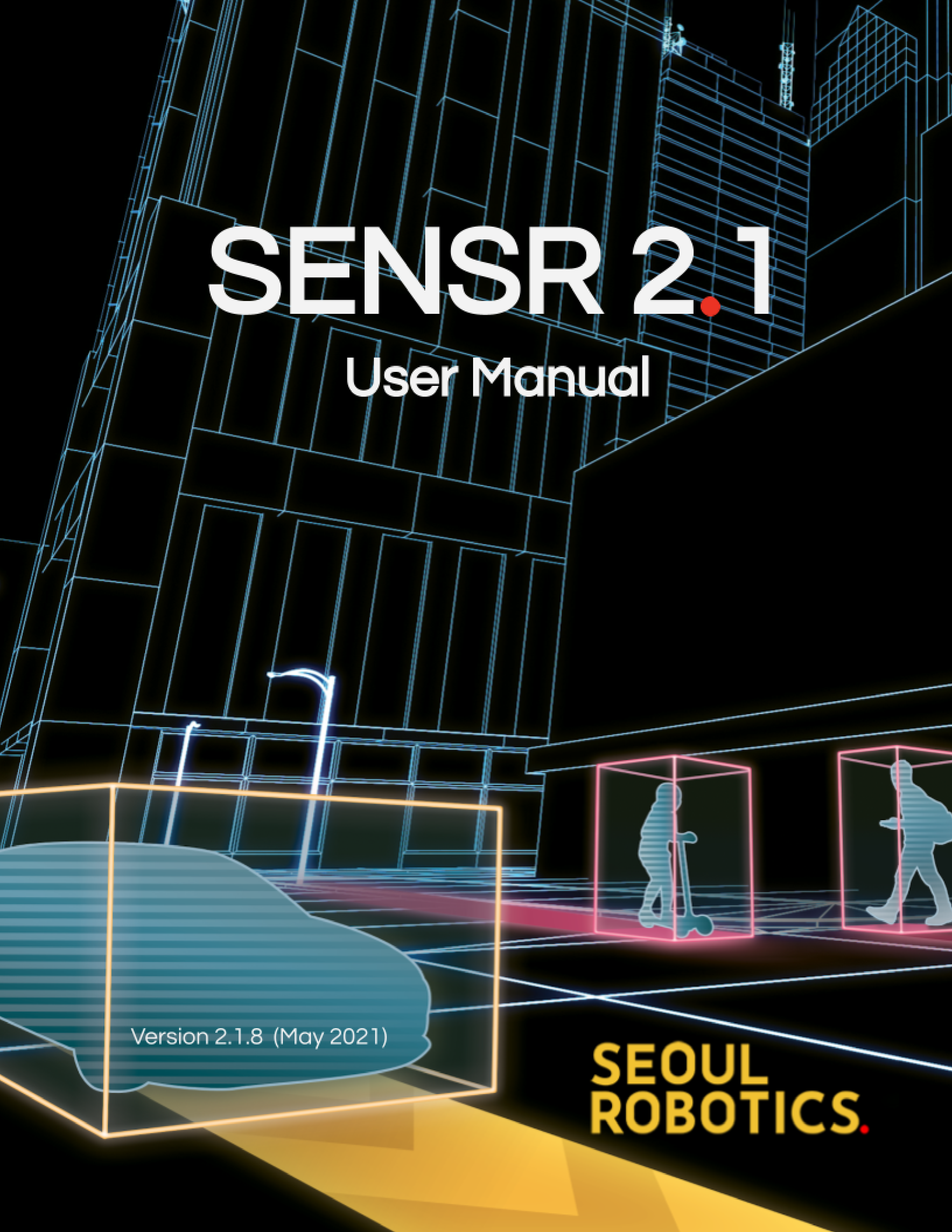


SENSR 2.1

User Manual

Version 2.1.8 (May 2021)

SEOUL ROBOTICS.

