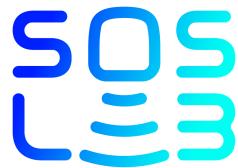




# ML LiDAR User Guide

Version 1.1.0



May 2, 2022

## Contents

<b>1 ML Hardware Configuration</b>	<b>3</b>
1.1 Mechanical Interface . . . . .	3
1.1.1 Included Components . . . . .	3
1.1.2 Exterior Mechanical Dimensions . . . . .	4
1.2 Electrical Interface . . . . .	6
1.2.1 Cable Connection . . . . .	6
1.2.2 IP/Port Information . . . . .	7
1.2.3 Power Adapter Information . . . . .	7
<b>2 ML Software - ML Manager</b>	<b>8</b>
2.1 ML Manager Setup . . . . .	8
2.1.1 Installation . . . . .	8
2.1.2 TCP/IP Setting . . . . .	9
2.2 ML Manager Tools . . . . .	11
2.2.1 Connection . . . . .	12
2.2.2 Data Load . . . . .	15
2.2.3 Device Setting . . . . .	16
2.2.3.1 Functions . . . . .	16
2.2.3.2 Update . . . . .	17
2.2.4 Point Cloud Setting . . . . .	20
2.2.5 Image Setting . . . . .	21
2.2.6 Grid . . . . .	22
2.2.7 X-Y / Y-Z / Z-X Axis . . . . .	23
2.3 ML Manager Viewer . . . . .	25
2.3.1 Point Cloud Viewer . . . . .	25
2.3.2 Image Viewer . . . . .	27
2.4 ML Manager Play Bar . . . . .	28
2.4.1 Play Controls . . . . .	28
2.4.2 Record . . . . .	29
2.4.3 Capture . . . . .	29
2.5 ML Log . . . . .	30
<b>3 Window API</b>	<b>31</b>
3.1 ML Window API . . . . .	31
3.1.1 API Build . . . . .	32
3.1.2 Example Code . . . . .	36
3.1.2.1 Connection . . . . .	36
3.1.2.2 Functions . . . . .	37
3.1.2.3 Data Streaming . . . . .	38
3.1.2.4 Record . . . . .	41

3.1.2.5	Data Conversion . . . . .	42
3.1.2.6	Multiple LiDAR Setup . . . . .	43
<b>4</b>	<b>Ubuntu API</b>	<b>46</b>
4.1	ML Ubuntu API . . . . .	46
4.1.1	API Build . . . . .	47
4.1.2	Example Code . . . . .	56
4.1.2.1	Connection . . . . .	56
4.1.2.2	Functions . . . . .	57
4.1.2.3	Data Streaming . . . . .	58
4.1.2.4	Record . . . . .	61
4.1.2.5	Data Conversion . . . . .	62
4.1.2.6	Multiple LiDAR Setup . . . . .	63
4.1.2.7	ROS . . . . .	66
<b>5</b>	<b>Appendix</b>	<b>71</b>
5.1	TCP Protocol Packet Structure . . . . .	71
5.2	UDP Protocol Packet Structure . . . . .	72
5.2.1	Single Echo Mode Packet Structure . . . . .	72
5.2.2	Multi Echo Mode Packet Structure . . . . .	73
5.3	API - Typedefs . . . . .	74
5.4	API - Structure / Class . . . . .	75

# 1 ML Hardware Configuration

## 1.1 Mechanical Interface

### 1.1.1 Included Components

ML은 다음과 같은 아이템들을 포함하여 제공합니다.

- ML-0 or ML-2 LiDAR sensor
- Sensor to Power cable/connector (4 meters)
- Sensor to Ethernet cable/connector (4 meters)
- Sensor AC/DC Power adapter/cable



(a) Power Cable/Connector



(b) Ethernet Cable/Connector



(c) AC/DC Power Adapter



(d) AC/DC Adapter Cable

Figure 1: ML LiDAR Components

### 1.1.2 Exterior Mechanical Dimensions

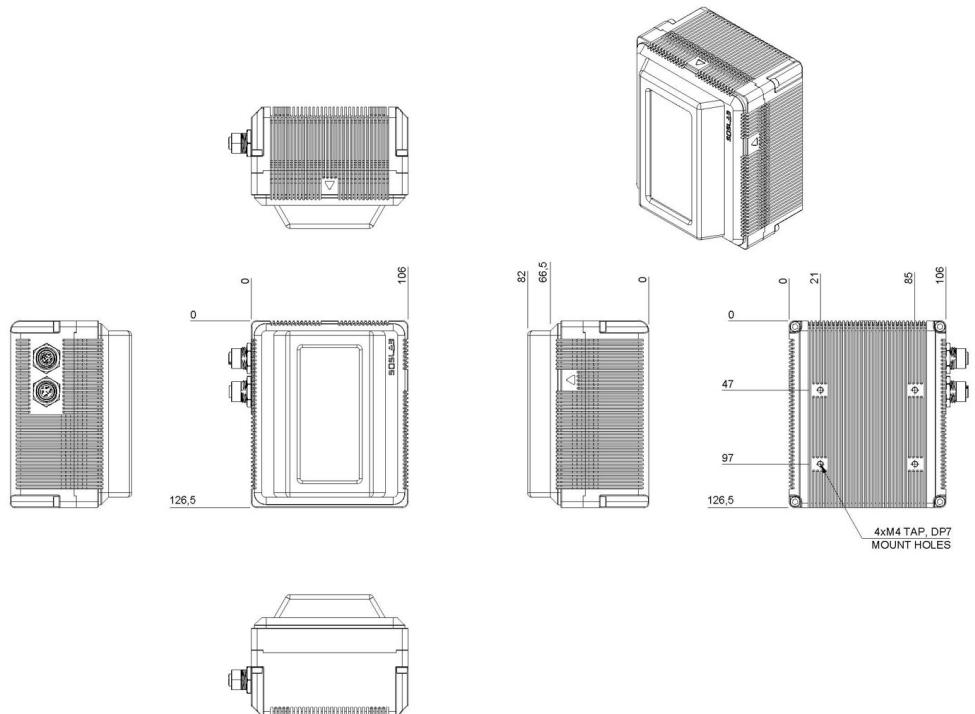


Figure 2: The sensor has  $4 \times M4$  mounting holes

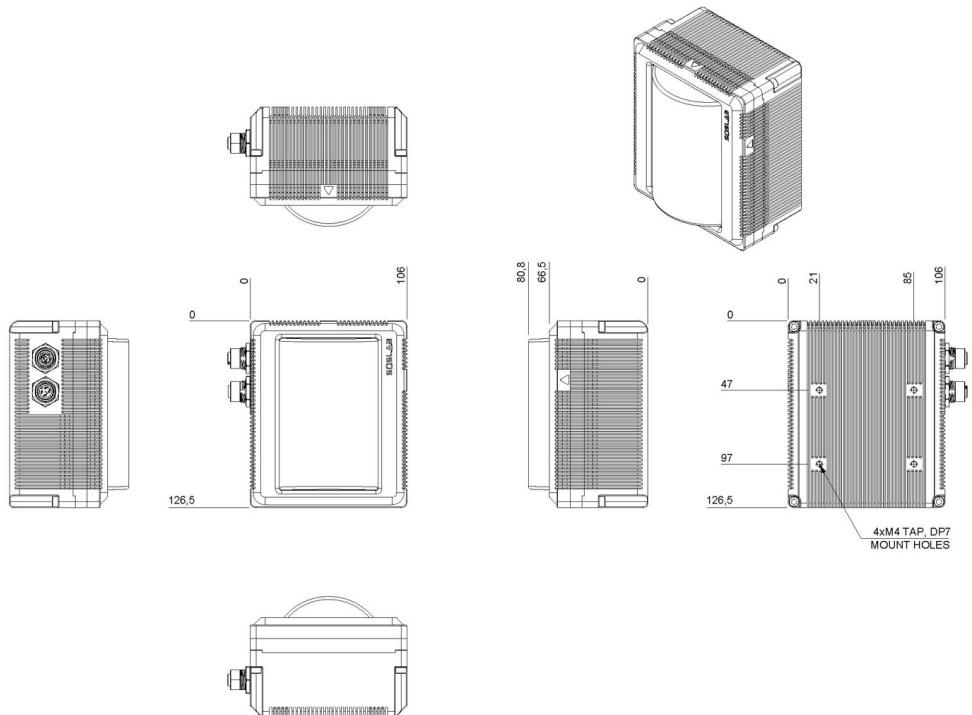


Figure 3: The sensor has  $4 \times M4$  mounting holes

## 1.2 Electrical Interface

### 1.2.1 Cable Connection

케이블을 연결하는 순서는 다음과 같습니다.

1. 제공된 센서와 이더넷 케이블을 연결합니다.
2. 제공된 센서와 파워 케이블을 연결합니다.
3. 이더넷 케이블과 PC를 연결합니다.
4. 파워 케이블과 파워 어댑터를 연결합니다.



Figure 4: ML-0/ML-2 Connector



Figure 5: ML-0/ML-2 Cable Connection

### **1.2.2 IP/Port Information**

제공된 센서의 기본 IP/Port 정보는 다음과 같습니다. Device Setup을 이용해 IP / Port 정보를 변경할 수 있습니다.

- Local IP : 192.168.1.15
- Local Port : 2000
- Device IP : 192.168.1.10
- Device Port : 2000

### **1.2.3 Power Adapter Information**

제공된 파워 어댑터의 정보는 다음과 같습니다. 아래 규격을 만족하지 않는 전원의 사용으로 인해 발생한 문제에 대해서는 제품의 신뢰성을 보증하지 않습니다.

- 정격입력 : 100-240V 50/60Hz 1.7A
- 정격출력: +12.0V 5.0A 60.0W

## 2 ML Software - ML Manager

### 2.1 ML Manager Setup

ML Manager는 장치 설정 및 데이터 시각화를 위한 소프트웨어입니다.

#### 2.1.1 Installation

ML Manager는 'ML Manager\_setup.exe' 설치 파일을 이용해 설치할 수 있습니다.  
설치 순서는 다음과 같습니다.

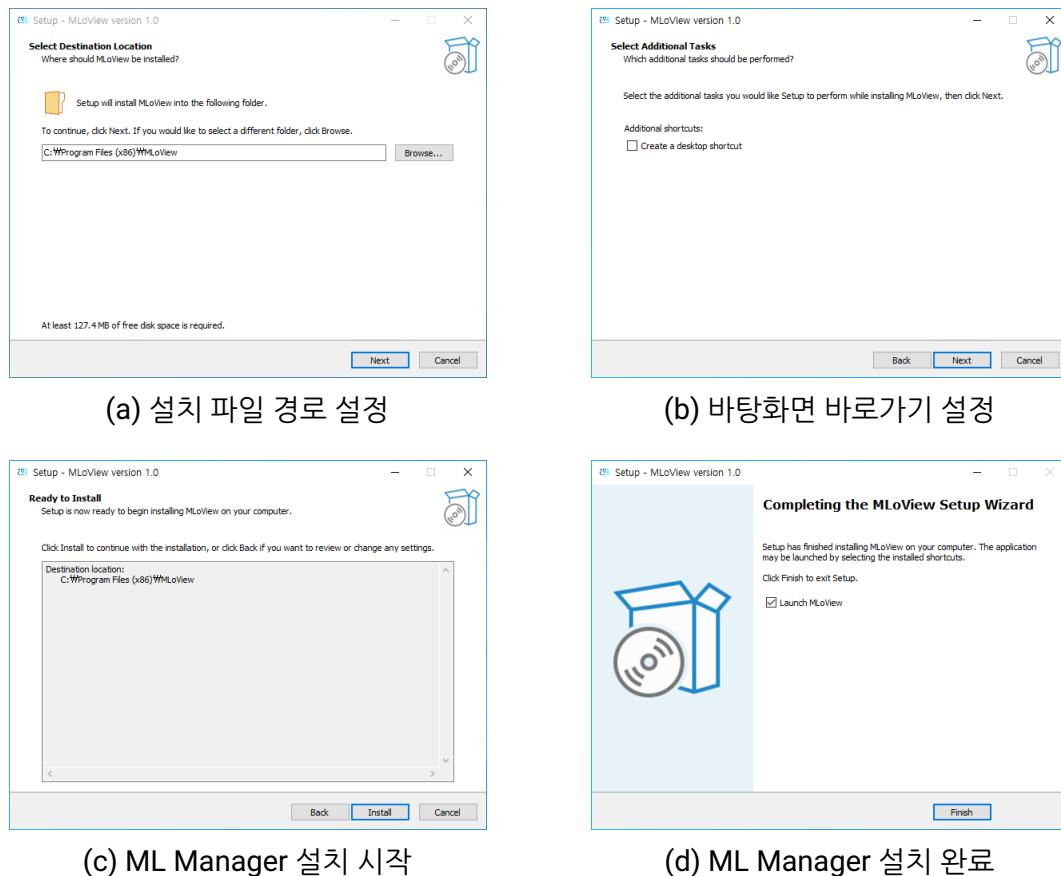


Figure 6: ML Manager Installation

위 과정을 통해 설치가 완료되면 아래 그림과 같이 ML Manager가 실행됩니다.

## 2.1.2 TCP/IP Setting

ML LiDAR와 PC의 통신을 위해 TCP/IP 설정을 진행합니다.

1. ML 전원 케이블을 연결하고, 네트워크 케이블로 ML LiDAR와 PC를 연결합니다.
2. 제어판 > 네트워크 및 인터넷 > 네트워크 및 공유 센터를 실행합니다.
3. 활성화 된 이더넷을 클릭 후 속성 창을 엽니다.
4. 인터넷 프로토콜 버전 4(TCP/IPv4) 선택 후 속성 버튼을 클릭합니다.
5. 속성 창에서 “다음 IP 주소 사용” 옵션을 선택합니다.
6. IP 주소(192.168.1.15)와 서브넷 마스크(255.255.255.0)를 아래 그림과 같이 설정 한 뒤, 확인 버튼을 클릭합니다.

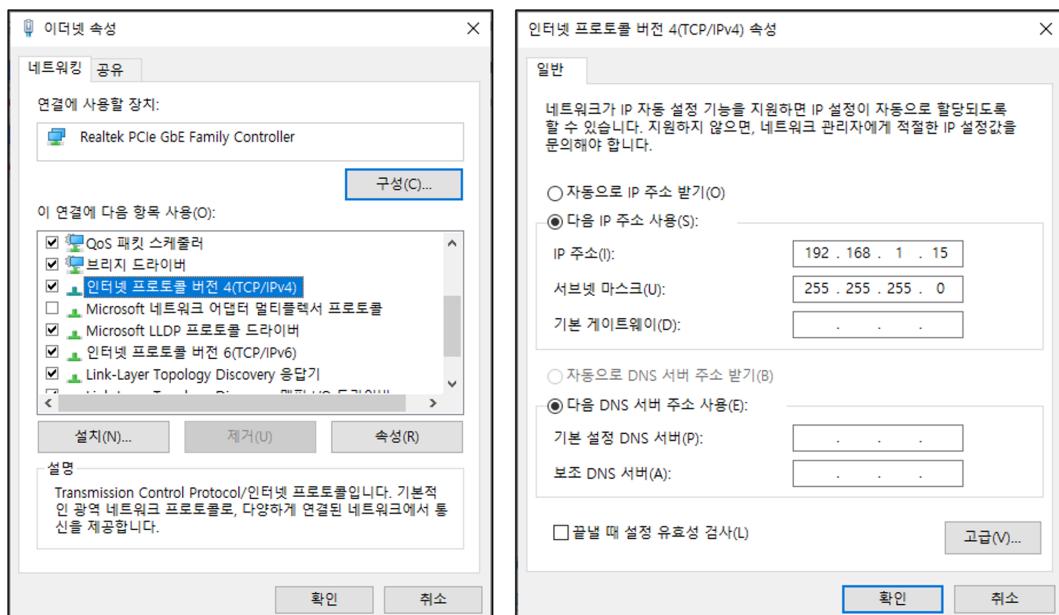
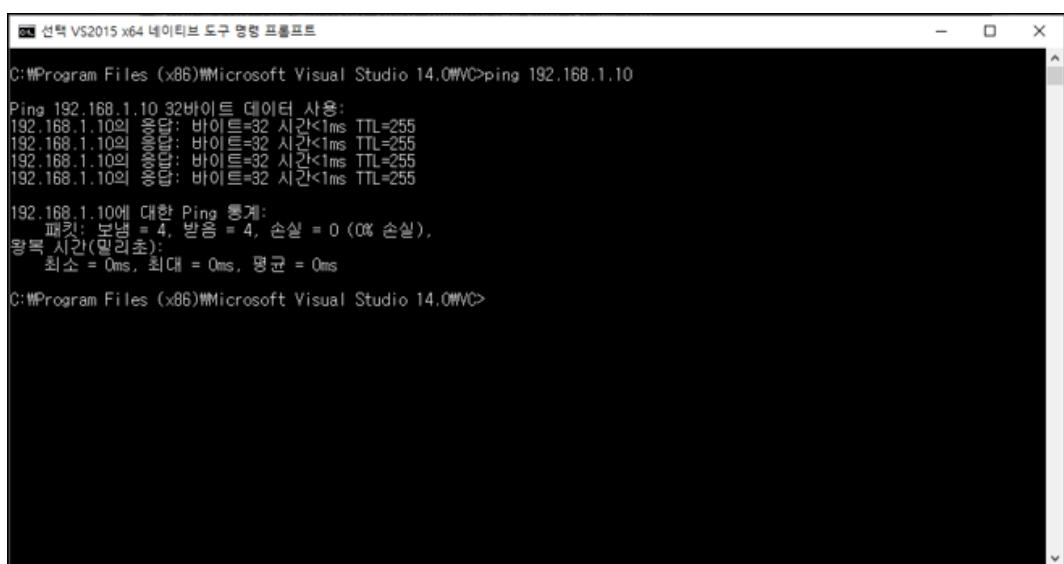


Figure 7: 인터넷 프로토콜 버전4(TCP/IPv4) 속성 창 설정

케이블 연결과 IP 설정이 완료되면 아래의 Ping 테스트 과정을 통해 ML LiDAR와 PC가 정상적으로 연결되었는지 확인이 가능합니다.

1. 윈도우 검색창에 'cmd' 명령어를 입력하여 명령 프롬프트를 실행합니다.
2. 명령어에 'ping 192.168.1.10'을 입력합니다.
3. ML LiDAR와 PC가 정상적으로 연결되었다면 아래와 같은 메시지가 출력됩니다.



```
선택 VS2015 x64 네이티브 도구 명령 프롬프트
C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC>ping 192.168.1.10

Ping 192.168.1.10 32바이트 데이터 사용:
192.168.1.10의 응답: 바이트=32 시간<1ms TTL=255

192.168.1.10에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms

C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC>
```

Figure 8: 명령 프롬프트 창을 이용한 ML LiDAR Ping 테스트

Ping 테스트 결과, 위 그림과 같이 응답이 없다면 케이블 연결과 이더넷 활성화 여부를 점검하여 다시 테스트를 진행합니다. 지속적으로 문제가 발생할 경우에는 고객 대응 팀([sales@soslab.co](mailto:sales@soslab.co))으로 문의 주시기 바랍니다.

## 2.2 ML Manager Tools

ML Manager의 왼쪽에 위치한 Tools에서는 ML LiDAR와 PC의 연결, 녹화된 ML LiDAR 데이터 불러오기, ML LiDAR 세팅, Point Cloud 및 Image Viewer 세팅, Grid 세팅, X-Y / Y-Z / Z-X Axis 카메라 이동의 기능들을 제공합니다.

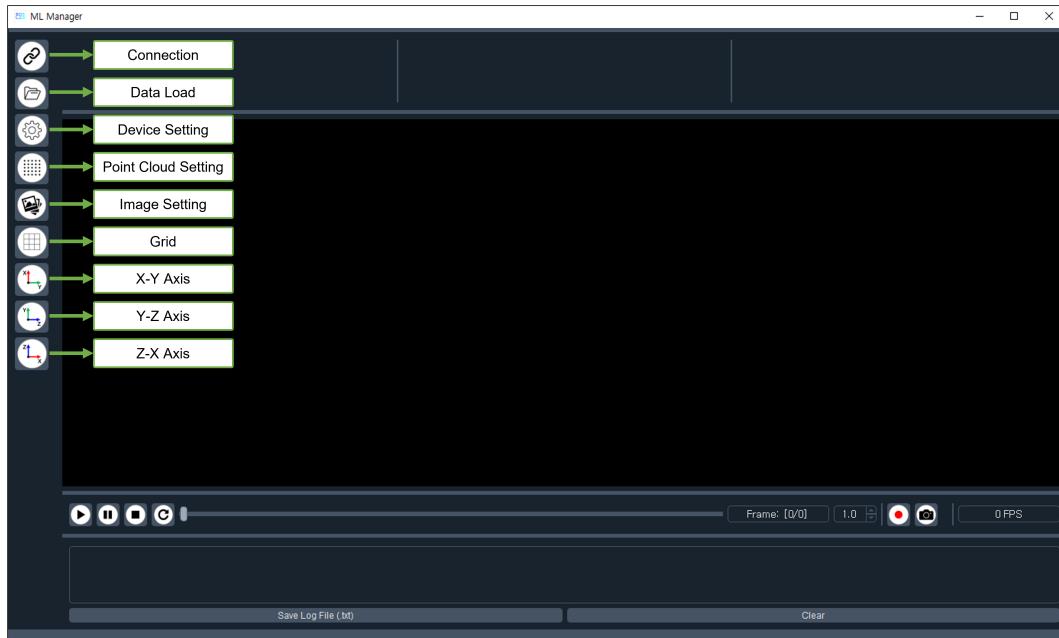


Figure 9: ML Manager Tools

### 2.2.1 Connection

Tools의 첫번째 메뉴 ☰ **Connection**에서는 PC (Local)와 Device (ML)의 IP 및 Port를 입력하여 장치를 연결 할 수 있습니다. ML LiDAR에서 기본으로 제공되는 설정 값은 다음과 같습니다

- **PC(Local)** - IP: 0.0.0.0, Port: 0
- **Device(ML)** - IP: 192.168.1.10, Port: 2000

Connection 아이콘을 클릭하면 아래의 그림과 같이 'ML Connection' 팝업창이 활성화됩니다. Connection 팝업창에서 IP 및 Port 값을 입력한 뒤, 'Connect' 버튼을 클릭하면 ML LiDAR와 PC가 연결됩니다.



