LIPScan 3D Scan SDK V2.0.4

User's Manual

July 2026

Version 1.1

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 arises from the use of the information. All contents are subject to constant revision to improve its reliability and may be changed without prior notice.

July 2026

LIPS
Sense The World

Contents

Revision History

Revision	Description	Date
V1.1	Content refactoring	July 2026
V1.0	New release	July 2024

1. System Requirements

Recommended System

The LIPScan™ 3D Scan SDK libraries and sample programs are designed and developed to be compatible with the Windows 10 operating system. Refer to the recommended hardware table below for specific hardware specifications necessary to ensure optimal performance with the LIPScan™ 3D Scan SDK.

Recommended environment

- Windows 10 64-bit system
- Microsoft Visual Studio 2019

Recommended Hardware

Processor	Intel® Core™ i5 processor or above	
RAM	4 GB or above	
GPU Memory	1 GB or above	
System	64-bit system (x64)	
Graphic Cards	Internal	Intel® HD Graphics 630
	Dedicated	NVIDIA GeForce GTX 1050Ti

Hardware compatibility

The LIPScan™ 3D Scan SDK is compatible with the following 3D cameras:

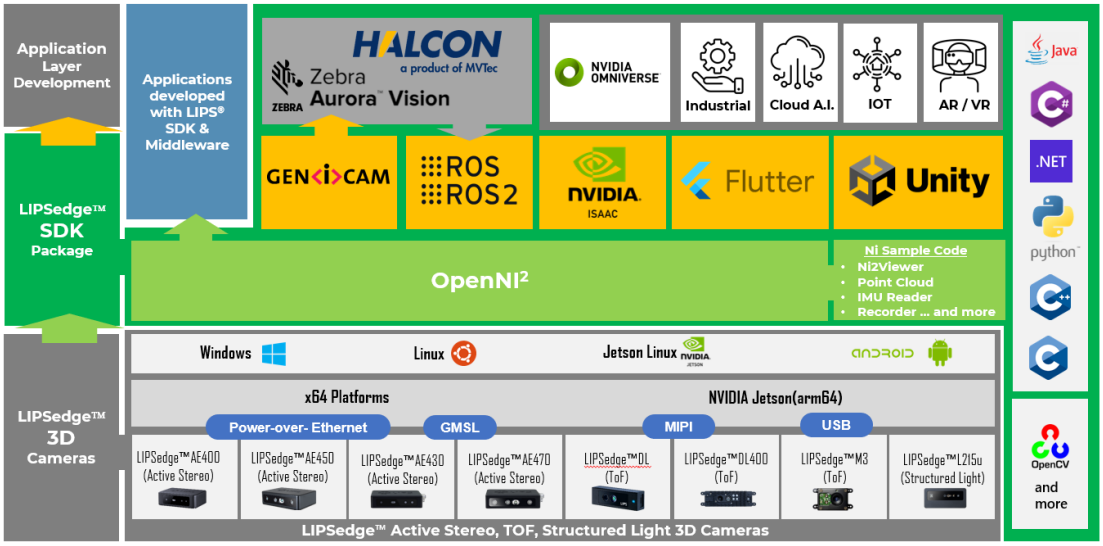
Series	Models
LIPSedge™	AE400
	AE430
	AE450
	AE470
	L210u
	L215u
Intel® RealSense™	Depth Camera D415
	Depth Camera D435
	Depth Camera D435i
	Depth Camera D455

2. SDK Structure

LIPS 3D camera / SDK offers a system for developing depth-sensing applications. As the LIPS system architecture illustrates, the system is comprised of the hardware layer and the software layer.

The hardware layer oversees data capture, transfer, and processes.

In the software layer, the captured data is fetched by the LIPS SDK (Software Development Kit) on the OS environment. Depending on the project complexity, wrappers and third-party utilities may be engaged before the data is eventually presented in the application layer for business applications.



The core of the system, the LIPS SDK, is comparable to a toolbox full of software modules comprised of middleware, libraries, wrappers and API, and miscellaneous programming languages / platforms for application development. With extensive wrapper support, LIPS SDK enables developers to access bottom layer data with APIs, thus eliminating the hassle of changing third-party functions. The result is a highly effective project scoping, monitoring, and execution workflow compatible with the fast-pacing AIoT market and machine vision demands.

3. Camera Connections

The LIPScan™ 3D Scan SDK supports Intel® RealSense™, LIPSedge™ AE series, and LIPSedge™ L210 / L215 series 3D cameras and SDK. Before using the LIPScan™ 3D Scan SDK, follow the instructions below to install the hardware and SDK for each camera and ensure that the 3D images can be viewed properly.

For details on Intel® RealSense™ series camera / SDK, refer to Intel's official website.

For details on LIPSedge™ AE series camera / SDK, refer to LIPSedge™ AE series User's Manual.

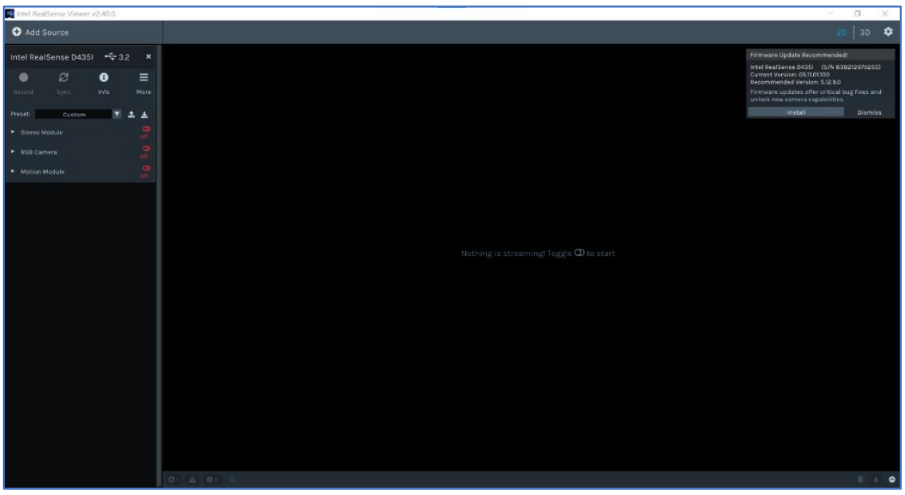
For details on LIPSedge™ L210u / L215u series camera / SDK, refer to LIPSedge™ L210u / L215u User's Manual.

A. Intel® RealSense™ D400 Series

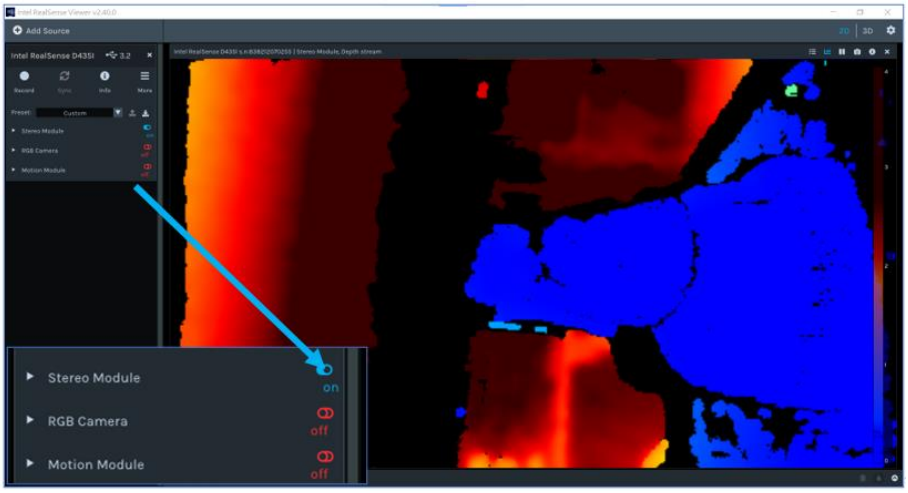
After installing Intel® RealSense™ SDK 2.0, the 3D depth image preview application is accessible at the following location:

Name	realsense-viewer.exe
Location	C:\Program Files (x86)\Intel RealSense SDK 2.0\tools

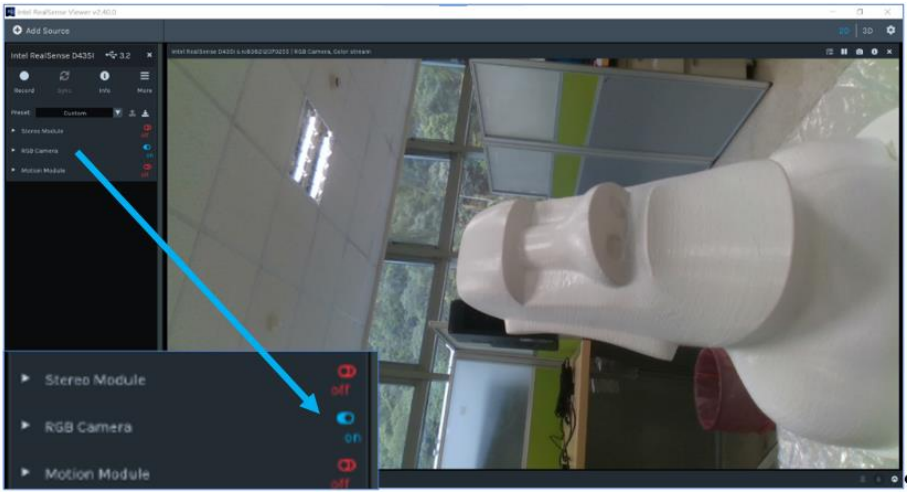
The interface of the application is as follows:



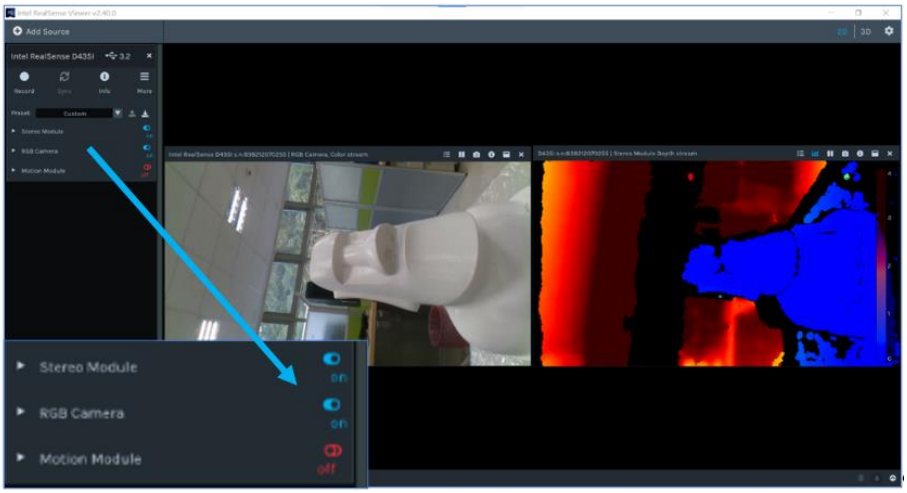
1. Toggle the ON / OFF switch of **Stereo Module** to see the live depth image.



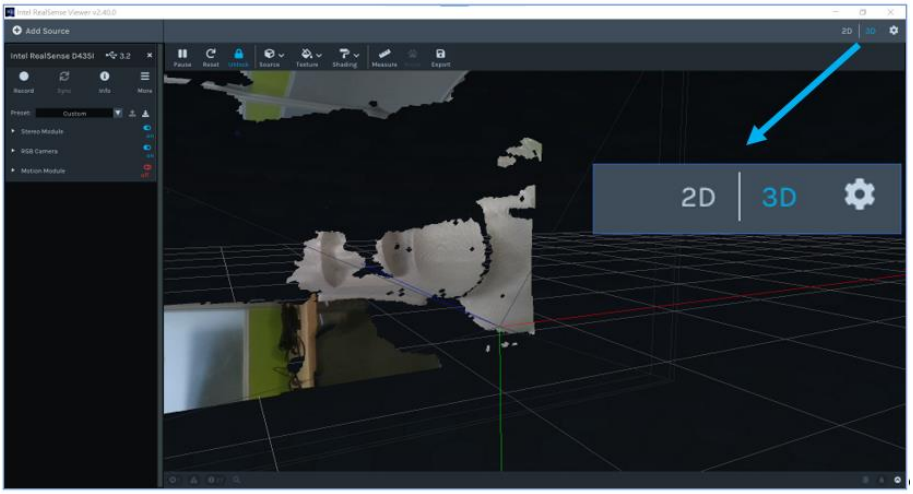
2. Toggle the ON / OFF switch of **RGB Camera** to see the live RGB image.



3. Besides individual streaming, the Stereo Module / RGB Camera also allows developers to view both image types simultaneously.



4. Click the 2D | 3D option at the upper right corner to switch between live depth and point cloud image.



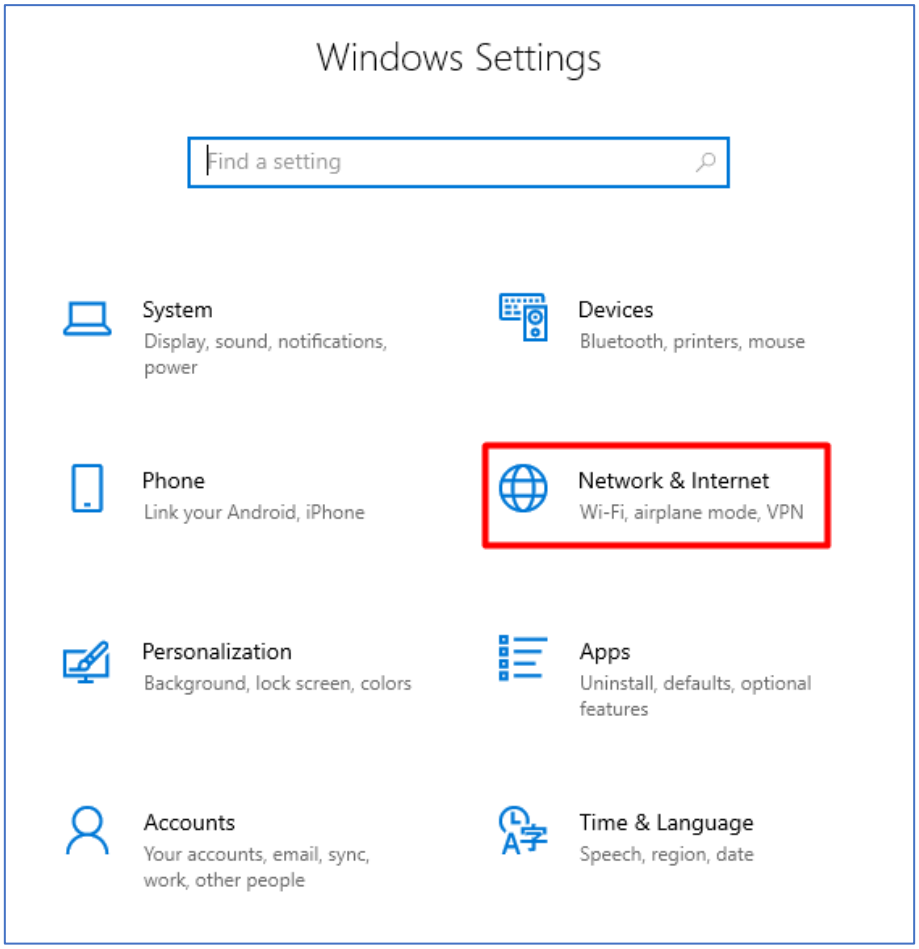
B. LIPSedge™ AE Series

The LIPSedge™ AE series cameras are powered and connected to the computer via Ethernet/PoE (Power over Ethernet). Before use, the computer needs to have Ethernet enabled and the relevant network settings properly configured in order to establish a connection with the camera and enable its functionality.

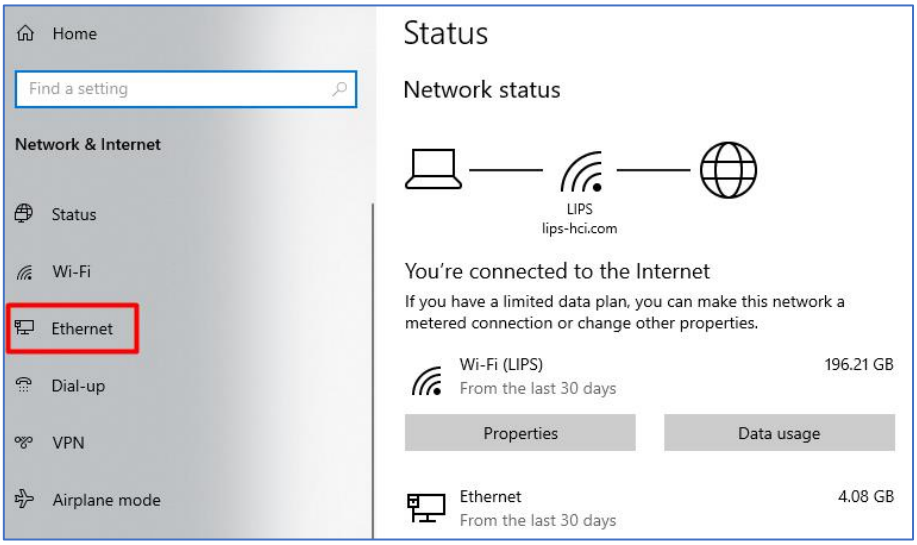
a. Network Configuration

Follow the steps below to complete the connection setup for the LIPSedge™ AE series on the host PC / laptop. Ensure that the host PC / laptop and the LIPSedge™ AE series camera is on the same subnet when configuring the network settings.