Contents

1. Introduction

SENSR 3D Tracking Software is an all-in-one object tracking solution that simplifies the 3D sensor experience and lowers the barrier for companies wanting to implement insights from 3D data into the systems and processes.

Its patented 3D computer vision system utilizes Machine Learning to allow for live detection, tracking and classification of objects in true 3D space.

SENSR's open platform architecture supports over 70 types of 3D sensors and is part of a quickly expanding ecosystem of sensor, analytics and integration partners, ready to be used in traffic safety, retail analytics, smart cities and many other markets.
It is the first industrial-grade software solution that provides scalable 3D perception for companies that seek to efficiently commercialize LiDAR solutions across industries.

SENSR helps you deploy LiDAR of your choice with the latest 3D AI, and helps you integrate multiple brands of LiDARs with ease. SENSR detects, classifies, tracks, and predicts objects and other real-time segmentation from raw 3D points, on the computer of your choice.

SENSR is a window to your LiDAR

2. Software Installation

2.1 System Requirements

If you ordered one of the Seoul Robotics SENSR Discovery kits, congratulations, your system is pre-installed, pre-configured and ready to be used. You can jump straight to chapter 4, the sensor setup.

If you intend to install the software on your own provided hardware, a couple of reference examples of minimum hardware requirements are as follows:

Single Sensor, producing < 400,000 points per second

- Intel Celeron J1900, Quad-core, 2.0GHz
- 8GB Memory
- 32GB HDD (preferably solid state)
- 2x 1Gbps Ethernet
- Ubuntu 18.05.4 LTS

Single or multiple sensors, producing < 3,000,000 points per second total

- NVidia Xavier NX, 6-core Carmel, ARM v8.2
- 8GB Memory
- 128GB M.2 2280 SSD
- Jetpack 4.3 and above

Up to 8 sensors, each producing < 350,000 points per second, or 6,000,000 total

- Intel 8th+ Gen i5 (i7 preferred) processor, >2.0Ghz
- 8GB Memory
- 64GB HDD (preferably solid state)
- 2x 1Gbps Ethernet (6x preferred)
- Ubuntu 18.04.5 LTS

Above 8 sensors

Please contact Seoul Robotics for recommendations.

The exact hardware requirements for optimal performance are dependent on the number of connected sensors, the amount of data produced by each sensor, the total area coverage and the maximum number of simultaneous tracked objects in the scene. The above are our minimum recommendations.

Operating System Support

SENSR will run on the following Operating Systems

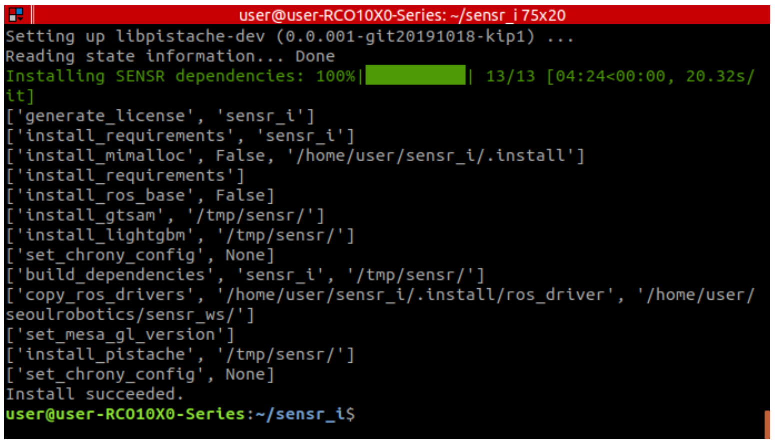
1. Ubuntu 18.04.5 LTS
2. Jetpack 4.3 and above for NVidia Xavier NX

2.2 Software Installation Process

If you did not purchase a pre-installed kit, you will have received the software through a downloadable link. For a fresh install do the following steps:

Note: Make sure you have an active internet connection during the installation process. The software automatically downloads the latest sets of supporting software and tools during the installation.

