If you encounter any errors during the development process, it is recommended to refer to the release notes of the respective external libraries for debugging purposes.

Library	Release Notes
Intel® RealSense™ SDK V 2.50.0	https://github.com/IntelRealSense/librealsense/wiki/Release-Notes (Scroll down to find the Intel® RealSense™ SDK 2.50.0 section)
NVIDIA® TensorRT V8.5.3	https://docs.nvidia.com/deeplearning/tensorrt/release-notes/index.html#rel-8-5-3
Intel® OpenVino 2022.1	https://github.com/openvinotoolkit/openvino/releases/tag/2022.1.0
OpenCV V4.5.5	https://github.com/opencv/opencv/wiki/ChangeLog#version455

V. Python Support

For Linux, use the default Python 3 package in the Ubuntu repository. The Python support for the LIPSense™ 3D Body Pose SDK differs based on the operating system. To use the SDK on Windows, download and install Python 3.6. On Linux, it is recommended to utilize the default Python 3 package available in the Ubuntu repository.

Operating Systems		Supported Python Versions
Windows		3.6
Linux	Ubuntu 18.04	3.6
	Ubuntu 20.04	3.8
	Ubuntu 22.04	3.10

VI. Unity Support

The LIPSense™ 3D Body Pose SDK supports Unity exclusively for the Windows version. We recommend utilizing Unity 2020.3 for optimal compatibility. Proceed to download and install Unity accordingly.

LIPSense™ 3D Body Pose SDK

User’s Guide

Welcome to the LIPSense™ 3D Body Pose SDK User's Guide! This comprehensive document offers a detailed, step-by-step guide on how to effectively utilize the LIPSense™ 3D Body Pose SDK and set up the development platform on your PC or laptop.

1. Overview

The LIPSense™ 3D Body Pose SDK offers an exceptional full-body skeletal-tracking feature designed for advanced interactive technologies. With its powerful 3D depth camera, the SDK provides robust skeletal-tracking capabilities, allowing for the tracking of up to 18 joints. Impressively, the SDK enables real-time tracking of multiple individuals, with support for up to 10 people simultaneously. Our proprietary algorithm has been meticulously refined to accurately capture the nuances of a moving human body, including the intricacies of swinging and staggering, ensuring optimal skeletal detection performance.

The LIPSense™ 3D Body Pose SDK leverages the power of the Intel® OpenVINO™ toolkit, enabling developers to harness a wide range of features across multiple platforms. These include CNN-based deep learning capabilities, cross-platform hardware acceleration through a unified API, and optimized code for OpenCV integration. To enhance both the visualization experience and the SDK's CPU usage efficiency, the SDK provides support for various inference engine optimizers, such as Intel® OpenVINO™ or TensorRT, depending on your graphics cards.

Features

- Full-body skeletal tracking
- Max. joint-tracking capability of 18 joints (eyes * 2, nose, ears * 2, neck, shoulder * 2, elbow * 2, wrist * 2, hip * 2, knee * 2, ankle * 2)
- Max. people-tracking capability of 10 people
- Proprietary algorithm for optimal skeletal detection performance
- Capture of swinging and staggering motions
- Support for Unity and multiple programming languages / platforms
- Support for multiple 3D camera mechanical positions (90° clockwise / 90° counterclockwise / 180°)
- Intel® OpenVINO™ compatibility for Intel® HD graphics inference engine acceleration
- NVIDIA® TensorRT compatibility for NVIDIA® graphics inference engine acceleration
- Optimized Intel® HD graphics support
- Compatibility with LIPSedge™ AE series
- Compatibility with LIPSedge™ DL (Linux only)
- Compatibility with Intel® RealSense™ D400 series

1.2 Camera Accessories

Before proceeding with the installation process, ensure you have the following accessories (sold separately) prepared:

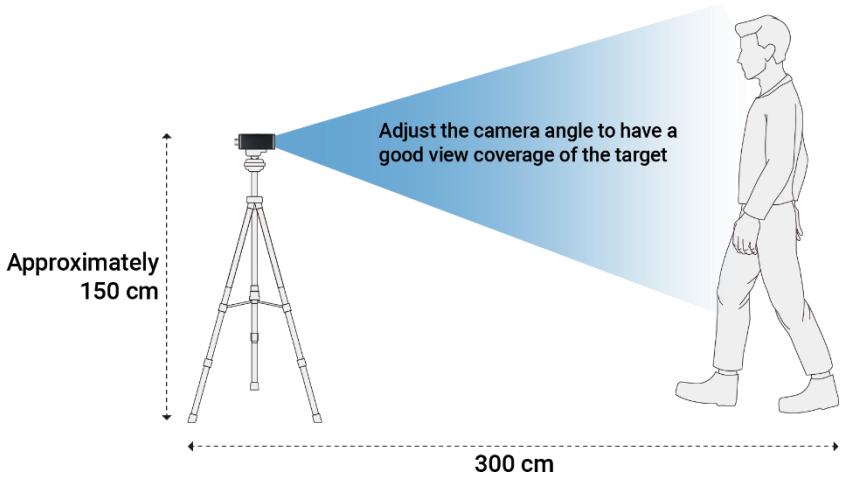
- LIPSedge™ DL 3D Camera / LIPSedge™ AE Series Ruggedized 3D Camera / Intel® RealSense™ Depth Camera D400 series
- Camera tripod
- PC/Laptop with Intel® OpenVINO™ compatible CPU, Intel® integrated graphics card, or NVIDIA graphics card

For additional accessories related to the LIPSedge™ AE series, refer to *section 1.3 Camera Accessories in the LIPSedge™ AE400 / AE450 User's Guide*.

2. Detection Requirements

2.1 Hardware Requirements

To ensure optimal recognition results, it is important to meet the following requirements:



- **Camera Working Distance:** Maintain an approximate distance of 3 meters (300 cm) between the camera and the recognition target.
- **Camera Position:** Install the camera on a **tripod**, ensuring that its **field of view (FOV) encompasses the full body of the target**. The LIPSense™ 3D Body Pose SDK supports various mechanical positions, including 90° clockwise, counterclockwise, and 180° rotations in addition to the recommended position of the Intel® RealSense™ Depth Camera D400.
- **Camera Height:** Position the camera at **around 150 cm** above the ground.
- **Illumination:** Ensure sufficient lighting during skeleton detection. AVOID installing the camera in locations with **direct sunlight** or complete darkness.
- **Background:** Keep the background of the recognition target as simple as possible, such as a clean white background, to minimize depth noises.

By complying with these requirements, you can enhance the recognition results and overall system performance. Once all the criteria are fulfilled, connect the camera to your PC / laptop.

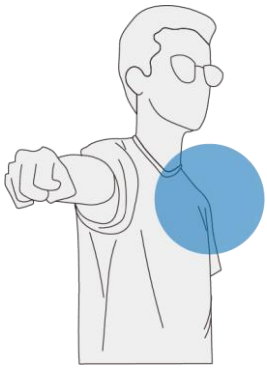
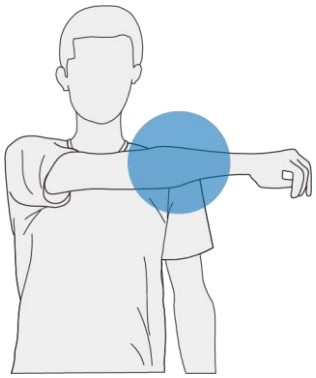
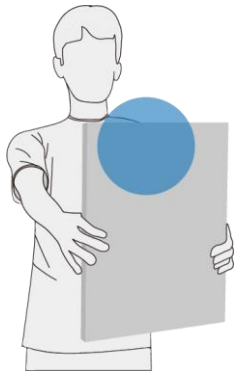
- For LIPSedge™ AE series users, refer to section *2. Hardware Installation* in the LIPSedge™ ***LIPSedge™ AE400 / AE450 User's Guide*** for detailed installation instructions.
- For LIPSedge™ DL users, refer to section *2. Hardware Installation* in the **LIPSedge DL SDK User's Guide** for step-by-step installation instructions.
- For Intel® RealSense™ Depth Camera users, refer to the **Get Started with Real Sense Depth Camera** for instructions on installation and setup.

Following these resources will assist you in successfully setting up the camera for optimal usage with the LIPSense™ 3D Body Pose SDK.

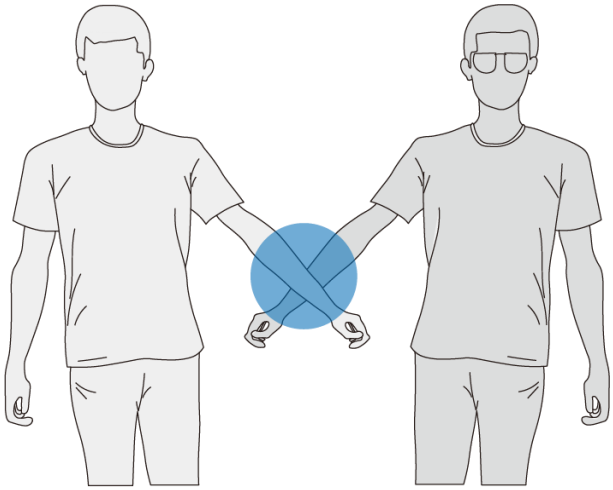
2.2 Pose Requirements

To optimize recognition results, it is recommended to avoid poses where obstacles obstruct or overlap with the recognition target's joints, as this can lead to suboptimal recognition accuracy.

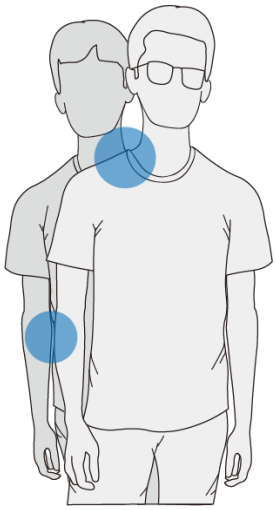
Arm positions (single target)

In these scenarios, when the shoulder joints are obstructed by another limb or an obstacle, it can result in inaccurate tracking outcomes.		
		