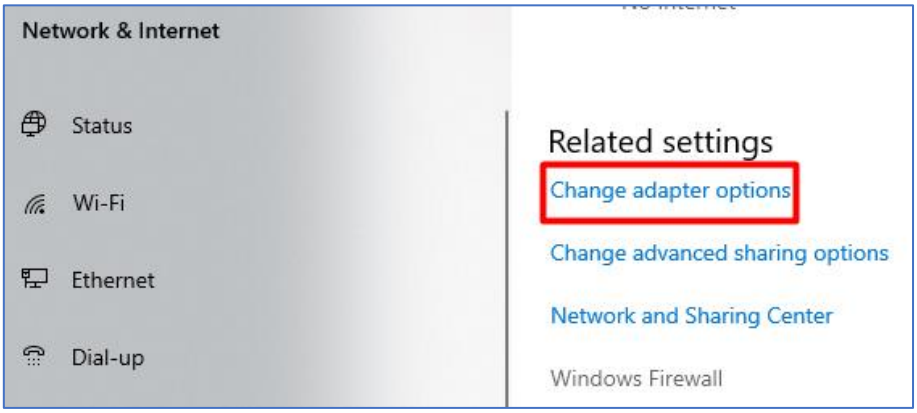
1. Open **Windows Settings** and click **Network & Internet**.



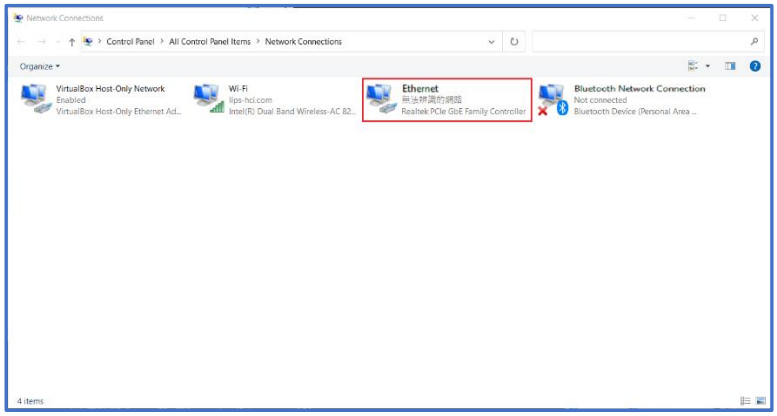
2. On the left menu, click **Ethernet**.



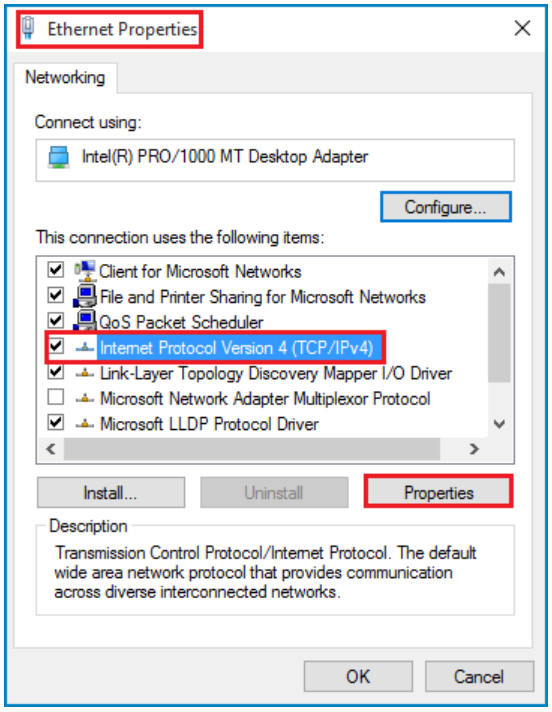
3. Click **Change adapter options**.



4. Right-click **Ethernet** and select **Properties**.



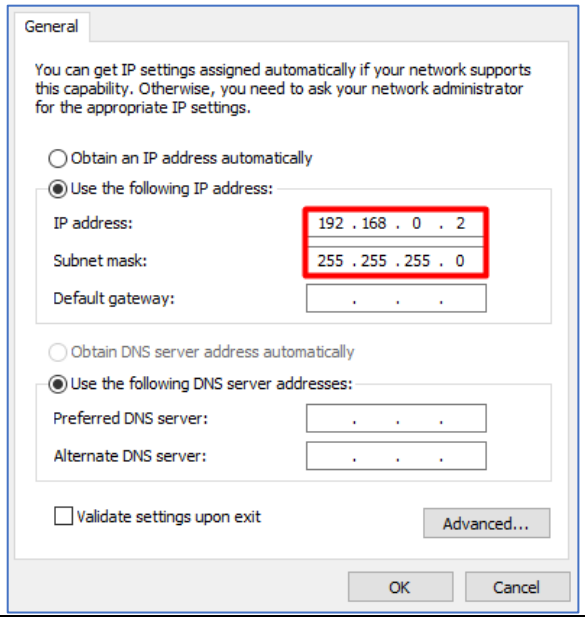
5. Select **Internet Protocol Version 4 (TCP / IPv4)** and click **Properties**.



6. Type the following network settings and click **OK**.

   IP Address：192.168.0.2

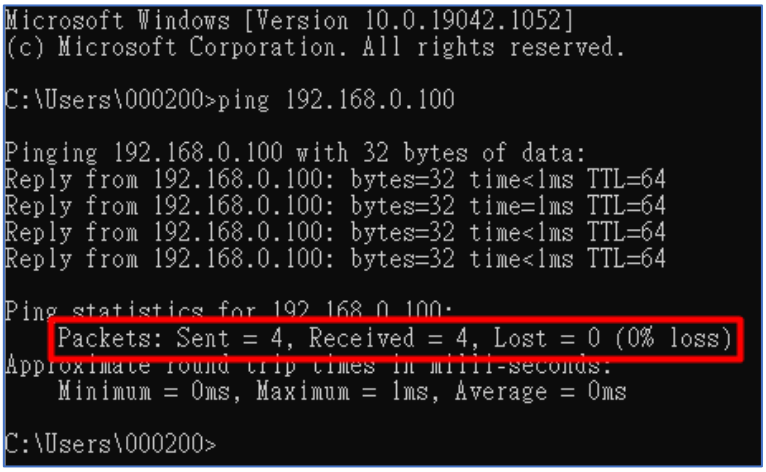
   Subnet mask：255.255.255.0



Note: The local device must fall under the same subnet as the LIPSedge™ AE series camera for communication. In this example, since the default IP address of a LIPSedge™ AE series camera is 192.168.0.100, it is recommended to set up a network ID of 192.168.0 and a host ID of 2.

---

7. In the **Command Prompt**, type the following command: **ping 192.168.0.100** and press **Enter**. When the ping packets sent equals to the ping packets received, the camera connection is properly set up.



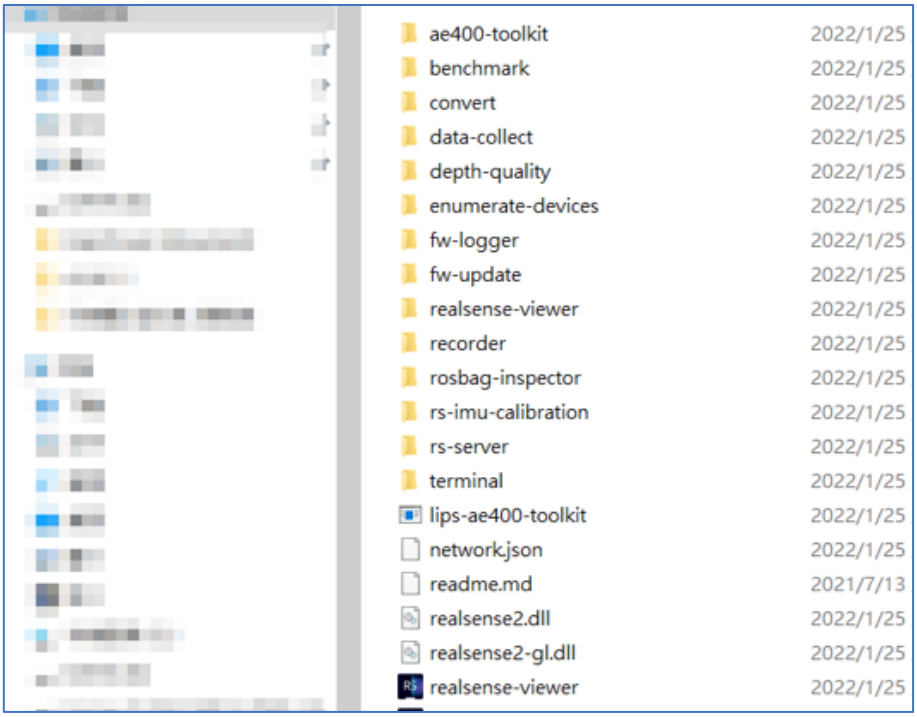
b. Camera Streaming

To preview 3D images using the LIPSedge™ AE series camera, install the LIPSedge™ AE series SDK. For detailed instructions, please refer to the LIPSedge™ AE series user manual.

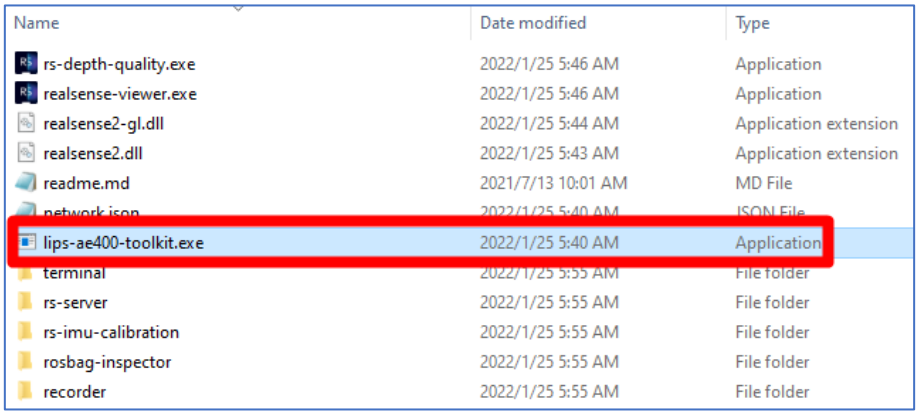
After installing the SDK, a **network.json** file is available in the following directory. The network.json file contains the connection settings for the LIPSedge™ AE series camera. Before establishing a connection, ensure that the camera's connection settings are correctly configured. To achieve this, use the **LIPSedge™ AE series Toolkit** located at the following directory to automatically scan for available connections with the LIPSedge™ AE series cameras.

File name	Network.json
Name	LIPS_AE400_Toolkit.exe
Location	[Download Directory]\tools\

The automatic scanning program is as follows:

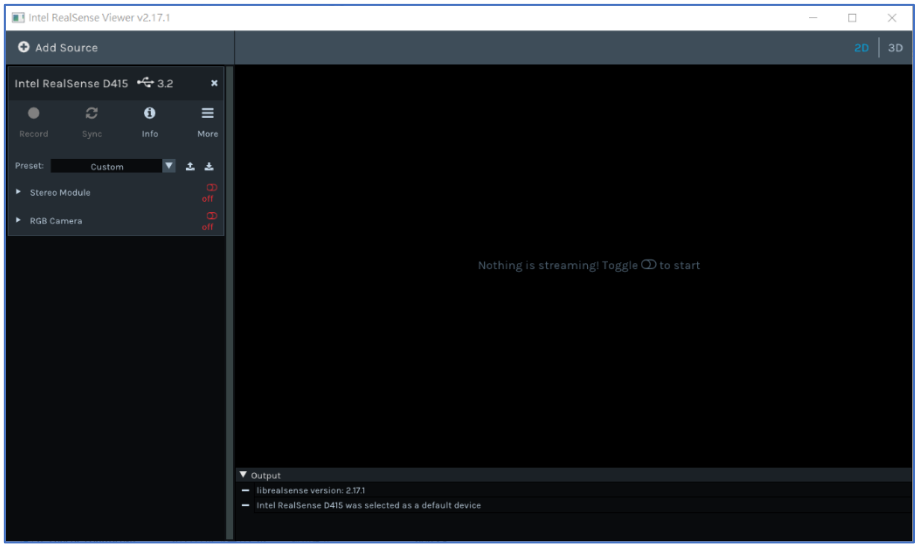


1. Click **LIPSedge™ AE400/ AE430 / AE450 Toolkit.**

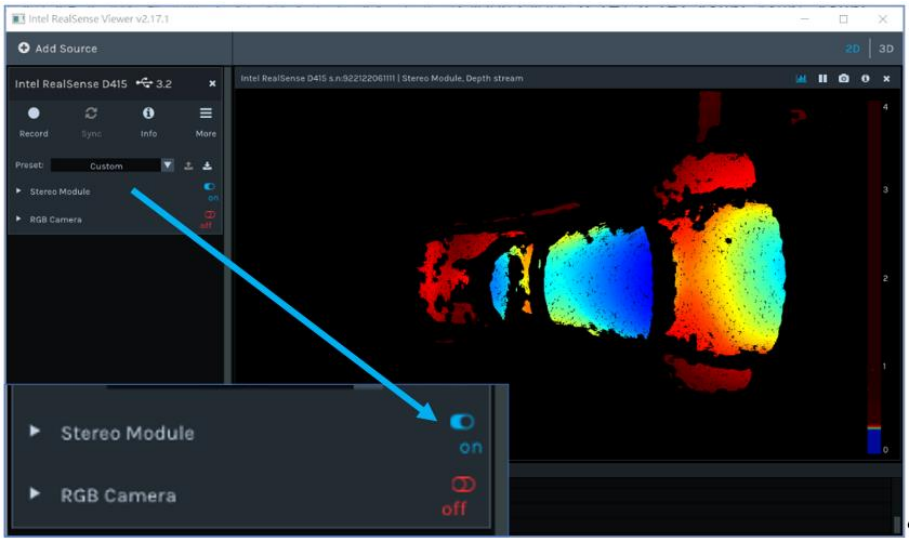


2. The toolkit automatically scans for available cameras.

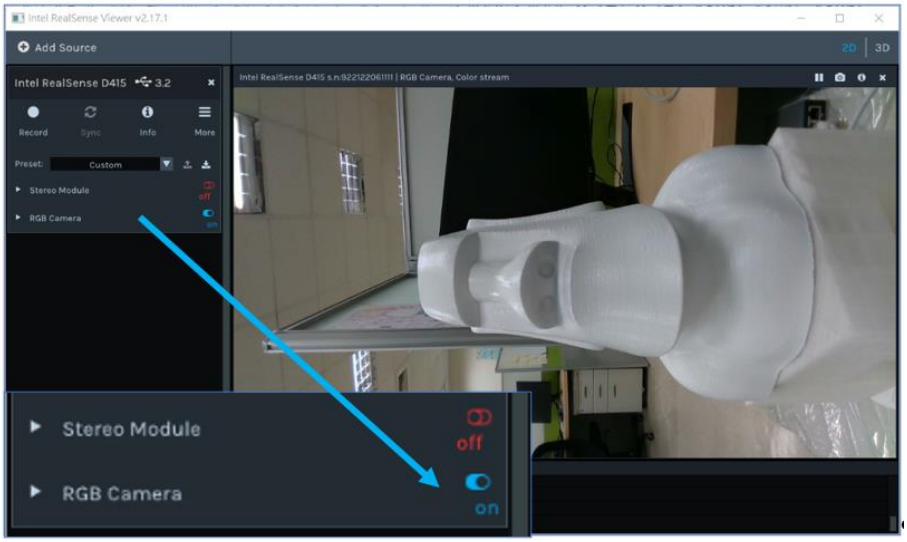
3. Return to the location of LIPSedge™ AE series SDK and start **realsense-viewer.exe**, the camera's image streaming program. Select **Run as administrator** when executing the program.



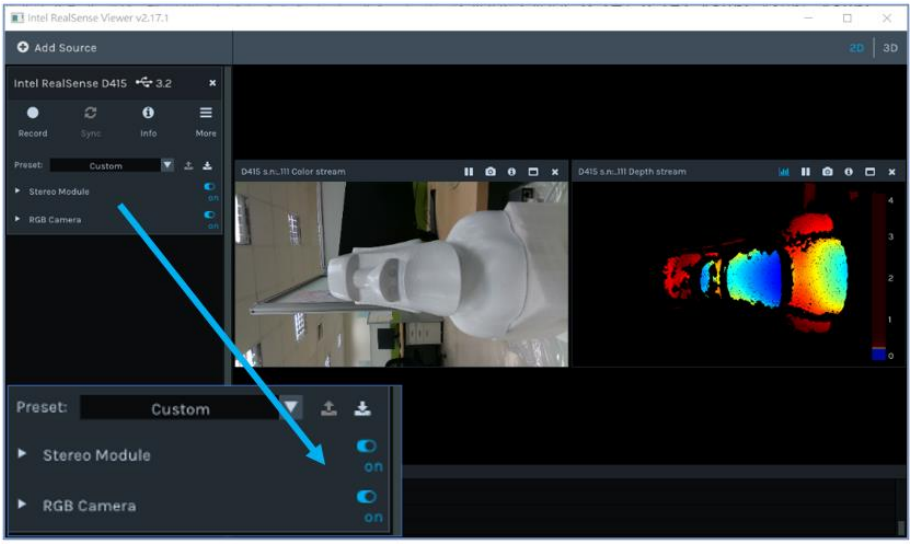
4. Toggle **ON / OFF** under **Stereo Module**. The viewer shows the real-time **depth image** of the LIPSedge™ AE series camera.