1. Copy the .zip or .tar file into a folder on the target computer, for example the 'Downloads' folder
2. Extract the file
3. The extracted file will have created a folder called 'sensr_2.x.x' with the version number of the software. Enter this folder.
4. You will see the main 'sensr' folder. This is the main folder for the SENSR software. Copy or move this to the final destination. The recommended location is in the home directory.
5. Open a terminal window (CTRL-ALT-T)
    a. Navigate to the 'sensr' folder. If you placed it in home, type '`cd sensr`'
    b. Run the install script by typing '`./install.sh`'
    c. You will be asked for the password. This is the standard password for the computer login.
    d. When the install is complete, you should see the following in the terminal:



```
user@user-RCO10X0-Series: ~/sensr_i 75x20
Setting up libpistache-dev (0.0.001-git20191018-kip1) ...
Reading state information... Done
Installing SENSR dependencies: 100%|██████████| 13/13 [04:24<00:00, 20.32s/
it]
['generate_license', 'sensr_i']
['install_requirements', 'sensr_i']
['install_mimalloc', False, '/home/user/sensr_i/.install']
['install_requirements']
['install_ros_base', False]
['install_gtsam', '/tmp/sensr/']
['install_lightgbm', '/tmp/sensr/']
['set_chrony_config', None]
['build_dependencies', 'sensr_i', '/tmp/sensr/']
['copy_ros_drivers', '/home/user/sensr_i/.install/ros_driver', '/home/user/
seoulrobotics/sensr_ws/']
['set_mesa_gl_version']
['install_pistache', '/tmp/sensr/']
['set_chrony_config', None]
Install succeeded.
user@user-RC010X0-Series:~/sensr_i$
```

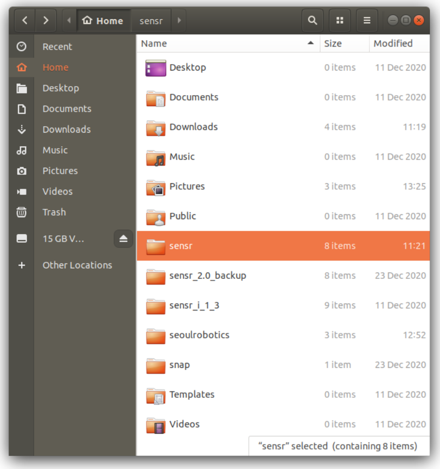
The final recommended step is to reboot the computer and continue to Chapter 3, License Activation.

2.3 Software Upgrade Process

If you already have a version of SENSR installed on the computer and you received an update, you can do the following:

1. It is strongly recommended to make a backup of your existing version by moving or renaming the main installation folder. For example, you can rename 'sensr' to 'sensr_backup'.
2. Follow steps 1 to 4 of the installation process above.
3. Unless specifically mentioned in the release notes, you should **NOT** need to run the install.sh script. If unsure, it is safe to run the script.
4. Navigate to your backup folder and copy the license file (.lic extension) to your new installation. You can now skip the License Activation step in Chapter 3.

For example:
This installation contains the latest version in the 'sensr' folder and has moved the previous version to the 'sensr_2.0_backup' folder.