Arm positions (multiple target)

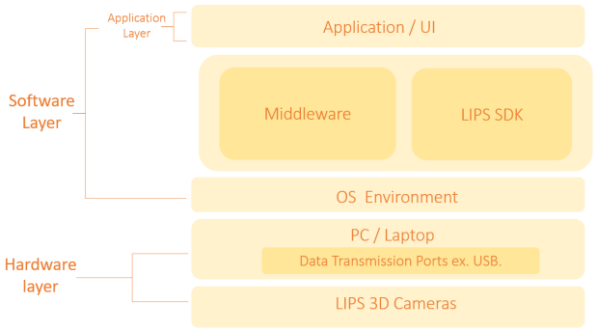
In this case, one person's arm is blocking another person's wrist.


In these cases, the joints are overlapping with each other.

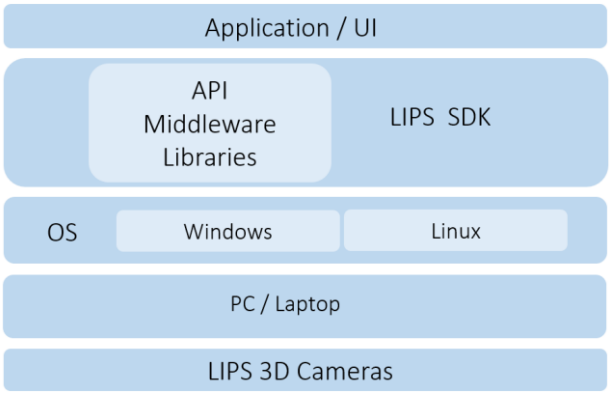


3. System Architecture

The LIPS 3D camera / SDK provides a comprehensive system for developing depth-sensing applications. The system architecture of LIPS consists of two main layers: the hardware layer and the software layer. The hardware layer is responsible for data capture, transfer, and processing. It handles the necessary functions related to capturing depth data from the environment. On the other hand, the software layer involves the LIPS SDK (Software Development Kit) running on the operating system environment. The SDK fetches the captured data from the hardware layer and provides the necessary tools and functions for developers to work with. Depending on the complexity of the project, additional wrappers and third-party utilities may be utilized to enhance the data processing capabilities. Finally, the processed data is presented in the application layer, where it can be utilized for various business applications.



The central component of the system, the LIPS SDK, can be likened to a toolbox equipped with software modules and APIs specifically designed for application development. With the support of the LIPS SDK, developers gain access to low-level data through APIs, eliminating the need to rely on third-party functions. This streamlined approach allows for efficient project scoping, monitoring, and execution workflows that are compatible with the rapidly evolving AIoT market and the demands of machine vision applications.



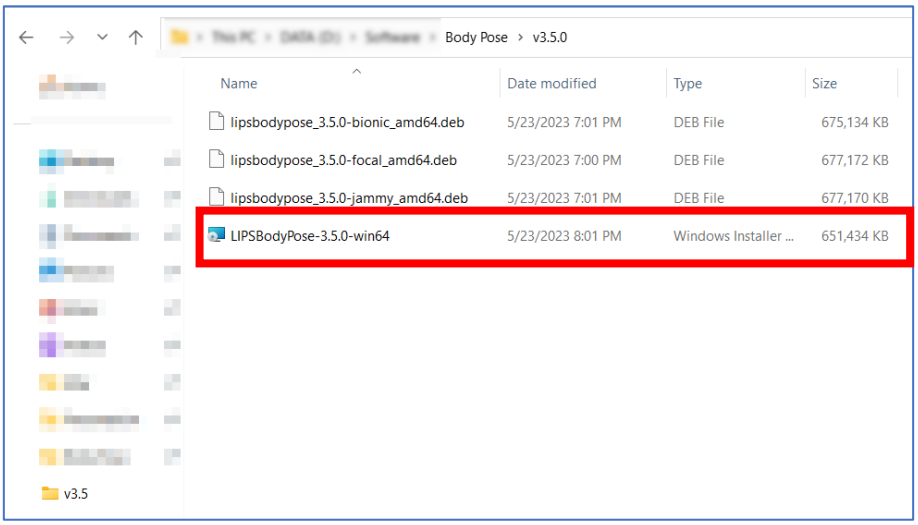
LIPS
Sense The World

Windows

4. SDK Installation

LIPS Corp. has released versions of the LIPSense™ 3D Body Pose SDK for specific systems on the official LIPS Corp. websites. You can download and install the LIPSense™ 3D Body Pose SDK according to your host platform.

1. Go to our website: https://www.lips-hci.com/developer-documentation.
2. On the menu, select the mode LIPSense™ 3D Body Pose Middleware.
3. Select and download the installation file according to your operating system.
4. Unzip the downloaded file and double-click the **LIPSBodyPose-vx.y.z-win64.msi** to install the SDK.



5. Click **Next**.

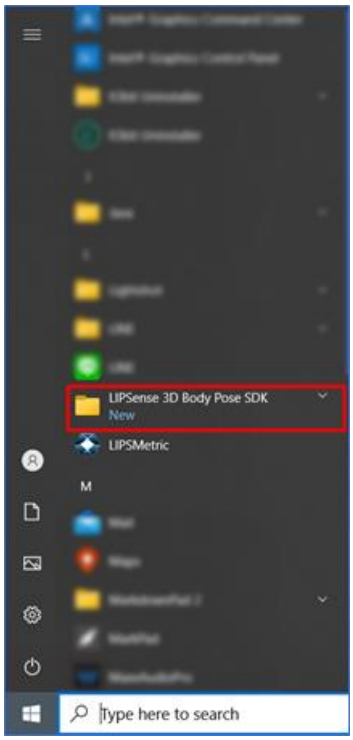
6. Review the license agreement and click **Next**.

7. Choose a **location** for the LIPSense™ 3D Body Pose SDK and click **Next**.

8. Click **Install**.

9. Click **Finish**.

10. Once the installation is completed, you can access LIPSense™ 3D Body Pose SDK though the **Windows Start Menu**.



After you have installed LIPSense™ 3D Body Pose SDK, additional configuration is needed for smooth operation of LIPSense™ 3D Body Pose SDK. For details, refer to *Additional Configurations*.

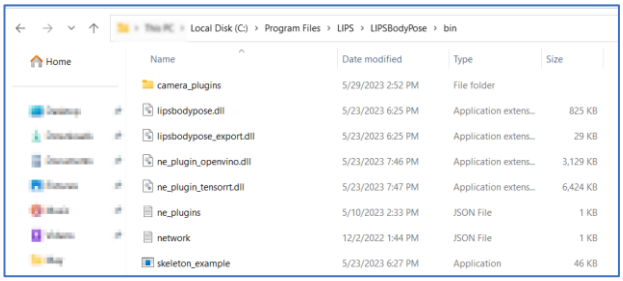
After installing LIPSense™ 3D Body Pose SDK, additional configuration is required to ensure smooth operation of the SDK. For more information, refer to the *Additional Configurations* section.

5. Example Application

LIPS Corp. provides a **Skeleton Example** application to demonstrate the interaction possibilities of LIPSense™ 3D Body Pose SDK. Before launching the example application, make sure that your camera is properly connected and it is recommended to use the following operating system and compiler:

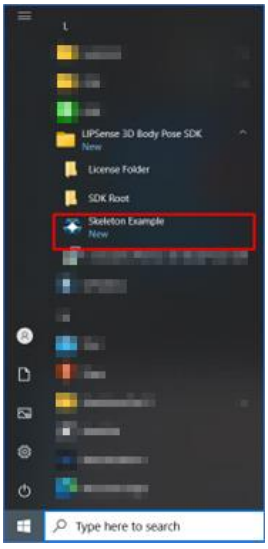
- Windows 10 64-bit system or later
- Microsoft Visual Studio 2017 with Microsoft Visual C++ Redistributable 2017

Note: For **LIPSedge™ AE400 / AE450** users, make sure the properly configured **network.json** (camera network configuration file) is in the same folder as skeleton_example.exe to start the application. The default location of LIPSense™ 3D Body Pose SDK is C:\Program Files\LIPS\LIPSBodyPose\bin.

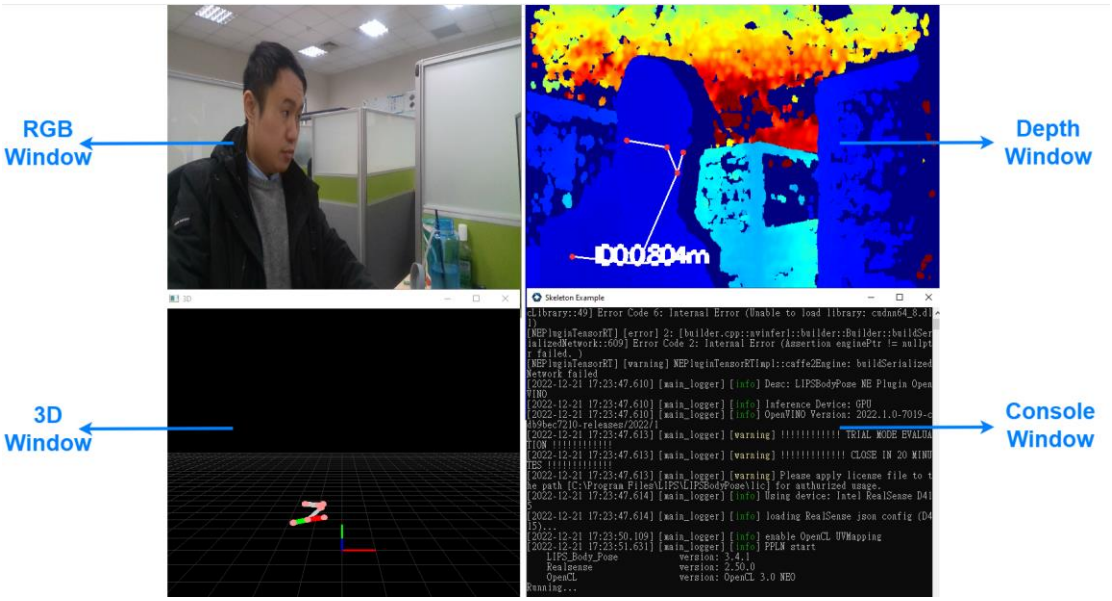


---

1. From Windows Start Menu, find LIPSense™ 3D Body Pose SDK and start **Skeleton Example.**



2. 4 windows pop up:
   - **Console Window**: Displays the camera information, error logs and status.
   - **RGB / Depth / 3D Windows**: Displays the live skeleton detection results in RGB / Depth / 3D images.