Figure 10: Connection 실행화면

ML Manager에서 ML LiDAR와 PC를 처음 연결하면, 아래의 Windows 보안 경고창이 나타납니다. 경고창에서 아래 그림과 같이 통신 허용 관련 체크박스를 모두 체크 한 뒤, 아래의 ‘액세스 허용’ 버튼을 클릭하면 프로그램이 정상적으로 동작합니다.

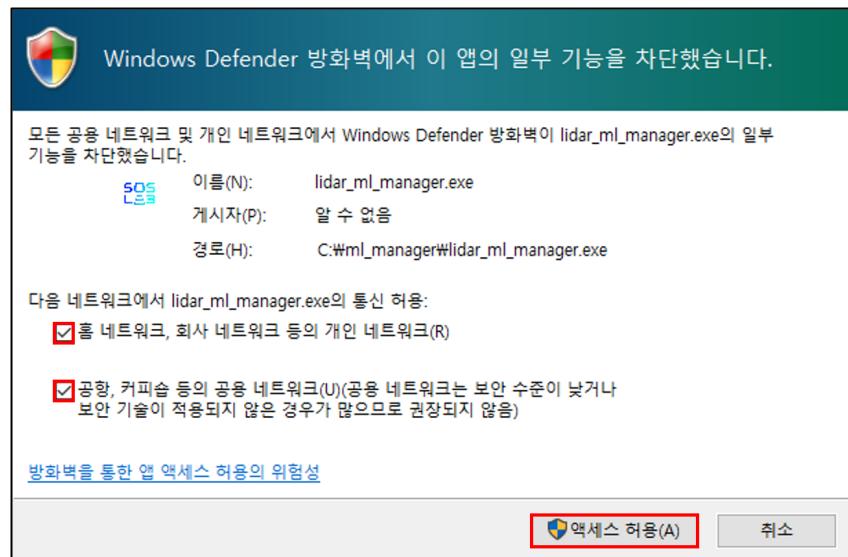


Figure 11: Windows 보안 경고 창

ML LiDAR와 PC가 연결이 완료되면 실시간으로 ML LiDAR 데이터가 획득되는 Stream Mode가 실행되며 아래 그림과 같이 ML Manager의 Point Cloud Viewer 및 Image Viewer에서 데이터가 가시화됩니다.

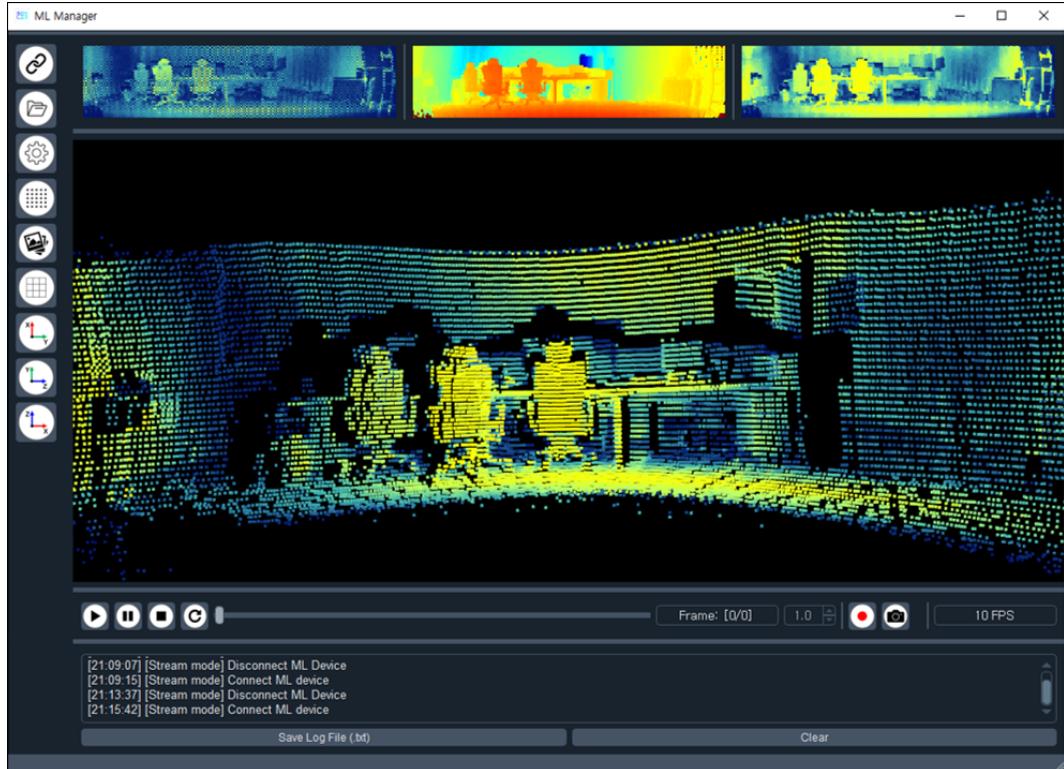


Figure 12: ML Manager 데이터 가시화

## 2.2.2 Data Load

Tools의 두번째 메뉴  **Data Load**에서는 녹화된 ML LiDAR 데이터 (\*.bin)를 불러올 수 있습니다. Data Load 아이콘을 클릭하면 아래의 그림과 같이 'Load ML Data' 창이 활성화됩니다. 활성화 된 창에서 녹화된 ML LiDAR 데이터 (\*.bin) 파일을 선택한 뒤, '열기' 버튼을 클릭하면 녹화된 파일을 불러올 수 있습니다.

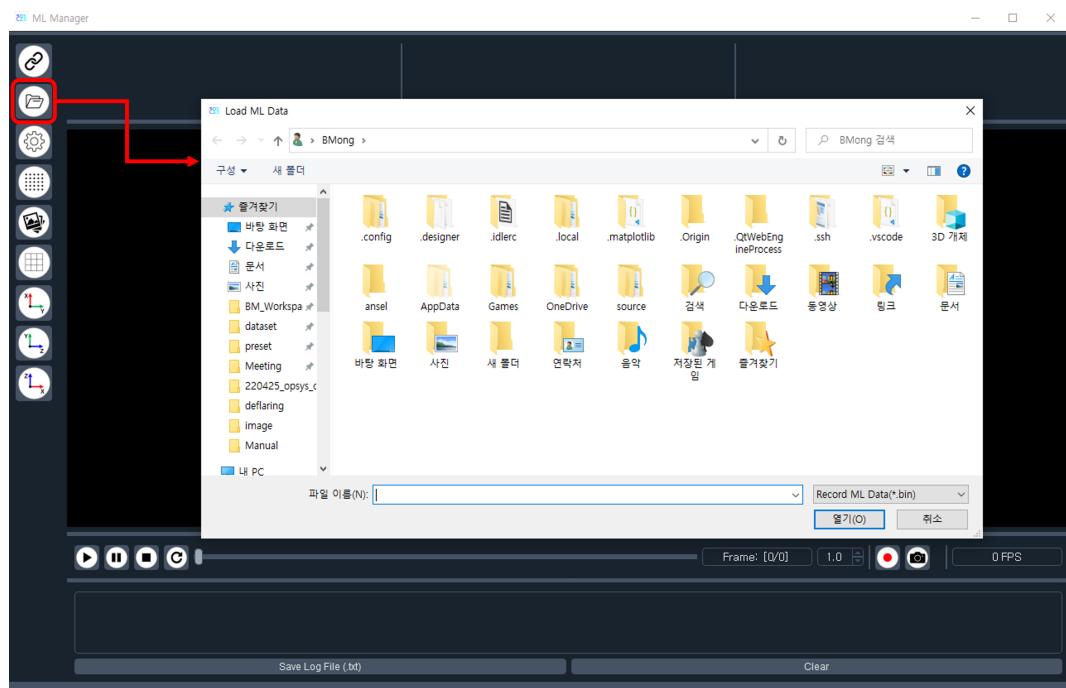


Figure 13: Data Load 실행화면

불러온 ML LiDAR 데이터는 Play Mode로 데이터가 재생되며 **2.4.1 Play Controls**에 설명된 기능들을 통해 재생과 관련된 세팅들을 변경할 수 있습니다.

### 2.2.3 Device Setting

Tools의 세번째 메뉴 **Device Setting**은 ML LiDAR의 Data Processing 세팅, ML LiDAR의 IP/Port 변경, 펌웨어 업데이트, ML LiDAR 재부팅 기능을 제공합니다.

#### 2.2.3.1 Functions

**Functions**에서 제공하는 Data Processing 항목은 다음과 같습니다

- **Line Filling**: ML-2 근거리에서 발생하는 Line 데이터 손실을 보정합니다.
- **Zigzag Filling**: ML-0에서 발생하는 Zigzag 데이터 손실을 보정합니다.
- **Deflaring**: Retro 객체 주변에서 발생하는 Flaring 노이즈를 제거합니다.
- **Real Ambient**: ML 신호가 제거된 Ambient 이미지를 획득합니다.
- **Sync Mode**: 다중 센서 연결을 위한 동기화 기능을 설정합니다.
- **False Positive Rate**: LiDAR의 False Positive Rate을 설정합니다.

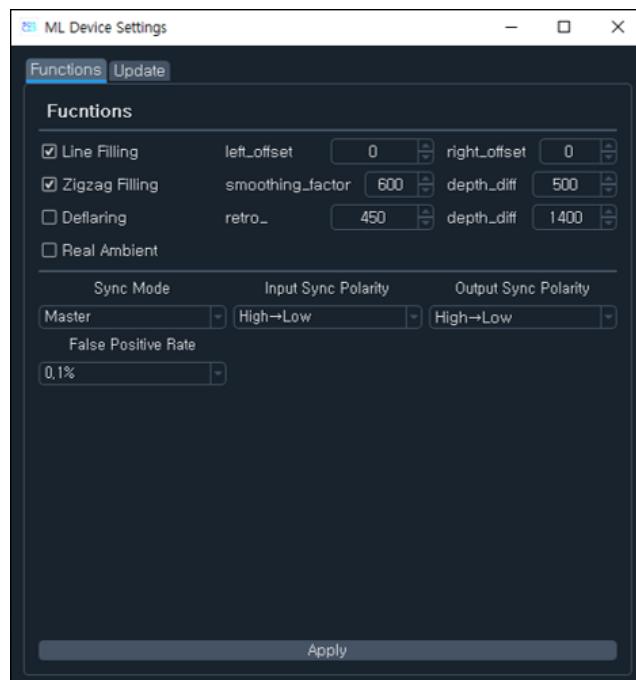


Figure 14: Device Setting - Functions 실행화면

### 2.2.3.2 Update

**Update**는 Message Log, Connection Info, Device Update로 구성됩니다.

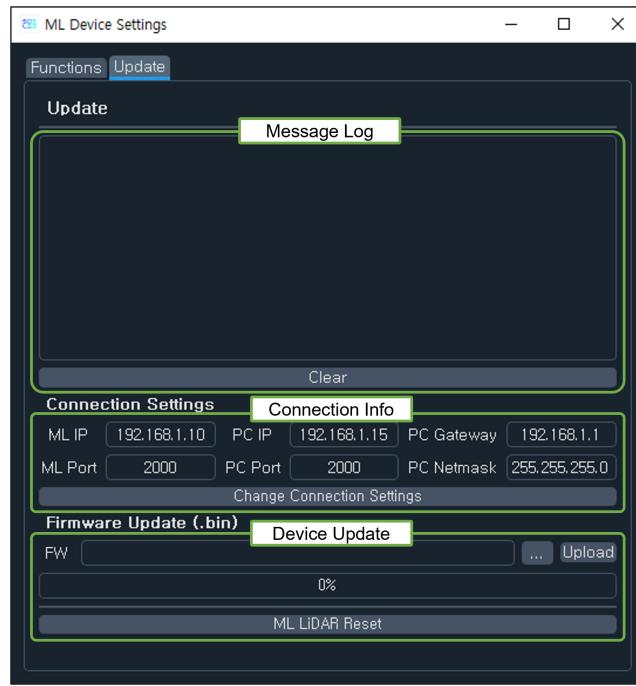


Figure 15: Device setting - Update 화면

**Message Log**에서는 ML LiDAR의 현재 상태를 출력합니다.

- **0::DO:::** ML LiDAR와 PC 연결
- **0::RO:::** ML LiDAR RAM 정보 업데이트
- **0::MO:::** ML LiDAR Flash 정보 업데이트

**Connection Info**에서는 ML LiDAR 및 PC의 IP/Port를 변경할 수 있습니다. 변경된 IP/Port 내역은 "C:/사용자/사용자이름/ML\_connect\_setting.txt" 경로에 저장됩니다.

1. 변경하고자하는 IP/Port 값을 입력합니다.
2. 'Change Connection Settings' 버튼을 클릭합니다.
3. 완료 창이 팝업되면 자동으로 ML LiDAR와의 연결이 해제됩니다.

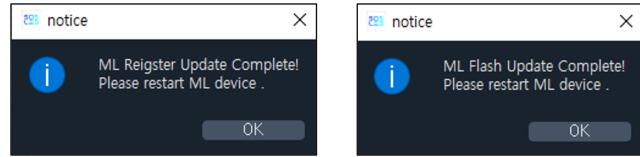


Figure 16: IP/Port 변경 완료 창

4. ML LiDAR의 전원 케이블을 해제하고 재연결합니다.

5. 변경한 IP/Port로 Connection 기능을 통해 재연결합니다.

**Device Update**에서는 ML LiDAR의 펌웨어 업데이트/재부팅 기능을 제공합니다.  
ML LiDAR의 펌웨어 업데이트 방법은 다음과 같습니다.

1. '...' 버튼을 클릭하여 ML LiDAR 펌웨어 파일(\*.bin)을 선택합니다.

2. 'Upload' 버튼을 클릭하여 펌웨어 파일을 확인 한 뒤, 'OK' 버튼을 클릭하여 펌웨어 업데이트를 진행합니다.

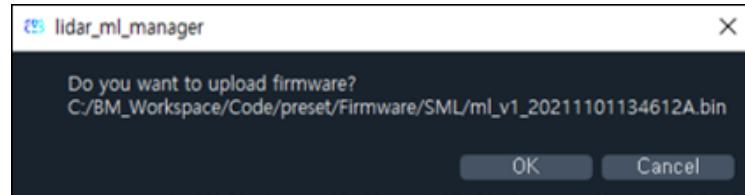


Figure 17: Firmware 파일 경로 확인 창

3. 완료 창이 팝업되면 자동으로 ML LiDAR와의 연결이 해제됩니다.

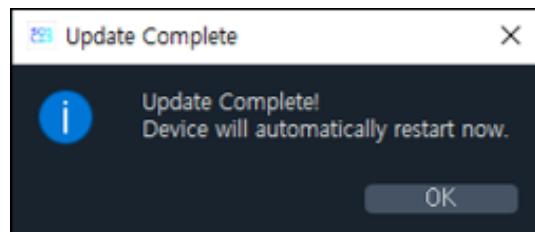


Figure 18: Firmware Update 완료 창

4. ML LiDAR의 전원 케이블을 해제하고 재연결합니다.

5. Connection 기능을 통해 재연결합니다.

ML LiDAR의 재부팅 방법은 다음과 같습니다.

1. 'ML LiDAR Reset' 버튼을 클릭합니다.
2. 완료 창이 팝업되면 자동으로 ML LiDAR와의 연결이 해제됩니다.

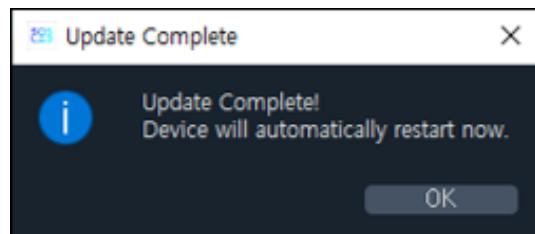


Figure 19: ML LiDAR Reset 완료 창

3. Figure 8의 ML LiDAR Ping 테스트를 통해 ML LiDAR와 PC가 정상적으로 연결이 확인되면 ML Manager의 Connection을 통해 ML LiDAR와 PC를 재연결합니다.

#### 2.2.4 Point Cloud Setting

Tools 네번째 메뉴  **Point Cloud Setting**에서는 ML Manager의 Point Cloud Viewer에 가시화 되는 Point의 가시화 속성을 변경할 수 있습니다.

- **Point Color:** Point 색상을 선택합니다. (White, Red, Green, Blue, Ambient, Depth, Intensity)
- **Point Size:** Point 크기를 설정합니다. (1 ~ 10)
- **Min Distance:** 가시화 되는 Point의 최소거리를 설정합니다. (0 ~ 200)
- **Max Distance:** 가시화 되는 Point의 최대거리를 설정합니다. (1 ~ 500)

속성을 변경한 뒤, 아래의 'Apply' 버튼을 클릭하면 Point Cloud Viewer에 속성 값이 반영됩니다.

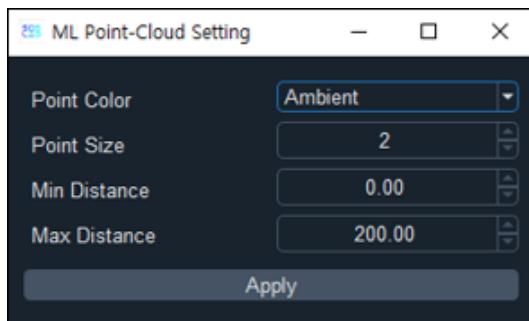


Figure 20: Point cloud setting 화면

### 2.2.5 Image Setting

Tools 다섯번째 메뉴  **Image Setting**에서는 ML Manager의 Image Viewer에 가시화 되는 Image의 가시화 속성을 변경할 수 있습니다.

- **Image Type:** 가시화 속성을 변경할 데이터를 선택합니다. (Ambient, Depth, Intensity)
- **Auto Normalization:** 입력되는 데이터의 최대/최소 값으로 이미지 정규화를 수행합니다.
- **Min Value:** 이미지 정규화의 최소 값을 설정합니다. (0 ~ 1000000)
- **Max Value:** 이미지 정규화의 최대 값을 설정합니다. (0 ~ 1000000)
- **Histogram Equalization:** 이미지 Histogram Equalization 기능을 수행합니다.
- **Grey:** 이미지를 흑백처리합니다.

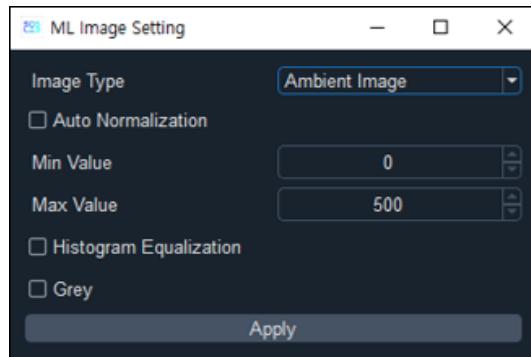


Figure 21: Image setting 화면

### 2.2.6 Grid

Tools 여섯번째 메뉴  **Grid**를 통해 ML Manager의 Point Cloud Viewer에 Grid 형태의 평면 좌표계를 가시화할 수 있습니다. Grid는 5m 간격으로 0m ~ 50m 까지 표현됩니다.