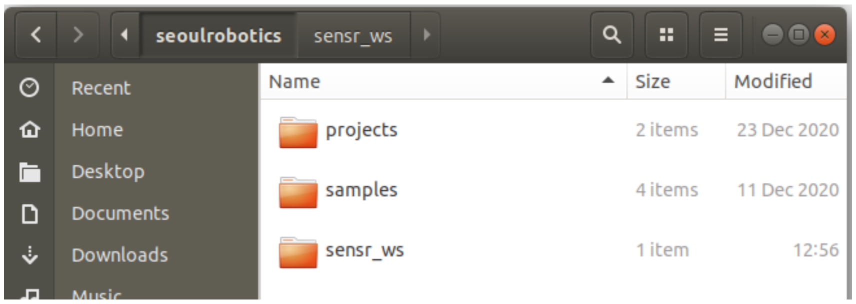


Name	Size	Modified
Desktop	0 items	11 Dec 2020
Documents	0 items	11 Dec 2020
Downloads	4 items	11:19
Music	0 items	11 Dec 2020
Pictures	3 items	13:25
Public	0 items	11 Dec 2020
sensr	8 items	11:21
sensr_2.0_backup	8 items	23 Dec 2020
sensr_i_1_3	9 items	11 Dec 2020
seoulrobotics	3 items	12:52
snap	1 item	23 Dec 2020
Templates	0 items	11 Dec 2020
Videos	0 items	11 Dec 2020

"sensr" selected (containing 8 items)

2.4 Verifying the installation

The installation script will create a folder called 'seoulrobotics' in the home directory. Inside that folder there should be three folders.



Name	Size	Modified
projects	2 items	23 Dec 2020
samples	4 items	11 Dec 2020
sensr_ws	1 item	12:56

Projects

This is where all project settings and project data replays are stored. If you want to copy an existing project configuration to a new computer, you can copy the contents of this folder to do so.

Samples

This folder provides one or more data sample files to work and test with. The calibration example in the calibration chapter of this manual will use these files so you can follow along with the exact same steps and data.

Sensr_ws

This folder contains the SENSR shared workspace and sensor drivers. Please do not modify or delete anything in this folder as its contents are critical to the system.

2.5 Advanced Installations, Remote Installations

SENSR is designed to be remotely accessible. The software consists of two components: the **algo node** and the **master**.

It is strongly recommended to start with a single computer installation before attempting to install a remote installation and distributed systems.

2.5.1 Algo Nodes

The **Algo Node**, short for 'algorithm node' is the process where all the heavy number crunching takes place. Sensors connect to algo nodes which process this information and send this processed information to the **Master**. An algo node is able to reduce the raw data bandwidth coming from the sensors by over 95%. An algo node does not have a visual interface and can be configured to start automatically when the computer is powered.

2.5.2 Master Server

The **Master Server** performs several functions.

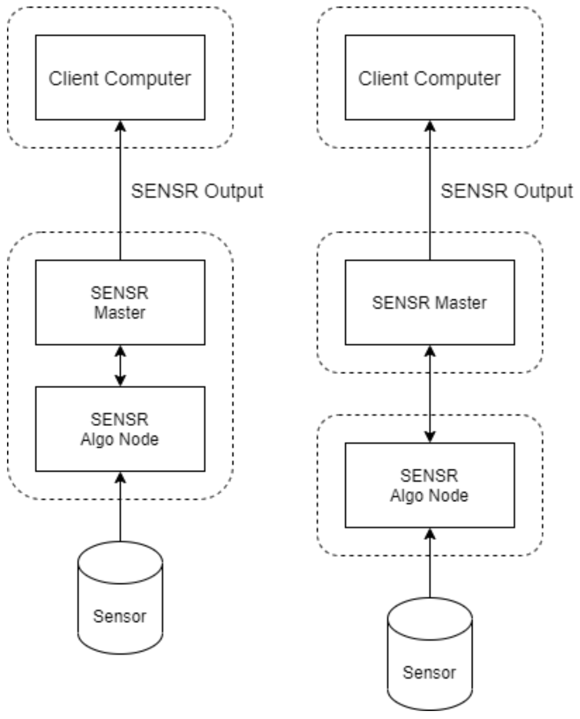
1. It receives and processes information from the algo node.
2. It manages the Graphical User Interface, which contains all tools needed to control the system such as project management, sensor setup and calibration.
3. It serves as the single point of contact for data and interaction for client systems.

2.6 Setting up a remote Algo Node

It is possible to install an Algo Node on a different computer than the Master Server. There are a couple of reasons why this is useful.