Note: Certain poses cause less optimal recognition results. Avoid those poses during the motion tracking process. For details, refer to *2.2 Pose Requirement, LIPSense™ 3D Body Pose SDK User's Guide*.

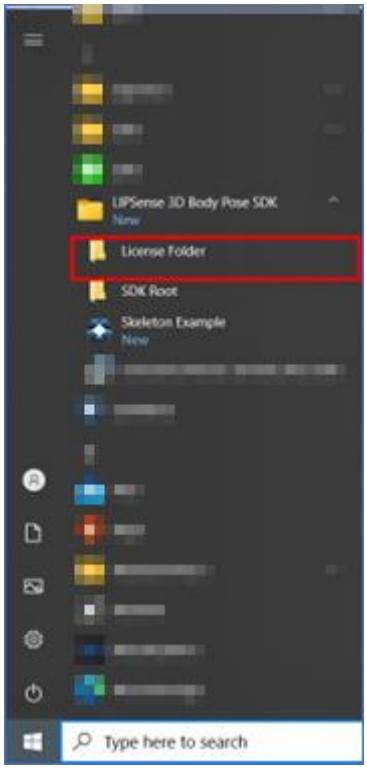
---

6. License File Settings

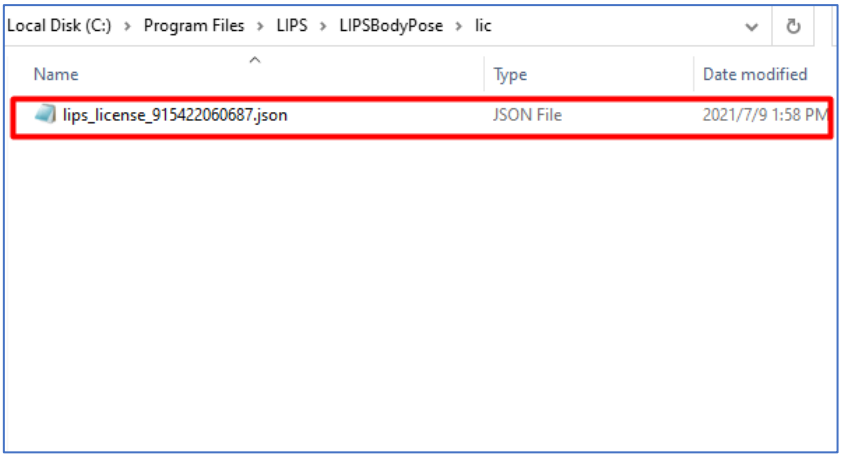
The example application of LIPSense™ 3D Body Pose SDK has a 20-minute time limit in Trial mode. To remove the time limit, contact info@lips-hci.com to obtain a license file. Next, follow the instructions to add the license file's location to your environment.

1. Obtain the license file from LIPS Corp.

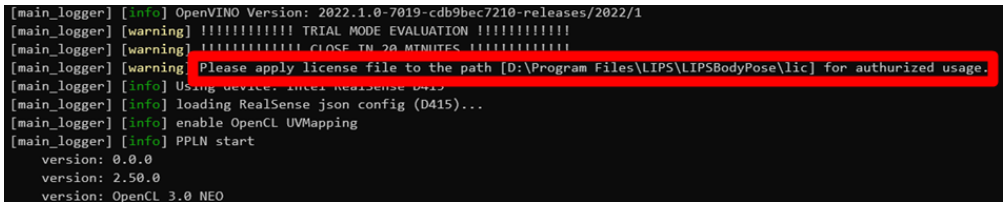
2. From **Windows Start Menu**, select **License Folder**.



3. Move the license file provided by LIPS Corp. to the designated location.
   Installation Directory \LIPS\LIPSBodyPose\lic



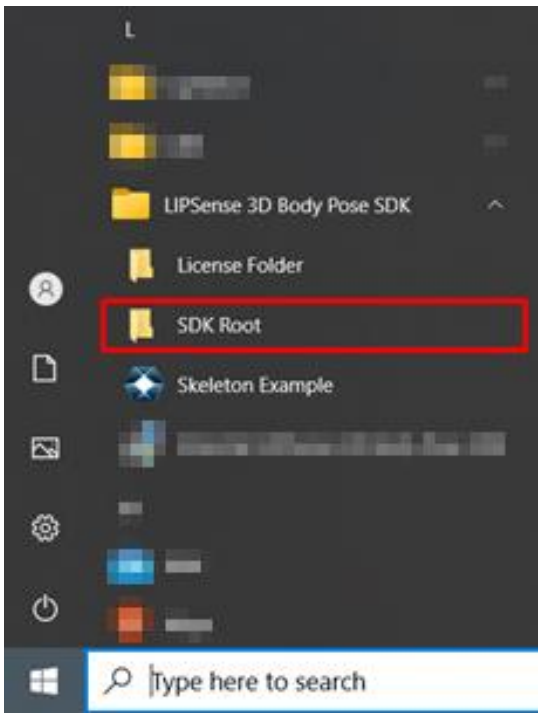
4. If the license file is not found, a warning message will appear in the program's console window indicating the location to store the license file. Obtain the license file and follow steps 1-2 to install it.



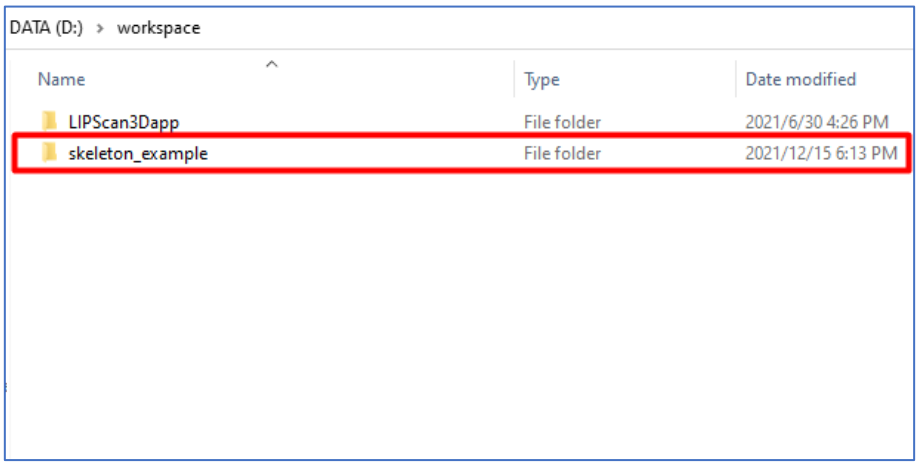
7. Source Code Compilation

LIPS Corp. provides a **source code package** of the example application for developers who wish to customize it. This package allows developers to add, remove, or modify features and compile the source code based on their individual development needs.

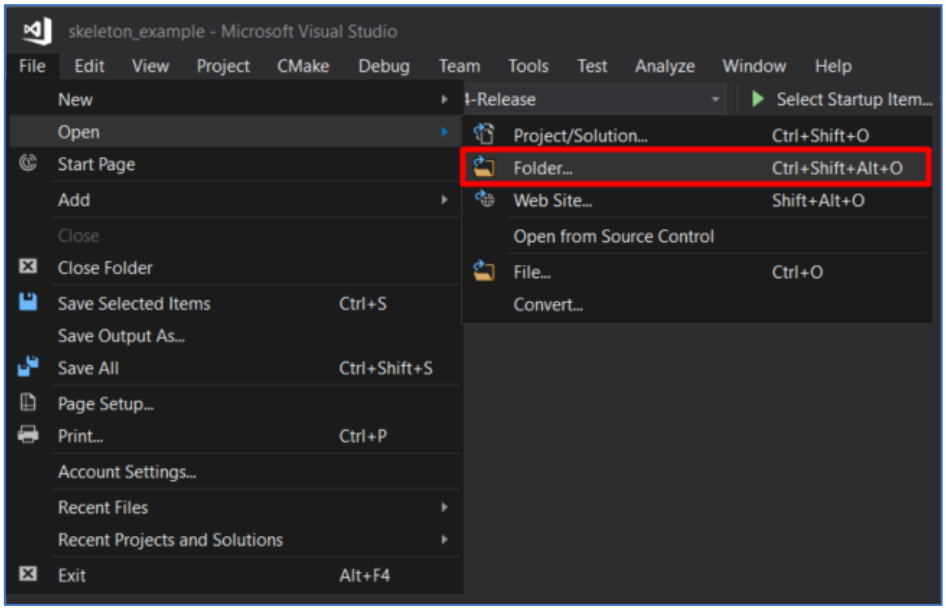
1. From **Windows Start Menu**, navigate to the location of **LIPSense™ 3D Body Pose SDK's location** and select "**SDK Root**".



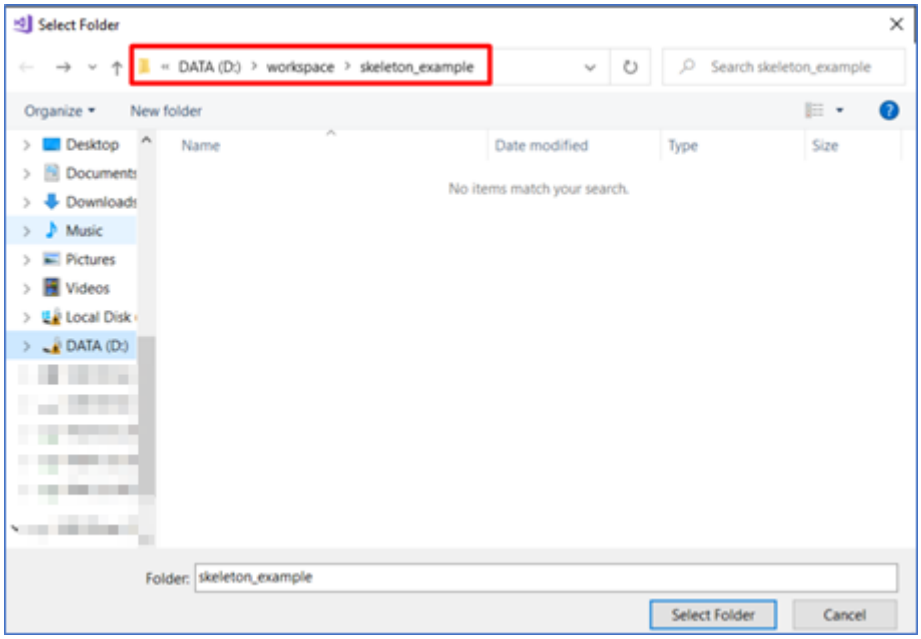
2. From there, copy the "**skeleton_example**" folder to the desired location.