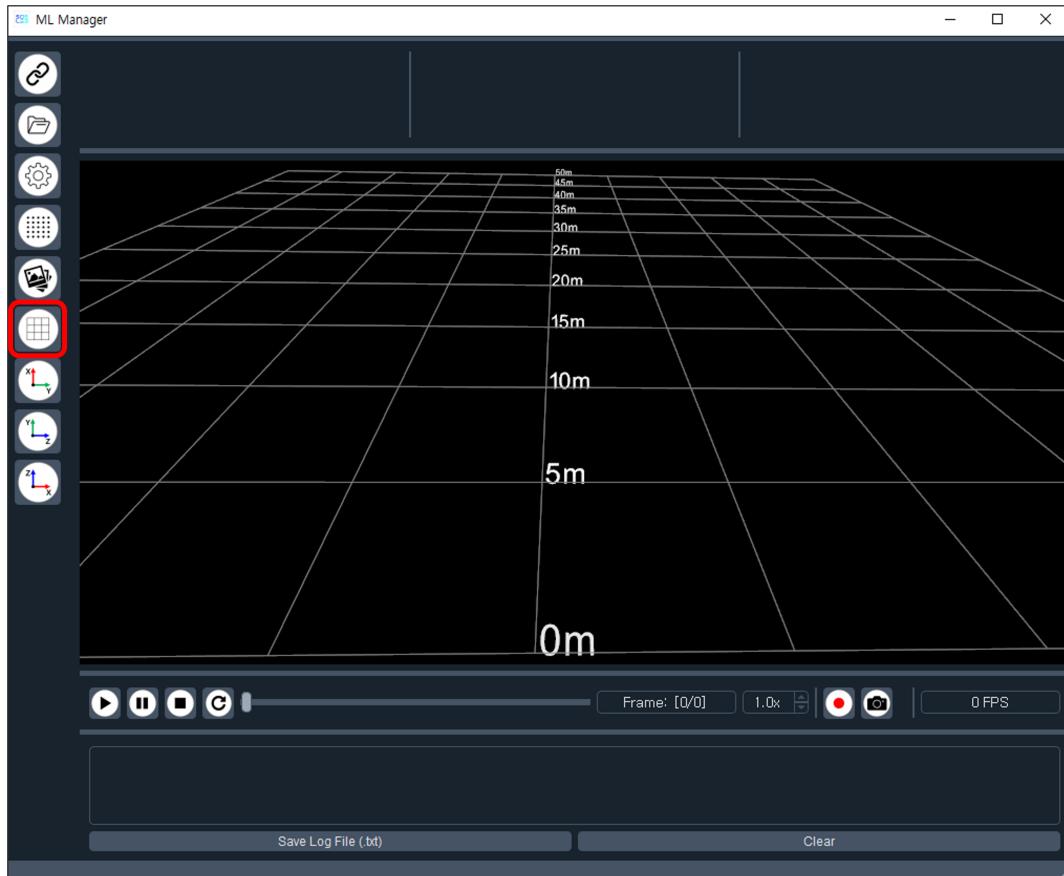


Figure 22: Grid 활성화 화면

### 2.2.7 X-Y / Y-Z / Z-X Axis

Tools 일곱번째부터 아홉번째의  **X-Y Axis** /  **Y-Z Axis** /  **Z-X Axis**는 ML Manager의 Point Cloud Viewer의 카메라 시점을 각각 X-Y / Y-Z / Z-X 평면으로 변경해주는 기능을 제공합니다.

- **X-Y Axis**: ML을 기준으로 Point Cloud 데이터를 위에서 바라보는 Top View를 의미합니다.
- **Y-Z Axis**: ML을 기준으로 Point Cloud 데이터를 정면에서 바라보는 Front View를 의미합니다.
- **Z-X Axis**: ML을 기준으로 Point Cloud 데이터를 측면에서 바라보는 Side View를 의미합니다.

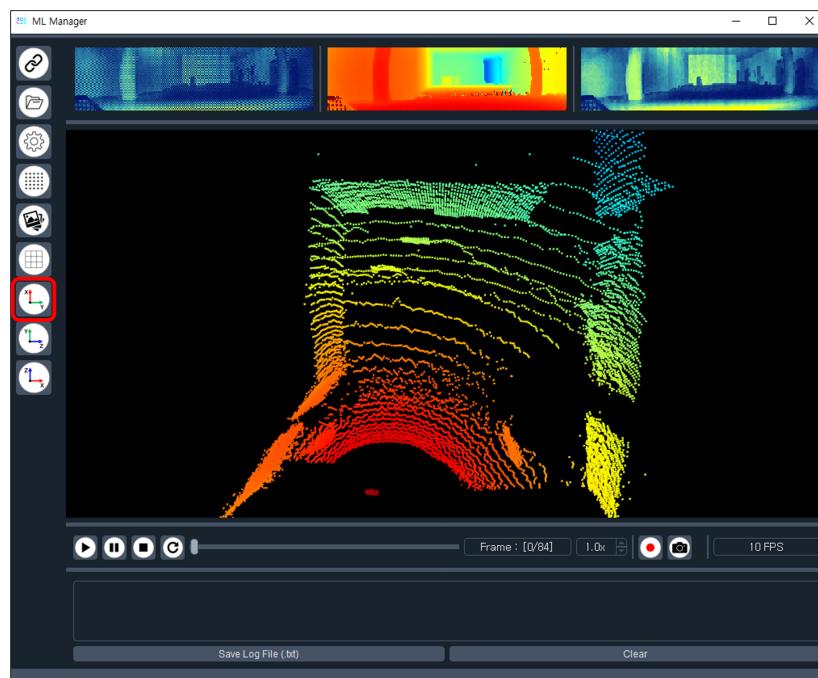


Figure 23: X-Y Axis 카메라 시점 변경

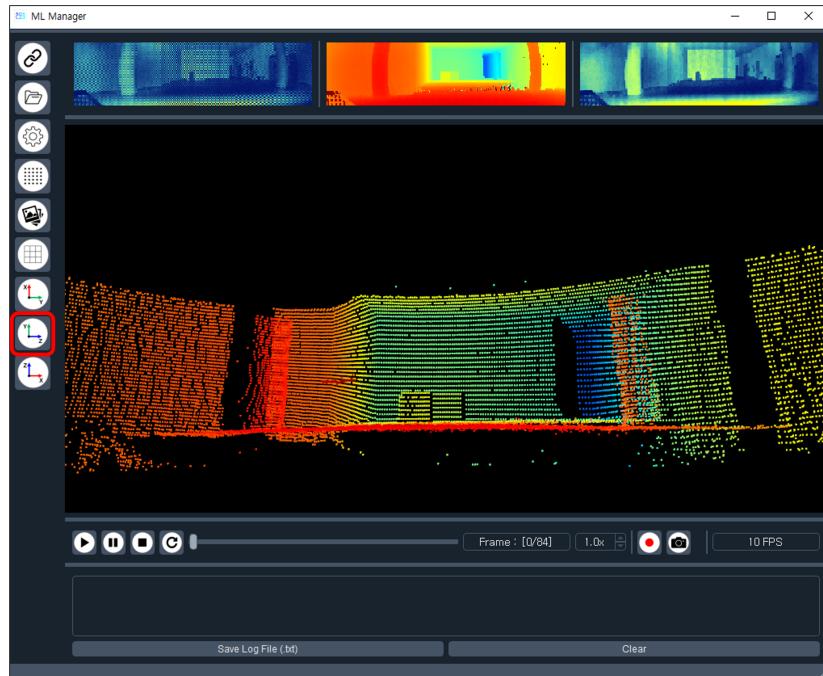


Figure 24: Y-Z Axis 카메라 시점 변경

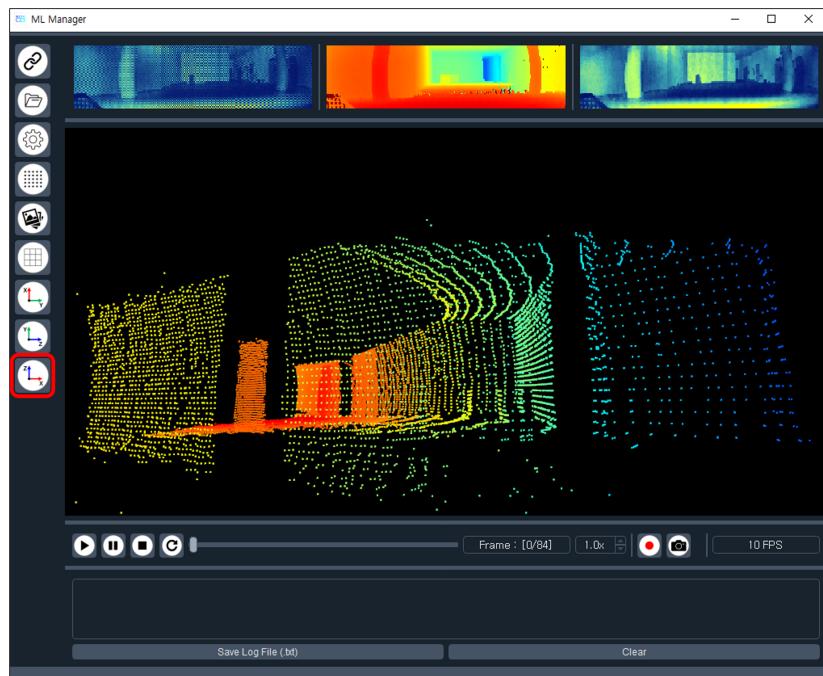


Figure 25: Z-X Axis 카메라 시점 변경

## 2.3 ML Manager Viewer

ML Manager의 중앙에 위치한 Viewer는 Point Cloud를 가시화하기 위한 하단의 Point Cloud Viewer와 Ambient / Depth / Intensity Image를 가시화하기 위한 상단의 Image Viewer로 구성됩니다.

### 2.3.1 Point Cloud Viewer

Point Cloud Viewer에서는 ML LiDAR에서 획득되는 Point Cloud 데이터를 가시화합니다. Point Cloud 데이터는 아래와 같은 3차원 좌표계에서  $(x, y, z, r, g, b)$  형태로 표현됩니다. 여기서  $(x, y, z)$ 는 한 점에 대한 3차원 위치를 의미하며,  $(r, g, b)$ 는 한 점에 대한 RGB 색상 값을 의미합니다.

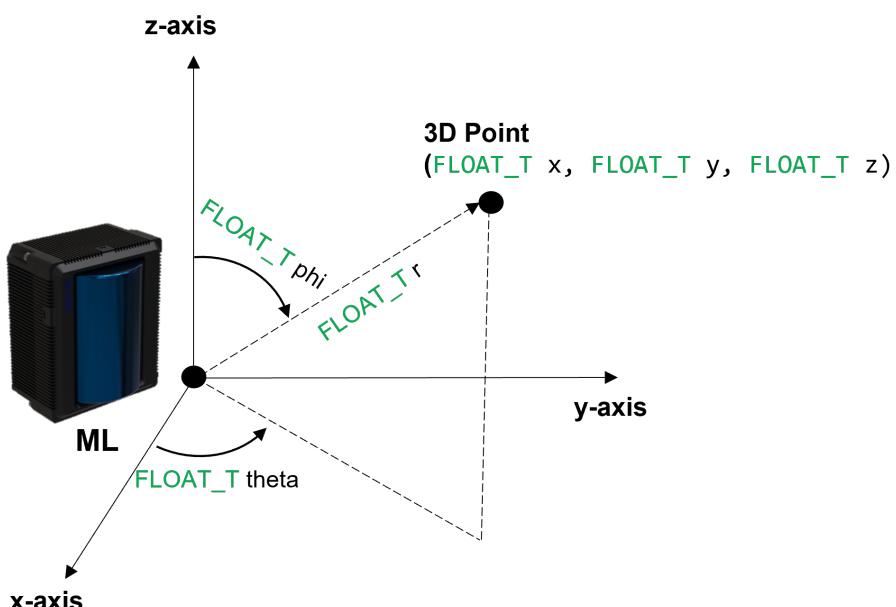


Figure 26: ML LiDAR Coordinate System

Point Cloud Viewer의 조작 방법은 다음과 같습니다.

- <마우스 좌 클릭 + 이동>: 점 군에 대한 시점 회전
- <마우스 휠 버튼 드래그>: 점 군의 확대/축소
- <마우스 우클릭 + 이동>: 점 군에 대한 중심 이동

Point Cloud Viewer는 그림과 같이 ML Manager의 중앙 하단에 위치하며, ML LiDAR에서 획득되는 Point Cloud 데이터를 가시화합니다.

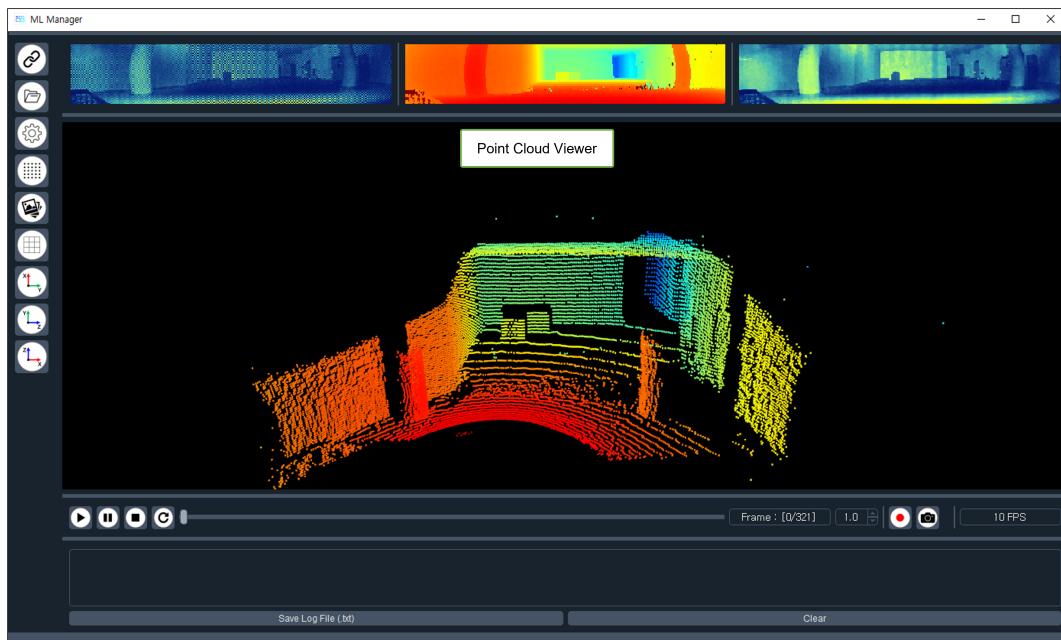


Figure 27: Point Cloud Viewer 화면

### 2.3.2 Image Viewer

Image Viewer에서는 ML LiDAR에서 획득되는 Ambient / Depth / Intensity 이미지를 가시화합니다.

- **Ambient Image**: ML LiDAR 수신부의 각 픽셀에서 획득된 주변광에 의한 신호 세기를 이미지로 표현한 데이터입니다.
- **Depth Image**: ML LiDAR 수신부의 각 픽셀에서 계산되는 거리 값을 이미지로 표현한 데이터입니다.
- **Intensity Image**: ML LiDAR 수신부의 각 픽셀에서 획득된 ML LiDAR의 신호 세기를 이미지로 표현한 데이터입니다.

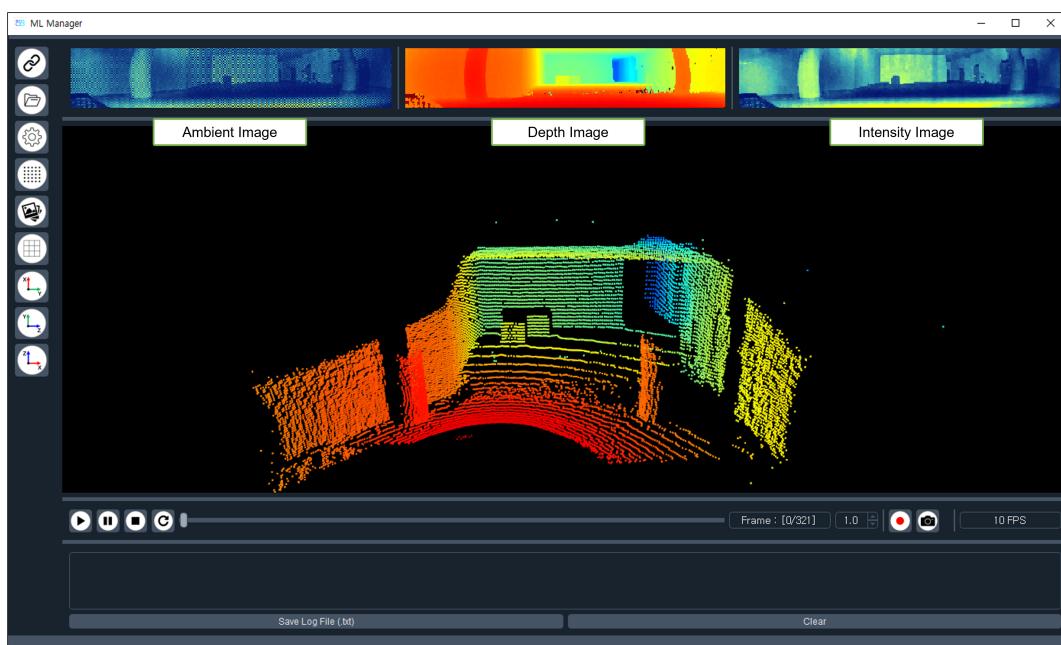


Figure 28: Image Viewer 화면

## 2.4 ML Manager Play Bar

ML Manager의 Viewer 아래에 위치한 Play Bar에서는 ML LiDAR의 데이터를 재생, 녹화, Capture 등의 기능을 제공합니다.

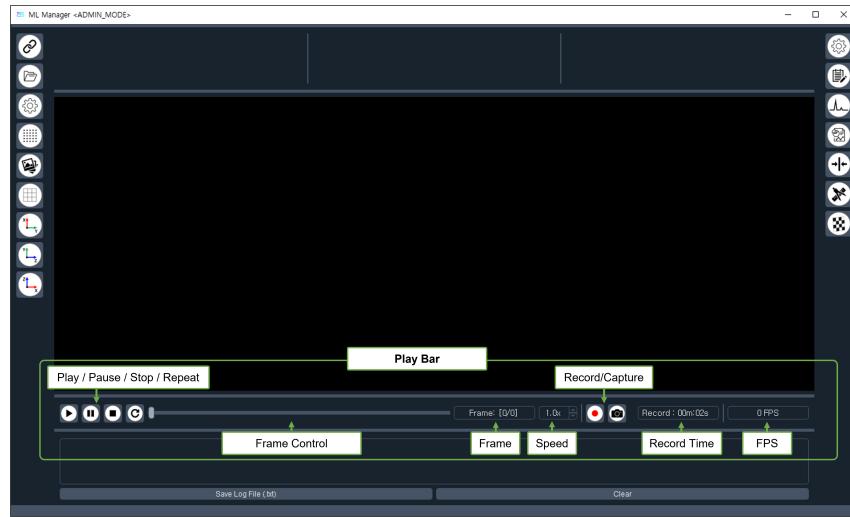


Figure 29: Play Bar 화면

### 2.4.1 Play Controls

④ 을 통해 불러온 ML LiDAR 데이터를 재생하는 기능은 다음과 같습니다

- **Play** ▶: ML LiDAR 녹화 파일을 재생합니다.
- **Pause** ■: ML LiDAR 녹화 파일 재생을 일시정지합니다.
- **Stop** ▨: ML LiDAR 녹화 파일 재생을 정지합니다.
- **Repeat** ◎: ML LiDAR 녹화 파일 재생의 반복 여부를 선택합니다.
- **Frame Control**: ML LiDAR 녹화 파일에서 Frame을 선택/이동 합니다.
- **Frame**: ML LiDAR 녹화 파일의 현재 Frame과 전체 Frame을 보여줍니다.
- **Speed**: ML LiDAR 녹화 파일 재생 속도를 조절합니다. (0.1 ~ 10)
- **FPS**: ML Manager Viewer에 출력되는 ML LiDAR의 Stream 속도 또는 녹화 파일의 재생 속도(FPS)를 보여줍니다.

## 2.4.2 Record

● 버튼을 클릭하면 Stream 되고 있는 ML LiDAR의 연속적인 데이터를 \*.bin 파일로 형태로 녹화할 수 있습니다. 녹화를 시작하면 아이콘 오른쪽에 녹화시간이 표시되며, 기존 아이콘은 ■으로 변경됩니다. 변경된 ■ 버튼을 다시 클릭하면 녹화를 중지할 수 있습니다.

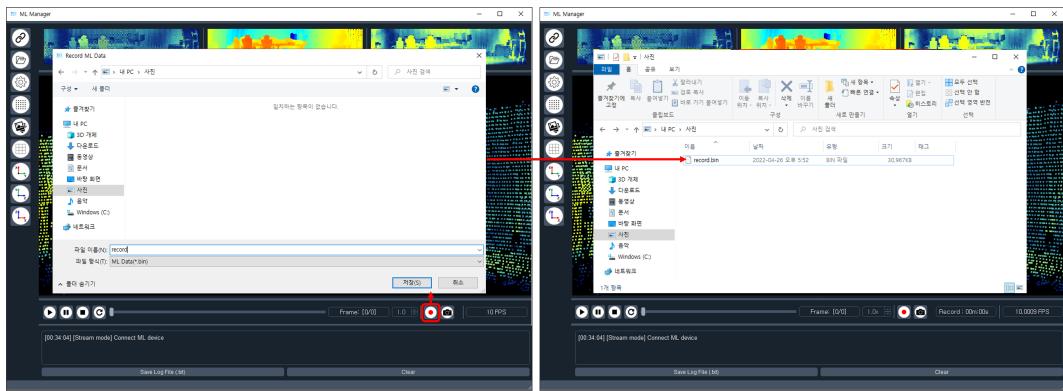


Figure 30: ML LiDAR 데이터 Record 화면

## 2.4.3 Capture

● 버튼을 클릭하면 Stream 되고 있는 ML LiDAR의 한 장면에 대한 Ambient / Depth / Intensity 이미지(\*.png), Point Cloud(\*.pcd) 형태로 저장할 수 있습니다.