First, this allows for remote access to multiple sites from a single operations computer. Each site can be managed as an independent project (see Projects chapter)

Second, this allows for a large site to be spread across multiple computers to balance the load of processing many Lidar sensors In this instance a single Master is connected to multiple Algo Nodes simultaneously.



2.6.1 Installing a dedicated Algo node

If you intend for a computer to be run as Algo Node only, which means it will not ever directly be used to visualize the point cloud, you can install the SENSR software with a special flag

- Type: `./install.sh --mode=algo`

Note that this additional parameter is not required. It is possible for a regular installation to be used as a remote Algo Node

2.6.2 Algo Node Authentication

Before the Master and the Algo Node can communicate, the Master will have to register with each algo node. To do this, an SSH based authentication must be established.

Before you attempt to establish the SSH authentication, first make sure that the network configuration of the Master and Algo Nodes is set up so the computers can ping each other. This means that both need to be on the same network and be part of the same subnet.

Create the SSH key on the Master (done once)

- Open a terminal (CTRL-ALT-T shortcut)
- Type: `ssh-keygen`
- Use the default location and just hit enter when asked for a pass-phrase

If the prompt asks you to overwrite a previous SSH key, you must answer 'no'. Otherwise any previously set up communication channels may be lost.

Copy the SSH key to the Algo node (done once for each Algo Node)

- Find the IP address and the user login of the Algo Node. In this example we will use 10.0.0.240 and the user login is 'discovery'
- Type: `ssh-copy-id discovery@10.0.0.240`

It may show a message like this:

```
The authenticity of host '10.0.0.240 (10.0.0.240)' can't be established.
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
Are you sure you want to continue connecting (yes/no)? yes
```

- Type: `yes`

Next it will ask you for the password of the login of the Algo Node

- Type in the password of the login of the Algo Node

You should now see a message like this:

```
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh discovery@10.0.0.240'"
and check to make sure that only the key(s) you wanted were added.
```

It is recommended to try to log in with the given instructions. If the login succeeds, the connection and authentication is set up correctly.

If you run into any issues, a in-depth explanation of how to configure SSH authentication can be found here at this link: https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server