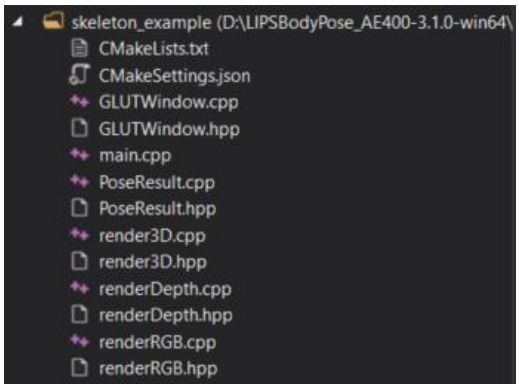
3. Start **Microsoft Visual Studio 2017**. From the top bar, select **File** > **Open** > **Folder**.



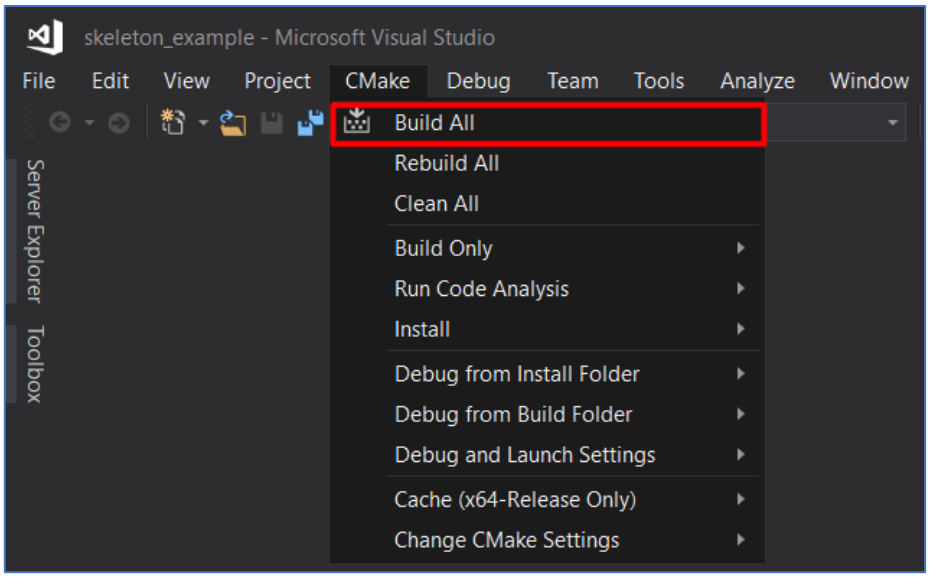
4. Select the "**skeleton_example**" folder.



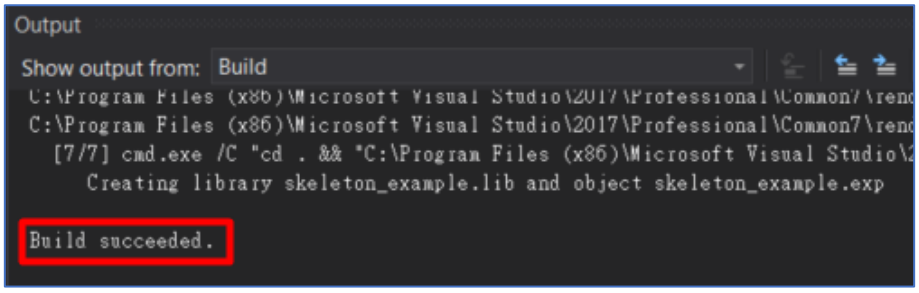
Note: Make sure the **skeleton_example** folder appears in the upper right menu of the Microsoft Visual Studio Community 2017.



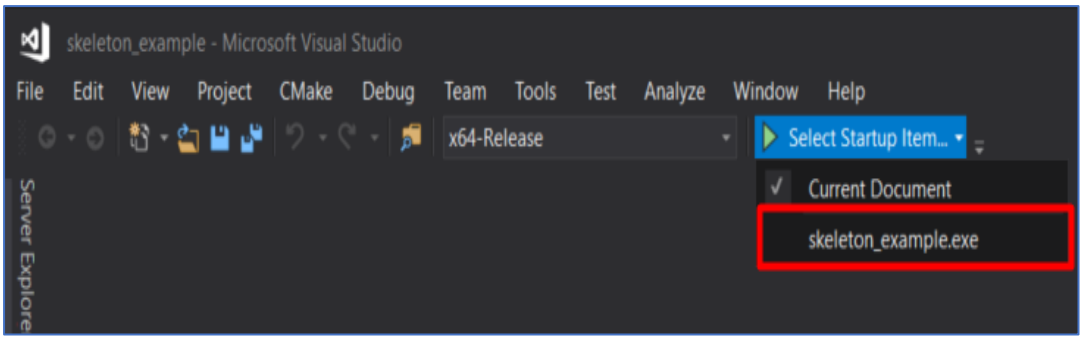
5. From the top bar, select **Cmake > Build All** to compile the **skeleton_example.**



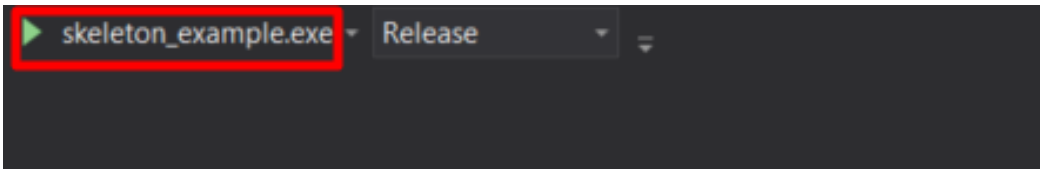
6. Make sure the **Build succeeded** message appears in the Output window to confirm a successful compilation.



7. From the top bar, click the dropdown list for **Select Startup Items** and choose **skeleton_example.exe**.

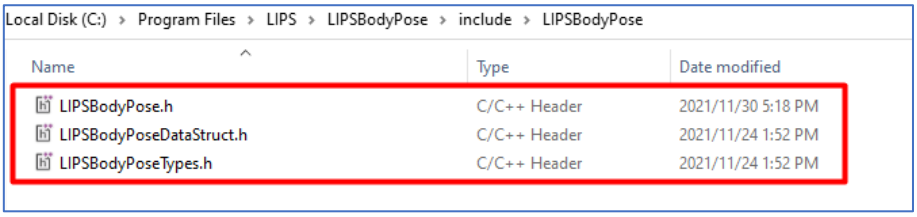


8. Click the **Arrow sign** to execute the built example application.



8. Core Functions

When integrating new functionalities, you can access the core function data in the **SDK headers**. The header files are located atLIPS\LIPSBodyPose\include\ LIPSBodyPose.