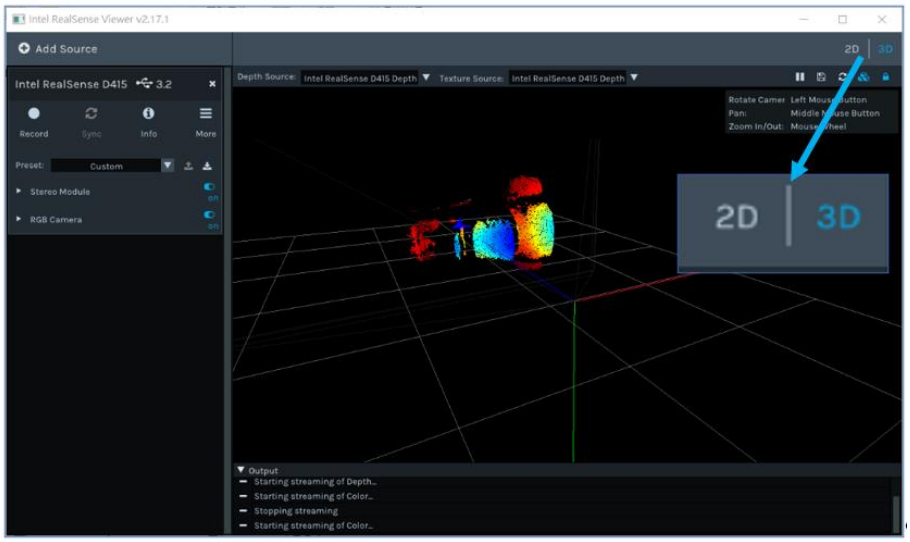
5. Toggle **ON / OFF** under **RGB Camera**. The viewer shows the real-time **RGB image** of the LIPSedge™ AE series camera.



5. Besides individual streaming, the Stereo Module / RGB Camera also allows developers to view both image types simultaneously.



6. Click the **2D | 3D** option on the upper right corner to view the **3D point-cloud** version of the 3D image.



C. LIPSedge™ L210u / L215u Series

To preview 3D images using the LIPSedge™ L series camera, install the LIPSedge™ L series SDK. For detailed information, contact our sales representative at info@lips-hci.com. LIPS Corp. will provide the latest version of the LIPSedge™ L series user manual.

After the installation of the SDK, which includes the OpenNI framework, go to the following directory to access the camera preview program:

Name	NIViewer.exe
Location	C:\Program Files\OpenNI2\Tools

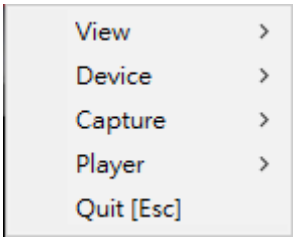
The NiViewer is only available for previewing LIPSedge™ L210u / L215u images and does not support configuring camera profiles. Before previewing, set up the corresponding configuration files based on the specific usage scenario. Failure to configure the profile files may result in incorrect images display.

For configuration file references, prefer to

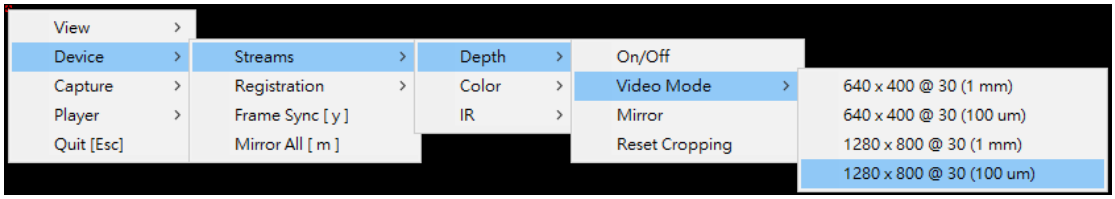
- LIPScan3D SDK\Config_files\L210\1280x720
- LIPScan3D SDK\Config_files\L210\640x480

directory in the SDK installation folder, which contains the **L210_config_640x400.json** and **L210_config_1280x800.json** files.

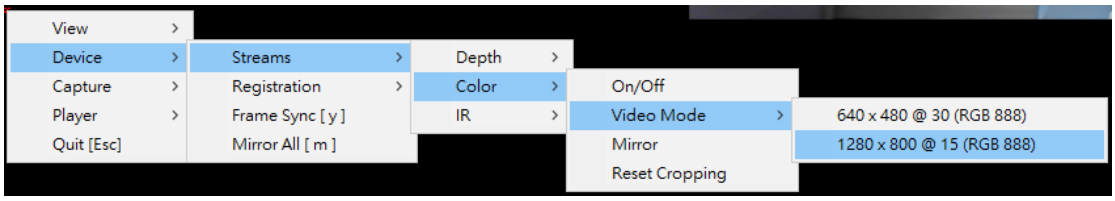
1. Right-click on **NIViewer**, a menu appears.



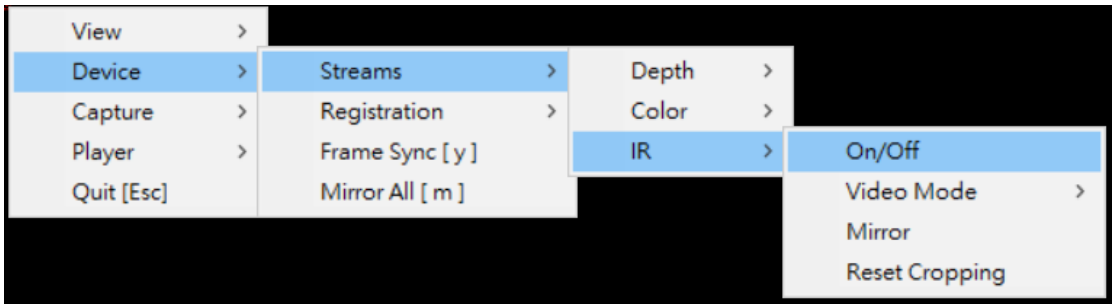
2. Select **Device** > **Streams** > **Depth** > **Video Mode** for desired depth image resolution / precision.



3. Select **Device** > **Streams** > **Color** > **Video Mode** for desired RGB image resolution / precision.



4. Select **Device** > **Streams** > **IR** > **On / Off** to adjust IR settings.



4. Source Code Compilation

This section details how to compile an example application of the LIPScan™ 3D Scan from the source code package and adjust the dependency configuration.

Glossary	
Terms	Definition
LIPScan™ 3D Scan SDK	The software development kit for LIPScan™ 3D Scan that includes the source code. dependent libraries, tools.
Example application	The LIPScan™ 3D Scan application built from the source code. The example application includes the functionality of the LIPScan™ 3D Scan. Developers can add / remove / modify the functions by programming.

A. Recommended Build Environment

LIPScan™ 3D Scan SDK and example application are compatible with Microsoft Visual Studio 2019, employing the **Release configuration** and **x64 platform**. It is recommended to use the same development environment, configuration, and platform when programming for LIPScan™ 3D Scan SDK. Refer to the table provided below for the third-party SDKs and their corresponding versions utilized by LIPScan™ 3D Scan SDK.

Library Names	Library Versions	Dynamic Libraries	Static Libraries	Header File Directory
Boost	1.69.0	N/A	N/A	3rd party/boost_1_69_0
OpenCV	4.5.5	opencv_core455.dll opencv_highgui455.dll opencv_imgcodecs455.dll opencv_imgproc455.dll opencv_videoio455.dll	opencv_core455.lib opencv_imgproc455.lib opencv_highgui455.lib	3rd party/opencv-4.5.5/include
L210u SDK	1.0.2.0	OpenNI2.dll OPENNI2/Drivers/…	N/A	N/A
Intel® RealSense™ SDK 2.0	2.50.0	realsense2.dll	N/A	N/A
AE400 SDK	2.43.0	realsense2.dll	N/A	N/A
LIPScan3D	2.0.4	librealsense.dll liblipscan3d.dll libgmp-10.dll CCCoreLib.dll camera/AE400/DepthCamAE400.dll camera/AE400/DepthCamAE430_470.dll camera/D415/DepthCamD415.dll camera/L210/DepthCamL210.dll	librealsense.lib liblipscan3d.lib	include

B. Environment Configuration (SDK)

a. SDK Installation

To obtain the LIPScan™ 3D Scan SDK, contact info@lips-hci.com. A download link will be provided. The SDK is compressed in a .7z format, which is decompressible by using the 7-Zip application avaiable at https://www.7-zip.org/. Once the download link is received and the 7-Zip application is ready, follow the instructions below to install the LIPScan™ 3D Scan SDK.:

1. Download and install 7-Zip from https://www.7-zip.org/.

2. From the download link, download and extract the LIPScan™ 3D Scan SDK.

3. Extracted SDK folder is as follows

- 3rd party: contains the boost library and opencv.
- bin: contains the dynamic library files (.dll) of the SDK.
- camera: contains dynamic library files (.dll) that support camera types. The placement path is the application (.exe) directory (or in the case of debugging, it is placed in the Visual Studio solution (.sln) directory).
- Config_files: contains configuration files (.json) for supported camera types and models in the SDK.
- Include: contains API header files (.h) for the SDK.
- lib: contains static libraries (.lib).
- License: contains license declaration files for third-party libraries.

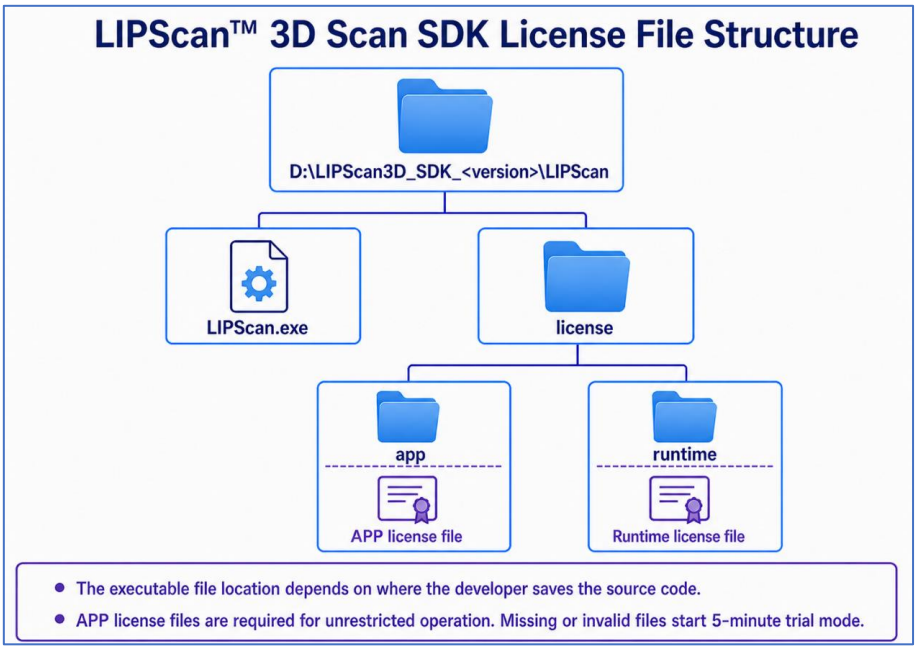
b. License File Settings

LIPScan™ 3D Scan SDK requires **two license types** for unrestricted operation: an **SDK license** and a **runtime license**. The SDK license authorizes the SDK to operate on the specified device, while the runtime license authorizes the developer-selected device to run the application developed with the SDK.

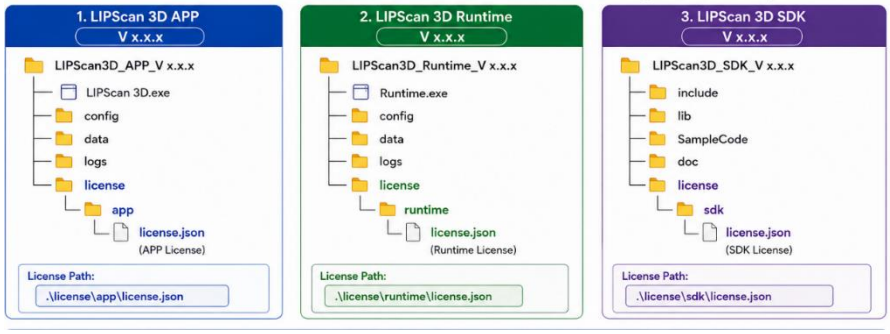
The executable application must be compiled from its source code. The executable file location may vary depending on where the developer saves it.

After the compiler successfully compiled executable from the source code, place the **license** folder in the same application directory as LIPScan.exe. Within this folder, place the SDK license file in `license\sdk` and the runtime license file in `license\runtime`. For example: If the executable is in D:\LIPScan3D_SDK_<version>\LIPScan\LIPScan.exe,

- then place the SDK license in D:\LIPScan3D_SDK_<version>\LIPScan\license\sdk\<SDK license file>
- runtime license in D:\LIPScan3D_SDK_<version>\LIPScan\license\runtime\<runtime license file>

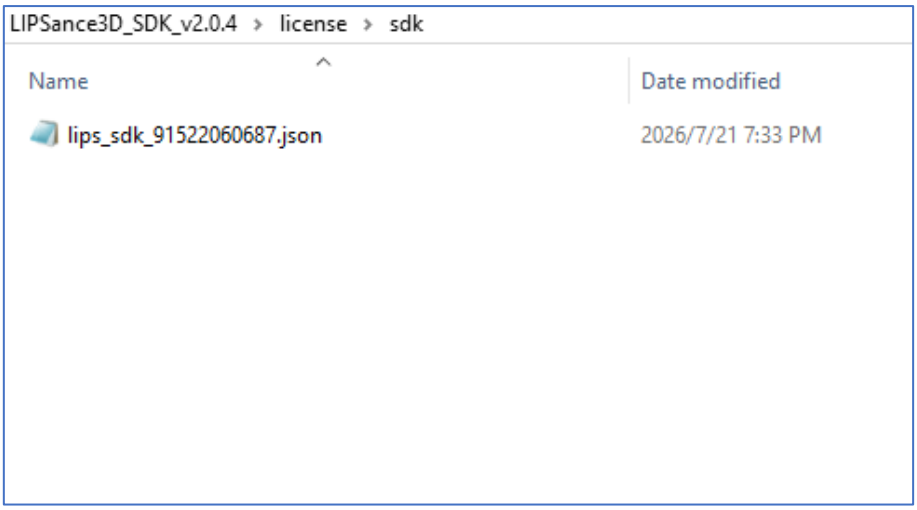


For details about the file structure, refer to the infographics below:

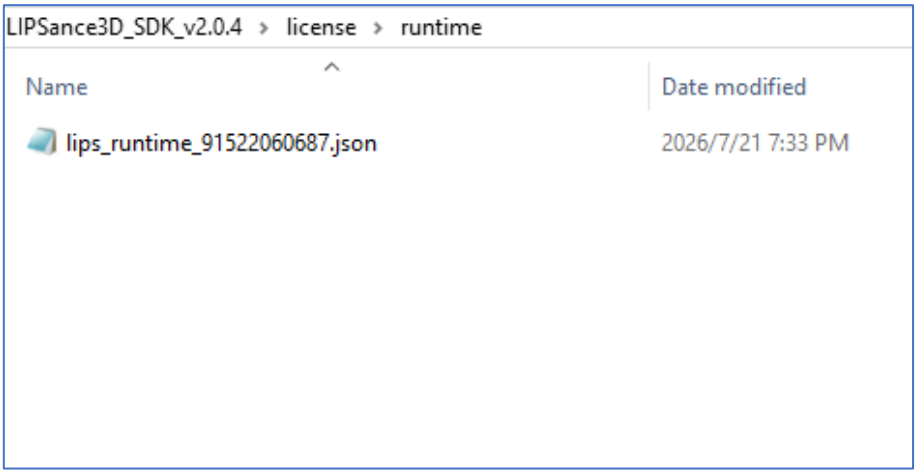


LIPScan™ 3D Scan SDK checks the SDK license location while the application is running. If either required license file is missing, invalid, or unreadable, the SDK enters a 5-minute trial mode. When the trial period expires, a termination notice is displayed and the application closes.

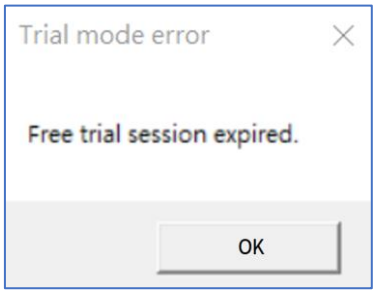
1. Place the **lips_license.json** file under the following location. In this example, a lips_license_915422060687.json is used.
   Location: D:\LIPScan3D_SDK_<version number>\license\sdk\



2. Optioanlly, place the **lips_runtime.json** file under the following location when the developer had developed applications using the SDK. In this example, a lips_runtime_915422060687.json is used.
   Location: D:\LIPScan3D_SDK_<version number>\license\runtime\



3. When the files is absent, the following notice appears when the time limit is met and the applicated terminates upon clicking **OK**. Once terminated, the camera images and 3D model becomes inaccessible.