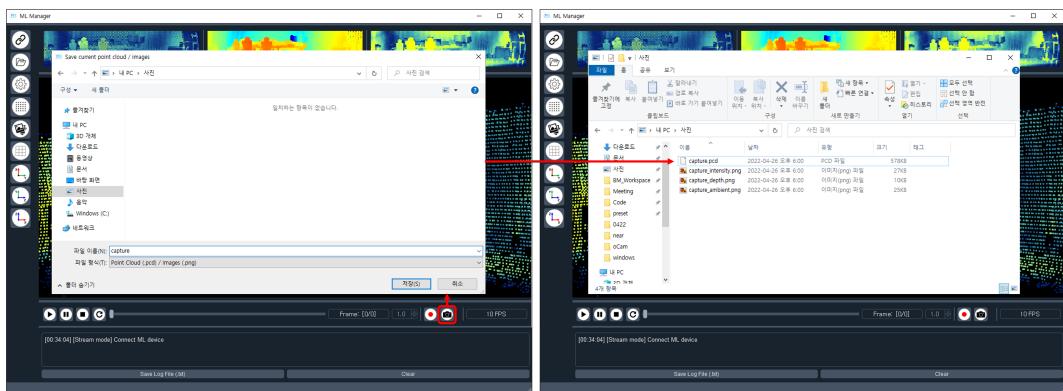


Figure 31: ML LiDAR Capture 화면

## 2.5 ML Log

ML Manager의 Play Bar 아래에 위치한 ML Log에서는 ML Manager의 사용 기록이 출력됩니다. 출력되는 내용은 다음과 같습니다

- **ML LiDAR** 연결 및 해제: [Stream mode] Connect ML device
- **ML LiDAR** 데이터 불러오기: [Play mode] Connect File : [파일 경로]
- **ML LiDAR Functions** 업데이트: [Device Setting] ML setting is updated
- **ML LiDAR Flash** 정보 업데이트: [Device Setting] ML flash is updated
- **ML LiDAR Firmware** 업데이트: [Device Setting] ML Firmware is updated

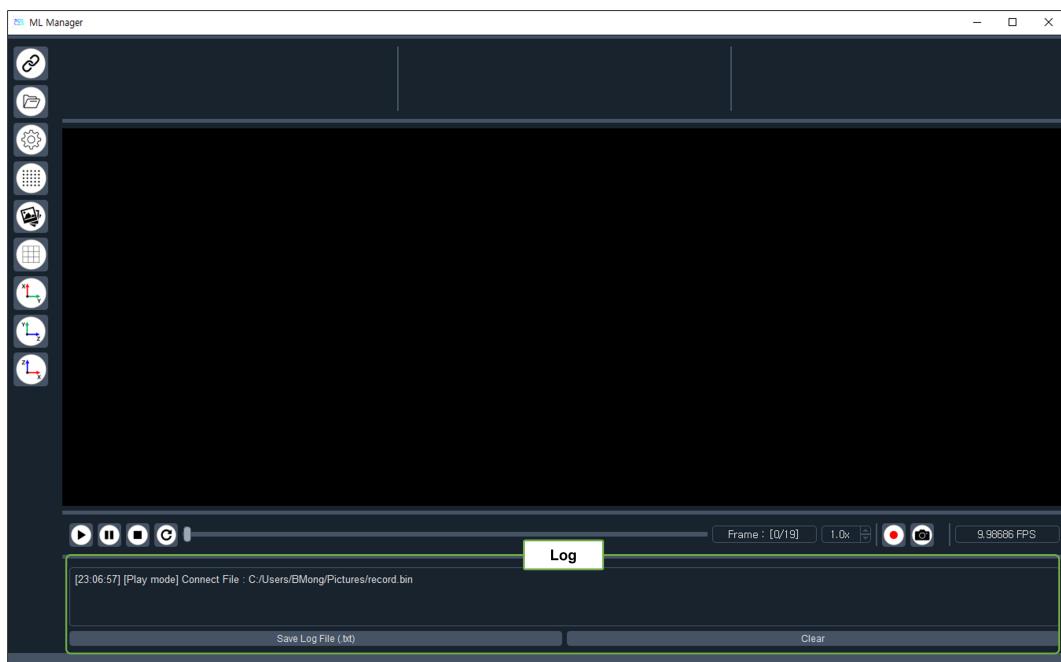


Figure 32: ML Log 화면

## 3 Window API

### 3.1 ML Window API

ML Window API는 libsosslab\_core, libsosslab\_ml 2가지로 구성되어 있으며, 32 bit / 64 bit 환경에서 사용 가능한 lib/dll 파일과 아래의 예제 코드를 함께 제공합니다.

- **test\_ml** : ML LiDAR 연결 예제 코드
- **test\_multi\_ml** : 다중 ML LiDAR 연결 예제 코드
- **test\_record** : ML LiDAR 데이터 녹화 예제 코드
- **test\_data\_conversion** : ML LiDAR 녹화 데이터 파일 변환 예제 코드

test\_core 예제 코드와 test\_record 예제 코드는 제공되는 API를 통해 사용 가능합니다. test\_ml 예제 코드와 test\_multi\_ml 예제 코드는 데이터 가시화 과정이 포함되므로, 아래의 추가적인 외부 라이브러리 설치가 필요합니다. 예제 코드를 위한 외부 라이브러리는 아래 버전을 사용하길 권장합니다.

- **[test\_ml 예제 코드]** OpenCV : 4.2.0 version
- **[test\_multi\_ml 예제 코드]** PCL : 1.9.1 version
- **[test\_multi\_ml 예제 코드]** Boost : 1.62 version
- **[test\_multi\_ml 예제 코드]** VTK : 8.2.0 version

### 3.1.1 API Build

Visual Studio의 새로운 프로젝트를 생성하여 ML API 연동하는 방법은 다음과 같습니다.

1. Visual Studio를 열어 API 사용을 위한 새로운 빈 프로젝트를 생성합니다.

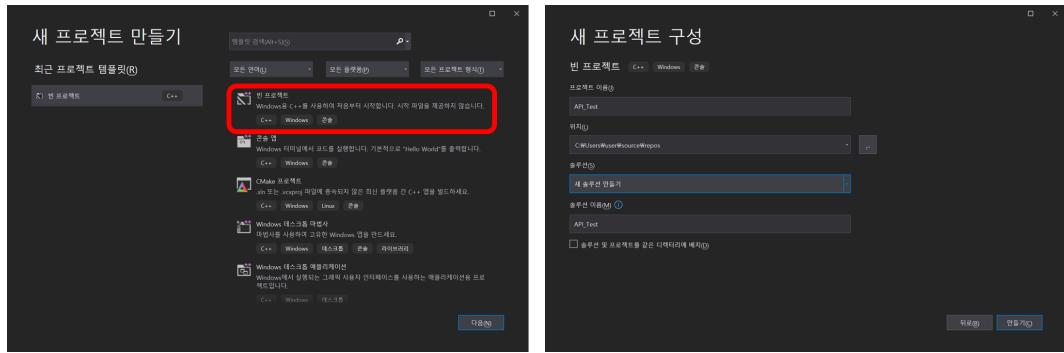


Figure 33: 1. 빈 프로젝트 생성

2. Build Setting을 위하여 Project에 main.cpp 생성 후, 아래 그림과 같이 프로젝터 속성을 클릭 또는 단축키(Alt + Enter)를 입력하여 엽니다.

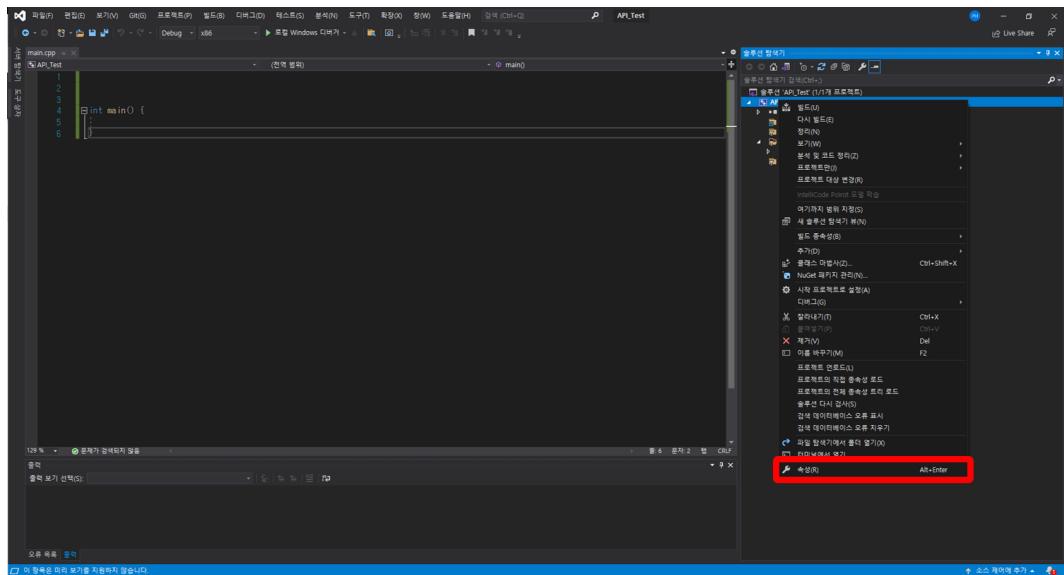


Figure 34: 2. 프로젝터 속성 선택

3. 프로젝트 속성 페이지에서 구성과 플랫폼을 ML API의 솔루션 구성과 플랫폼을 동일하게 변경합니다. ex) ML\_API가 32bit Release일 경우, 프로젝터 구성은 Release로 변경하고, 플랫폼은 Win32 또는 x86으로 변경합니다.

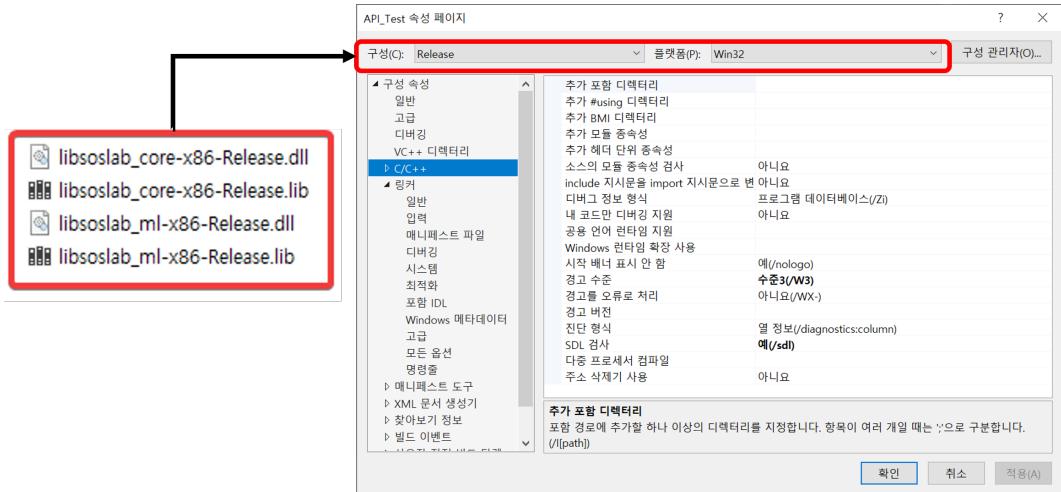


Figure 35: 3. ML API 빌드 환경 설정

4. ML API Header 파일 경로 설정을 위해 속성 페이지 왼쪽 C/C++ - 일반 항목에서 추가 포함 디렉터리 항목에 ML API Header 경로를 입력합니다.

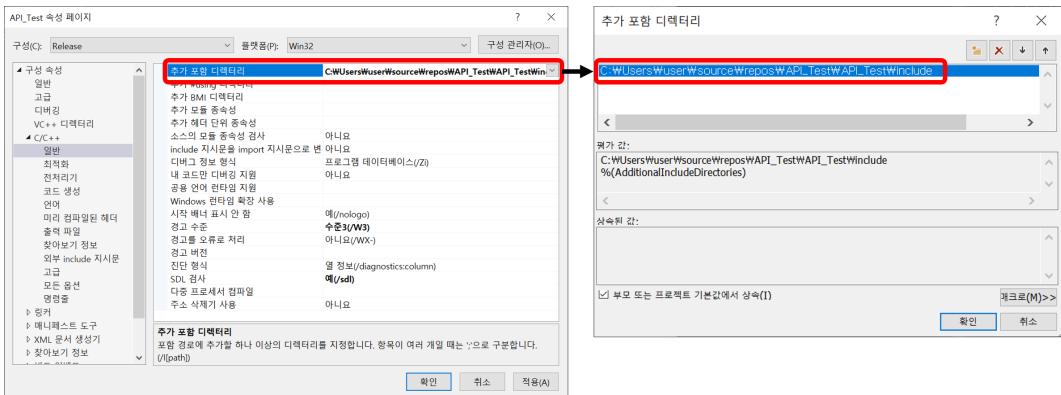


Figure 36: 4. ML API header 파일 경로 설정

5. API 정적 라이브러리(.lib) 경로 설정을 위해 속성 페이지 왼쪽 링커 – 일반 항목에서 추가 라이브러리 디렉터리 항목에 API 정적 라이브러리(.lib) 경로를 입력합니다. 그리고 링커 – 입력 항목에서 추가 종속성 항목에 정적 라이브러리 파일명을 입력합니다.

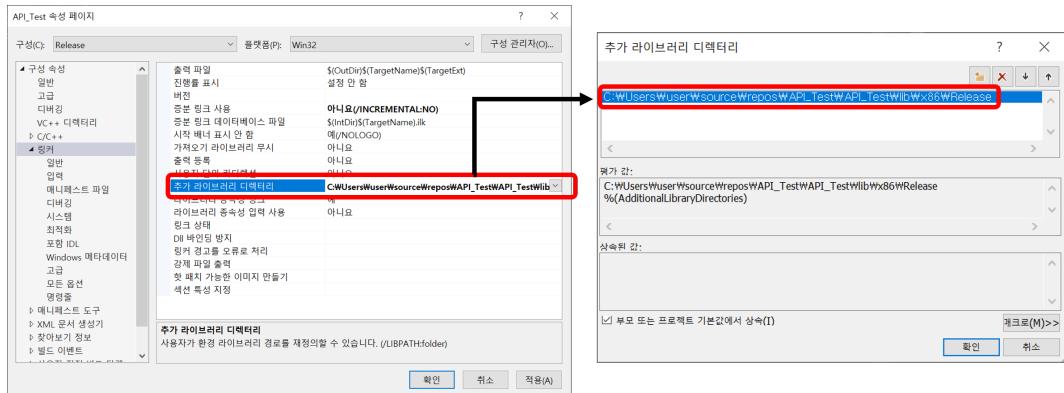


Figure 37: 5-1. ML API 정적 라이브러리(.lib) 경로 설정

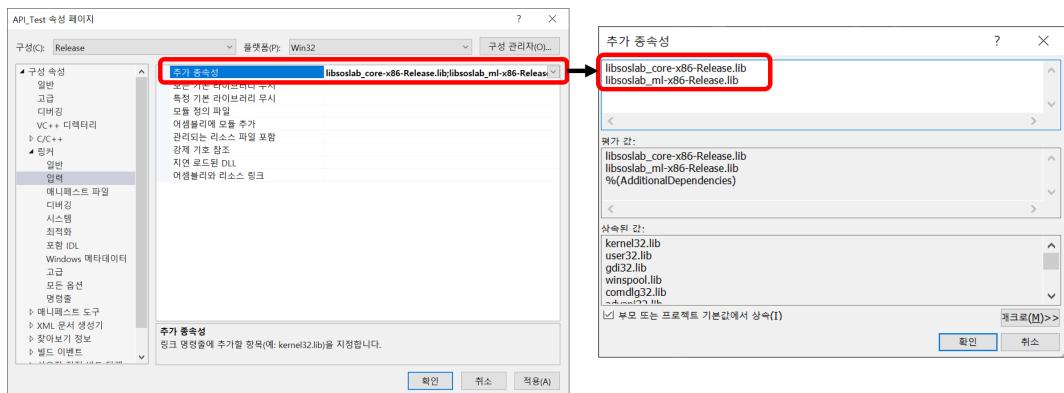


Figure 38: 5-2. ML API 정적 라이브러리(.lib) 추가

6. 프로젝트 아이콘에 오른쪽 버튼을 클릭하여 파일 탐색기에서 폴더 열기 항목을 통해 현재 프로젝트 폴더를 열고, ML API 동적 라이브러리를 프로젝트 폴더로 이동합니다.

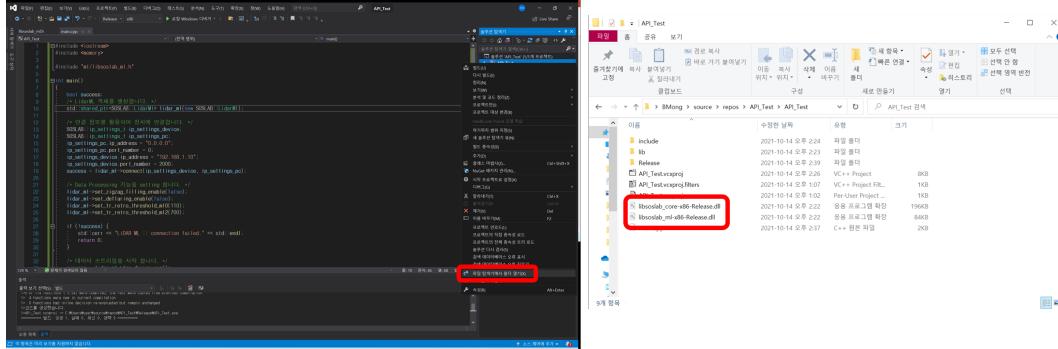


Figure 39: 6. ML API 동적 라이브러리(.dll) 추가

7. 제공된 예제 코드를 Build하여 출력창에 '성공 1, 실패 0'이 출력되면 Build가 완료됩니다.

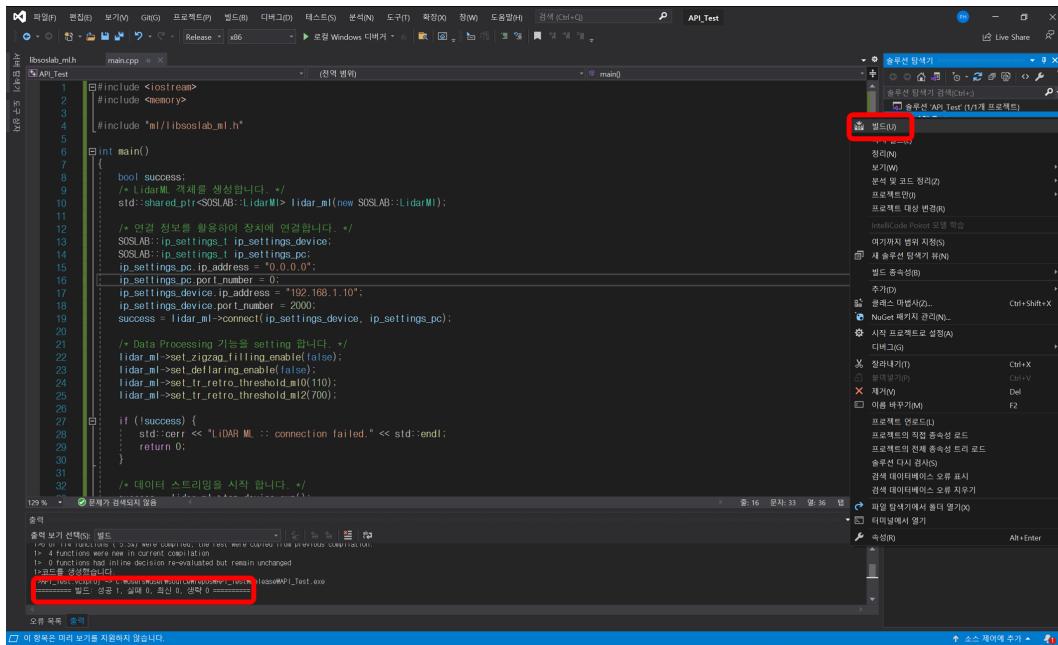


Figure 40: 7. ML API Build 완료

### 3.1.2 Example Code

#### 3.1.2.1 Connection

ML LiDAR와 PC의 통신(연결)을 위한 ML API 예제 코드입니다.

```
1  bool success;
2  std::shared_ptr<SOSLAB::LidarMl> lidar_ml(new SOSLAB::LidarMl);
3  SOSLAB::ip_settings_t ip_settings_device;
4  SOSLAB::ip_settings_t ip_settings_pc;
5  ip_settings_pc.ip_address = "0.0.0.0";
6  ip_settings_pc.port_number = 0;
7  ip_settings_device.ip_address = "192.168.1.10";
8  ip_settings_device.port_number = 2000;
9  success = lidar_ml->connect(ip_settings_device,
10    → ip_settings_pc);
11  if (!success) {
12      std::cerr << "LiDAR ML :: connection failed." << std::endl;
13      return 0;
14  }
15  success = lidar_ml->tcp_device_run();
16  if (!success) {
17      std::cerr << "LiDAR ML :: start failed." << std::endl;
18      return 0;
19  }
std::cout << "LiDAR ML :: Streaming started!" << std::endl;
```

- **2 line** : Local(PC)과 ML LiDAR와의 통신을 위한 변수를 **class SOSLAB::LidarMl** lidar\_ml로서 선언합니다.
- **3-8 line** : Local(PC)과 ML LiDAR의 SOSLAB::ip\_settings\_t 변수를 선언합니다. 예제 코드에서는 ip\_setting\_device, ip\_setting\_pc라는 이름으로 선언합니다. 선언된 변수에 IP 및 Port 값과 ML device의 IP 및 Port 값을 구조체의 ip\_address, port\_number 변수 값에 대입합니다.
- **9-13 line** : 정의된 Local, ML ip\_settings\_t 변수를 connect() 함수의 입력 변수로 ML LiDAR와 연결합니다. 입력 변수 순서는 ML LiDAR IP setting, Local(pc) IP setting 순서입니다. 이후, connect 함수의 반환 값으로 연결 성공 여부를 **bool** 변수형으로 얻습니다.
- **14-19 line** : ML LiDAR와의 연결이 완료되면 tcp\_device\_run() 함수를 통해 Scanning을 시작합니다. 함수의 반환값으로 scanning 시작 여부를 **bool** 변수형으로 얻습니다.