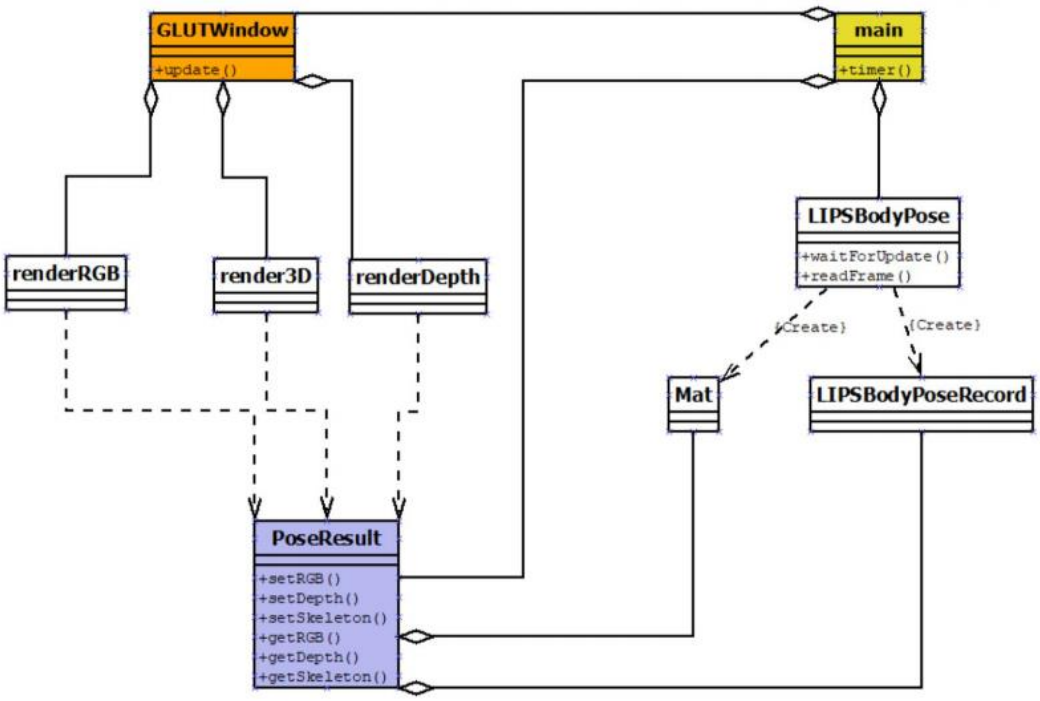
8.1 Function Features

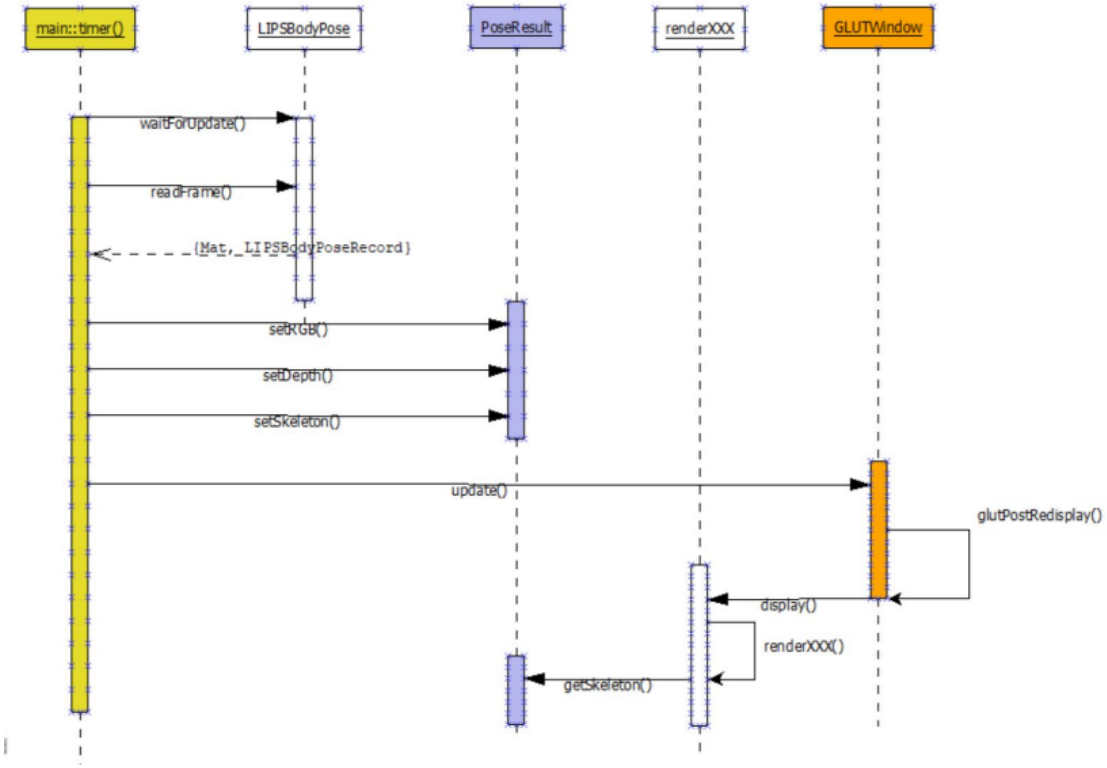
The features of each core functions are listed below:

No.	File Name	Features
1.	**GLUTWindow.cpp**	The interface of GLUT window.
2.	**renderRGB.cpp**	Renders the color image for the GLUT window
3.	**renderDepth.cpp**	Renders the depth image for the GLUT window
4.	**render3D.cpp**	Renders the 3D skeletal image for the GLUT window
5.	**PoseResult.cpp**	Saves detection results to be passed to different methods for generating motion tracking images.
6.	**main.cpp**	Uses a timer function to determine when to renew images.

8.2 Class / Sequence Diagram

The following class / sequence diagrams illustrate the structure of these functions.





8.3 Main Program

The main program can be accessed at LIPSense 3D Body Pose SDK > skeleton_example > main.cpp. The functions defined in the main.cpp file is listed below:

```
lips:LIPSBodyPose::waitForUpdate()
```

This function holds incoming raw motion tracking data until all data are ready for further access. It should be called before invoking **`lips::BodyPose::readFrame()`**, which is responsible for reading the skeletal frame of the target.

```
lips::LIPSBodyPose::readFrame(cv::Mat&,
cv::Mat&, std::shared_ptr<LIPSBodyPoseRecord>&)
```

This function processes raw motion capture data into images. The data returned from **`lips::LIPSBodyPose::waitForUpdate()`** will be captured by **`readFrame()`**, allowing developers to access key information such as RGB and depth images.

```
shared_ptr<LIPSBodyPoseRecord> humans_information;
cv::Mat matRGB;
cv::Mat matDepth;

try
{
    // Wait for pose engine result
    if( pPoseEngine->waitForUpdate() )
    {
        pPoseEngine->readFrame( matRGB, matDepth, humans_information );
```

The class **`LIPSBodyPoseRecord`** records the 2D/3D coordinates of joint positions, along with the number of detected targets and their respective IDs.

The 2D coordinates of joint positions, saved in **`Coord2D`**, consist of the x and y axes, which are normalized to the range [0:1]. To represent the joint position in the pixel coordinate system, multiply the x and y axes by the image resolution, such as [x * 1280, y * 720].

The 3D coordinates are already aligned with the world coordinate system, so there is no need for the normalization process.

```
LIPSBodyPose::LIPSBodyPose(
const bool is_mirror = true,
const LogTo log = LOG_NONE,
const bool enable_crash_handle = false,
const DeviceType device_type = DeviceType::AUTO,
const NeuralEngine neural_engine =
NeuralEngine::AUTO,
const RotationAngle rotation_angle =
RotationAngle::ROTATE_NONE
)
```

Here are the descriptions of the provided parameters:

- **`is_mirror`**: This parameter flips images and skeletal image predictions horizontally if set to **`True`**.
- **`Output`**: It controls the destination of log entries. The default value is **`LOG_NONE`**. Refer to the **`Lips::Logto enum`** section for other possible values.
- **`enable_crash_handle`**: Set this value to **`True`** or **`False`** to enable or disable error log entries upon emergency program termination. Signal handlers include **`SIGFPE, SIGEGV, SIGILL, SIGINT`**, and **`SOGTERM`**. If you want to implement your own signal handler, set the value to **`False`**.
- **`device_type`**: This parameter allows you to select the camera device to be connected. Refer to the **`Lips::DeviceType enum`** section for other possible values.
- **`neural_engine`**: It specifies the device to perform the neural network inference. For example, when **`NeuralEngine::NVIDIA`** is assigned, the NVIDIA GPU will be used if the required software and hardware environments are satisfied. Refer to the **`lips::NeuralEngine enum`** section for other possible values.
- **`rotation_angle`**: This parameter specifies the camera's mechanical position. Refer to the **`lips::RotationAngle enum`** section for other possible values.

lips::LogTo enum

- **LOG_NONE**: Turns error log **OFF**.
- **LOG_FILE**: Prints error log to pose.log.
- **LOG_CONSOLE**: Prints error log to the console (standard output).
- **LOG_BOTH**: Prints error log to both pose.log and the console.

lips::DeviceType enum

- **AUTO**: Automatically detects the available camera.
- **INTEL_REALSENSE**: Selects Intel® RealSense™ cameras.
- **LIPSEDGE_AE**: Selects LIPSedge™ AE series cameras.
- **LIPS_OPENNI**: Selects LIPS OpenNI-compatible cameras, such as LIPSedge™ DL.

lips::NeuralEngine enum

- **AUTO**: Automatically detects the available graphics card.
- **INTEL**: Uses the Intel® OpenVINO™ compatible HD graphics card as the default graphics card for neural network inference.
- **NVIDIA**: Uses the NVIDIA graphics card as the default graphics card for neural network inference.

lips::RotateAngle enum

- **ROTATE_NONE**: The camera remains in a neutral position.
- **ROTATE_90_CLOCKWISE**: The camera rotates 90°clockwise.
- **ROTATE_90_COUNTERCLOCKWISE**: The camera rotates 90°counterclockwise.
- **ROTATE_180**: The camera rotates 180°.

9. Error Message (Windows)

In case you come across an [error] message such as "ocl content creation failure" and "unsupported realsense distortion coeff," please disregard it, as we will modify it to [warning] in the upcoming version.

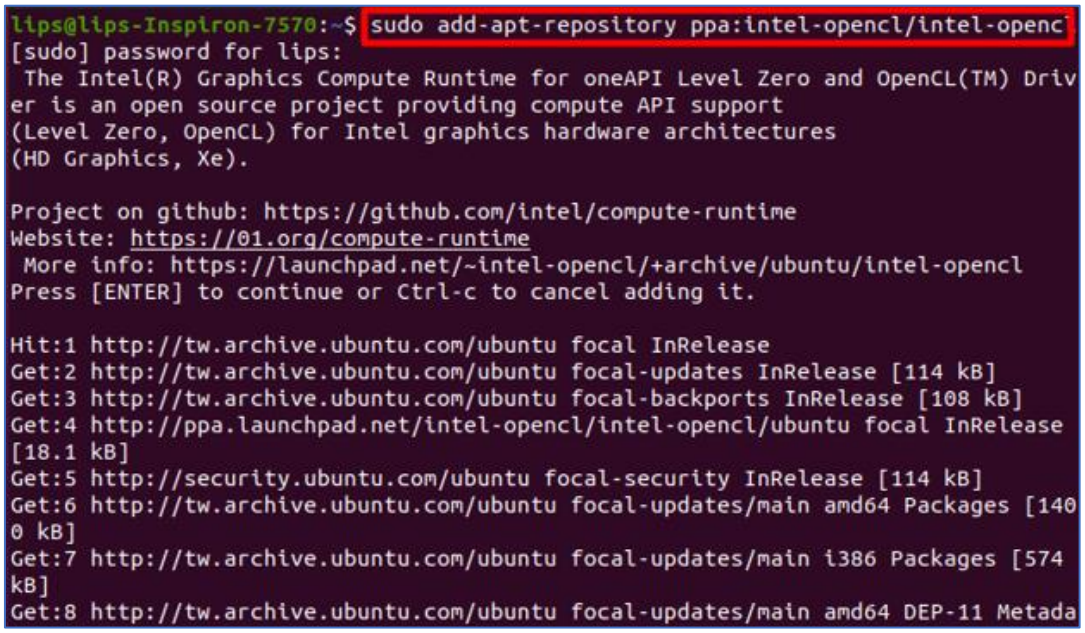
Linux

10. SDK Installation

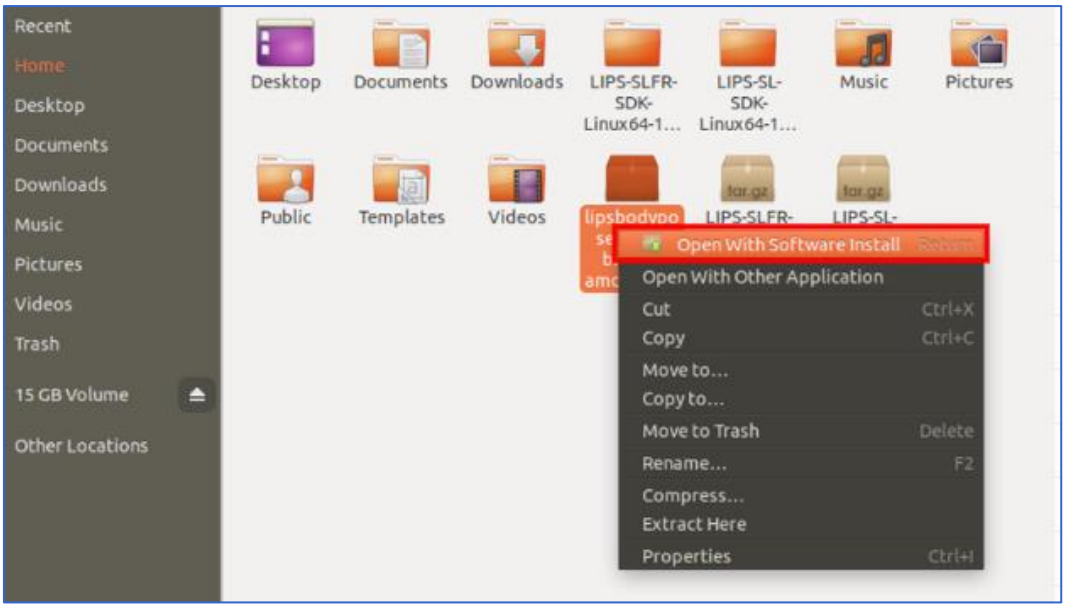
LIPS Corp. provides an installation assistant to install LIPSense™ 3D Body Pose SDK and its dependent components except for OpenCL libraries, which requires a separate installation process. Follow the instructions to download and complete the installation.