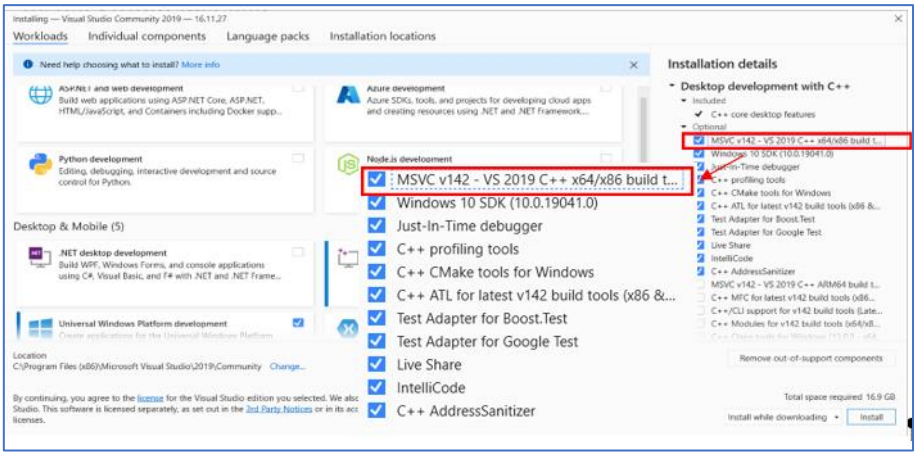
C. Environment Configuration (Compiler)

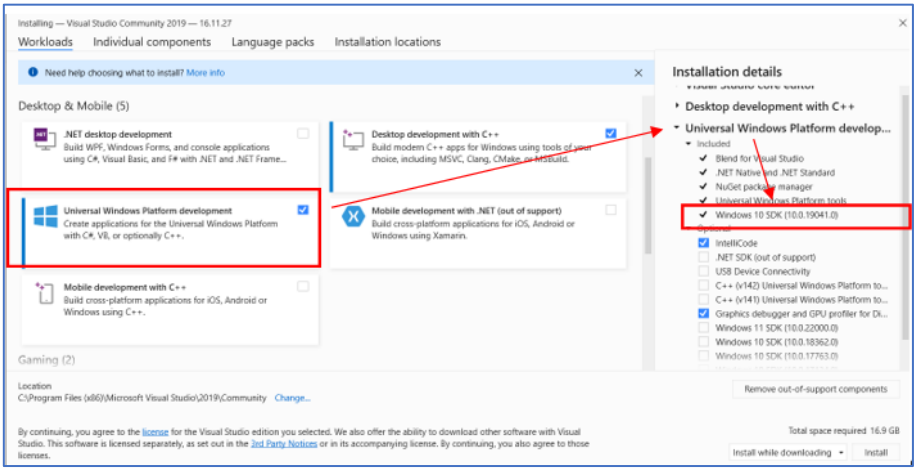
a. Workload Settings

Prior to starting the compilation of LIPScan™ 3D Scan SDK example applications, it is crucial to install the following Microsoft Visual Studio workloads:

- MSVC v142 - VS 2019 C++ x64/x86
- Windows 10 SDK (10.0.17763.0)

1. Follow the figures below to install the **workloads**:



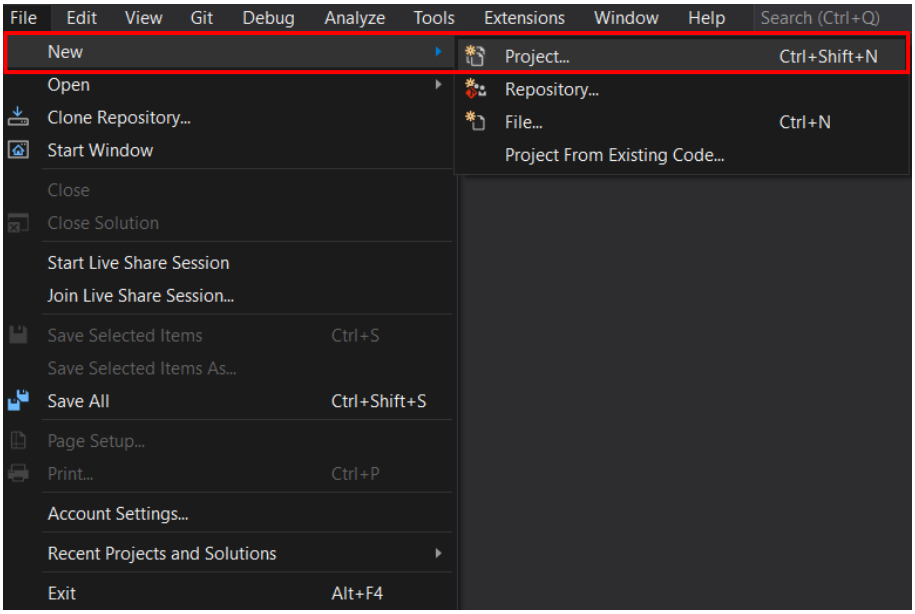


If a 30-day trial license is required, sign in with a Microsoft account to access a free license.

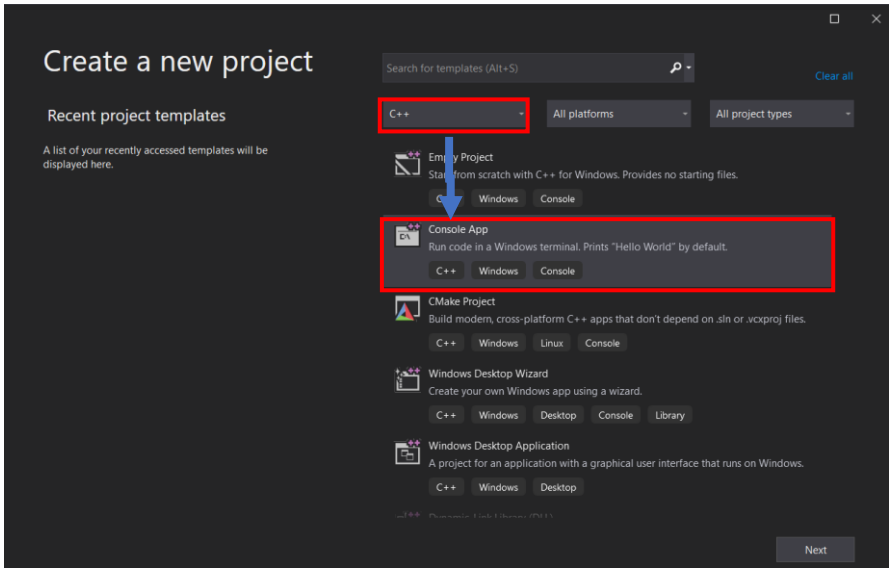
b. Building from Source Code

Follow the instructions below to complete the configuration of the example application, platform, and environment settings for the LIPScan™ 3D Scan SDK library required for the compilation process. Once the compilation configuration is successful, **build the example application**. This example is based on a C++ console application.

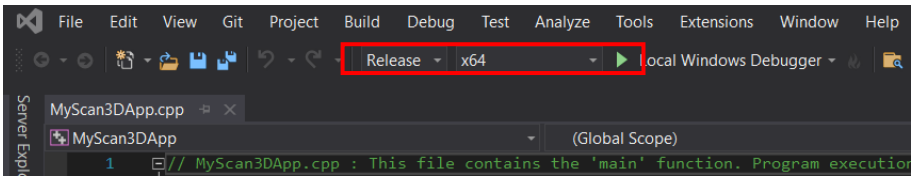
1. Open Microsoft Visual Studio 2019 and go to **File > New > Project**.



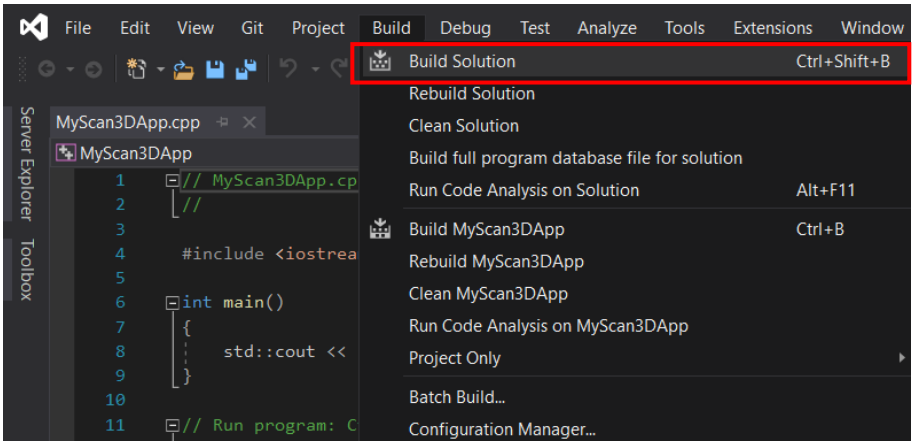
2. Select **C++ > Console App** and assign the project name / the location for place the example application project. For example, "D:\workspace".



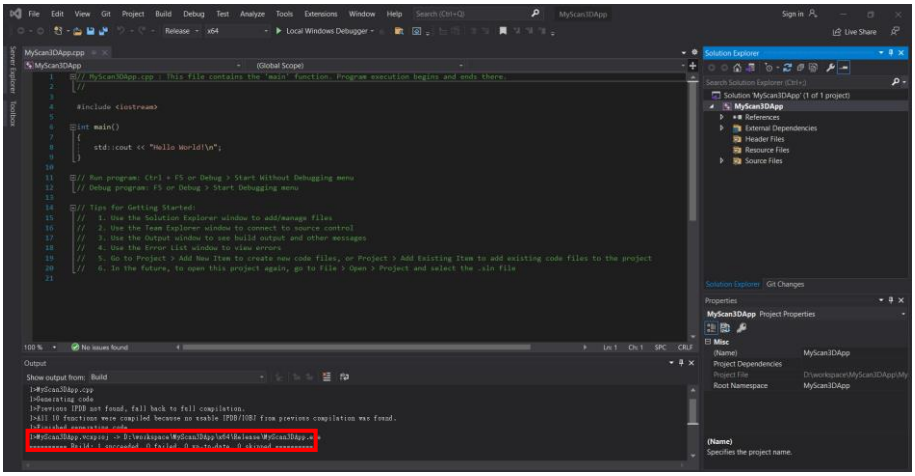
3. After adding the project, select the **solution configuration**:
   - **Solution Configuration**: Release
   - **Solution Platform**: x64



4. Click on the menu **Build > Build Solution**.



5. In the case of a successful build, the output window will display the message: **1 succeeded, 0 failed, 0 up-to-date, 0 skipped**.



Successful build is **insufficient** for running LIPScan™ 3D Scan example application. To run LIPScan™ 3D Scan example application, refer to the following chapters to **complete subsequent configuration settings**.

c. Dependency Configuration

The example application completed in the initial build is NOT immediately executable. This is due to its reliance on dependencies from the LIPScan™ 3D Scan SDK, which are essential to establish a functional camera connection and process image data. To ensure the example application is readily executable, follow the instructions to set up the dependency configuration and rebuild the source code.

The relevant dependency settings include:

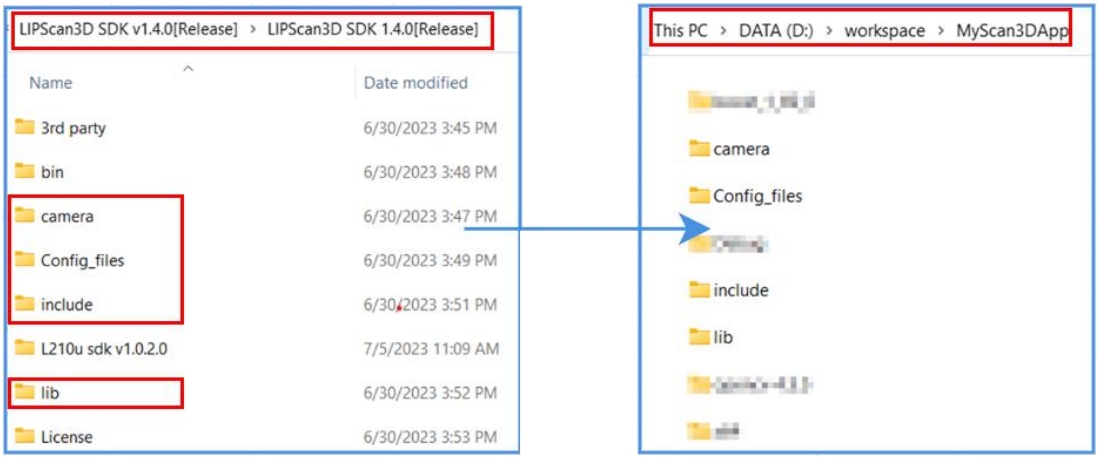
- Camera retrieval configuration: Setting files that specifies the camera intended for connection and the image mode for specific scenario.
- Binary and dynamic library configuration: Setting files that call third-party frameworks or libraries for image processing.
- Additional include directory configuration: Refer to *4-C-c. Compiler Property Settings*.

[Camera Retrieval Configuration]

During runtime, the LIPScan™ 3D Scan example application utilizes a **camera.json** file identifying the connected camera model. This file contains an identification string crucial for establishing the camera connection. To set up the identification string that matches the camera intended for connection, refer to *5-a. Camera Retrieval Configuration*.

1. Build the LIPScan™ 3D Scan.

2. Go to LIPScan™ 3D Scan SDK's location and copy the following folders, which contain the camera dependencies, to the LIPScan™ 3D Scan example application's location.
   - **camera**
   - **Config_files**
   - **include**
   - **lib**

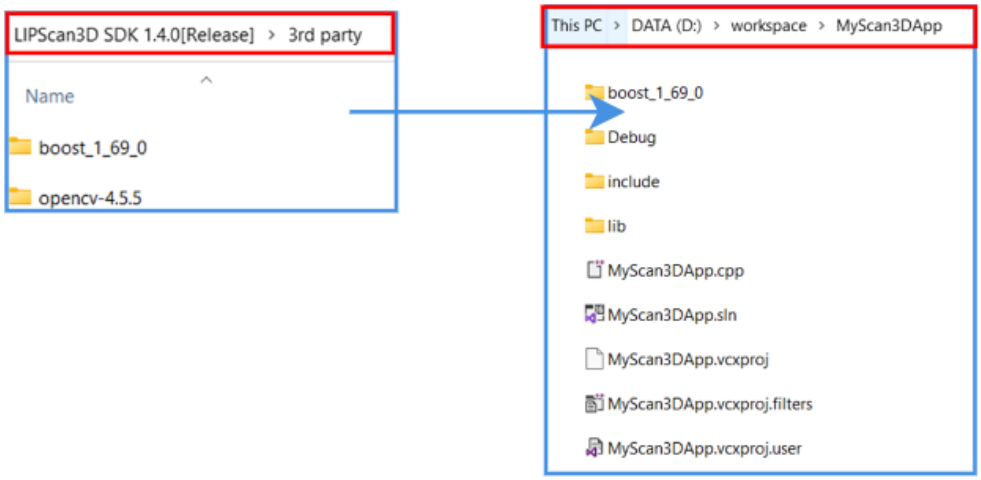


3. Refer to *5-a. Camera Retrieval Configuration* for camera identification string setup.

[OpenCV Dependencies]

Follow the instructions to move the OpenCV dependencies to the LIPScan™ 3D Scan example application's location.

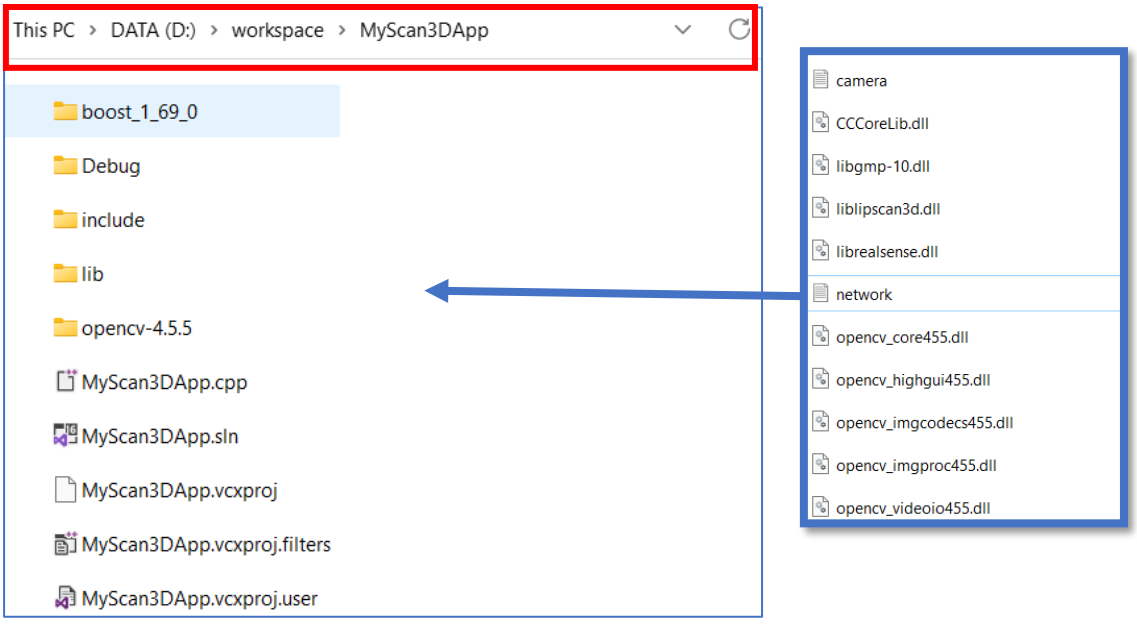
4. From **LIPScan™ 3D Scan SDK > 3rd party**, copy the following folders to the LIPScan™ 3D Scan example application's location.
   - **LIPScan™ 3D Scan SDK \3rdparty\boost_1_69_0**
   - **LIPScan™ 3D Scan SDK \3rdparty\opencv-4.5.5**



[Binary and dynamic library configuration]

Follow the instructions to move the binary and dynamic library dependencies to the LIPScan™ 3D Scan example application's location.

5. From **LIPScan™ 3D Scan SDK > bin**, copy the following folders to the LIPScan™ 3D Scan example application's location.
   - camera
   - CCCoreLib.dll
   - libgmp-10.dll
   - liblipscan3d.dll
   - librealsense.dll
   - network
   - opencvcore455.dll
   - opencv_highgui455.dll
   - opencv_imgcodec455.dll
   - opencv_imgproc455.dll
   - opencv_videoio455.dll

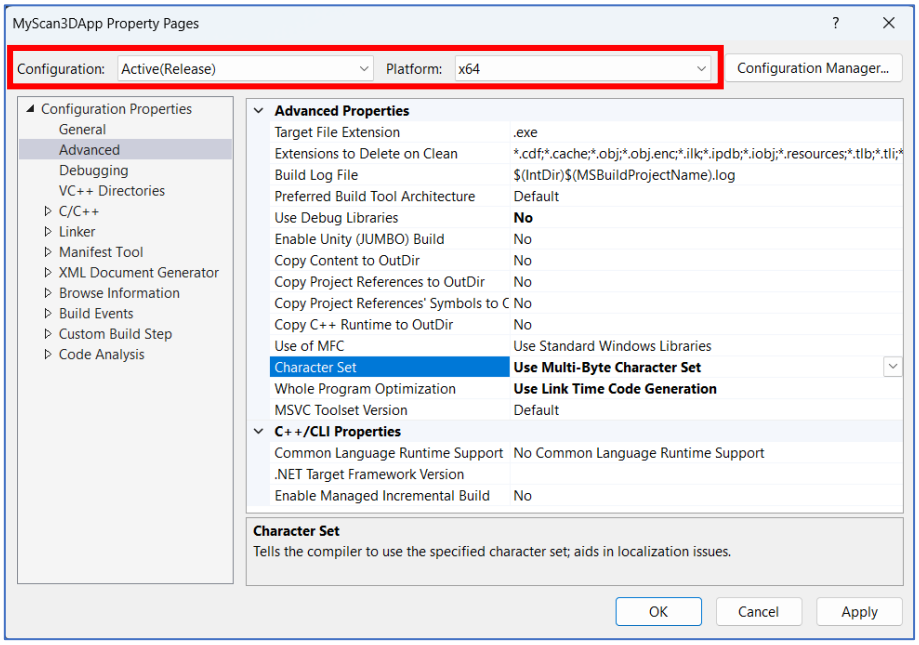


d. Compiler Property Settings

Follow the instructions to set up the compiler's property, including character set and additional libraries.