### 3.1.2.2 Functions

ML API에는 ML 데이터 향상을 위한 Data Processing 모듈이 기본적으로 포함됩니다. 아래의 코드는 각 모듈의 사용여부와 모듈에서 사용하는 Parameter를 설정하는 예제 코드입니다. 해당 코드를 사용하기 위해서는 Scanning을 시작하는 함수(tcp\_device\_run(), start()) 전에 호출하면 됩니다. Functions에 대한 자세한 설명은 Appendix에 기술되어 있습니다.

```
1 lidar_ml->set_line_filling_enable(bool enable);  
2 lidar_ml->set_zigzag_filling_enable(bool enable);  
3 lidar_ml->set_deflaring_enable(bool enable);  
4 lidar_ml->set_real_ambient_enable(bool enable);  
5 lidar_ml->set_line_filling_left_offset(int val);  
6 lidar_ml->set_line_filling_right_offset(int val);  
7 lidar_ml->set_zigzag_depth_diff(int val);  
8 lidar_ml->set_zigzag_smoothing_factor(int val);  
9 lidar_ml->set_deflaring_retro_threshold(int val);  
10 lidar_ml->set_deflaring_depth_diff(int val);
```

- **1 line** : ML LiDAR의 Data Processing - Line Filling 기능을 ON(**true**)/OFF(**false**)하는 set\_line\_filling\_enable() 함수를 호출합니다.
- **2 line** : ML LiDAR의 Data Processing - Zigzag Filling 기능을 ON(**true**)/OFF(**false**)하는 set\_zigzag\_filling\_enable() 함수를 호출합니다.
- **3 line** : ML LiDAR의 Data Processing - Deflaring 기능을 ON(**true**)/OFF(**false**)하는 set\_deflaring\_enable() 함수를 호출합니다.
- **4 line** : ML LiDAR의 Data Processing - Real Ambient 기능을 ON(**true**)/OFF(**false**)하는 set\_real\_ambient\_enable() 함수를 호출합니다.
- **5-6 line** : Line Filling 기능의 left, right offset 파라미터를 설정합니다. **int** 값을 입력으로 받습니다.
- **7-8 line** : Zigzag Filling 기능의 smoothing factor와 depth\_diff 파라미터를 설정합니다. **int** 값을 입력으로 받습니다.
- **9-10 line** : Deflaring 기능의 retro\_와 depth\_diff 파라미터를 설정합니다. **int** 값을 입력으로 받습니다.

### 3.1.2.3 Data Streaming

"test\_ml" 프로젝트는 ML LiDAR Data를 획득하는 예제 코드입니다. ML LiDAR Data는 scene\_t 형태로 획득되며, 총 4개의 1차원 데이터 배열(Ambient, Intensity, Depth, Point Cloud)로 구성되어 있습니다. 자세한 scene\_t 구조체 구조에 대한 자세한 설명은 Appendix에 기술되어 있습니다.

```
1 while (keyinput_checker(cv::waitKey(1))) {  
2     SOSLAB::LidarMl::scene_t scene;  
3     if (lidar_ml->get_scene(scene)) {  
4         std::vector<uint16_t> ambient = scene.grey_image;  
5         std::vector<uint16_t> intensity = scene.intensity_image;  
6         std::vector<uint32_t> depth = scene.depth_image;  
7         std::vector<SOSLAB::point_t> pointcloud =  
8             scene.pointcloud;  
9     }  
}
```

- **2 line** : ML LiDAR Data를 저장할 SOSLAB::LidarMl::scene\_t 변수를 scene 이름으로 선언합니다.
- **3 line** : scene\_t 변수를 입력으로 받는 SOSLAB::LidarMl class의 get\_scene() 함수를 통해서 scene 변수에 ML LiDAR Data를 획득합니다.
- **4-6 line** : ML LiDAR Data를 저장한 scene\_t scene 구조체로부터 Ambient, Intensity, Depth, Point Cloud 데이터를 획득합니다.

아래의 코드는 Ambient Data를 이미지로 변환 및 가시화하는 코드입니다.

```
1 std::string grey_viz_name = "Grey Image";
2 cv::Mat grey_image_raw(scene.rows, scene.cols, CV_16SC1,
3   ↳ scene.grey_image.data());
4 cv::Mat grey_image_norm;
5 cv::normalize(grey_image_raw, grey_image_norm, 0, 255,
6   ↳ cv::NORM_MINMAX)
7 cv::Mat grey_rgb;
8 grey_image_norm.convertTo(grey_rgb, CV_8U);
9 cv::applyColorMap(grey_rgb, grey_rgb, cv::COLORMAP_VIRIDIS);
10 cv::imshow(grey_viz_name, grey_rgb);
```

- **2 line**: scene\_t 구조체의 Ambient Data를 이미지로 가시화하는 과정입니다. cv::Mat 형식으로 변환하기 위해 Data Type을 **uint16\_t**(CV\_16SC1)으로 설정합니다.
- **3-4 line** : 이미지 가시화를 위하여 Ambient Data의 최소, 최대 값 기준으로 0~255 값으로 정규화 합니다.
- **5-8 line** : 1 Channel 이미지를 3 Channel Colormap (cv::COLORMAP\_VIRIDIS)으로 변환 후, imshow 함수를 통해 이미지를 가시화합니다.

아래의 코드는 Point Cloud Data를 텍스트 파일(.txt)로 변환하는 코드입니다.

```
1 std::string title = std::to_string(scene.timestamp[0]) +
2   → "_pc.txt";
3 std::ofstream write_pointcloud(title);
4 for (int i = 0; i < scene.pointcloud.size(); i++) {
5   double x = scene.pointcloud[i].xyz.x;
6   double y = scene.pointcloud[i].xyz.y;
7   double z = scene.pointcloud[i].xyz.z;
8   double intensity = scene.intensity_image[i];
9   write_pointcloud << x << ", " << y << ", " << z << ", " <<
10    → intensity << std::endl;
11 }
12 write_pointcloud.close();
```

- 1 line : ML LiDAR Data 획득 시간 scene.timestamp[0]을 파일명으로 정의합니다.
- 3-8 line : 획득되는 Point Cloud Data를 저장하기 위하여 SOSLAB::scene\_t 구조체에서 std::vector<SOSLAB::point\_t> 배열로 저장된 Point Cloud Data의 3차원 좌표(x,y,z)와 scene.intensity\_image의 반사강도 intensity 값을 std::ofstream를 통해 텍스트 파일(.txt)로 저장합니다.

### 3.1.2.4 Record

"test\_record" 프로젝트는 ML LiDAR Data를 녹화하는 예제 코드입니다.

```
1 std::string save_directory = "./log/"
2 #ifdef _WIN32
3     if (_access(save_directory.c_str(), 0)) {
4         if (_mkdir(save_directory.c_str())) return
5             false;
6     }
7 #endif // _WIN32
8 #ifdef __linux__
9     mkdir(save_directory.c_str(), S_IRWXU | S_IRWXG |
10           S_IROTH | S_IXOTH);
11 #endif
12 bool retval = lidar_ml->start_recording(save_directory);
13 int frame_number = 0;
14 int logging_data_size = 10;
15 while (frame_number < logging_data_size) {
16     SOSLAB::LidarMl::scene_t scene;
17     if (lidar_ml->get_scene(scene)) {
18         std::size_t height = scene.rows;
19         std::size_t width = scene.cols;
20         frame_number++;
21     }
22 }
lidar_ml->stop_recording();
```

- **1-9 line** : 녹화 파일의 저장 위치(save\_directory)를 설정하고 해당 폴더가 없으면 생성합니다.
- **10 line** : SOSLAB::LidarMl의 start\_recording(save\_directory) 함수를 호출하여 녹화를 시작합니다.
- **11-20 line** : scene\_t 변수를 입력으로 받는 SOSLAB::LidarMl의 get\_scene함수를 통해서 scene 변수에 ML LiDAR Data를 획득합니다.
- **21 line** : SOSLAB::LidarMl의 stop\_recording() 함수를 호출하여 녹화를 중지합니다.

### 3.1.2.5 Data Conversion

"test\_data\_conversion" 프로젝트는 녹화된 ML LiDAR Data를 \*.pcd 파일로 변환하는 예제 코드입니다.

```
1 int main(int argc, char** argv) {
2     bool success;
3     std::shared_ptr<SOSLAB::LidarMl> lidar_ml(new
4         SOSLAB::LidarMl);
5     if (argc > 1) {
6         std::string path(argv[1]);
7         lidar_ml->connect_file(path);
8         lidar_ml->binary2file("./");
9     }
10    lidar_ml->disconnect();
}
```

- **5-6 line** : 녹화된 ML LiDAR 데이터에 대한 파일 경로를 `std::string`으로 변환하여 `SOSLAB::LidarMl`의 `connect_file(path)` 함수에 입력합니다.
- **7 line** : `SOSLAB::LidarMl`의 `binary2file("./")` 함수를 통해 설정된 경로에 녹화된 ML LiDAR 데이터를 \*.pcd 파일로 변환합니다.

### 3.1.2.6 Multiple LiDAR Setup

"test\_multi\_ml" 프로젝트는 다중 ML LiDAR Data 가시화 예제 코드입니다. 각 ML LiDAR의 네트워크 케이블을 공유기에 모두 연결한 뒤, 공유기와 PC 사이의 LAN 케이블을 연결하여 다중 ML LiDAR와 PC를 연결합니다.

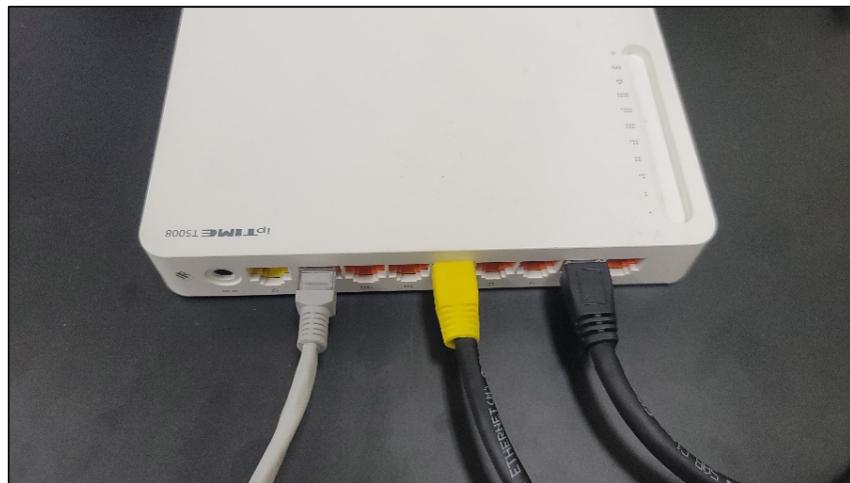


Figure 41: 2개의 ML 연결 예시

다중 ML LiDAR Data 가시화를 위해서 ML LiDAR의 Device Setting을 통해 ML LiDAR의 IP 및 PC PORT 값이 중복되지 않도록 다르게 변경합니다. 아래는 2개의 ML LiDAR Data를 가시화하기 위한 세팅 예시입니다.

- 1번 ML LiDAR Device Setting
  - **ML LiDAR IP** : 192.168.1.10
  - **PC Port** : 2000
- 2번 ML LiDAR Device Setting
  - **ML LiDAR IP** : 192.168.1.11
  - **PC Port** : 2001

```

1  bool success = true;
2  int num_lidar = 1;
3  std::vector<std::shared_ptr<SOSLAB::LidarMl>>
    ↳ lidar_ml_list(new SOSLAB::LidarMl);
4  int standard_port = 2000;
5  for (int i = 0; i < num_lidar; i++) {
6      SOSLAB::ip_settings_t ip_settings_pc;
7      SOSLAB::ip_settings_t ip_settings_device;
8      ip_settings_pc.ip_address = "192.168.1.15";
9      ip_settings_pc.port_number = standard_port + i;
10     ip_settings_device.ip_address = "192.168.1.1" +
11         ↳ to_string(i);
12     ip_settings_device.port_number = standard_port;
13     success = lidar_ml_list[i]->connect(ip_settings_device,
14         ↳ ip_settings_pc);
15     if (!success) {
16         std::cerr << "LiDAR ML #" + std::to_string(i) + "
17             ↳ ::connection failed" << std::endl;
18         return 0;
19     }
20     for (int i = 0; i < num_lidar; i++) {
21         success &= lidar_ml_list[i]->tcp_device_run();
22         if (!success) {
23             std::cerr << "LiDAR ML #" + std::to_string(i) + " ::"
24                 ↳ start failed" << std::endl;
25         }
26     }
}

```

- **2 line** : ML LiDAR와 Local(PC)의의 통신을 위한 SOSLAB::LidarMl 객체를 연결할 ML LiDAR의 개수( num\_lidar)에 따라서 초기화 합니다.

- **4-15 line** : 예제 코드에서는 초기 값 ML IP(192.168.1.10), PC port(2000)에서 ML LiDAR 개수에 따라 ML IP, PC port의 마지막 값을 1씩 증가시켜 PC와 연결합니다.

**Example :** 2개의 ML 연결

1번 ML LiDAR IP = "192.168.1.10", PC port = "2000"

2번 ML LiDAR IP = "192.168.1.11", PC port = "2001"

- **16-18 line** : 모든 ML LiDAR를 연결한 뒤, ML LiDAR 개수에 맞게 tcp\_device\_run() 함수를 호출합니다.

Multiple ML LiDAR 예제에서는 Point Cloud를 가시화하기 위해 PCL과 VTK Li-brary을 사용합니다. 아래 그림은 2개의 ML LiDAR Data를 가시화한 예시로써, 1 번 ML LiDAR에서 획득한 Point Cloud를 흰색, 2번 ML LiDAR에서 획득한 Point Cloud를 빨간색으로 가시화한 결과입니다.

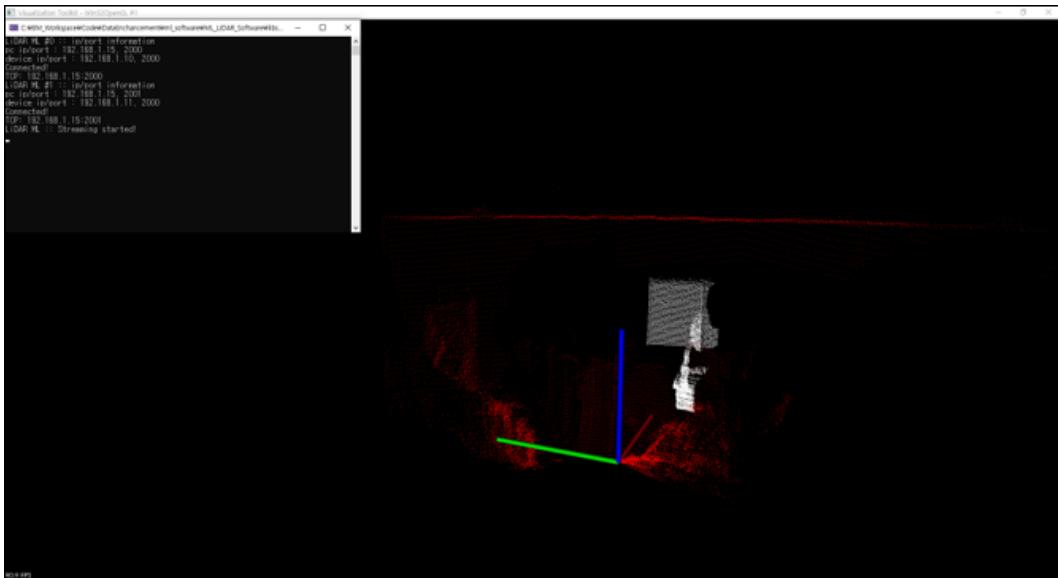


Figure 42: 다중 ML LiDAR 연결 및 가시화 예시

## 4 Ubuntu API

### 4.1 ML Ubuntu API

ML Ubuntu API는 Ubuntu 18.04 및 ROS Melodic 버전에서 사용 가능합니다. ML Ubuntu API는 libsosslab\_core, libsosslab\_ml 2가지로 구성되어 있으며, 64 bit 환경에서 사용 가능한 so 파일과 아래의 예제 코드를 함께 제공합니다.

- **test\_ml** : ML LiDAR 연결 예제 코드
- **test\_multi\_ml** : 다중 ML LiDAR 연결 예제 코드
- **test\_record** : ML LiDAR 데이터 녹화 예제 코드
- **test\_data\_conversion** : ML LiDAR 녹화 데이터 파일 변환 예제 코드

test\_core 예제 코드와 test\_record 예제 코드는 제공되는 API를 통해 사용 가능합니다. test\_ml 예제 코드와 test\_multi\_ml 예제 코드는 데이터 가시화 과정이 포함되므로, 아래의 추가적인 외부 라이브러리 설치가 필요합니다. 예제 코드를 위한 외부 라이브러리는 아래 버전을 사용하길 권장합니다.

- **[test\_ml 예제 코드]** OpenCV : 4.2.0 version
- **[test\_multi\_ml 예제 코드]** PCL : 1.9.1 version
- **[test\_multi\_ml 예제 코드]** Boost : 1.62 version
- **[test\_multi\_ml 예제 코드]** VTK : 8.2.0 version

#### 4.1.1 API Build

Ubuntu 18.04의 OpenCV Package 설치 과정입니다.

OpenCV Package에 필요한 Library를 아래 Script를 통해 설치합니다.

---

[OpenCV Package를 위한 사전 Library 설치]

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential cmake
$ sudo apt-get install pkg-config
$ sudo apt-get install libjpeg-dev libtiff5-dev libpng-dev
$ sudo apt-get install libavcodec-dev libavformat-dev
  ↳ libswscale-dev libxvidcore-dev libx264-dev libxine2-dev
$ sudo apt-get install libv4l-dev v4l-utils
$ sudo apt-get install libgstreamer1.0-dev
  ↳ libgstreamer-plugins-base1.0-dev
$ sudo apt-get install libgtk2.0-dev
$ sudo apt-get install mesa-utils libgl1-mesa-dri
  ↳ libgtkg12.0-dev libgtkglext1-dev
$ sudo apt-get install libatlas-base-dev gfortran libeigen3-dev
```

---

Library 설치가 완료되면 아래 Script를 통해 OpenCV Package를 설치합니다.