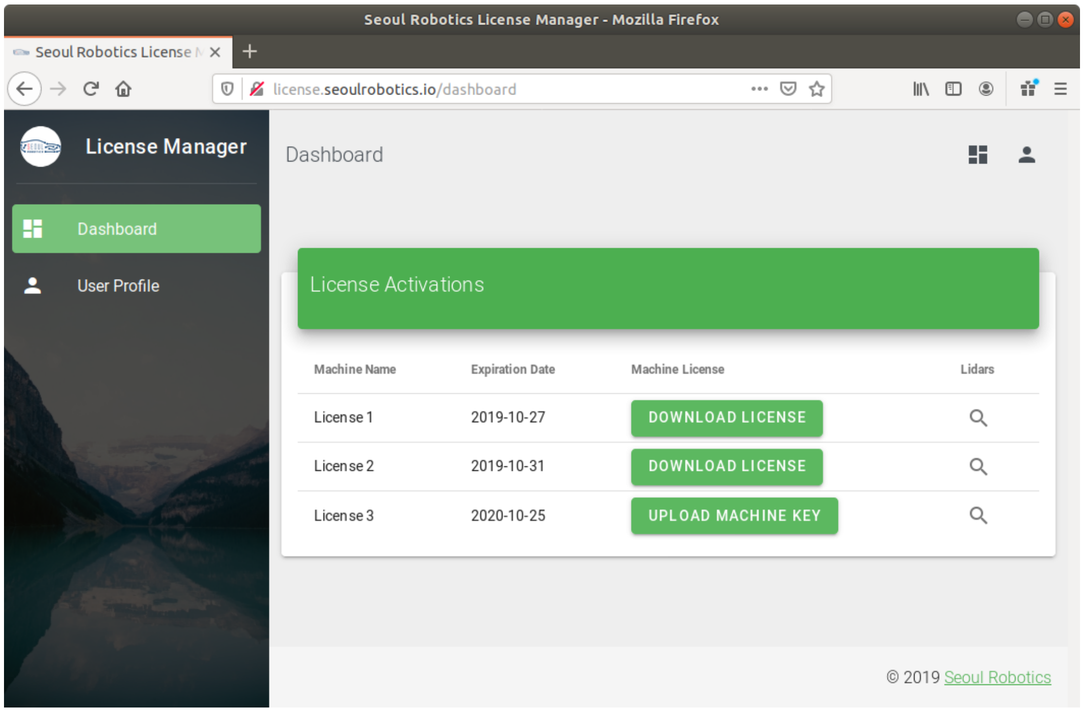
Set up the remote algo node in SENSR

Follow instructions in chapter 6.3.3 for the remote algo node connection.

3. License Activation

1. Find generated **license_key.json** file in SENSR folder
2. Go to Seoul Robotics License Portal here: http://license.seoulrobotics.io/
3. Sign in to your account

   *** NOTE**: You have to set a new password when you sign-in for the first time.
4. Go to **Dashboard** and check available license



Machine Name	Expiration Date	Machine License	Lidars
License 1	2019-10-27	DOWNLOAD LICENSE	🔍
License 2	2019-10-31	DOWNLOAD LICENSE	🔍
License 3	2020-10-25	UPLOAD MACHINE KEY	🔍

5. Click the **UPLOAD MACHINE KEY** button of the license you want to activate and upload the **license_key.json** file that was generated during installation.

   *** NOTE**: You can register only one machine to your license.
6. It is recommended to change the license name to something recognizable. To do this, click on the license name and enter a new name.
7. When the **UPLOAD MACHINE KEY** button changes to **DOWNLOAD LICENSE**, click the button and download your license file (.lic) to your SENSR folder.

   *** NOTE**: There is a known issue of upload machine key failure with some Firefox versions. If your **UPLOAD MACHINE KEY** doesn't work, use other browsers such as Google Chrome or update your Firefox (run `sudo apt update && sudo apt install firefox` in your terminal)

8. Make sure the license file (.lic) is in the same folder as the license_key.json and the SENSR.sh file.

Your license is tied to your hardware. It is not possible to transfer a license from one machine to another. This is also the case for some Virtual Machine environments.

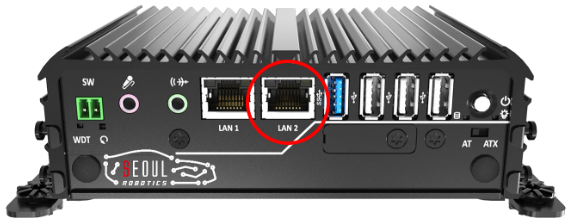
If you need to transfer SENSR to a different machine, please contact us at support@seoulrobotics.org or contact your account manager directly and we can reset your license.

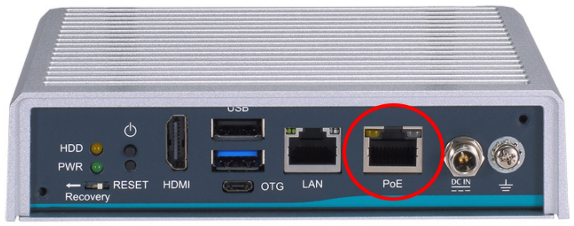
4. LiDAR Connection Setup

There are many ways to connect a LiDAR to a computer that runs SENSR software. Please consult the sensor manual for instructions on how to prepare your type of sensor for connection.