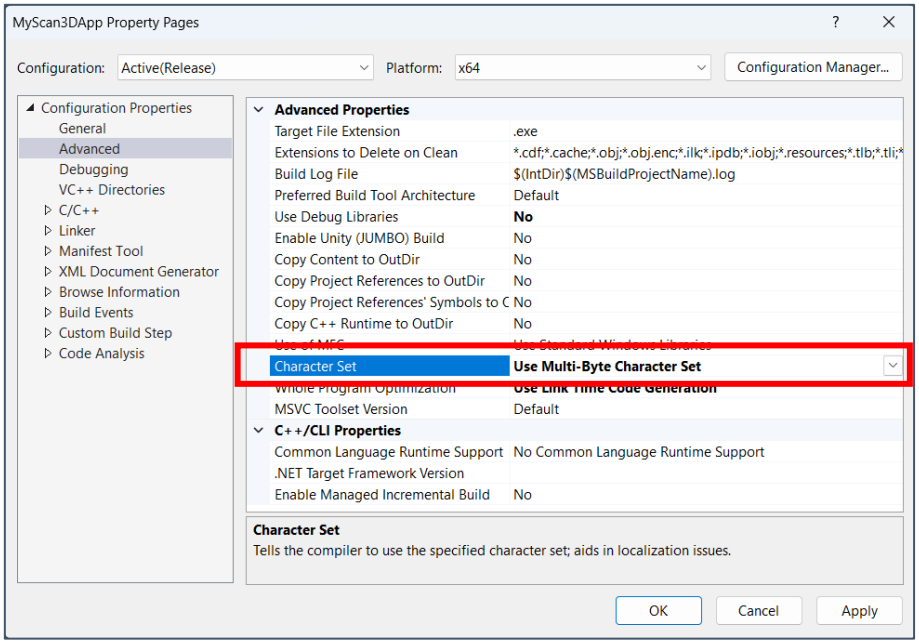
1. On the right menu, right-click the example application and select **Properties**. A dialog box pops up.

2. On the top menu, select the following options:
   - **Configuration**: Release
   - **Platform**: x64



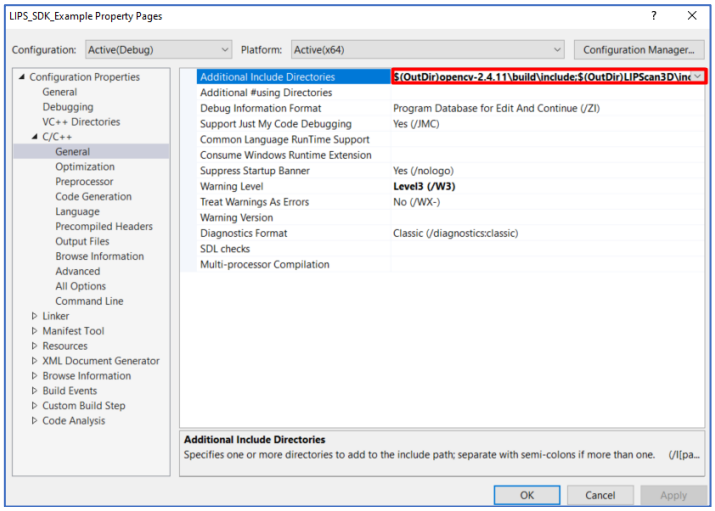
[Character Set Settings]

3. On the left menu, click **Configuration Properties > Advanced > Character Set** and select **Use Multi-Byte Character Set**.



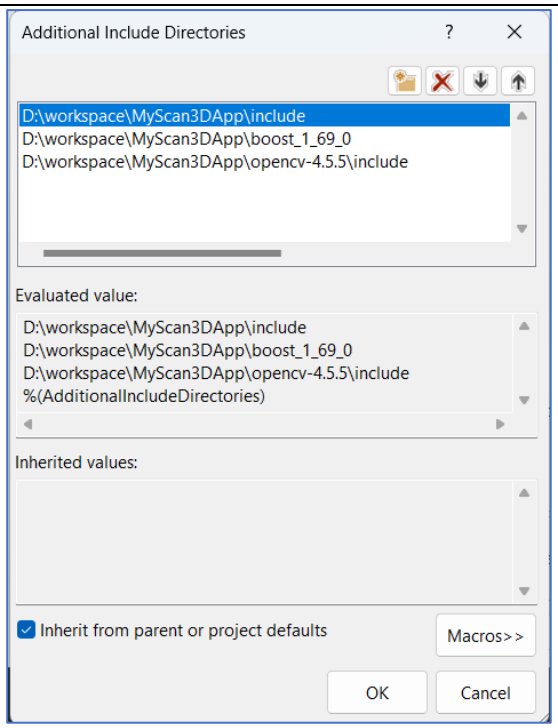
[C / C++ Settings]

4. On the left menu, go to **Configuration Properties** > **C / C++** > **General** and expand **Additional Include Directories**.



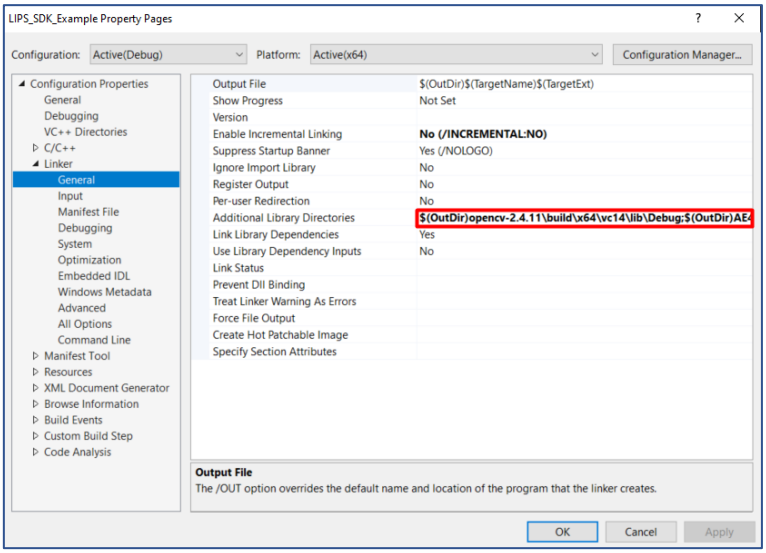
5. Add the following paths to **Additional Include Directories** and click **Apply**.

Additional Include Directories
D:\workspace\[LIPScan™ 3D Scan example application]\include
D:\workspace\[LIPScan™ 3D Scan example application]\boost_1_69_0
D:\workspace\[LIPScan™ 3D Scan example application]\opencv-4.5.5\include %(Additional Include Directories)



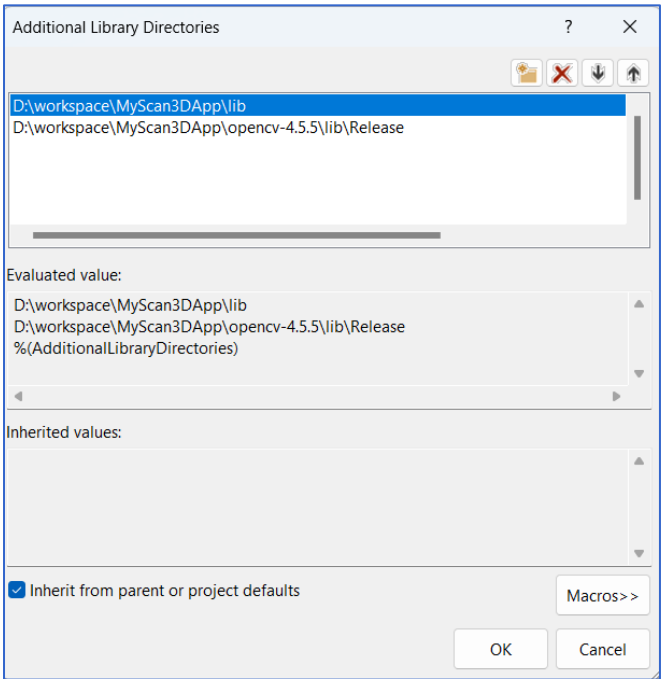
[Linker Settings]

6. On the left menu, go to **Configuration Properties** > **Linker** > **General** and expand **Additional Library Directories**.



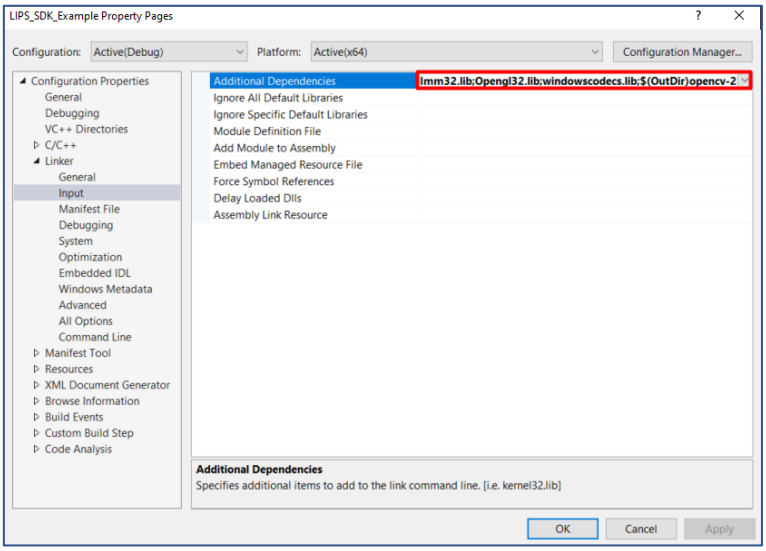
7. Add the following paths to **Additional Library Directories** and click **Apply**.

Additional Library Directory
D:\workspace\[LIPScan™ 3D Scan example application]\lib
D:\workspace\[LIPScan™ 3D Scan example application]\lib\Release



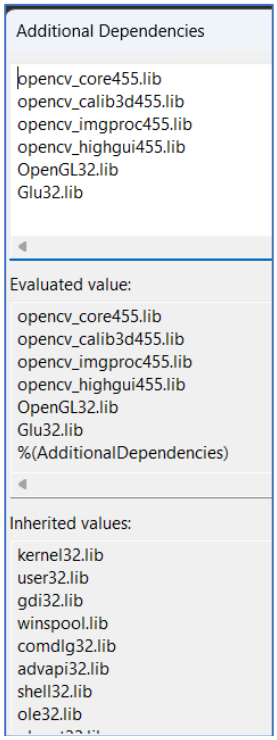
[Other Dependency Settings]

8. Under **Linker**, select **Input** > **Additional Dependencies**.



9. Add the following paths to **Additional Dependencies** and click **Apply**.

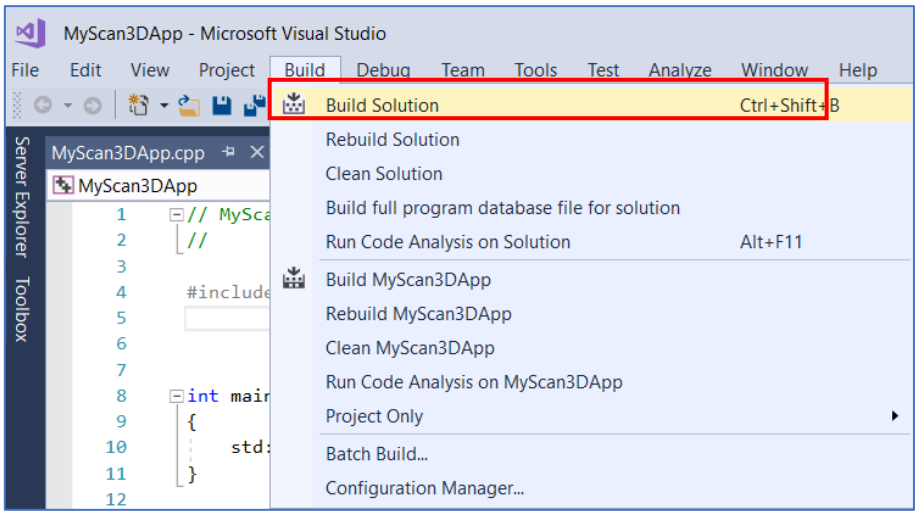
- opencv_core455.lib
- opencv_calib3d455.lib
- opencv_imgproc455.dll
- opencv_highgui455.lib
- OpenGL32.lib
- Glu32.lib



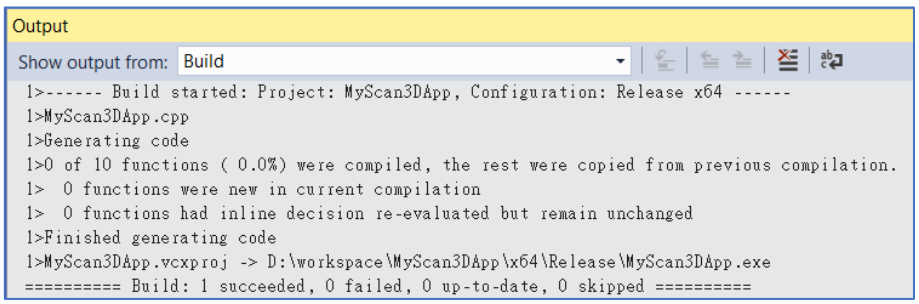
[Rebuild the Example Application]

Once the initial build is finished and the dependencies are in place, rebuild the LIPScan™ 3D Scan example application in **release mode** for a readily executable application. To achieve this, make sure to refer to *4-C-b. Building from Source Code* to adjust the solution configuration.

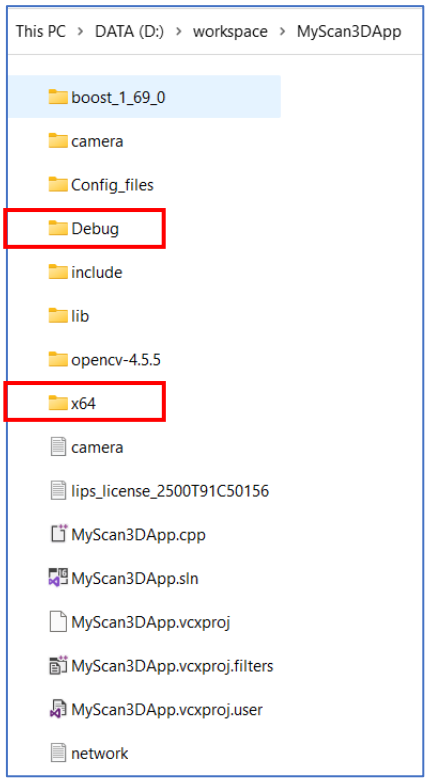
10. Click on the menu **Build > Build Solution**.



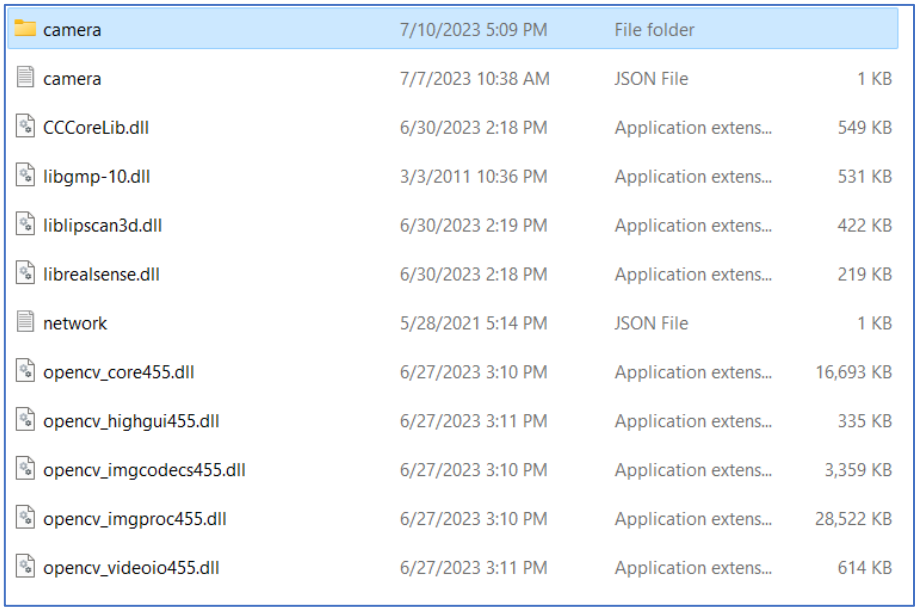
11. In the case of a successful build, the output window will display the message: **1 succeeded, 0 failed, 0 up-to-date, 0 skipped**.