```
[OpenCV 설치]
$ cd ~
$ mkdir opencv && cd opencv
$ wget -O opencv.zip
→ https://github.com/opencv/opencv/archive/4.2.0.zip
$ unzip opencv.zip
$ cd opencv-4.2.0
$ mkdir build && cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D WITH_TBB=OFF \
-D WITH_IPP=OFF \
-D WITH_1394=OFF \
-D BUILD_WITH_DEBUG_INFO=OFF \
-D BUILD_DOCS=OFF \
-D INSTALL_C_EXAMPLES=ON \
-D BUILD_EXAMPLES=OFF \
-D BUILD_TESTS=OFF \
-D BUILD_PERF_TESTS=OFF \
-D WITH_QT=OFF \
-D WITH_GTK=ON \
-D WITH_OPENGL=ON \
-D WITH_V4L=ON \
-D WITH_FFMPEG=ON \
-D WITH_XINE=ON \
-D OPENCV_GENERATE_PKGCONFIG=ON ../
$ time make -j4
$ sudo make install
$ cd ~
$ sudo nano .bashrc
- bashrc 파일 맨 아래 줄에 아래 내용 추가
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
$ source ~/.bashrc
$ sudo ldconfig
```

Ubuntu 18.04의 ROS Melodic 버전 설치 과정입니다.

예제 코드("ROS\_API\_CodeEX")를 사용하기 위해서는 ROS 설치가 필요합니다.

1. Ubuntu 18.04 설치 후, Terminal을 실행하여 아래 Script를 순서대로 입력하여 ROS를 설치합니다.

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
↪ $(lsb_release -sc) main" >
↪ /etc/apt/sources.list.d/ros-latest.list'
$ sudo apt install curl
$ curl -s
↪ https://raw.githubusercontent.com/ros/rosdistro/master/
↪ ros.asc | sudo apt-key add -
$ sudo apt update
$ sudo apt install ros-melodic-desktop-full
```

2. ROS 설치 후, ROS 환경 설정 파일(setup.bash)을 자동으로 Ubuntu에 추가하기 위하여 아래 Script를 순서대로 입력합니다.

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

3. 환경 설정이 완료되면, ROS 설치여부를 확인하기 위해 3개의 Terminal에 아래 Script를 각각 입력합니다.

```
[Terminal #1]  
$ roscore  
[Terminal #2]  
$ rosrun turtlesim turtlesim_node  
[Terminal #3]  
$ rosrun turtlesim turtle_teleop_key
```

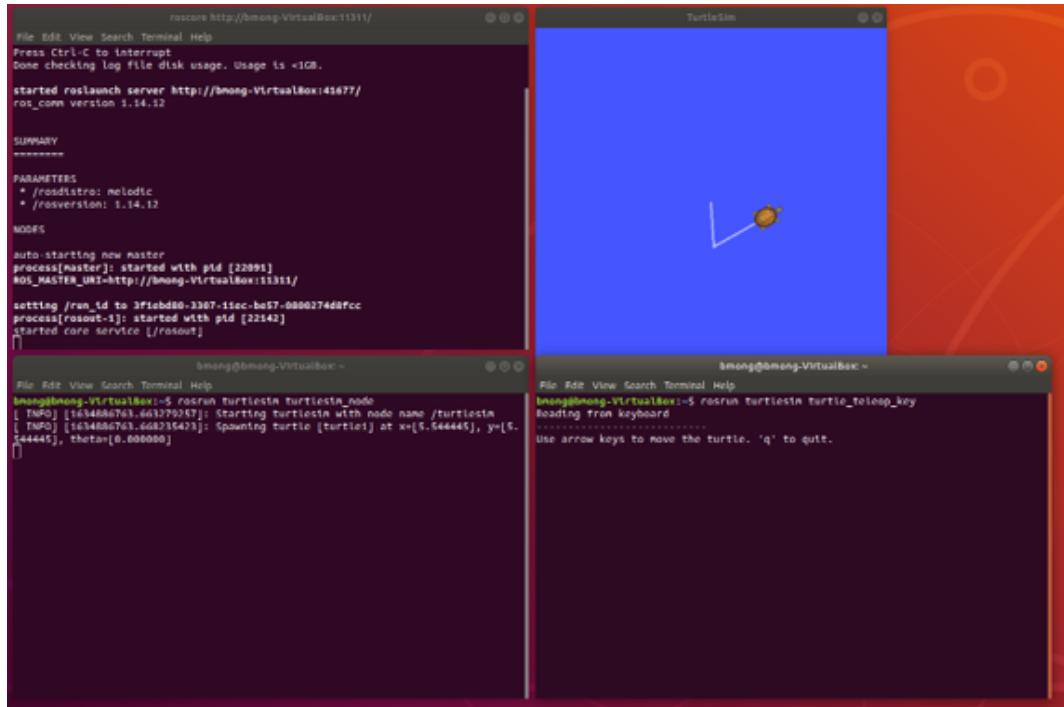


Figure 43: ROS 설치여부 확인 테스트 결과

4. 예제 코드에서 사용되는 Point Cloud 가시화를 위하여 pcl\_ros package를 추가적으로 설치합니다.

```
$ sudo apt install ros-melodic-pcl-ros
```

5. Home에 catkin\_ws 폴더를 생성한 뒤, ROS 예제 폴더("ROS\_API\_CodeEX") 안의 src 폴더를 아래 그림과 같이 Home/catkin\_ws 위치로 이동합니다.

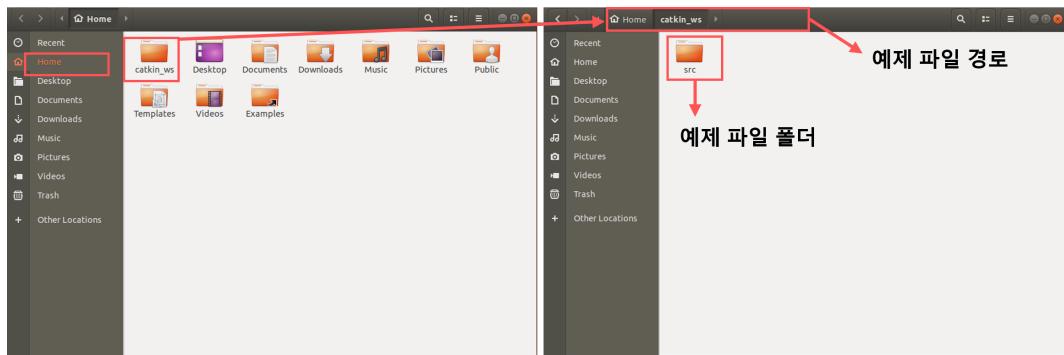


Figure 44: ROS 예제 코드 위치 변경

6. Terminal의 작업 Direcotry를 catkin\_ws 폴더로 변경한 뒤, “catkin\_make”를 입력하여 ML Package를 생성합니다.

```
$ cd ~/catkin_ws
$ catkin_make
```

build 결과

예제 코드 build 성공

```
bmong@bmong-VirtualBox: ~/catkin_ws
File Edit View Search Terminal Help
bmong@bmong-VirtualBox:~/catkin_ws$ catkin_make

bmong@bmong-VirtualBox: ~/catkin_ws
# define DEG2RAD(x) ((x)*0.017453293)
In file included from /home/bmong/catkin_ws/src/ml/include/lbsoslab_ml.h:12:0,
                 from /home/bmong/catkin_ws/src/ml/src/mathn.cpp:6:
/home/bmong/catkin_ws/src/ml/include/soslab_typedef.h:49:0: warning: "RAD2DEG" redefined
#define RAD2DEG      (180.0 / M_PI)
In file included from /usr/include/pcl-1.8/pcl/PCLHeader.h:11:0,
                 from /usr/include/pcl-1.8/pcl/pcl_point_cloud.h:48:
                 from /opt/ros/melodic/include/pcl_ros/point_cloud.h:50:
                 from /home/bmong/catkin_ws/src/ml/src/mathn.cpp:5:
/usr/include/pcl-1.8/pcl/pcl_macros.h:143:0: note: this is the location of the previous definition
#define RAD2DEG(x) ((x)*57.29578)

In file included from /home/bmong/catkin_ws/src/ml/src/mathn.cpp:5:
/opt/ros/melodic/include/pcl_ros/point_cloud.h:503:27: warning: variable template is only available with -std=c++14 or -std=gnu++14
  constexpr static bool pcl_uses_boost = true;
                                         ^
[100%] Linking CXX executable /home/bmong/catkin_ws/devel/lib/ml/ml
[100%] Built target ml
bmong@bmong-VirtualBox:~/catkin_ws$
```

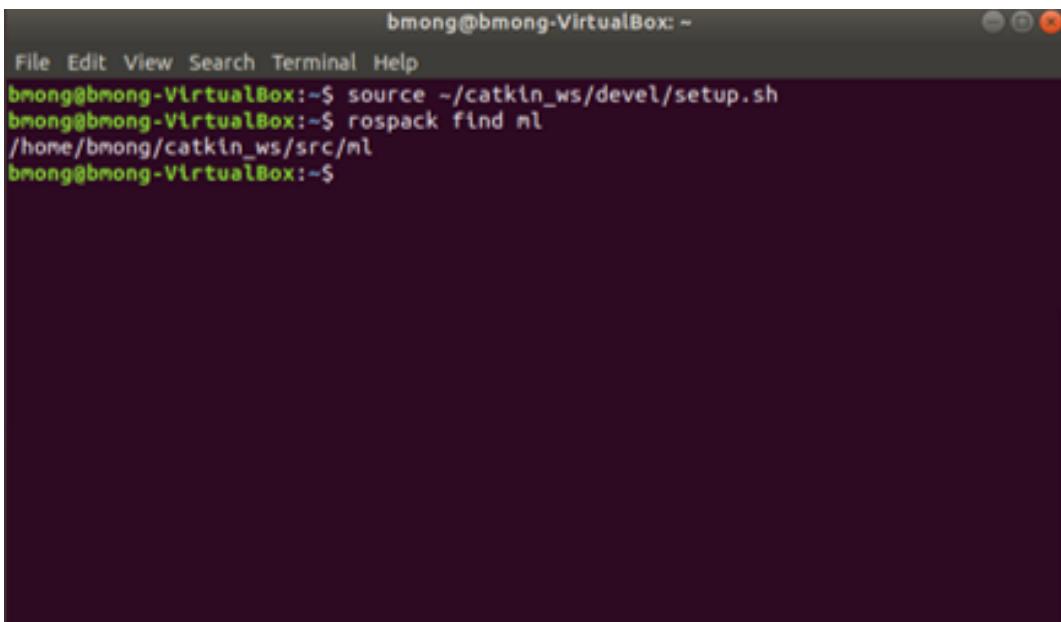
Figure 45: catkin\_make를 통해 ML 예제 코드 Build

7. ML Package를 ROS 환경에 추가하기 위해 아래 Script를 입력합니다.

---

```
$ source ~/catkin_ws/devel/setup.sh  
$ rospack find ml
```

---



A screenshot of a terminal window titled "bmong@bmong-VirtualBox: ~". The window shows the following command sequence:

```
bmong@bmong-VirtualBox:~$ source ~/catkin_ws/devel/setup.sh  
bmong@bmong-VirtualBox:~$ rospack find ml  
/home/bmong/catkin_ws/src/ml  
bmong@bmong-VirtualBox:~$
```

Figure 46: ML Package를 ROS 환경에 추가

8. ML LiDAR와 PC의 연결을 위해 Setting창의 Network 항목에서 아래와 같이 IPv4 Network를 설정합니다.

- IPv4 - Manual
- Address : 192.168.1.15
- NetMask : 255.255.255.0

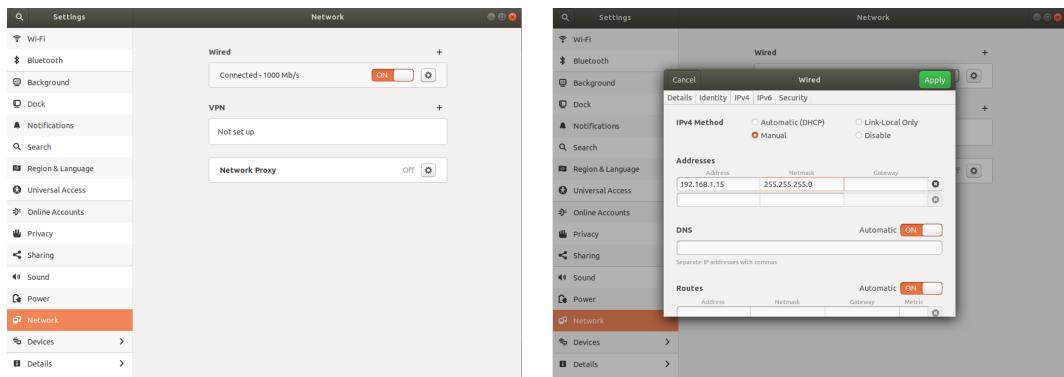


Figure 47: IPv4 Network 설정

9. Network 설정 후, 아래 Script를 입력하여 ML LiDAR와 연결 합니다. 정상적으로 ML LiDAR가 연결 되면 아래 그림과 같이 Terminal 창에 Lidar ML :: Streaming started! 출력을 확인할 수 있습니다.

```
$ cd ~/catkin_ws/src/ml/launch  
$ roslaunch ml.launch
```

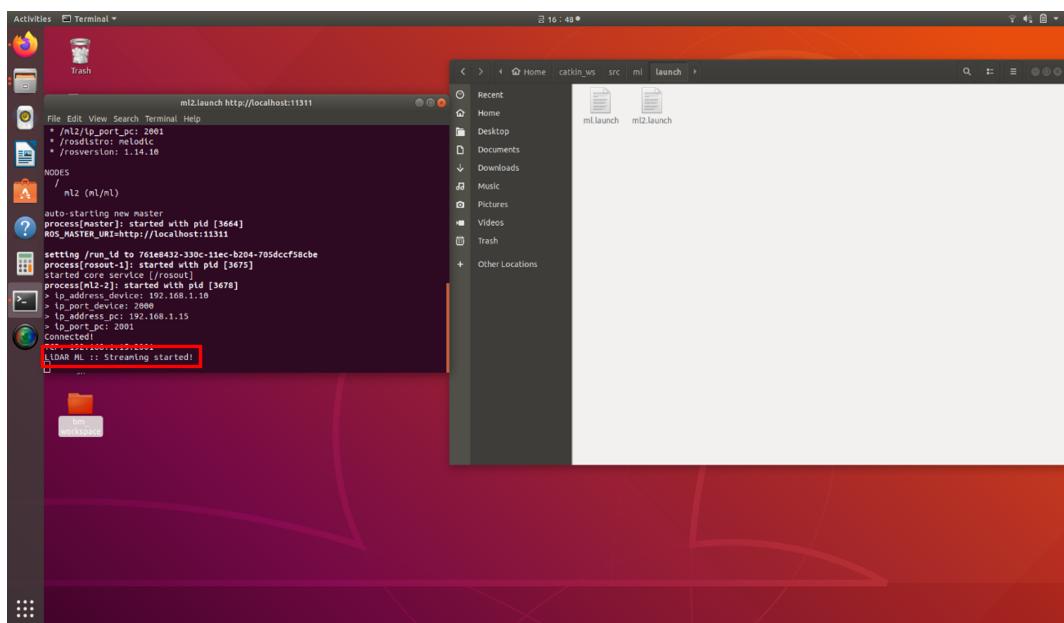


Figure 48: ROS 환경에서의 ML LiDAR 연결 화면

## 4.1.2 Example Code

### 4.1.2.1 Connection

ML LiDAR와 PC의 통신(연결)을 위한 ML API 예제 코드입니다.

```
1  bool success;
2  std::shared_ptr<SOSLAB::LidarMl> lidar_ml(new SOSLAB::LidarMl);
3  SOSLAB::ip_settings_t ip_settings_device;
4  SOSLAB::ip_settings_t ip_settings_pc;
5  ip_settings_pc.ip_address = "0.0.0.0";
6  ip_settings_pc.port_number = 0;
7  ip_settings_device.ip_address = "192.168.1.10";
8  ip_settings_device.port_number = 2000;
9  success = lidar_ml->connect(ip_settings_device,
10    → ip_settings_pc);
11  if (!success) {
12      std::cerr << "LiDAR ML :: connection failed." << std::endl;
13      return 0;
14  }
15  success = lidar_ml->tcp_device_run();
16  if (!success) {
17      std::cerr << "LiDAR ML :: start failed." << std::endl;
18      return 0;
19  }
std::cout << "LiDAR ML :: Streaming started!" << std::endl;
```

- **2 line** : Local(PC)과 ML LiDAR와의 통신을 위한 변수를 **class SOSLAB::LidarMl** lidar\_ml로서 선언합니다.
- **3-8 line** : Local(PC)과 ML LiDAR의 SOSLAB::ip\_settings\_t 변수를 선언합니다. 예제 코드에서는 ip\_setting\_device, ip\_setting\_pc라는 이름으로 선언합니다. 선언된 변수에 IP 및 Port 값과 ML device의 IP 및 Port 값을 구조체의 ip\_address, port\_number 변수 값에 대입합니다.
- **9-13 line** : 정의된 Local, ML ip\_settings\_t 변수를 connect() 함수의 입력 변수로 ML LiDAR와 연결합니다. 입력 변수 순서는 ML LiDAR IP setting, Local(pc) IP setting 순서입니다. 이후, connect 함수의 반환 값으로 연결 성공 여부를 **bool** 변수형으로 얻습니다.
- **14-19 line** : ML LiDAR와의 연결이 완료되면 tcp\_device\_run() 함수를 통해 Scanning을 시작합니다. 함수의 반환값으로 scanning 시작 여부를 **bool** 변수형으로 얻습니다.

#### 4.1.2.2 Functions

ML API에는 ML 데이터 향상을 위한 Data Processing 모듈이 기본적으로 포함됩니다. 아래의 코드는 각 모듈의 사용여부와 모듈에서 사용하는 Parameter를 설정하는 예제 코드입니다. 해당 코드를 사용하기 위해서는 Scanning을 시작하는 함수(tcp\_device\_run(), start()) 전에 호출하면 됩니다. Functions에 대한 자세한 설명은 Appendix에 기술되어 있습니다.

```
1 lidar_ml->set_line_filling_enable(bool enable);  
2 lidar_ml->set_zigzag_filling_enable(bool enable);  
3 lidar_ml->set_deflaring_enable(bool enable);  
4 lidar_ml->set_real_ambient_enable(bool enable);  
5 lidar_ml->set_line_filling_left_offset(int val);  
6 lidar_ml->set_line_filling_right_offset(int val);  
7 lidar_ml->set_zigzag_depth_diff(int val);  
8 lidar_ml->set_zigzag_smoothing_factor(int val);  
9 lidar_ml->set_deflaring_retro_threshold(int val);  
10 lidar_ml->set_deflaring_depth_diff(int val);
```

- **1 line** : ML LiDAR의 Data Processing - Line Filling 기능을 ON(**true**)/OFF(**false**)하는 set\_line\_filling\_enable() 함수를 호출합니다.
- **2 line** : ML LiDAR의 Data Processing - Zigzag Filling 기능을 ON(**true**)/OFF(**false**)하는 set\_zigzag\_filling\_enable() 함수를 호출합니다.
- **3 line** : ML LiDAR의 Data Processing - Deflaring 기능을 ON(**true**)/OFF(**false**)하는 set\_deflaring\_enable() 함수를 호출합니다.
- **4 line** : ML LiDAR의 Data Processing - Real Ambient 기능을 ON(**true**)/OFF(**false**)하는 set\_real\_ambient\_enable() 함수를 호출합니다.
- **5-6 line** : Line Filling 기능의 left, right offset 파라미터를 설정합니다. **int** 값을 입력으로 받습니다.
- **7-8 line** : Zigzag Filling 기능의 smoothing factor와 depth\_diff 파라미터를 설정합니다. **int** 값을 입력으로 받습니다.
- **9-10 line** : Deflaring 기능의 retro\_와 depth\_diff 파라미터를 설정합니다. **int** 값을 입력으로 받습니다.

#### 4.1.2.3 Data Streaming

"test\_ml" 프로젝트는 ML LiDAR Data를 획득하는 예제 코드입니다. ML LiDAR Data는 scene\_t 형태로 획득되며, 총 4개의 1차원 데이터 배열(Ambient, Intensity, Depth, Point Cloud)로 구성되어 있습니다. 자세한 scene\_t 구조체 구조에 대한 자세한 설명은 Appendix에 기술되어 있습니다.

```
1 while (keyinput_checker(cv::waitKey(1))) {  
2     SOSLAB::LidarMl::scene_t scene;  
3     if (lidar_ml->get_scene(scene)) {  
4         std::vector<uint16_t> ambient = scene.grey_image;  
5         std::vector<uint16_t> intensity = scene.intensity_image;  
6         std::vector<uint32_t> depth = scene.depth_image;  
7         std::vector<SOSLAB::point_t> pointcloud =  
8             scene.pointcloud;  
9     }  
}
```

- **2 line** : ML LiDAR Data를 저장할 SOSLAB::LidarMl::scene\_t 변수를 scene 이름으로 선언합니다.
- **3 line** : scene\_t 변수를 입력으로 받는 SOSLAB::LidarMl class의 get\_scene() 함수를 통해서 scene 변수에 ML LiDAR Data를 획득합니다.
- **4-6 line** : ML LiDAR Data를 저장한 scene\_t scene 구조체로부터 Ambient, Intensity, Depth, Point Cloud 데이터를 획득합니다.