1. Connect your camera to your PC / laptop.
2. Go to our website: https://www.lips-hci.com/developer-documentation.
3. On the menu, select the mode LIPSense™ 3D Body Pose Middleware.
4. Select and download the installation file according to your operating system.
5. Open the **Terminal** and install the **OpenCL library** by entering the following command:

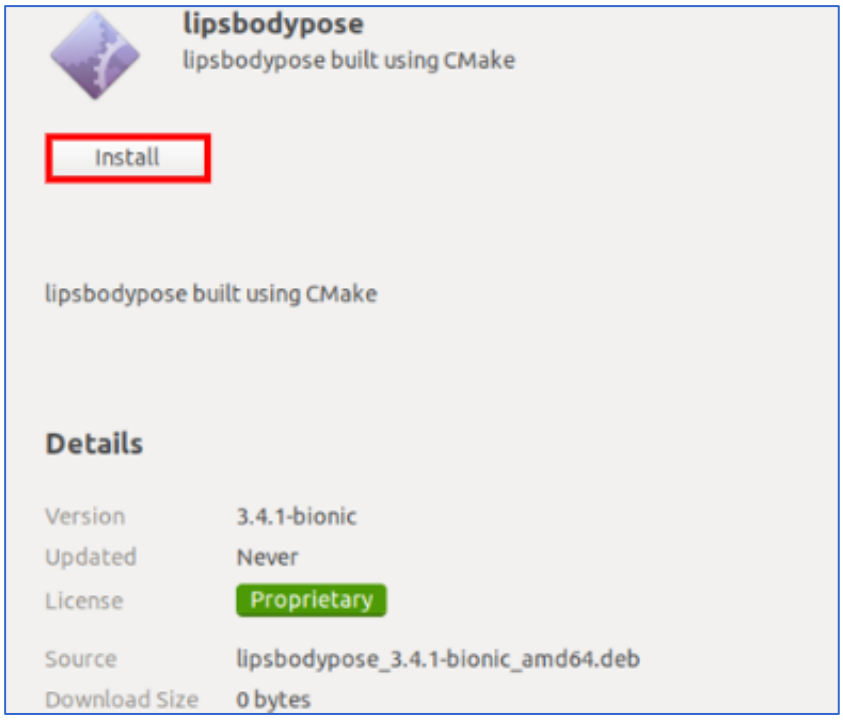
```
sudo add-apt-repository ppa:intel-opencl/intel-opencl
```



6. Return to the system Window.
7. Right-click the downloaded file and select **Open with Software Install**.



8. Click **Install**.



9. Restart your system to apply the environment variables.

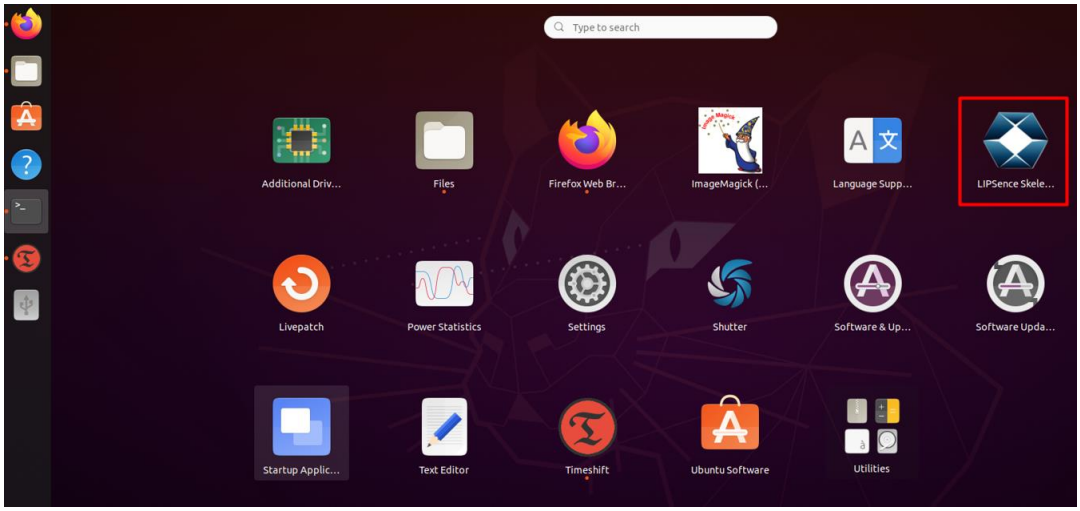
After you have installed LIPSense™ 3D Body Pose SDK, additional configuration is required for smooth operation of LIPSense™ 3D Body Pose SDK. For details, refer to *Additional Configurations*.

11. Example Application

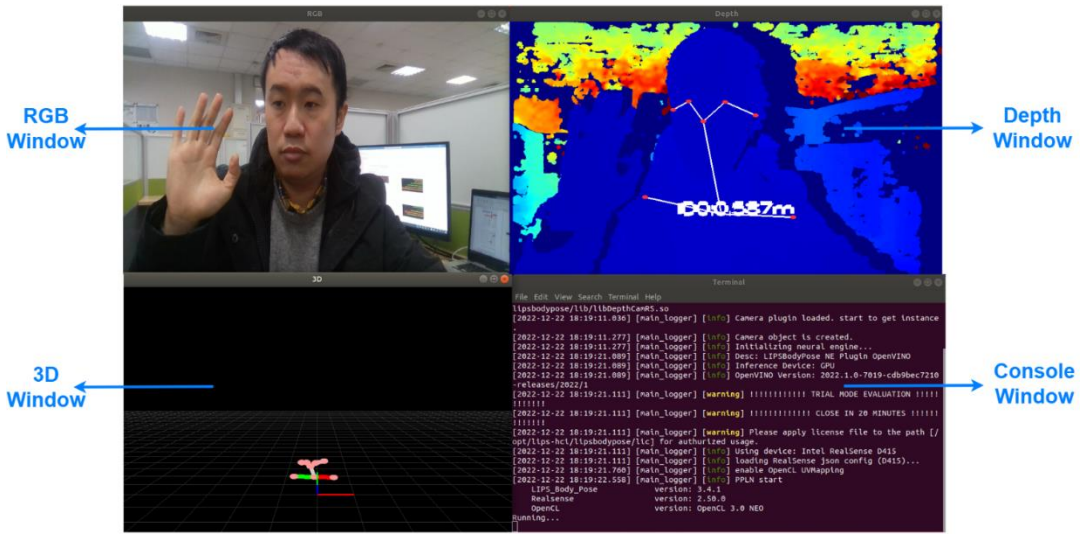
Once installed, **LIPSense™ 3D Body Pose SDK** and its configuration files will be available in your system at **/opt/lips-hci/lipsbodypose**.

LIPS Corp. provides a **Skeleton Example** application that demonstrates the interaction possibilities of LIPSense™ 3D Body Pose SDK. Before launching the example application, ensure that your camera is properly connected. You can find the **LIPSense Skeleton Example** in the **Applications** section located at the bottom left corner.

1. Click the **LIPSense Skeleton Example** app to start the streaming.



2. 4 windows pop up: Four windows will pop up:
   - **Console Window**: Displays camera information, error logs, and status.
   - **RGB Window**: Displays live skeleton detection results in RGB images.
   - **Depth Window**: Displays live skeleton detection results in depth images.
   - **3D Window**: Displays live skeleton detection results in 3D images.



Note:

1. To achieve optimal recognition results during the motion tracking process, it is advisable to avoid certain poses. For more information on pose requirements, refer to *Section 2.2 Pose Requirements, LIPSense™ 3D Body Pose SDK User's Guide*.
2. For detailed information about the core functions of the program, refer to *8. Core Functions, Windows, LIPSense™ 3D Body Pose SDK User's Guide*.

---

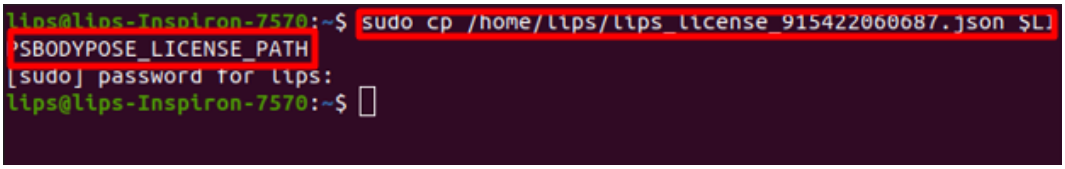
12. License File Settings

LIPSense™ 3D Body Pose SDK's example application comes with a 20-minute time limit in the Trial mode. To remove the time limit, contact info@lips-hci.com to obtain a license file. Next, follow the instructions to add the license file's location to your environment.

1. Obtain the license file from LIPS Corp.

2. Move the license file provided by LIPS Corp. to the designated location.
   Location: /opt/lips-hci/lipsbodypose/lic

```
sudo cp [Current ocation of the license file]
$LIPSBODYPOSE_LICENSE_PATH
```

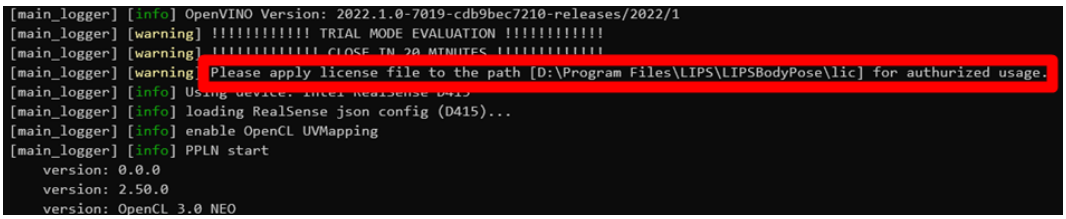


Tip: When specifying the location of the license file, you have the option to use the abbreviated form $LIPSBODYPOSE_LICENSE_PATH.