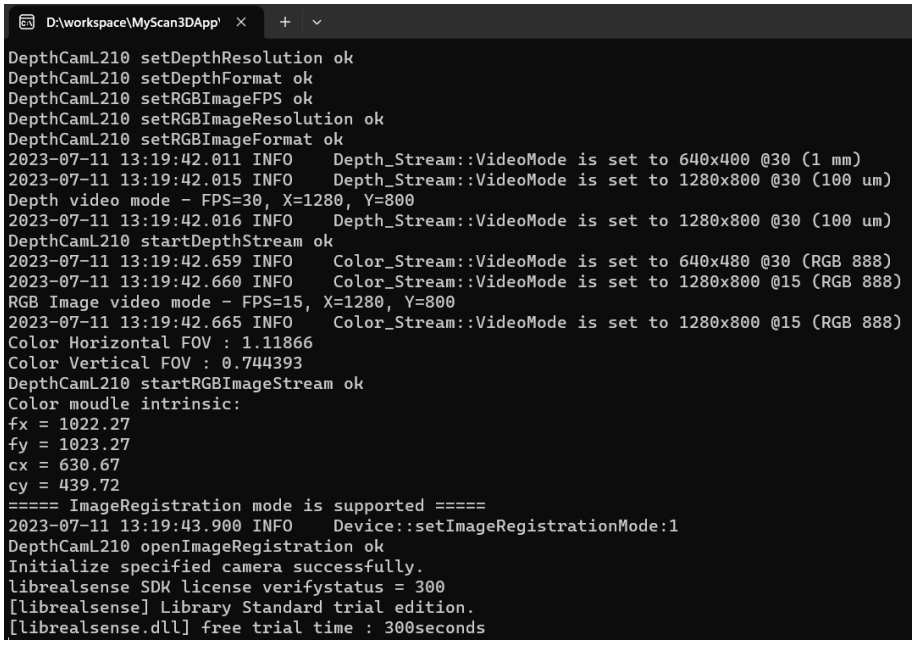
12. Once the executable built is completed, 2 new folders will be created:
    - Debug
    - x64



13. In the LIPScan™ 3D ScanSDK folder, make a copy of the following folder and files to the ..\LIPScan™ 3D Scan SDK\x64\Release.
    - ..\LIPScan™ 3D Scan\camera
    - All files within ..\LIPScan™ 3D Scan\bin



14. Once the build is successful, go to the installation directory of the example application. In this example, the installation directory is D:\workspace\MyScan3DApp\x64\Release.



e. Temporary Debugging Build

The example application, constructed under the **Release configuration**, is equipped with pre-defined functionalities. Developers enjoy the flexibility to alter or append new features to this application. While programming, developers might require a temporary build for debugging. To accommodate this need, developers can construct a temporary example application using the **Debug configuration**.

1. On the top menu, change the **solution configuration**:
   - **Solution Configuration**: Debug
   - **Solution Platform**: x64

2. Paste the source code to Microsoft Visual Studio 2019. Make the modification needed and build the project in Debug configuration.

```
#include <iostream>
#include <Windows.h>
#include <configParser.h>
#include <lips_camera.h>
#include <lips_scanner.h>

int main()
{
	lips::Camera* camera;
	lips::ScannerProcess* scanner;
	int device_slot = -1;
	char device_name[512];
	camera = new lips::Camera(device_slot, device_name); //L210 Camera
can not connected in debug process
	if (device_slot != -1)
		std::cout << "Connected Camera : " << device_name << endl;
	std::cout << "Hello LIPScan3D SDK!\n";
	// Initialzie camera with configuration file
	std::string config_file =
"D:\\workspace\\MyScan3DApp\\Config_files\\L210\\1280x720\\L210_config_1280
x800.json"; //Change into your own directory
	bool bInit = camera->InitCamera(config_file.c_str());
	if (!bInit)
		std::cout << "Initialize camera configuration failed." <<
endl;
	//Preview camera live color image
	cv::Mat color_img(800, 1280, CV_8UC3);
	cv::Mat depth_img(800, 1280, CV_16UC1); //For AE series and Intel
Real Sense please change into (720.1280)
	do
	{
		camera->GetFrame((unsigned char*)color_img.data, (unsigned
short*)depth_img.data);
		cv::imshow("Color Frame", color_img);
		cv::waitKey(1);
	} while (bInit);
	system("pause");
```

5. Application Development

LIPScan™ 3D Scan SDK enables developers to implement the following functions to the example application:

- Real-time camera image preview
- Reconstruction of 3D mesh models
- Loading and saving of triangle mesh model files
- Accessing triangle mesh model data (points/lines/faces)
- Analyzing the accuracy and precision of triangle mesh models
- Exporting files for error analysis of triangle mesh models

The functionalities require the following configuration parameter files to be properly set up. Verify that they are in place and are properly set up before executing the LIPScan™ 3D Scan example application:

Configuration	Function
Camera retrieval configuration	**Location**: ..\camera.json
	Identifies the camera intended for connection.
Distance mode configuration	**Location**: \Config_files\camera_model
	Identifies the scanning configuration in relation to object size and measurement distance, called a distance mode, for supported camera models.
3D scanning resolution setting	Identifies the resolution for the 3D scanning process.
License file settings	The authentication information is specified by the camera serial number. For detailed information, refer to *4-A-c. License File Settings*.

A. Camera Retrieval Configuration

During runtime, the LIPScan™ 3D Scan example application utilizes a **camera.json** file identifying the connected camera model. The location of the file:

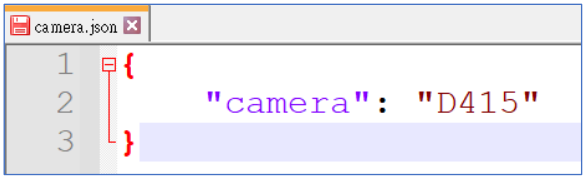
- **Location:**..\[LIPScan™ 3D Scan example application]\bin

This file contains an identification string crucial for establishing the camera connection. If the connected camera does not align with the parameters specified in camera.json, the application will terminate. Below is a list of parameters representing each supported model:

Camera Type	camera.json parameters
Intel® RealSense™ D415 / D435 / D435i / D455	D415
LIPSedge™ AE400 / 450	AE400
LIPSedge™ AE430/470	AE430
LIPSedge™ L210u / 215u	L210

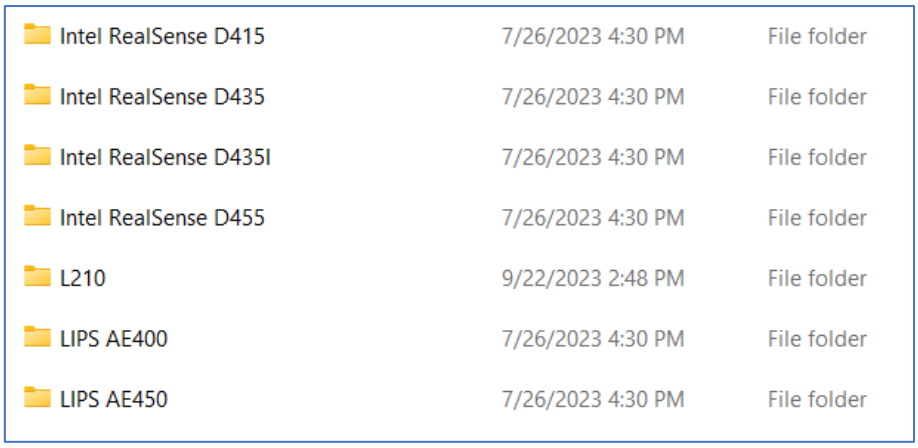
The camera.json is located within LIPScan™ 3D Scan SDK folder hence needs to be moved to the example application's location. Follow the instructions to complete camera connection configuration.

1. Open camera.json and confirm the camera identification string aligns with the camera intended for connection.



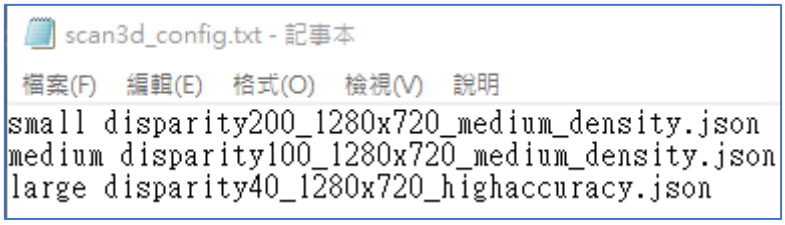
B. Distance Mode Configuration

Once the LIPScan™ 3D Scan example application retrieved the camera, it then loads the associated configuration files. In this example, default configuration file settings for the **7 camera models** series are provided:



Each camera model configuration folder contains **2 types of files**:

- **configuration file (.json)**: The camera view configuration file.
- **scan3d_config.txt**: Specifies the distance modes applied by the camera for various ranges:
  - **Small**: Suitable for the measurement of a smaller object at a closer range.
  - **Medium**: Suitable for the measurement of a medium-sized object at a further range.
  - **Large**: Suitable for the measurement of a larger object at a far range.



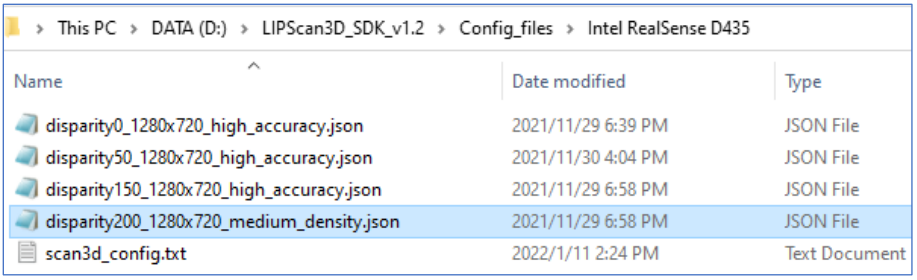
During 3D scanning, LIPScan™ 3D Scan example application retrieves the corresponding camera configuration file (.json) from **scan3d_config.txt** based on the scanning mode selected.

[Configuration File Addition]

When existing camera configuration doesn't align with measurement need, developers can create a new configuration file tailored for specific measurement needs and update it in the **scan3d_config.txt**.

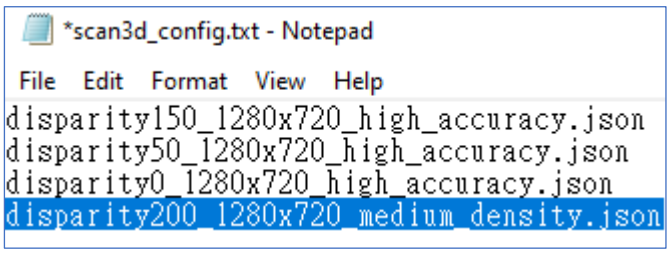
1. Create a configuration file. For details on how to create a configuration file, refer to *5-c, 3D Scanning Resolution Setting.*

2. Place the configuration file in:
   - Location: \ [LIPScan™ 3D Scan SDK] \Config_files\ [Camera Model]:



Note: When using Intel® RealSense™ D435, LIPScan™ 3D currently supports1280 * 720 / 640 * 480 resolution.

3. Open scan3d_config.txt and add the configuration file's name to scan3d_config.txt:



C. 3D Scanning Resolution Setting

The supported 3D imaging distance varies according to the resolution of each supported model. As a result, it is required for the developers to set up the 3D scanning resolution configuration according to specific needs and save the settings as a configuration file for optimal depth image formation.

According to the test data, the higher the camera resolution, the farther the minimum depth image formation distance. As the result, scanning a medium sized or larger object at a short distance (Distance mode – small) is NOT recommended. Mismatch between the object size / resolution setting may cause incomplete scanning results.

a. Intel® RealSense™ D400 Series

When the Intel® RealSense™ camera's resolution is defined, it is required to set up its disparity shift parameter, which determines the camera's depth image formation distance.

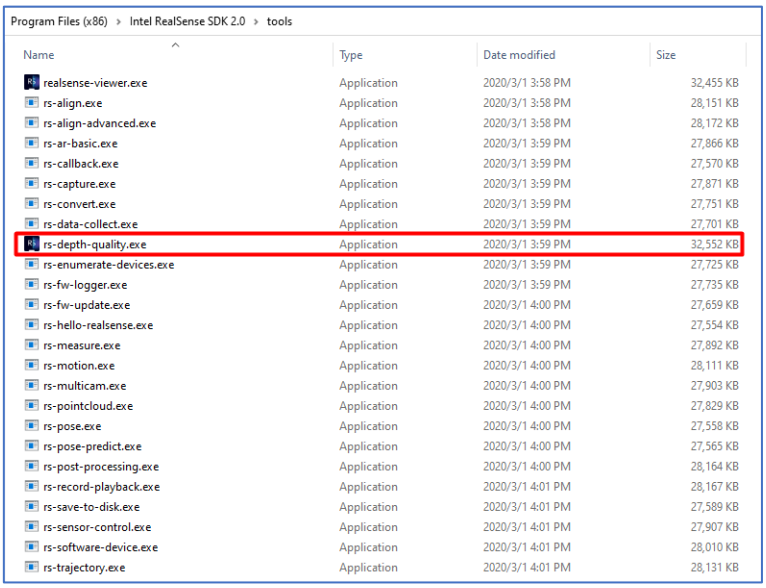
Take Intel® RealSense™ D415 as an example, when camera resolution is set at 1280 * 720, and the disparity shift parameter is set at 200, the camera will have a min-Z of 150.911 (mm) and a max-Z of 245.985 (mm). This value means that the depth image will be formed within the range of 150.911 - 245.985 (mm). As the result, it is required for each distance mode to have a corresponding disparity shift setting.

Based on the test results, the following settings for Intel® RealSense™ D400 models is recommended:

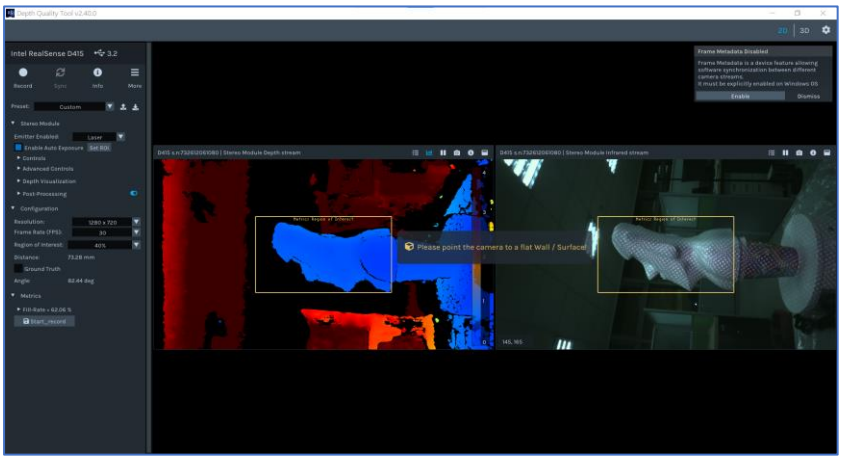
Device	Distance Modes	Disparity Shift	Min-Z(min)	Max-Z(max)
D415	Small	200	150.9107746	245.9845627
	Medium	100	217.6854537	491.9691253
	Large	40	296.366943	1229.922813
D435 / 435i	Small	150	116.1034626	213.6303711
	Medium	50	182.071339	640.8911133
	Large	30	205.4138184	1068.151855
D455	Small	200	188.186418	306.7438614
	Medium	150	222.2781604	408.9918152
	Large	60	329.832109	1022.479538

To generate and export a camera configuration file based on the adjustment of the disparity shift, use the rs-depth-quality.exe. For adding the camera configuration to the distance mode configuration file, refer to *5-b. Distance Mode Configuration*.

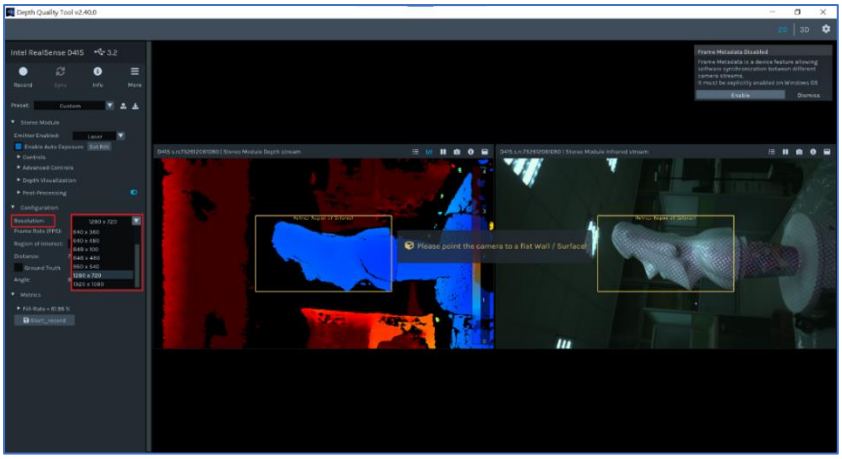
- **Application**: rs-depth-quality.exe
- **Location**: C:\Program Files (x86)\Intel RealSense SDK 2.0\tools