아래의 코드는 Ambient Data를 이미지로 변환 및 가시화하는 코드입니다.

```
1 std::string grey_viz_name = "Grey Image";
2 cv::Mat grey_image_raw(scene.rows, scene.cols, CV_16SC1,
3   ↳ scene.grey_image.data());
4 cv::Mat grey_image_norm;
5 cv::normalize(grey_image_raw, grey_image_norm, 0, 255,
6   ↳ cv::NORM_MINMAX)
7 cv::Mat grey_rgb;
8 grey_image_norm.convertTo(grey_rgb, CV_8U);
9 cv::applyColorMap(grey_rgb, grey_rgb, cv::COLORMAP_VIRIDIS);
10 cv::imshow(grey_viz_name, grey_rgb);
```

- **2 line**: scene\_t 구조체의 Ambient Data를 이미지로 가시화하는 과정입니다. cv::Mat 형식으로 변환하기 위해 Data Type을 **uint16\_t**(CV\_16SC1)으로 설정합니다.
- **3-4 line** : 이미지 가시화를 위하여 Ambient Data의 최소, 최대 값 기준으로 0~255 값으로 정규화 합니다.
- **5-8 line** : 1 Channel 이미지를 3 Channel Colormap (cv::COLORMAP\_VIRIDIS)으로 변환 후, imshow 함수를 통해 이미지를 가시화합니다.

아래의 코드는 Point Cloud Data를 텍스트 파일(.txt)로 변환하는 코드입니다.

```
1 std::string title = std::to_string(scene.timestamp[0]) +
2   → "_pc.txt";
3 std::ofstream write_pointcloud(title);
4 for (int i = 0; i < scene.pointcloud.size(); i++) {
5   double x = scene.pointcloud[i].xyz.x;
6   double y = scene.pointcloud[i].xyz.y;
7   double z = scene.pointcloud[i].xyz.z;
8   double intensity = scene.intensity_image[i];
9   write_pointcloud << x << ", " << y << ", " << z << ", " <<
10    → intensity << std::endl;
11 }
12 write_pointcloud.close();
```

- 1 line : ML LiDAR Data 획득 시간 scene.timestamp[0]을 파일명으로 정의합니다.
- 3-8 line : 획득되는 Point Cloud Data를 저장하기 위하여 SOSLAB::scene\_t 구조체에서 std::vector<SOSLAB::point\_t> 배열로 저장된 Point Cloud Data의 3차원 좌표(x,y,z)와 scene.intensity\_image의 반사강도 intensity 값을 std::ofstream를 통해 텍스트 파일(.txt)로 저장합니다.

#### 4.1.2.4 Record

"test\_record" 프로젝트는 ML LiDAR Data를 녹화하는 예제 코드입니다.

```
1 std::string save_directory = "./log/"
2 #ifdef _WIN32
3     if (_access(save_directory.c_str(), 0)) {
4         if (_mkdir(save_directory.c_str())) return
5             false;
6     }
7 #endif // _WIN32
8 #ifdef __linux__
9     mkdir(save_directory.c_str(), S_IRWXU | S_IRWXG |
10           S_IROTH | S_IXOTH);
11 #endif
12 bool retval = lidar_ml->start_recording(save_directory);
13 int frame_number = 0;
14 int logging_data_size = 10;
15 while (frame_number < logging_data_size) {
16     SOSLAB::LidarMl::scene_t scene;
17     if (lidar_ml->get_scene(scene)) {
18         std::size_t height = scene.rows;
19         std::size_t width = scene.cols;
20         frame_number++;
21     }
22 }
lidar_ml->stop_recording();
```

- **1-9 line** : 녹화 파일의 저장 위치(save\_directory)를 설정하고 해당 폴더가 없으면 생성합니다.
- **10 line** : SOSLAB::LidarMl의 start\_recording(save\_directory) 함수를 호출하여 녹화를 시작합니다.
- **11-20 line** : scene\_t 변수를 입력으로 받는 SOSLAB::LidarMl의 get\_scene함수를 통해서 scene 변수에 ML LiDAR Data를 획득합니다.
- **21 line** : SOSLAB::LidarMl의 stop\_recording() 함수를 호출하여 녹화를 중지합니다.

#### 4.1.2.5 Data Conversion

"test\_data\_conversion" 프로젝트는 녹화된 ML LiDAR Data를 \*.pcd 파일로 변환하는 예제 코드입니다.

```
1 int main(int argc, char** argv) {
2     bool success;
3     std::shared_ptr<SOSLAB::LidarMl> lidar_ml(new
4         SOSLAB::LidarMl);
5     if (argc > 1) {
6         std::string path(argv[1]);
7         lidar_ml->connect_file(path);
8         lidar_ml->binary2file("./");
9     }
10    lidar_ml->disconnect();
}
```

- **5-6 line** : 녹화된 ML LiDAR 데이터에 대한 파일 경로를 `std::string`으로 변환하여 `SOSLAB::LidarMl`의 `connect_file(path)` 함수에 입력합니다.
- **7 line** : `SOSLAB::LidarMl`의 `binary2file("./")` 함수를 통해 설정된 경로에 녹화된 ML LiDAR 데이터를 \*.pcd 파일로 변환합니다.

#### 4.1.2.6 Multiple LiDAR Setup

"test\_multi\_ml" 프로젝트는 다중 ML LiDAR Data 가시화 예제 코드입니다. 각 ML LiDAR의 네트워크 케이블을 공유기에 모두 연결한 뒤, 공유기와 PC 사이의 LAN 케이블을 연결하여 다중 ML LiDAR와 PC를 연결합니다.

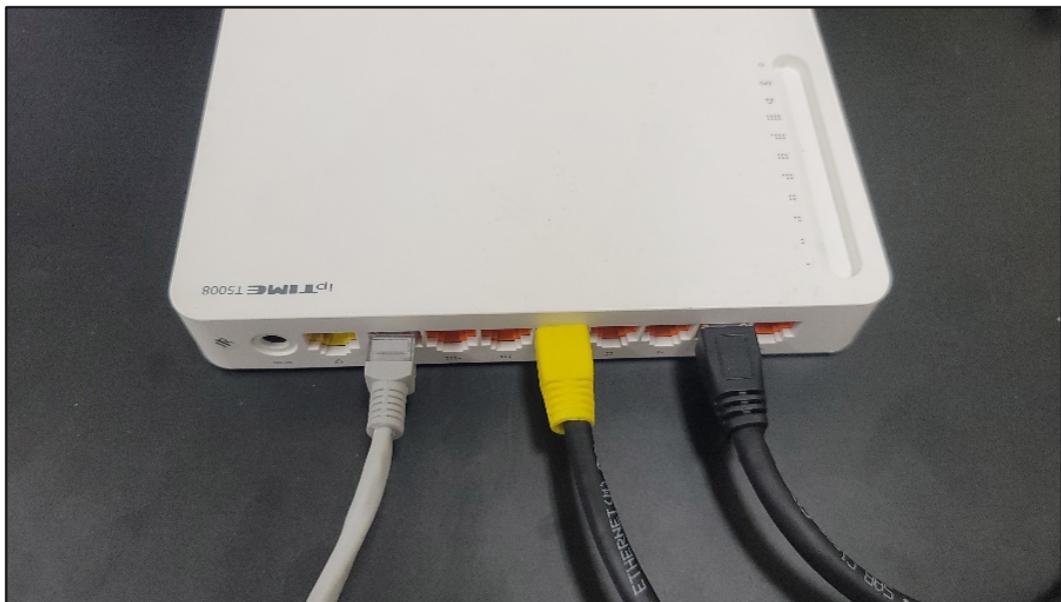


Figure 49: 2개의 ML 연결 예시

다중 ML LiDAR Data 가시화를 위해서 ML LiDAR의 Device Setting을 통해 ML LiDAR의 IP 및 PC PORT 값이 중복되지 않도록 다르게 변경합니다. 아래는 2개의 ML LiDAR Data를 가시화하기 위한 세팅 예시입니다.

- 1번 ML LiDAR Device Setting
  - **ML LiDAR IP** : 192.168.1.10
  - **PC Port** : 2000
- 2번 ML LiDAR Device Setting
  - **ML LiDAR IP** : 192.168.1.11
  - **PC Port** : 2001

```

1  bool success = true;
2  int num_lidar = 1;
3  std::vector<std::shared_ptr<SOSLAB::LidarMl>>
    ↵ lidar_ml_list(new SOSLAB::LidarMl);
4  int standard_port = 2000;
5  for (int i = 0; i < num_lidar; i++) {
6      SOSLAB::ip_settings_t ip_settings_pc;
7      SOSLAB::ip_settings_t ip_settings_device;
8      ip_settings_pc.ip_address = "192.168.1.15";
9      ip_settings_pc.port_number = standard_port + i;
10     ip_settings_device.ip_address = "192.168.1.1" +
11         ↵ to_string(i);
12     ip_settings_device.port_number = standard_port;
13     success = lidar_ml_list[i]->connect(ip_settings_device,
14         ↵ ip_settings_pc);
15     if (!success) {
16         std::cerr << "LiDAR ML #" + std::to_string(i) + "
17             ↵ ::connection failed" << std::endl;
18     }
19     return 0;
20 }
21 for (int i = 0; i < num_lidar; i++) {
22     success &= lidar_ml_list[i]->tcp_device_run();
23     if (!success) {
24         std::cerr << "LiDAR ML #" + std::to_string(i) + " ::"
25             ↵ start failed" << std::endl;
26     }
27 }
```

- **2 line** : ML LiDAR와 Local(PC)의의 통신을 위한 SOSLAB::LidarMl 객체를 연결할 ML LiDAR의 개수( num\_lidar)에 따라서 초기화 합니다.

- **4-15 line** : 예제 코드에서는 초기 값 ML IP(192.168.1.10), PC port(2000)에서 ML LiDAR 개수에 따라 ML IP, PC port의 마지막 값을 1씩 증가시켜 PC와 연결합니다.

**Example :** 2개의 ML 연결

1번 ML LiDAR IP = "192.168.1.10", PC port = "2000"

2번 ML LiDAR IP = "192.168.1.11", PC port = "2001"

- **16-18 line** : 모든 ML LiDAR를 연결한 뒤, ML LiDAR 개수에 맞게 tcp\_device\_run() 함수를 호출합니다.

Multiple ML LiDAR 예제에서는 Point Cloud를 가시화하기 위해 PCL과 VTK Library을 사용합니다. 아래 그림은 2개의 ML LiDAR Data를 가시화한 예시로써, 1번 ML LiDAR에서 획득한 Point Cloud를 흰색, 2번 ML LiDAR에서 획득한 Point Cloud를 빨간색으로 가시화한 결과입니다.

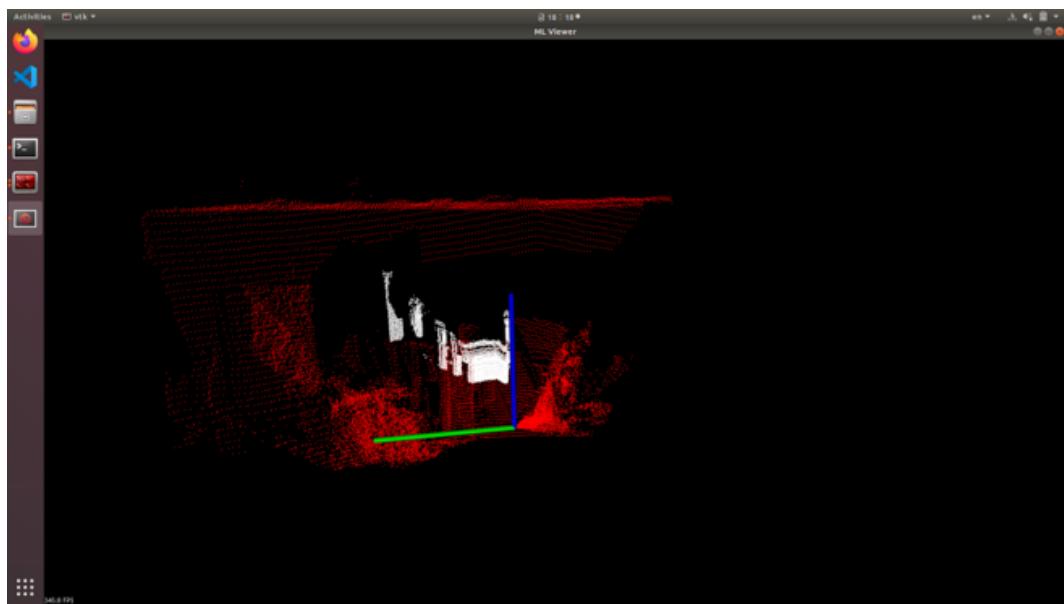


Figure 50: 다중 ML LiDAR 연결 및 가시화 예시

#### 4.1.2.7 ROS

아래의 코드는 Rviz에서의 가시화를 위해 ML LiDAR로부터 획득한 Ambient, Depth, Intensity 데이터를 변환하고 Publish하는 예제 코드입니다.

```
1 ros::NodeHandle nh("~");
2 image_transport::ImageTransport it(nh);
3 image_transport::Publisher pub_grey = it.advertise("grey", 1);
4 sensor_msgs::ImagePtr msg_grey;
5 cv::Mat grey_image_raw(height, width, CV_16UC1,
6   ↳ scene.grey_image.data());
7 cv::Mat grey_8b;
8 cv::Mat grey_rgb;
9 grey_image_raw.convertTo(grey_8b, CV_8UC1, 1.0/1.0);
10 grey_rgb = colormap(grey_8b);
11 if (pub_grey_color.getNumSubscribers() > 0) {
12   msg_grey = cv_bridge::CvImage(std_msgs::Header(), "rgb8",
13   ↳ grey_rgb).toImageMsg();
14   pub_grey_color.publish(msg_grey);
15 }
```

- **1-4 line** : Message를 보내는 Publisher Node를 생성하기 위해 ROS의 ImageTransport, ImagePtr 등의 객체를 생성합니다. Publisher는 grey\_color Topic으로 Ambient 데이터를 제공합니다.
- **5 line** : scene\_t 구조체의 Ambient 데이터를 가시화하기 위해 OpenCV의 cv::Mat 형식으로 변환하는 과정입니다. cv::Mat 초기화 입력 값으로 Raw Data의 Height, Width, Ambient 데이터 Type (CV\_16UC1)과 동일하게 입력하고, Raw Data 값을 4번째 인수로 입력합니다.
- **6-8 line** : 데이터 Type을 CV\_16UC1에서 CV\_8UC1으로 변경합니다.
- **9 line** : colormap 함수를 통하여 1 Channel에서 3 Channel 데이터로 변환합니다.
- **10-12 line** : OpenCV의 cv::Mat 객체를 Publisher Node의 Message 형태로 저장하기 위하여 ImagePtr로 변환하는 과정입니다. Topic이 grey\_color로 지정된 Publisher 객체의 publish 함수에 ImagePtr 변수를 입력 인수로 사용하여 Publish를 완료합니다.

아래 코드는 Rviz에서의 가시화를 위해 scene\_t의 Point Cloud를 Publish하는 예제 코드입니다.

```
1 typedef pcl::PointCloud<pcl::PointXYZRGB> PointCloud_T;
2 static const uint32_t DEFAULT_PUBLISHER_QUEUE_SIZE = 10;
3 static const char* DEFAULT_FRAME_ID = "map";
4 static const char* DEFAULT_PUBLISHER_TOPIC_NAME_LiDAR =
5     → "pointcloud";
6 std::string publisher_topic_name_lidar =
7     → DEFAULT_PUBLISHER_TOPIC_NAME_LiDAR;
8 ros::NodeHandle nh("~");
9 ros::Publisher pub_lidar =
10    → nh.advertise<PointCloud_T>(publisher_topic_name_lidar,
11    → uint32_t(DEFAULT_PUBLISHER_QUEUE_SIZE));
12 PointCloud_T::Ptr msg_pointcloud(new PointCloud_T);
13 msg_pointcloud->header.frame_id = DEFAULT_FRAME_ID
14 msg_pointcloud->width = scene.cols;
15 msg_pointcloud->height = scene.rows;
16 msg_pointcloud->points.resize(scene.pointcloud.size())
17 for (int col=0; col < width; col++) {
18     for (int row = 0; row < height; row++) {
19         int idx = col + (width * row);
20         // unit: m
21         msg_pointcloud->points[idx].x =
22             → scene.pointcloud[idx].xyz.x / 1000.0;
23         msg_pointcloud->points[idx].y =
24             → scene.pointcloud[idx].xyz.y / 1000.0;
25         msg_pointcloud->points[idx].z =
26             → scene.pointcloud[idx].xyz.z / 1000.0;
27         int intensity = int(intensity_norm_8b.at<uint8_t>(row,
28             → col));
29         msg_pointcloud->points[idx].r =
30             (uint8_t)(viridis_r[intensity] * 256.f);
31         msg_pointcloud->points[idx].g =
32             (uint8_t)(viridis_g[intensity] * 256.f);
33         msg_pointcloud->points[idx].b =
34             (uint8_t)(viridis_b[intensity] * 256.f);
35     }
36 }
37 pcl_conversions::toPCL(ros::Time::now(),
38     → msg_pointcloud->header.stamp);
39 pub_lidar.publish(msg_pointcloud);
```

- **1-7 line** : Message를 보내는 Publisher Node를 생성합니다. Publisher는 pointcloud라는 Topic으로 Point Cloud 데이터를 제공합니다.
- **8-12 line** : Message 변수를 정의하여 데이터 크기, 이름을 초기화합니다.
- **17-19 line** : ML LiDAR의 Point Cloud 3차원 좌표(x,y,z)를 msg\_pointcloud 변수에 복사합니다.
- **20-23 line** : ML LiDAR의 Intensity 데이터에 대응하는 Viridis color를 msg\_pointcloud 색상으로 복사합니다.
- **26-27 line** : Point Cloud가 Scanning된 Timestamp를 저장한 뒤, Topic0 pointcloud로 지정된 Publisher 객체의 publish 함수에 PointCloud\_T::Ptr 변수를 입력 인수로 사용하여 Publish를 완료합니다.

아래 Script는 ROS에서 ML LiDAR와 PC의 통신을 위한 예제입니다. Launch 파일에는 실행할 Package(ml)와 ML LiDAR 및 PC의 IP/Port 주소 정보가 포함되어 있습니다.

```

1 [Terminal #1]
2 $ cd ~/catkin_ws/src
3 $ catkin_make
4 $ source ~/catkin_ws/devel/setup.sh
5 $ cd ~/catkin_ws/src/ml/launch
6 $ roslaunch ml.launch
7 [Terminal #2]
8 $ rviz

```

- **1-3 line** : Terminal의 작업 Direcotry를 catkin\_ws/src 폴더로 변경한 뒤, catkin\_make를 입력하여 Build 과정을 수행합니다.
- **4 line** : Build된 Package(ml)를 등록합니다.
- **5-6 line** : Terminal의 작업 Direcotry를 launch 파일이 있는 폴더로 변경한 뒤, Launch 파일을 실행합니다.
- **8 line** : 새로운 Terminal에서 rviz script를 입력하여 Rviz 프로그램을 실행합니다.

아래 그림과 같이 Rviz가 실행되면 왼쪽 하단의 Add 버튼을 클릭합니다. 팝업 창의 상단 “By topic” Tab을 클릭하면 현재 Publish 되고 있는 전체 Topic을 확인할 수 있습니다.

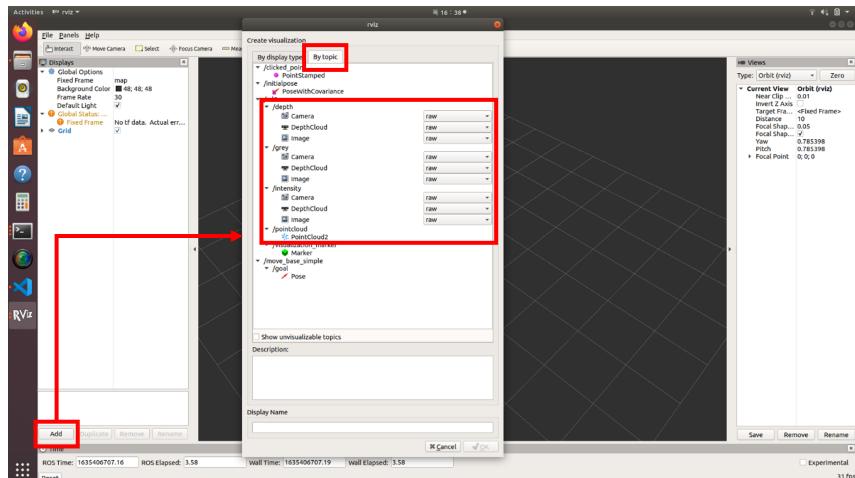


Figure 51: Rviz 실행 화면 및 Publish 된 Topic 확인

Ambient, Depth, Intensity 이미지는 각 Topic의 'Image'를 선택한 뒤 'OK' 버튼을 클릭하면 각 이미지들을 Rviz에서 가시화할 수 있습니다. Point Cloud는 '/point-cloud' Topic의 "PointCloud2"를 선택한 뒤 'OK' 버튼을 클릭면 Rviz에서 가시화할 수 있습니다. 아래 그림은 Ambient, Depth, Intensity Point Cloud 데이터를 가시화한 결과입니다.