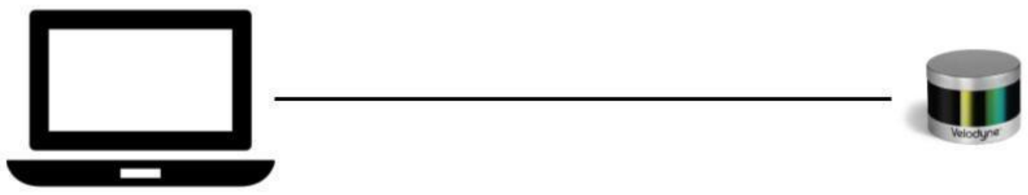
4.1 Discovery Kits

If you received one of our Discovery kits, which includes a LiDAR and a LiDAR Processing Unit (LPU), the LPU will be pre-configured for that specific LiDAR brand and model. You can plug the LiDAR directly into the second ethernet port, marked with either 'LAN 2' or 'PoE' depending on the model of the LPU.





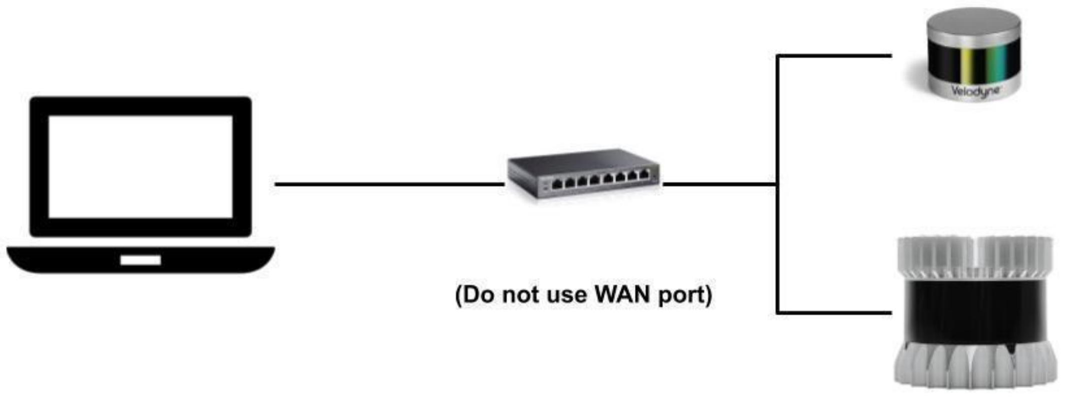
4.2 LiDAR Connection Without a Router



If you have your own computer and a single LiDAR, you can, in most cases, connect the LiDAR directly to your computer. Consult the LiDAR manual for connection settings and how to configure the network.

4.3 LiDAR Connection with a Router

If you have more sensors than available network ports on your computer, you may need a router to connect to the sensors. Plug one port into your computer and plug the sensors in the other available ports. Do not plug into any port marked 'Uplink' or 'WAN'.



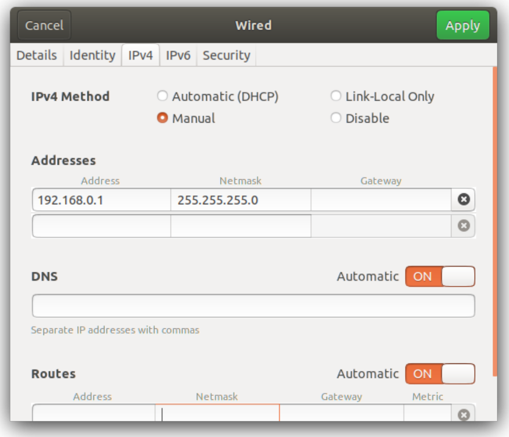
Some sensors are delivered with a static IP address, some with DHCP enabled and some require a DNS server to be active. Refer to your sensor manual on how to connect your sensor.

In case of Static IP support:

Make sure that the IP address of each device on the network is different. If you have received multiple sensors with the same default factory IP address, connect them one at a time and assign new, unique addresses to each of your sensors. Finally, set the IP address of your computer to a unique address.