1. Start **rs-depth quality** as the figure below.

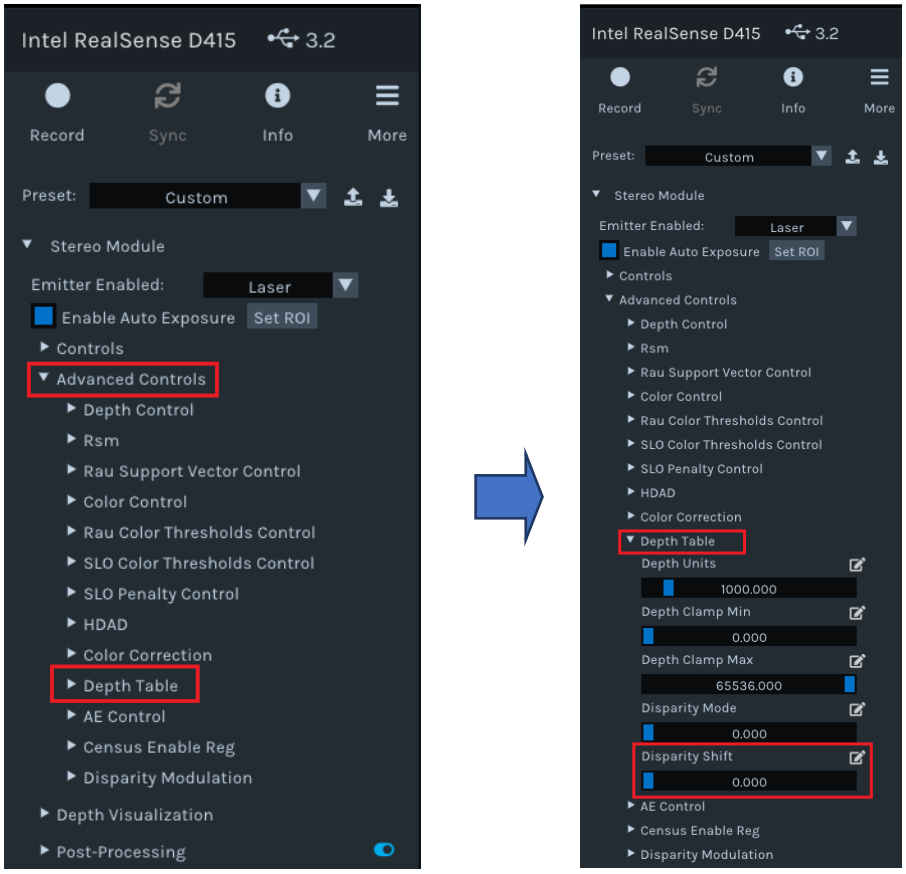


2. On the left menu, select **Resolution** and select the desired resolution from the drop-down list.

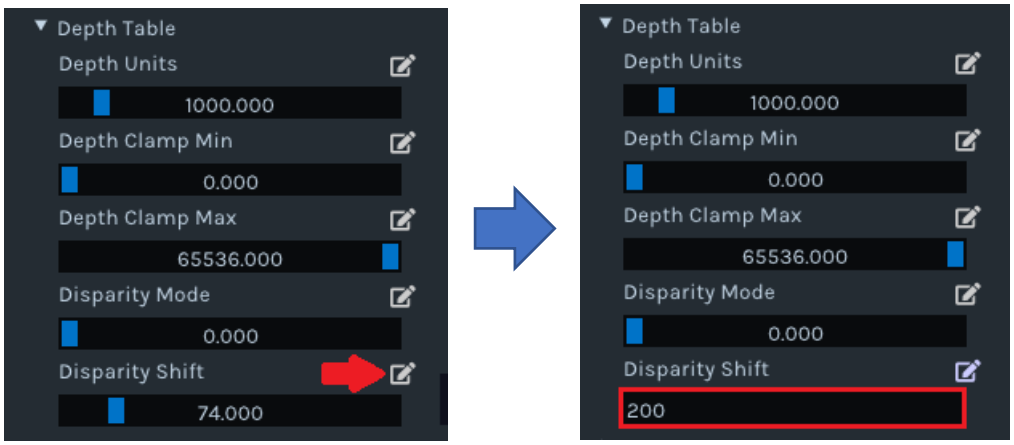


3. Select **Advanced Controls** and expand **Depth Table**.

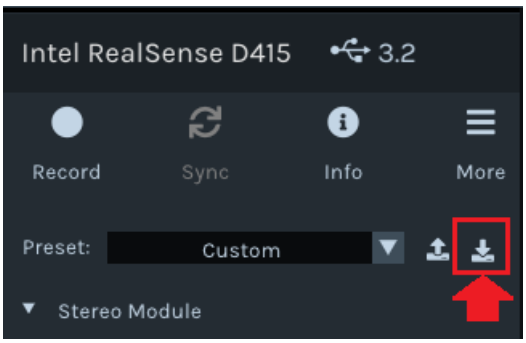
4. Toggle to adjust the **Disparity Shift** settings.



5. Alternatively, click the  icon on the upper right corner and type the **Disparity Shift** value.



6. Once the resolution and the disparity shift setting are completed, click the  icon to download the Intel® RealSense™ configuration file (.json).



b. LIPSedge™ L Series

To adjust the 3D scanning resolution settings for LIPSedge™ L210u / L215u, modifying the resolution / fps values in the LIPSedge™ L210u / L215u's configuration, which is available at the following location:

- File: L210_config_[Resolution].json
- Location: LIPScan3D SDK v2.0.4[Release]\Config_files\L210\[Resolution]

By default, the scanning mode for various resolutions are categorized by the folders named as the resolution.

- HD Scan: 1280 * 720 folder
- VGA Scan: 640 * 480 folder

Each resolution folder contains 2 types of files:

- configuration file (.json): The camera view configuration file. In the file, there are 4 parameters that forms a 3D image:
  - depth-stream-fps: The fps of the depth image streaming channel.
  - rgb-stream-fps
  - stream-height
  - stream-width
- scan3d_config.txt: Specifies the distance modes applied by the camera for various ranges:

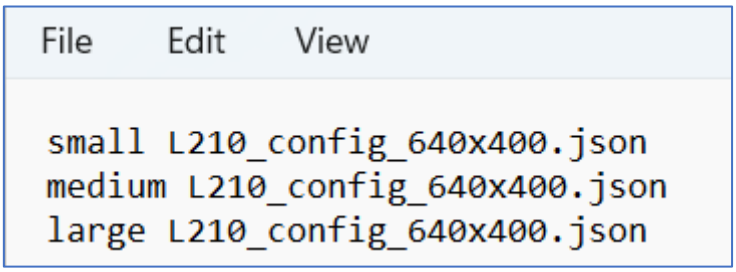
himax_config_1280x800.json

```
{
    "depth-stream-fps": "30",
    "rgb-stream-fps": "15",
    "stream-height": "800",
    "stream-width": "1280"
}
```

To set the scanning mode as VGA, modify the following values:

- stream-width: 640
- stream-height: 400

scan3d_config.txt contains a list of the types of configuration files for Small \ Medium \ Large distance modes. When setting up the configuration files, open scan3d_config.txt and assign L210_config_1280x800.json for all 3 of the distance modes.



D. Real-time 3D Reconstruction

After configuring the SDK application, developers can refer to the API call flow in the image below to implement real-time 3D scanning reconstruction.

The integration of SDK functions and concepts in the API call flow is as follows:

No.	Name		Description
1.	`Camera::InitCamera()`		Camera Initialization:
	Outcome		
	Pass the specified camera configuration file path as a parameter.		
	Return Value	True	The initialization is successful.
		False	The initialization is not successful.
	Note		
	During initialization, the SDK license file is automatically loaded or the trial mode is activated.		

No.	Name	Description
2.	`Camera::GetFrame()`	Get Real-time Camera Frames.
	Outcome	
	Retrieves real-time depth and color image data pointers.	
	Note	
	Depth images store depth values per pixel in 16 bits, with units in millimeters (mm).	

No.	Name	Description
3.	`ConvertDepthTo3DWorld()`	Convert Depth Map to 3D Point Cloud
	Outcome	
	Converts depth image pixel data to three-dimensional point coordinates.	
	Note	
	Units are in centimeters (cm).	

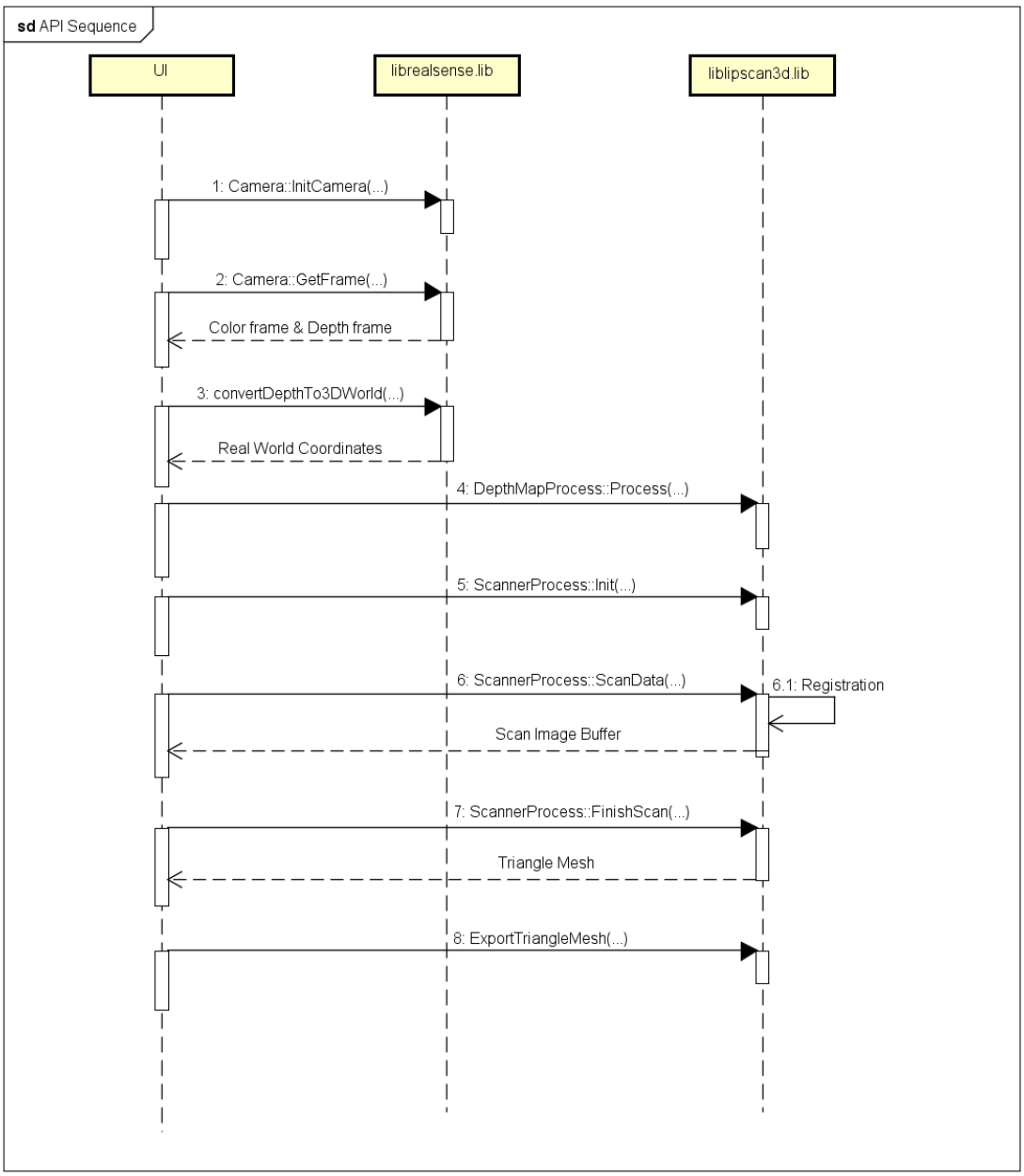
No.	Name	Description
4.	`DepthMapProcess::Process()`	Depth Map Processing
	Outcome	
	Takes depth map and 3D point cloud as parameters.	
	Note	
	For pixels outside the scanning range (e.g., medium mode scanning range = 20-45cm), sets the depth map value to 0 and generates an RGB-mapped depth image for better visualization of depth changes.	

No.	Name	Description
5.	`ScannerProcess::Init()`	Scanner Process Initialization
	Outcome	
	Passes parameters such as resolution width and height (matching the camera), camera model (e.g., D415), scanning mode (e.g., Medium), and camera serial number.	
	Initializes internal memory.	
	Note	
	If the input parameters do not match the camera configuration, unexpected anomalies may occur in the 3D reconstruction model.	

No.	Name	Description
6.	`ScannerProcess::ScanData()`	3D Point Cloud Positioning:
	Outcome	
	Passes color image data obtained from `Camera::GetFrame()` and 3D point cloud data obtained from `ConvertDepthTo3DWorld()`, as well as resolution width and height.	
	Performs 3D point cloud positioning and reconstruction.	
	Stores reconstructed mesh data in internal memory.	
	Note	
	The last parameter stores a 2D preview image of the 3D reconstruction.	

No.	Name	Description
7.	`ScannerProcess::FinishScan()`	Generate Triangle Mesh Model:
	Outcome	
	Converts the 3D reconstruction data from `ScannerProcess::ScanData()` into a triangle mesh model.	
	Return Value	
	Returns the generated model.	

No.	Name	Description
8.	`ExportTriangleMesh()`	Export Mesh Model File:
	Outcome	
	Saves the triangle mesh model data into a specified file format.	

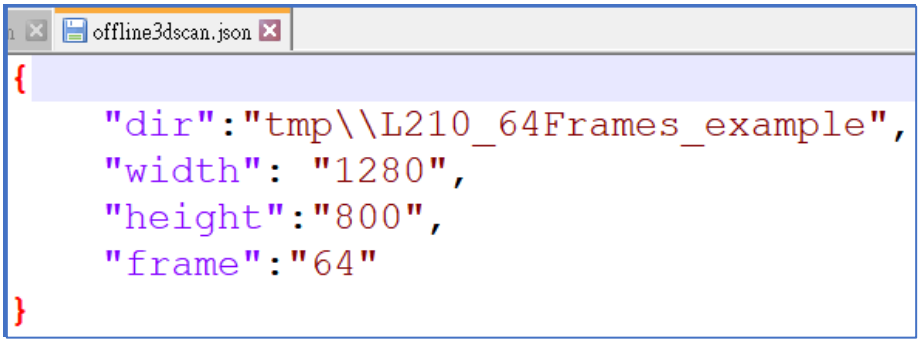

sd API Sequence
UI
librealsense.lib
liblipscan3d.lib
1: Camera::InitCamera(...)
2: Camera::GetFrame(...)
Color frame & Depth frame
3: convertDepthTo3DWorld(...)
Real World Coordinates
4: DepthMapProcess::Process(...)
5: ScannerProcess::Init(...)
6: ScannerProcess::ScanData(...)
6.1: Registration
Scan Image Buffer
7: ScannerProcess::FinishScan(...)
Triangle Mesh
8: ExportTriangleMesh(...)

E. Offline Mode 3D Reconstruction

The LIPScan™ 3D Scan SDK can also be used to develop non-real-time 3D reconstruction applications. The concept is to read depth images, color images, and 3D point cloud data from files and pass the data to `ScannerProcess::ScanData()` for 3D point cloud positioning. Finally, `ScannerProcess::FinishScan()` is used to generate triangle mesh data, and `ExportMeshFile()` is executed to save the triangle mesh model file.

a. Image Data Storage

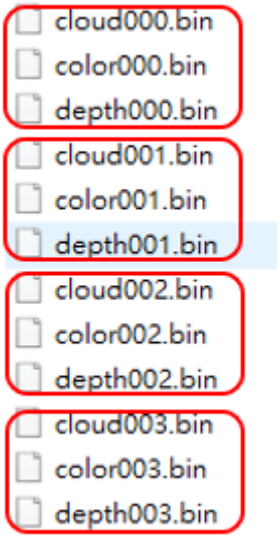
The most efficient way to save the camera's real-time image data is to output it as a binary file (.bin). The configuration file (named offline3dscan.json in this case) records the save path, resolution width and height, and file path, as shown in the image below.



The above image shows the contents of the configuration file for recording non-real-time 3D scans. Here are the explanations for the parameters:

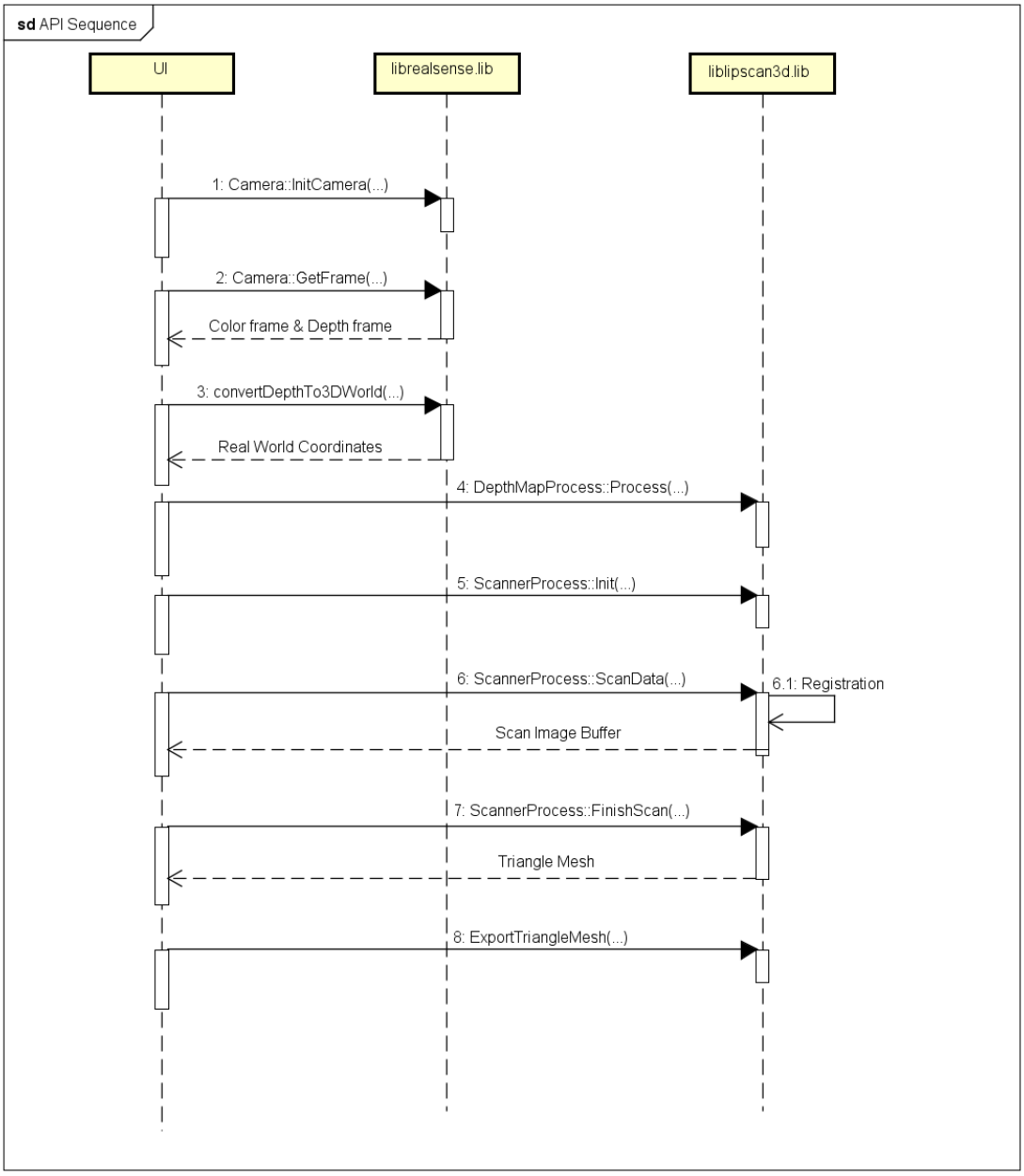
- **dir**: The directory where the SDK application outputs the color image, depth image, and point cloud files.
- **width**: The width of the image file resolution.
- **height**: The height of the image file resolution.
- **frame**: The number of image frames.