---

3. Verify that the license file is present in the specified location.

4. If the license file is absent, a warning message will appear in the program's console window indicating the location to store the license file. Obtain the license file and follow steps 1 and 2 to install the license file.

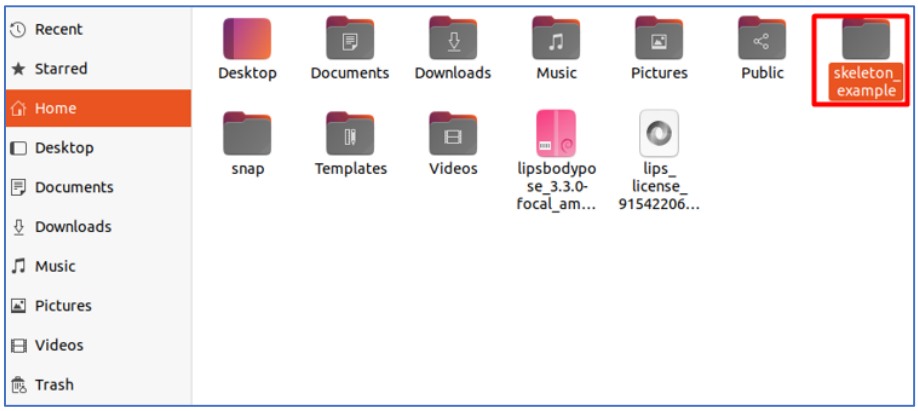


13. Source Code Compilation

For developers who would like to customize the application, LIPS Corp. provides the **source code package** of the example application. This package enables developers to add, remove, or modify features and compile the source code according to their individual development needs.

Note that Cmake tools are required for this process. Make sure **Cmake** is installed on your PC/laptop.

1. Copy the **skeleton_example** folder from the root location of LIPSense™ 3D Body Pose SDK to a location of your choice.



Tip: For Terminal users, when specifying the root location of LIPSense™ 3D Body Pose SDK, you can use the abbreviated $LIPSBODYPOSE_SDK_ROOT form.

---

2. Install the **Cmake** tools.

```
sudo apt-get install -y cmake build-essential
```

```
lips@lips-Inspiron-7570:~$ sudo apt-get install -y cmake build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cmake-data dpkg-dev
  fakeroot g++ g++-9 gcc gcc-9 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1
  libbinutils libc-dev-bin libc6-dev libcrypt-dev libctf-nobfd0 libctf0
  libcurl4 libfakeroot libgcc-9-dev libitm1 libjsoncpp1 liblsan0 libquadmath0
  librhash0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make
  manpages-dev
Suggested packages:
  binutils-doc cmake-doc ninja-build debian-keyring g++-multilib
  g++-9-multilib gcc-9-doc gcc-multilib autoconf automake libtool flex bison
  gcc-doc gcc-9-multilib gcc-9-locales glibc-doc libstdc++-9-doc make-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cmake
  cmake-data dpkg-dev fakeroot g++ g++-9 gcc gcc-9 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1
  libbinutils libc-dev-bin libc6-dev libcrypt-dev libctf-nobfd0 libctf0
  libcurl4 libfakeroot libgcc-9-dev libitm1 libjsoncpp1 liblsan0 libquadmath0
  librhash0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make
```

3. In a folder of your choice, **create a folder** for the compiled application and go to the folder.

```
cd [Desired folder location]/

mkdir [Compiled application location] && cd [Compiled application location]
```

```
lips@lips-Inspiron-7570:~$ cd skeleton_example/
lips@lips-Inspiron-7570:~/skeleton_example$ mkdir build && cd build
lips@lips-Inspiron-7570:~/skeleton_example/build$
```

4. Run **Cmake**

```
cmake ..
```

```
lips@lips-Inspiron-7570:~/skeleton_example/build$ cmake ..
-- The CXX compiler identification is GNU 9.3.0
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenCV: /opt/lips-hci/lipsbodypose/3rdparty/opencv (found version "4.5.
3")
CMake Warning (dev) at /usr/share/cmake-3.16/Modules/FindOpenGL.cmake:275 (messa
ge):
  Policy CMP0072 is not set: FindOpenGL prefers GLVND by default when
  available.  Run "cmake --help-policy CMP0072" for policy details.  Use the
  cmake_policy command to set the policy and suppress this warning.

  FindOpenGL found both a legacy GL library:

    OPENGL_gl_LIBRARY: /usr/lib/x86_64-linux-gnu/libGL.so

  and GLVND libraries for OpenGL and GLX:

    OPENGL_opengl_LIBRARY: /usr/lib/x86_64-linux-gnu/libOpenGL.so
    OPENGL_glx_LIBRARY: /usr/lib/x86_64-linux-gnu/libGLX.so
```

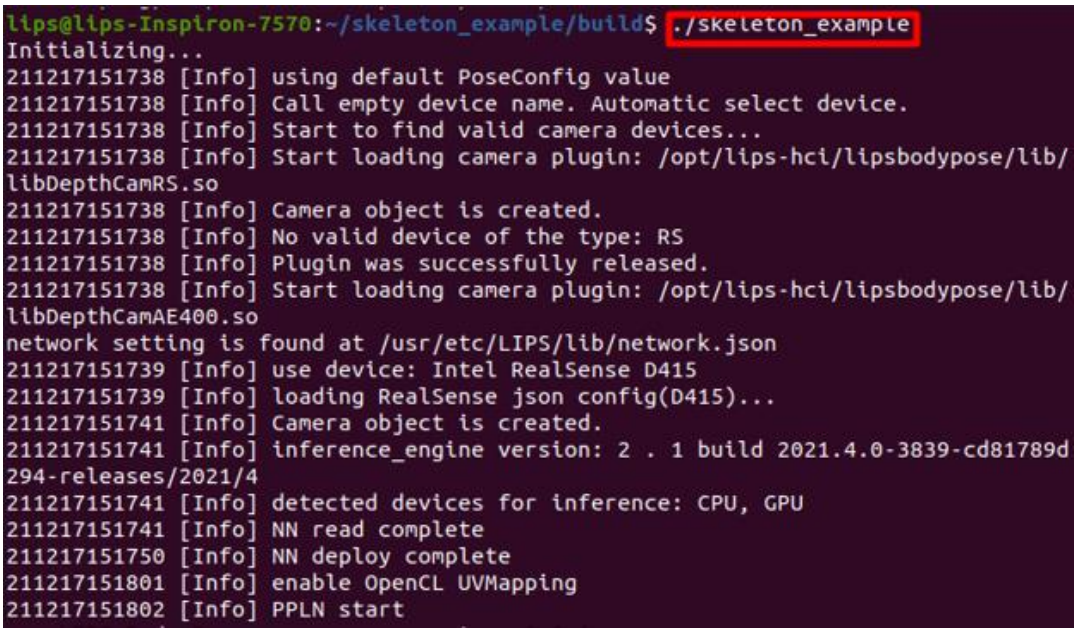
5. Compile the example application.

```
make
```

```
lips@lips-Inspiron-7570:~/skeleton_example/build$ make
Scanning dependencies of target skeleton_example
[ 14%] Building CXX object CMakeFiles/skeleton_example.dir/GLUTWindow.cpp.o
[ 28%] Building CXX object CMakeFiles/skeleton_example.dir/PoseResult.cpp.o
[ 42%] Building CXX object CMakeFiles/skeleton_example.dir/main.cpp.o
[ 57%] Building CXX object CMakeFiles/skeleton_example.dir/render3D.cpp.o
[ 71%] Building CXX object CMakeFiles/skeleton_example.dir/renderDepth.cpp.o
[ 85%] Building CXX object CMakeFiles/skeleton_example.dir/renderRGB.cpp.o
[100%] Linking CXX executable skeleton_example
[100%] Built target skeleton_example
lips@lips-Inspiron-7570:~/skeleton_example/build$
```

6. You can access the example application compiled by using the following command.

```
./skeleton_example
```



To access the core functions for application development, you can refer to the following files:

- For details of the core functions and class diagrams, consult the similar content in *8. Core Functions, Windows, LIPSense™ 3D Body Pose SDK User's Guide*

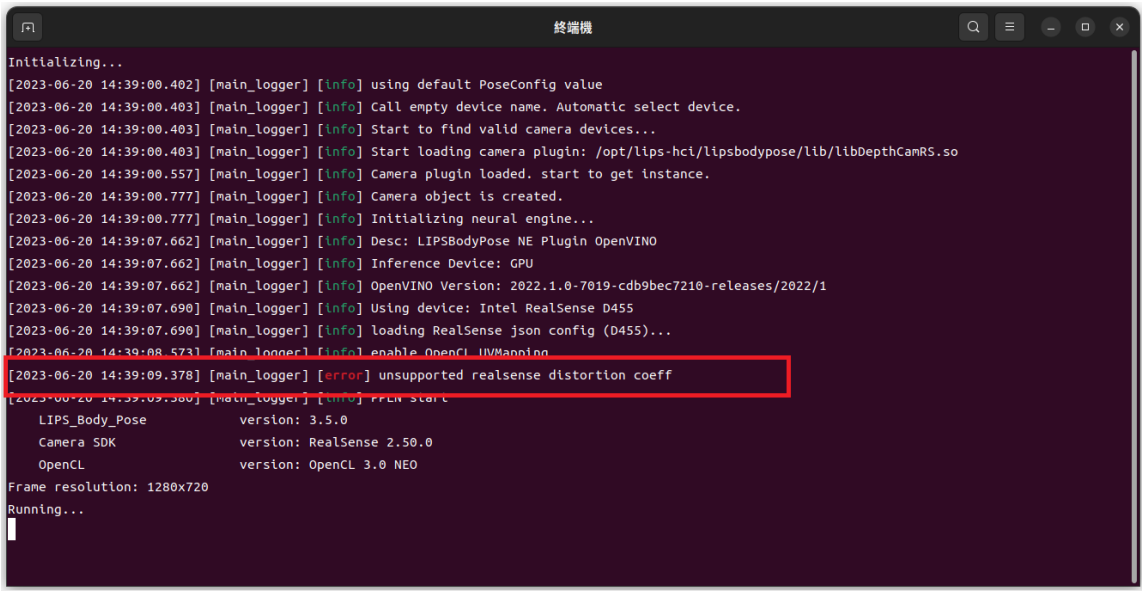
To access the source code of LIPSense™ 3D Body Pose SDK, navigate to the following directory:

- /opt/lips-hci/lipsbodypose/skeleton_example

To access the SDK header files, navigate to the following directory:

- /opt/lips-hci/lipsbodypose/include/LIPSBodyPose

14. Error Message (Linux)



In case you come across an [error] message such as "unsupported realsense distortion coeff," please disregard it, as we will modify it to [warning] in the upcoming version.