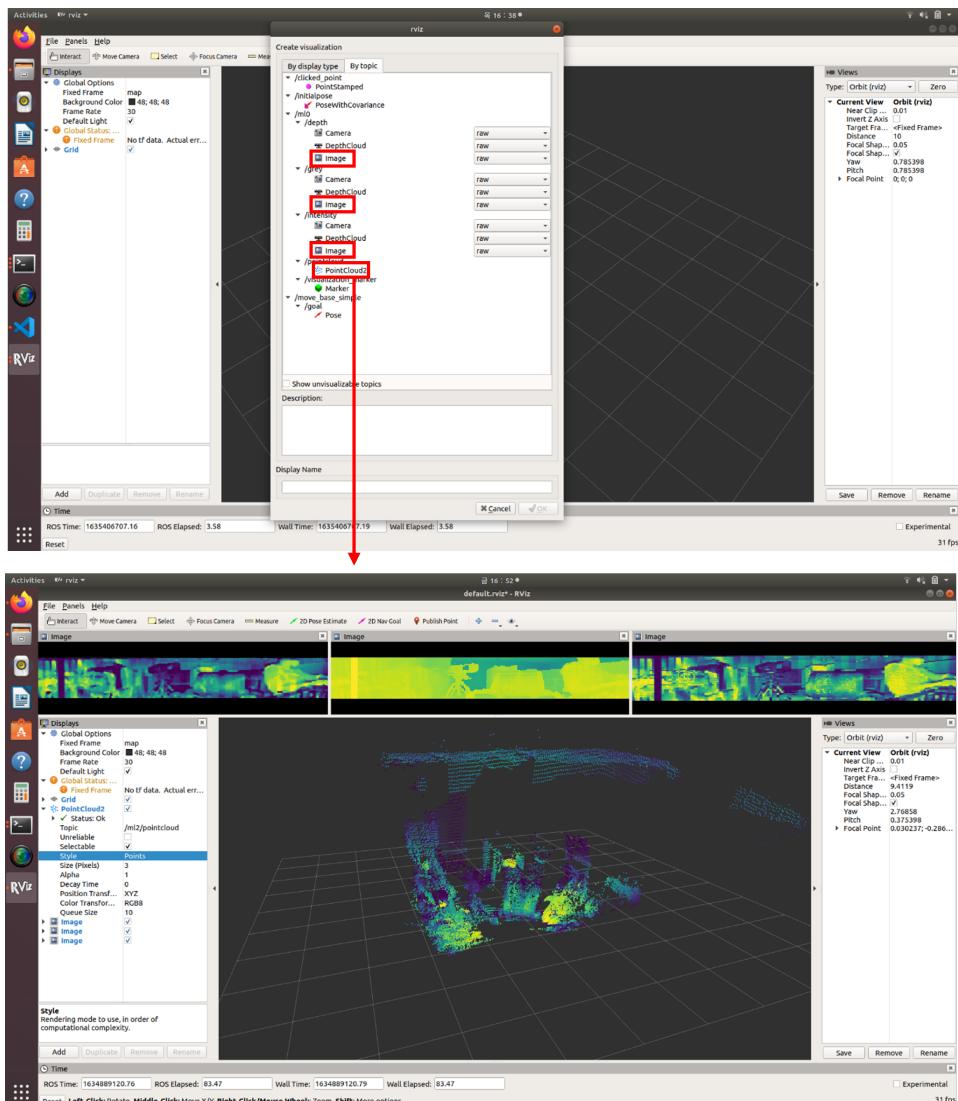


Figure 52: Rviz의 ML LiDAR Data 가시화 화면

## 5 Appendix

### 5.1 TCP Protocol Packet Structure

TCP 통신은 대화식 문자열 구조로 구성됩니다. PC에서 설정된 명령어를 전송하면 ML에서 응답하는 형태로 구성되어 있으며, ML에서 추가적인 설정이 필요할 경우, 질문 형태의 응답이 PC로 전송됩니다. 패킷 구조는 아래 Table과 같습니다.

TCP Protocol 패킷 구조					
Byte	Name	Sub Name	Type	Range	Description
0	DL	DL[0]	Number	0-65531	DATA 영역의 길이 (DL의 LSB)
1		DL[1]			DATA 영역의 길이 (DL의 MSB)
3	Main CMD				<b>PC → Device</b>
					S/N: System Mode Change to Normal, (DL == 0)
					S/P: System Mode Change to Production, (DL == 0)
4	CMD	SUB CMD	Character	ASCII 코드	S/R: System Mode Read, (DL==0)
					R/W: Register Write
					R/R: Register Read, (DL == 0)
					M/U: Memory Upload, (DL == 0)
					M/D: Memory Download, (DL == 0)
					M/V: Memory Verify,(DL == 0)
					F/U: Firmware Upload
					P/C: Production Mode Command
					D/R: Device Run (DL == 0)
					D/S: Device Stop (DL == 0)
					D/S: Device Stop (DL == 0)
					<b>Device → PC</b>
					E/P: Error Packet
					S/O: System Mode Change Success, (DL == 0)
					S/X: System Mode Change Fail
5-N	DATA	DATA	String or Binary	ASCII 코드 or Binary	S/N: System Mode is Normal
					S/P: System Mode is Production
					S/F: System Mode is Firmware Upload
					S/E: System Mode is Error
					R/O: Register Write Success, (DL == 0)
					R/X: Register Write Fail
					R/S: Register Send
					M/O: Memory Command Success, (DL == 0)
					M/X: Memory Command Fail
					P/R: Production Mode Report
					P/X: Production Mode Fail
					F/O: Firmware Upload Success, (DL == 0)
					F/A: Firmware Upload Packet Ack, (DL == 0)
					F/X: Firmware Upload Fail
					D/O: Device Command Success, (DL == 0)
					D/X: Device Command Fail
펌웨어를 제외한 모든 정보들은 문자열 형태로 전달					

## 5.2 UDP Protocol Packet Structure

UDP 통신의 기본 패킷 구조 아래 Table과 같습니다.

UDP Protocol 기본 패킷 구조					
Byte	Name	Type	Range	Description	
0~7	TS	UINT64	-	Timestamp. Unit: 10 [nsec]. Little Endian (Byte 0: Lowest Byte)	
8	PT	Character	ASCII코드	<b>Packet Type</b> 'R': Raw Data – Single Echo 'M': Raw Data – Multi Echo 'H': Histogram Data 'S': Sensing Data	
9	FC	UINT8	-	Frame Count	
10~11	WH	UINT16	-	Width × Height ([15:9] Height, [8:0] Width)	
12~13	CC	UINT16	-	Column Count (except "Sensing Data")	
14~15	DL	UINT16	-	Data Length	
16~N	DATA	-	-	It depends on "Packet Type"	

Raw Data 통신에 사용되는 패킷은 Single / Multi echo 모드로 구성됩니다.

### 5.2.1 Single Echo Mode Packet Structure

아래 Table은 UDP 통신에서 사용되는 Single echo 모드 패킷 구조입니다.

UDP Protocol 기본 패킷 구조 (Single Echo)					
Data Byte	Name	Sub Name	Type	Range	Description
0~7	TS	-	UINT64	-	Timestamp. Unit: 10 [nsec]. Little Endian (Byte 0: Lowest Byte)
8	PT	-	Character	'R' / 'r' - - - 700	Raw Data – Single Echo ('R':mm, 'r':bin) Frame Count Width × Height ([15:9]Height, [8:0] Width) Column ([15:9] Col. Info, [8:0] Col. Num.) Data Length (50 [CH]: 700)
16~19 20~23 24~25 26~29	ROW[0]	THETA PHI -	FLOAT FLOAT UINT16 DATA	-3.14 ~ 3.14 -3.14 ~ 3.14 - 0 ~ 262143	Polar Angle. Unit: Radian Azimuthal Angle. Unit: Radian Grey Image [31:18] Intensity [17:0] Radial Distance. Unit (mm / bin)
...	...	...	...	...	...
702~705 706~709 710~711 712~715	ROW[49]	THETA PHI -	FLOAT FLOAT UINT16 DATA	-3.14 ~ 3.14 -3.14 ~ 3.14 - 0 ~ 262143	Polar Angle. Unit: Radian Azimuthal Angle. Unit: Radian Grey Image [31:18] Intensity [17:0] Radial Distance. Unit (mm / bin)

### 5.2.2 Multi Echo Mode Packet Structure

아래 Table은 UDP 통신에서 사용되는 Multi echo 모드 패킷 구조입니다.

UDP Protocol 기본 패킷 구조 (Single Echo)					
Data Byte	Name	Sub Name	Type	Range	Description
0~7	TS	-	UINT64	-	Timestamp. Unit: 10 [nsec]. Little Endian (Byte 0: Lowest Byte)
8	PT	-	Character	'M' / 'm' - - - 900	Raw Data – Multi Echo ('M':mm, 'm':bin) Frame Count Width × Height ([15:9]Height, [8:0] Width) Column ([15:9] Col. Info, [8:0] Col. Number) Data Length (50[CH]: 900)
16~19	ROW[0]	THETA	FLOAT	-3.14 ~3.14	Polar Angle. Unit: Radian
20~23		PHI	FLOAT	-3.14 ~3.14	Azimuthal Angle. Unit: Radian
24~25		-	UINT16	-	Grey Image
26~29		DATA0	UINT32	0 ~262143	[31:18] Intensity [17:0] Radial Distance. Unit (mm / bin)
30~33		DATA1	UINT32	0 ~262143	[31:18] Intensity [17:0] Radial Distance. Unit (mm / bin)
...	...	...	...	...	...
898~901	ROW[49]	THETA	FLOAT	-3.14 ~3.14	Polar Angle. Unit: Radian
902~905		PHI	FLOAT	-3.14 ~3.14	Azimuthal Angle. Unit: Radian
906~909		-	UINT16	-	Grey Image
910~911		DATA0	UINT32	0 ~262143	[31:18] Intensity [17:0] Radial Distance. Unit (mm / bin)
912~915		DATA1	UINT32	0 ~262143	[31:18] Intensity [17:0] Radial Distance. Unit (mm / bin)

### 5.3 API - Typedefs

이름	SOSLAB::INT_T
선언	<b>typedef int32_t</b> SOSLAB::INT_T
<b>Header</b>	#include "soslab_typedef.h"
설명	integer 변수

이름	SOSLAB::UINT_T
선언	<b>typedef uint32_t</b> SOSLAB::UINT_T
<b>Header</b>	#include "soslab_typedef.h"
설명	unsigned int 변수형

이름	SOSLAB::FLOAT_T
선언	<b>typedef double</b> SOSLAB::FLOAT_T
<b>Header</b>	#include "soslab_typedef.h"
설명	double 변수형

이름	SOSLAB::hex_array_t
선언	<b>typedef std::vector&lt;uint8_t&gt;</b> SOSLAB::hex_array_t
<b>Header</b>	#include "soslab_typedef.h"
설명	16진수 배열 변수형

이름	SOSLAB::pointcloud_t
선언	<b>typedef std::vector&lt;point_t&gt;</b> SOSLAB::pointcloud_t
<b>Header</b>	#include "soslab_typedef.h"
설명	SOS LAB LiDAR Point Cloud 변수형

## 5.4 API - Structure / Class

### - Spherical/Cartesian Coordinate System Structure

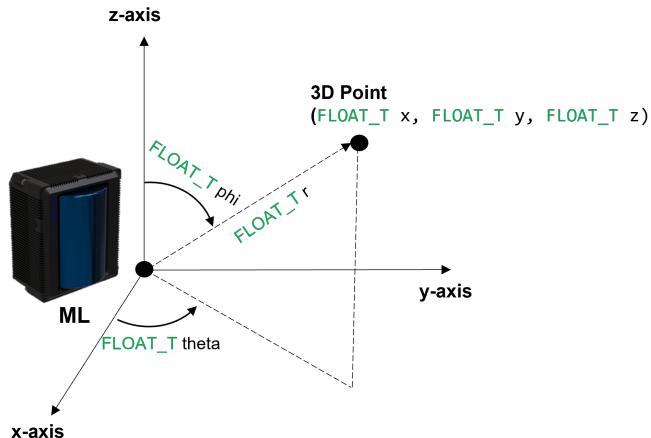


Figure 53: ML LiDAR Coordinate System

변수형	SOSLAB::spherical_point_t
선언	<b>typedef struct _SPHERICAL_POINT_T</b> spherical_point_t
<b>Header</b>	#include "soslab_typedef.h"
설명	Spherical Coordinate System 구조체
<b>Variables</b>	<b>Variables</b> 설명
FLOAT_T r	Spherical Coordinate System의 Range 값
FLOAT_T theta	Spherical Coordinate System의 Theta 값
FLOAT_T phi	Spherical Coordinate System의 Phi 값

변수형	SOSLAB::cartesian_point_t
선언	<b>typedef struct _CARTESIAN_POINT_T</b> cartesian_point_t
<b>Header</b>	#include "soslab_typedef.h"
설명	Cartesian Coordinate System 구조체
<b>Variables</b>	<b>Variables</b> 설명
FLOAT_T x	Cartesian Coordinate System의 x 값
FLOAT_T y	Cartesian Coordinate System의 y 값
FLOAT_T z	Cartesian Coordinate System의 z 값

### - Coordinate System/IP Setting Structure

변수형	SOSLAB::point_t
선언	<b>typedef struct _POINT_T</b> point_t
<b>Header</b>	#include "soslab_typedef.h"
설명	Coordinate system 구조체
<b>Variables</b>	<b>Variables</b> 설명
std::size_t index	Point Index
std::size_t timestamp_1us	Timestamp [1 ms 단위]
spherical_point_t rtp	Point의 Spherical Coordinate System 정보
cartesian_point_t xyz	Point의 Cartesian Coordinate System 정보
FLOAT_T pw	Point에 대응하는 Intensity 값

변수형	SOSLAB::ip_setting_t
선언	<b>typedef struct IP_ADDRESS_T</b> ip_setting_t
<b>Header</b>	#include "soslab_typedef.h"
설명	IP 및 Port 정보를 저장하는 구조체
<b>Variables</b>	<b>Variables</b> 설명
std::string ip_address	IP 주소
int port_number	Port 번호

## - ML Data Structure

변수형	SOSLAB::LidarMl::scene_t
선언	<b>typedef struct _ML1_SCENE_T point_t</b>
<b>Header</b>	#include "soslab_typedef.h"
설명	ML Data 구조체
<b>Variables</b>	<b>Variables</b> 설명
std::vector< <b>uint64_t</b> > timestamp	Timestamp [1 ms 단위]
<b>uint8_t</b> frame_id	Raw Sequence Data의 Index 값
std::size_t rows	Raw Data의 Height 값
std::size_t cols	Raw Data의 Width 값
FLOAT_T pw	Point에 대응하는 Ambient 값
std::vector< <b>uint16_t</b> > grey_image	Ambient 데이터 배열
std::vector< <b>uint32_t</b> > depth_image	Depth 데이터 배열
std::vector< <b>uint16_t</b> > intensity_image	Intensity 데이터 배열
std::vector<point_t> pointcloud	Point cloud 데이터 배열
std::vector< <b>uint32_t</b> > depth_image2	Multi echo에서 측정되는 Depth 데이터 배열
std::vector< <b>uint16_t</b> > intensity_image2	Multi echo에서 측정되는 Intensity 데이터 배열
std::vector<point_t> pointcloud2	Multi echo에서 측정되는 Point Cloud 데이터 배열

### - ML LiDAR Class : Connection

변수형	SOSLAB::LidarML
선언	<b>class LidarML</b>
<b>Header</b>	#include "ml/libssoslab_ml.h"
설명	ML LiDAR와 연결, 데이터 획득 등을 관리하는 Functions
<b>Functions</b>	
<b>bool connect(const ip_settings_t ml, const ip_settings_t local);</b>	
- 설명: Local(PC)에서 ML LiDAR로 접속하는 함수	
- Input: ML IP 주소와 Port 번호와 Local IP 주소와 Port 번호	
- Output: 접속 상태에 대해 <b>bool</b> 변수형으로 반환(접속되면 true 반환)	
<b>bool disconnect();</b>	
- 설명: Local(PC)에서 ML LiDAR로 접속을 해제하는 함수	
- Output: 접속 해제 상태에 대해 <b>bool</b> 변수형으로 반환(해제되면 true 반환)	
<b>bool start();</b>	
- 설명: ML LiDAR 시작하는 함수	
- Output: 시작 유무에 대해 <b>bool</b> 변수형으로 반환(시작되면 true 반환)	
<b>bool tcp_device_run();</b>	
- 설명: ML LiDAR를 시작하는 함수	
- Output: 시작 유무에 대해 <b>bool</b> 변수형으로 반환(시작되면 true 반환)	
<b>bool stop();</b>	
- 설명: ML LiDAR를 중단하는 함수	
- Output: 중단 유무에 대해 <b>bool</b> 변수형으로 반환(중단되면 true 반환)	
<b>bool tcp_device_stop();</b>	
- 설명: ML LiDAR를 중단하는 함수	
- Output: 중단 유무에 대해 <b>bool</b> 변수형으로 반환(중단되면 true 반환)	
<b>bool get_scene(scene_t&amp; scene);</b>	
- 설명: 입력 변수 <b>scene</b> 으로 Raw Data를 획득하는 함수	
- Input: <b>scene_t</b> 변수형	
- Output: ML Raw Data를 저장한 <b>scene_t</b> 변수	

## - ML LiDAR Class : Data Processing

변수형	SOSLAB::LidarML
선언	<code>class LidarML</code>
Header	<code>#include "ml/libssoslab_ml.h"</code>
설명	Data Processing 관련 Functions
<b>Functions</b>	
<code>void set_zigzag_filling_enable(bool enable);</code>	- 설명: Data Processing - Zigzag Filling 함수 - Input: <code>bool</code> 변수형: On(true)/Off(false)
<code>void set_deflaring_enable(bool enable);</code>	- 설명: Data Processing - Deflaring 함수 - Input: <code>bool</code> 변수형: On(true)/Off(false)
<code>void set_real_ambient_enable(bool enable);</code>	- 설명: Data Processing - Real Ambient 함수 - Input: <code>bool</code> 변수형: On(true)/Off(false)
<code>void set_line_filling_enable(bool enable);</code>	- 설명: Data Processing - Line Filling 함수 - Input: <code>bool</code> 변수형: On(true)/Off(false)
<code>void set_deflaring_retro_threshold(int val);</code>	- 설명: Data Processing - Deflaring 함수의 retro_threshold Parameter 설정 함수 - Input: <code>int</code> 변수형
<code>void set_deflaring_depth_diff(int val);</code>	- 설명: Data Processing - Deflaring 함수의 depth_diff Parameter 세팅 함수 - Input: <code>int</code> 변수형
<code>void set_zigzag_depth_diff(int val);</code>	- 설명: Data Processing - Zigzag Filling 함수의 depth_diff Parameter 세팅 함수 - Input: <code>int</code> 변수형
<code>void set_zigzag_smoothing_factor(int val);</code>	- 설명: Data Processing - Zigzag Filling 함수의 smoothing_factor Parameter 세팅 함수 - Input: <code>int</code> 변수형
<code>void set_line_filling_left_offset(int val);</code>	- 설명: Data Processing - Line Filling 함수의 left_offset Parameter 세팅 함수 - Input: <code>int</code> 변수형
<code>void set_line_filling_right_offset(int val);</code>	- 설명: Data Processing - Line Filling 함수의 right_offset Parameter 세팅 함수 - Input: <code>int</code> 변수형

### - ML LiDAR Class : Data Record

변수형	SOSLAB::LidarML
선언	<code>class LidarML</code>
Header	<code>#include "ml/libssoslab_ml.h"</code>
설명	ML LiDAR Data 녹화 관련 Functions
<b>Functions</b>	
	<code>void start_recording(std::string dir_path);</code>
- 설명	: ML LiDAR 데이터 녹화 기능을 시작하는 함수
- Input	: 녹화 파일 저장 경로
	<code>void stop_recording();</code>
- 설명	: ML LiDAR 데이터 녹화 기능을 정지하는 함수
	<code>void connect_file(const std::string filepath);</code>
- 설명	: ML LiDAR 데이터 녹화 파일을 Load 하는 함수
- Input	: 녹화 파일 저장 경로
	<code>void binary2file(std::string save_directory);</code>
- 설명	: ML LiDAR 데이터 녹화 파일을 pcd 파일로 변환하는 함수
- Input	: pcd 파일 저장 경로
	<code>bool get_scene(scene_t&amp; scene, uint64_t index);</code>
- 설명	: 입력 Scene과 Frame Index를 통해 해당 Frame의 Data를 획득하는 함수
- Input	: 입력 Scene (scene_t& scene), Frame Index ( <code>uint64_t index</code> )
- Output	: ML LiDAR 데이터의 scene_t 변수
	<code>void get_logging_file_size(uint64_t&amp; size);</code>
- 설명	: 총 Frame 수를 반환하는 함수
- Input	: <code>uint64_t</code> 변수를 입력으로 받아 전체 Frame 수를 반환