The output file formats for the depth image, color image, and point cloud are (.bin). It is recommended to differentiate the data types by naming the files as ColorXX.bin, DepthXX.bin, and CloudXX.bin:



b. Offline 3D reconstruction Execution

The implementation process for offline (non-real-time) 3D reconstruction follows the same flow as the real-time 3D scanning reconstruction described in *5-D. Real-time 3D Reconstruction*. Refer to the figure below.



The API execution flow for offline (non-real-time) 3D reconstruction using the SDK can be explained as follows:

1. Read the configuration file for offline 3D reconstruction data to obtain the source directory, resolution width and height, and frame count parameters.

2. Initialize the 3D reconstruction tool using `ScannerProcess::Init()`, with the resolution width and height obtained from the configuration file.

3. Sequentially pass the read color image, depth image, and point cloud data to the `ScannerProcess::ScanData()` function, with the number of iterations determined by the frame count parameter from the configuration file.

4. Construct the triangle mesh model using `ScannerProcess::FinishScan()` for 3D reconstruction.

5. Execute `ExportMeshFile()` to save the triangle mesh model as a specified file format.

6. By following this API execution flow, you can perform offline 3D reconstruction using the SDK by reading the configuration file, initializing the necessary tools, passing the data, and generating the desired output file.

F. Mesh Comparison Analysis

The LIPScan™ 3D Scan SDK provides an API for comparing a reconstructed mesh model (referred to as the comparison model) with a pre-existing reference mesh model (referred to as the reference model) to calculate accuracy and precision errors between them. The errors in the mesh models can be exported as a statistical file in .csv format.

This functionality module is included in the liblipscan3d library. The Professional Edition software provides an operating interface and a 300-second trial mode. To obtain the Professional Edition license file, contact info@lips-hci.com.

This section explains the implementation process for mesh comparison and analysis, presented in sequential order. The image below is a sequence diagram illustrating the mesh comparison process.

Before performing mesh comparison, the target object model file and reference object model file need to be loaded. During loading, the comparison model and reference model are displayed in the window without being aligned. Model alignment needs to be performed before comparison.

1. Initialize the mesh comparison analysis tool.
2. Load the target object file (.ply) using `MeshComparison::loadCompMesh()`, providing the file path as a parameter. If the file is loaded successfully, it returns true.
3. Load the reference object file (.ply) using `MeshComparison::loadRefMesh()`, providing the file path as a parameter. If the file is loaded successfully, it returns true.
4. Add alignment points for the target object (requires 3 points) using `MeshComparison::addAlignedPoint()`, providing the coordinates of the alignment points. To clear the alignment points, use `MeshComparison::clearAlignedPoints()` to remove all alignment points of the target object.
5. Add alignment points for the reference object (requires 3 points) using `MeshComparison::addReferencePoint()`, providing the coordinates of the alignment points. To clear the alignment points, use `MeshComparison::clearRefPoints()` to remove all alignment points of the reference object.
6. Complete the alignment point settings for the target and reference objects. Perform three-point alignment by calling `MeshComparison::Align()`. The three-point alignment principle involves calculating the rigid transformation matrix between the two objects. If the scale between the target and reference objects is different or the alignment point error is significant, it will result in noticeable errors in the three-point alignment.

7. After completing the three-point alignment for the target and reference objects, perform precise registration by calling `MeshComparison::Register()`. Precise registration requires setting two parameters: iteration convergence threshold and model sampling points. The iteration convergence threshold is recommended to be between 0.01 and 0.00001. A smaller value will result in longer computation time. The model sampling points are recommended to be between 20,000 and 50,000. A larger value will also result in longer computation time.

8. After completing the precise registration for the target and reference objects, compute the mesh distance (error) between the two objects by calling `MeshComparison::ComputeMeshDistance()`.

9. Generate a rendered model that visualizes the distance (error) between the two objects. Convert the mesh distance (error) into color information by calling `MeshComparison::GenCompMeshDistanceColors()`.

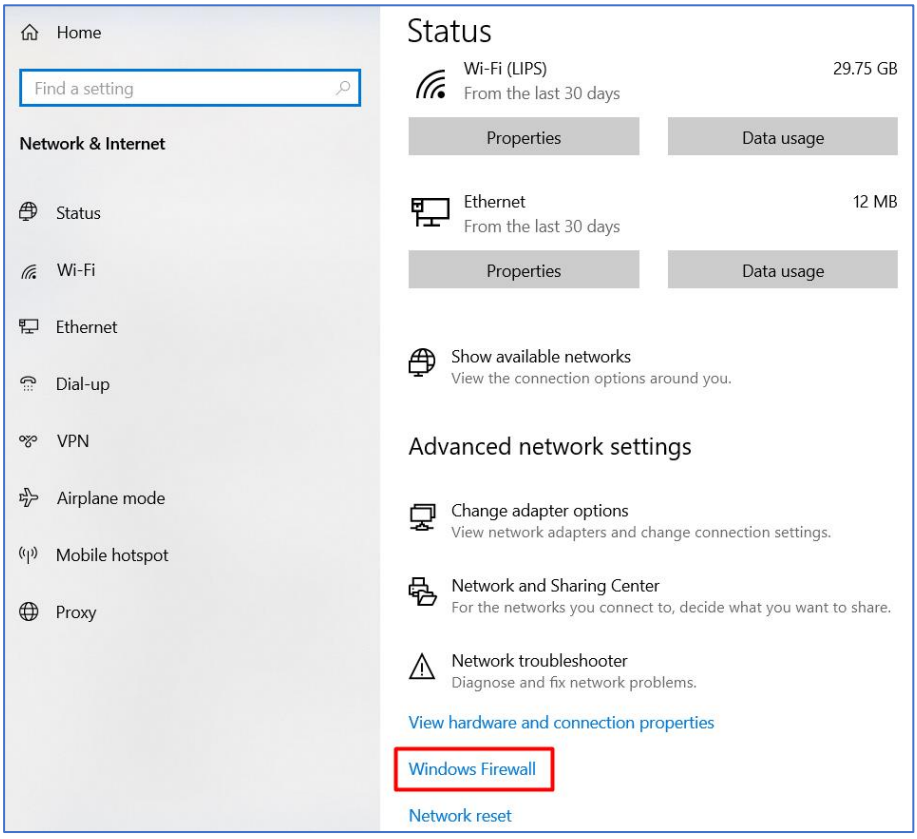
10. Export the mesh distance (error) between the two objects to a CSV file by calling `MeshComparison::ExportToCSV()`. This allows importing the data into Excel for further analysis, such as examining whether the error distribution follows a Gaussian normal distribution.

6. Appendix

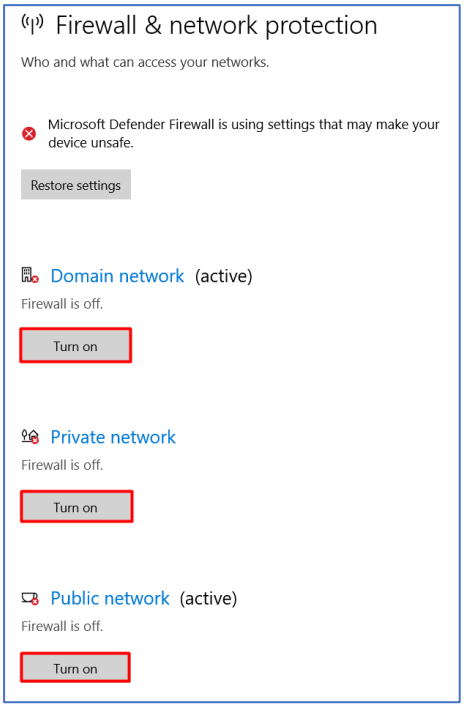
A. Firewall Settings for LIPSedge™ AE series

Windows firewall settings may block LIPSedge™ AE400 / 430 / 450 / AE470 Toolkit from detecting available cameras in the same subnet. In that case, adjust the Windows firewall settings to allow LIPSedge™ AE400 /430 / 450 / AE470 Toolkit through the firewall.

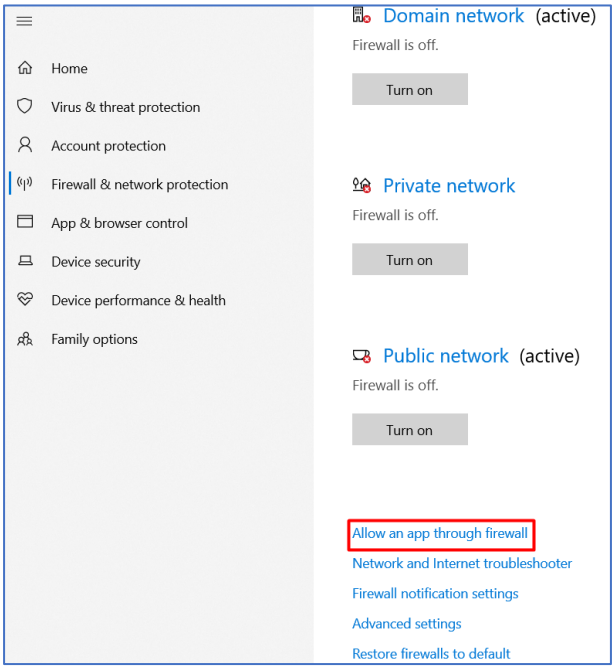
1. Click  at **Windows toolbar** and select **Network & Internet > Windows Firewall**.



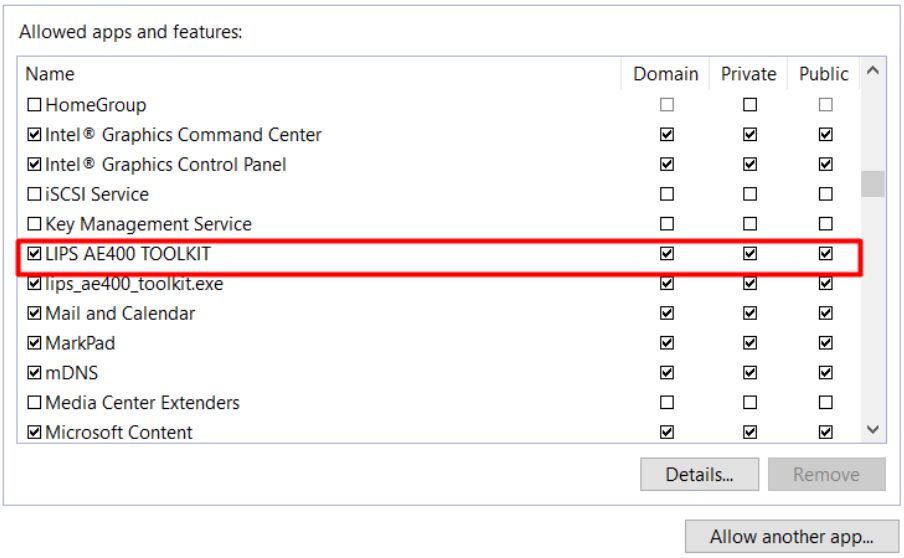
2. Under **Firewall & network protection**, disable the firewall settings of **Domain Network**, **Private Network**, and **Public Network**.



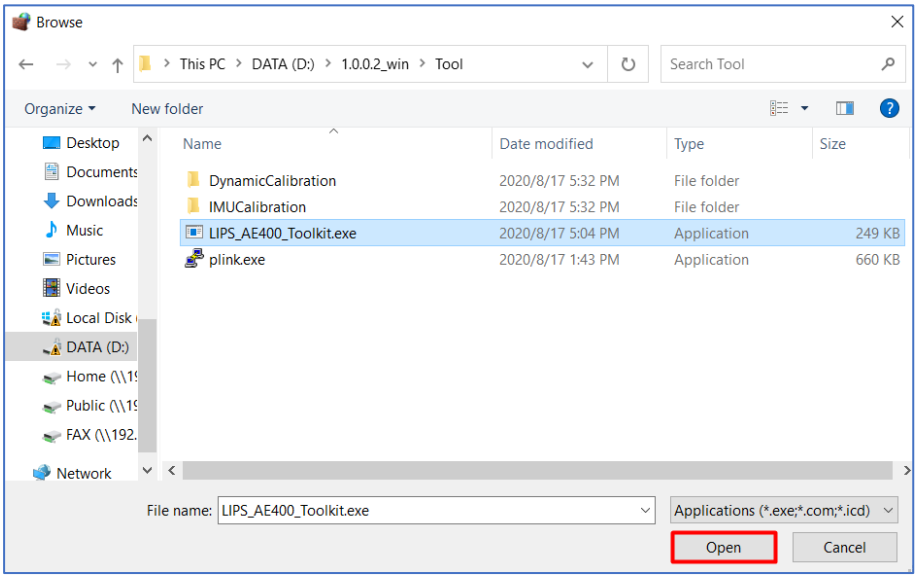
3. If for any reason the firewall settings couldn't be disabled, scroll to the bottom of **Firewall & network protection**, and click **Allow an app through firewall**.



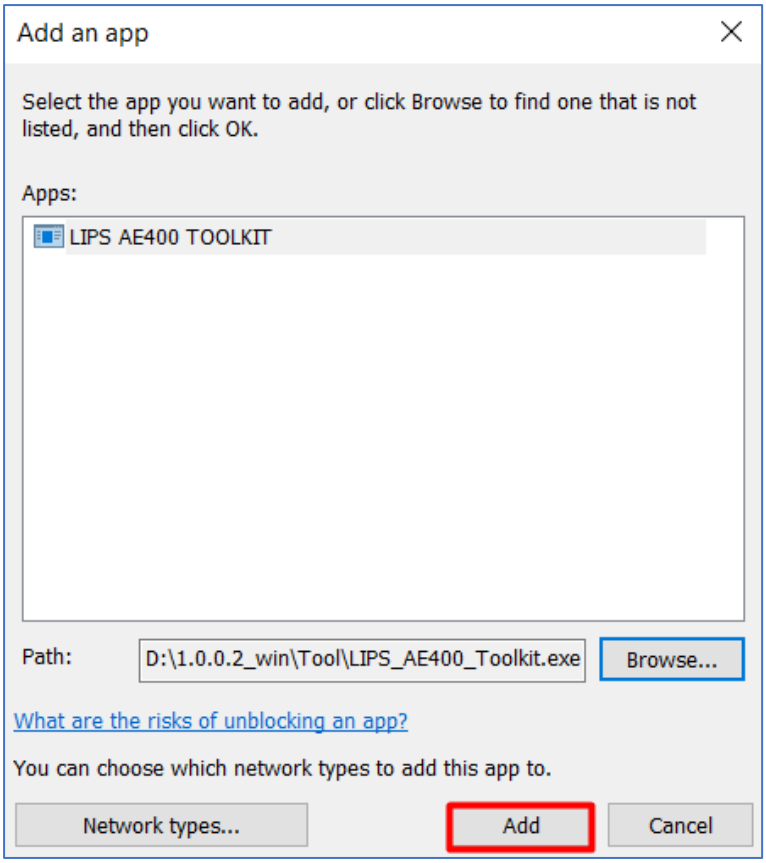
4. From the **Allowed apps and features** list, select **LIPS AE400 TOOLKIT**.



5. If the **LIPS AE400 TOOLKIT** option is not found in the list, click **Allow another app**, and find the application.



6. Select **LIPS AE400 TOOLKIT** and click **Add**.



B. Regulatory Compliance Notice

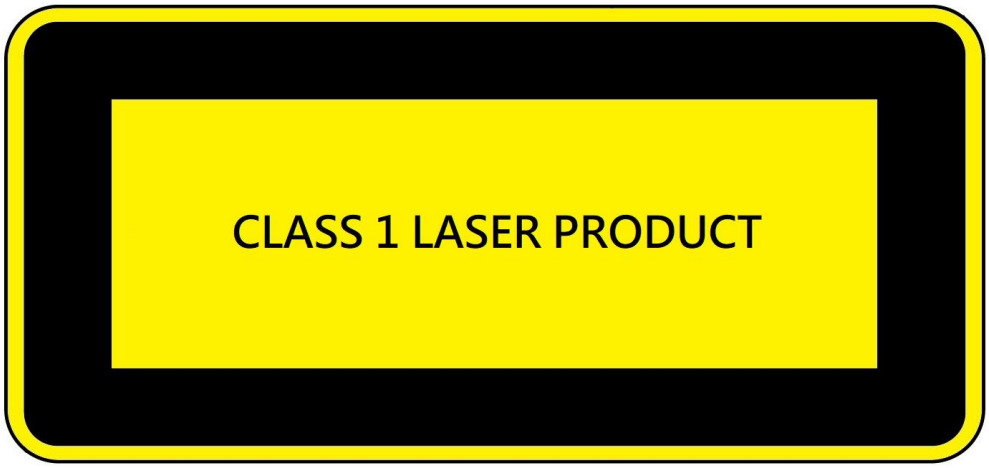
FCC Compliance

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Caution: Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Label Notice



This device complies with part 15 of the FCC Rules. Operation is subject to the following conditions:

(1) This device may not cause harmful interference, and
(2) this device must accept any interference received, including interference that may cause undesired operation.

CE Compliance

This is a Class B product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

RoHS Compliance

All lead-free products offered by the company comply with the requirements of the European law on the Restriction of Hazardous Substances (RoHS) directive, which means our manufacture processes and products are strictly “lead-free” and without the hazardous substances cited in the directive.

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