Additional Configurations

15. Additional Hardware Packages

Since LIPSense™ 3D Body Pose SDK interacts with various hardware components, including the graphics card, additional packages may be necessary for your device to function properly with the SDK. Install these packages after completing the installation of LIPSense™ 3D Body Pose SDK

15.1 NVIDIA GPUs

For NVIDIA GPU users, install the packages based on your OS by following the official instructions.

Library	Installation Guide
CUDA Toolkit 11.x	https://developer.nvidia.com/cuda-11.0-download-archive
cuDNN 8	https://docs.nvidia.com/deeplearning/cudnn/install-guide/index.html

Supported API Wrappers

16. Python API

The Python support for LIPSense™ 3D Body Pose SDK may vary depending on your operating system (OS). For detailed information, refer to *Notes for Programmers V, Python Support, LIPSense™ 3D Body Pose SDK User's Guide*, which provides instructions for downloading the corresponding Python packages.

The functions for Python API are defined as below:

```
lipsbodypose.lipsbodypose(
is_mirror = true,
device_type = lipsbodypose.AUTO,
neural_engine = lipsbodypose.NeuralEngine_AUTO,
rotation_angle = lipsbodypose.ROTATE_NONE
)
```

The function returns instances of the active LIPSense™ 3D Body Pose SDK pipeline. For more information about the arguments, refer to *8. Core Functions, Windows, LIPSense™ 3D Body Pose SDK User's Guide.*

```
lipsbodypose.lipsbodypose.readFrame()
```

The query function returns the updated predictions of **`RGB`**, **`depth`**, **`skeleton2d`**, and **`skeleton3d`**.

- **`RGB`**: Stores live RGB image data as a numpy array of HD resolution.
- **`Depth`**: Stores live 3D image data as a numpy array of HD resolution.
- **`Skeleton2d`**: Stores the skeletal image data as 2D images.
- `Skeleton3d`: Stores the skeletal image data as 3D images.

When calling the function at maximum speed, you can achieve a maximum throughput of up to 30 frames per second (fps).

The **skeleton2d** and skeleton3d structures are nested with a shape of [P, K, C], where:

- P represents the number of detected people.
- K represents the variable for skeletal joints, which is equal to 18.
- C represents the variable for the dimension of the skeletons, where 2 represents 2D and 3 represents 3D.

16.1 Python Examples

```
import lipsbodypose

is_mirror = True
device_type = lipsbodypose.DeviceType_AUTO
neural_engine = lipsbodypose.NeuralEngine_AUTO
rotation_angle = lipsbodypose.RotationAngle_ROTATE_NONE

engine = lipsbodypose.lipsbodypose( is_mirror, device_type, neural_engine, rotation_angle )

while True:
rgb, depth, body2D, body3D = engine.readFrame()
# display …
```

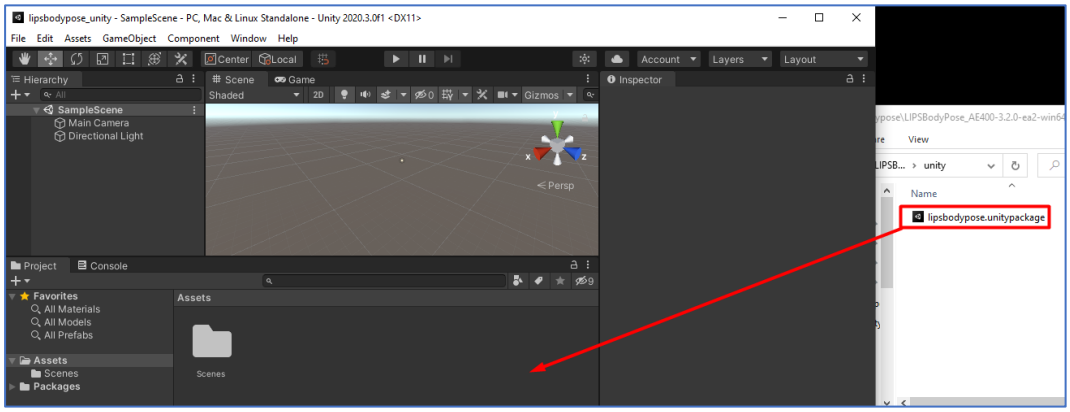
17. Unity API (C#)

LIPSense™ 3D Body Pose SDK supports the Unity C# API. For more details, refer to the Notes for Programmers VI, Unity Support, LIPSense™ 3D Body Pose SDK User's Guide.

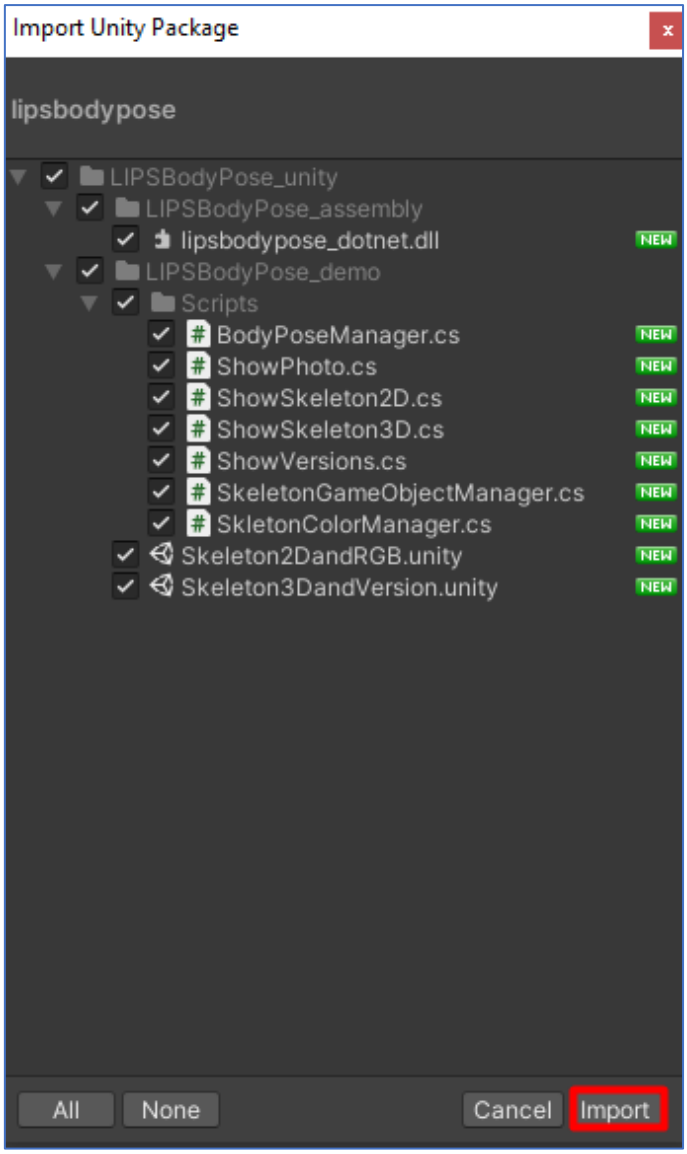
17.1 SDK Import

The LIPSense™ 3D Body Pose SDK includes a Unity package located at **LIPSense™ 3D Body Pose SDK > Unity**. Follow the instructions below to import the LIPSense™ 3D Body Pose SDK into Unity for software development.

1. Start **Unity** and drag the **lipsbodypose.unitypackage** to your unity project, an **Import Unity Package** window pops-up.



2. Click **Import**.



17.2 Unity Examples

The Unity C# API is similar to the C++ API as the definition below shows:

```
lips_dotnet.LIPSBodyPose(
[bool isMirror = true],
[bool enableCrashHandle = false],
[LIPS.DeviceType deviceType = LIPS.DeviceType.AUTO],
[LIPS.NeuralEngine neuralEngine = LIPS.NeuralEngine.AUTO],
[LIPS.RotationAngle rotationAngle = LIPS.RotationAngle.ROTATE_NONE]
)
```

The function returns instances of active LIPSense™ 3D Body Pose SDK pipeline.

```
bool lips_dotnet.LIPSBodyPose:ReadFrame()
```

The function queries a new frame with skeleton data from the pipeline and stores it in an internal buffer. When calling the function at maximum speed, you can achieve a maximum throughput of up to 30 frames per second (fps).

After calling `readFrame()`, you can use the functions below to acquire the data you need.

```
void lips_dotnet.LIPSBodyPose.GetImageSize(ref int width, ref int height)
byte[] lips_dotnet.LIPSBodyPose.GetColorImage()
ushort[] lips_dotnet.LIPSBodyPose.GetDepthImage()
lips_dotnet.LIPSBodyPoseRecord
lips_dotnet.LIPSBodyPose.GetBodyPoseRecord()
```

A series of getter functions are provided to fetch the skeletal data.

```
int lips_dotnet.LIPSBodyPoseRecord.GetId(int who)

int lips_dotnet.LIPSBodyPoseRecord.GetNumberPeople()

lips_dotnet.Coord2D    lips_dotnet.LIPSBodyPoseRecord.Keypoint2d(int who, int part)

lips_dotnet.Coord3D    lips_dotnet.LIPSBodyPoseRecord.Keypoint3d(int who, int part)
```

LIPS CORPORATION

2F, No. 100, Ruiguag Road, Neihu District,
Taipei City 114, Taiwan
Tel.: + 886-2-8791-6998
Fax: +886-2-8791-8996
Official Website: https://www.lips-hci.com/
E-Mail: info@lips-hci.com