For example:

Computer: IP 192.168.1.1, Netmask 255.255.255.0
Sensor A: IP 192.168.1.10, Netmask 255.255.255.0
Sensor B: IP 192.168.1.11, Netmask 255.255.255.0
And so on.



IPv4 Method: Automatic (DHCP) [✓ Manual] Link-Local Only Disable

Addresses

Address	Netmask	Gateway
192.168.0.1	255.255.255.0	

DNS — Automatic ON

Separate IP addresses with commas

Routes — Automatic ON

Address	Netmask	Gateway	Metric

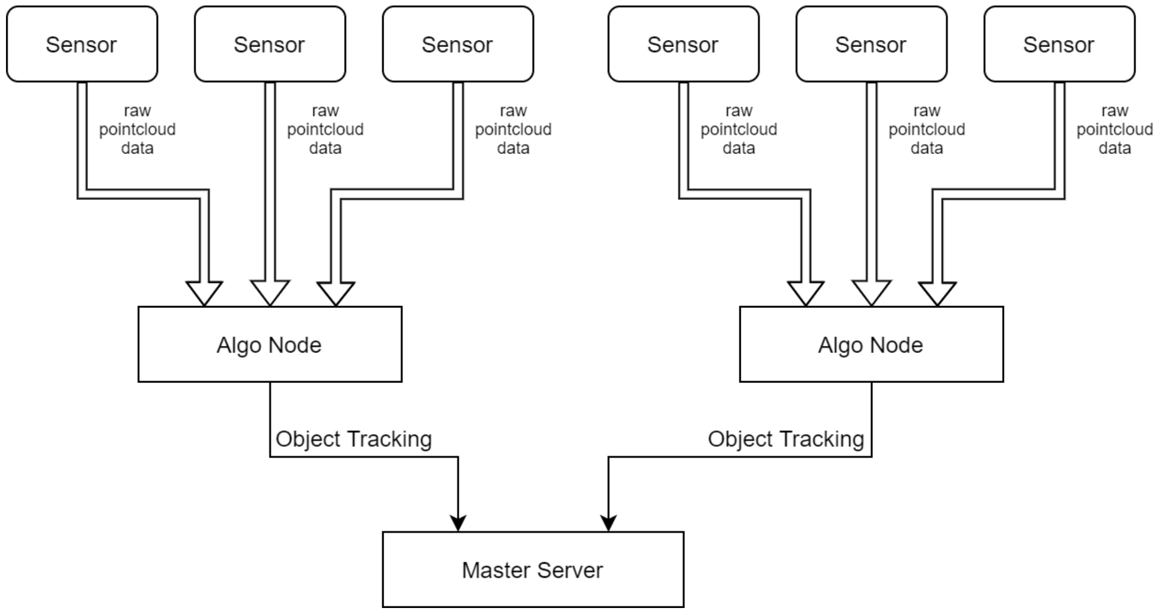
You can use the Wireshark tool (**$ sudo apt-get install wireshark**) for checking LiDAR data packets.

4.4 Larger installations

In larger installations you may need multiple routers to handle the connections to all the sensors.

With SENSR 2.1, Seoul Robotics introduced distributed computing. This makes it possible to connect a large number of sensors and distribute the load of processing the data over multiple computers. Each of these computers is called an '*Algo Node*' (See chapter 2.5).

In addition to processing the raw data from the sensors, these Nodes reduce the network bandwidth by up to 95%. These Nodes report their processed data to a single computer called the '*Master Server*', which is where the system is configured and monitored and from which the output data is streamed.



Be aware that LiDAR sensors generate large amounts of data. The bandwidth used by a 32-beam sensor can be up to 50Mbps. Consult your LiDAR manual for the bandwidth for your units. As a general rule of thumb, a standard 1 Gbps network should not be loaded above half of its capacity to ensure that data is not delayed. This means that a router that is rated for 1 Gbps cannot handle more than 10 standard 32-beam LiDAR units.

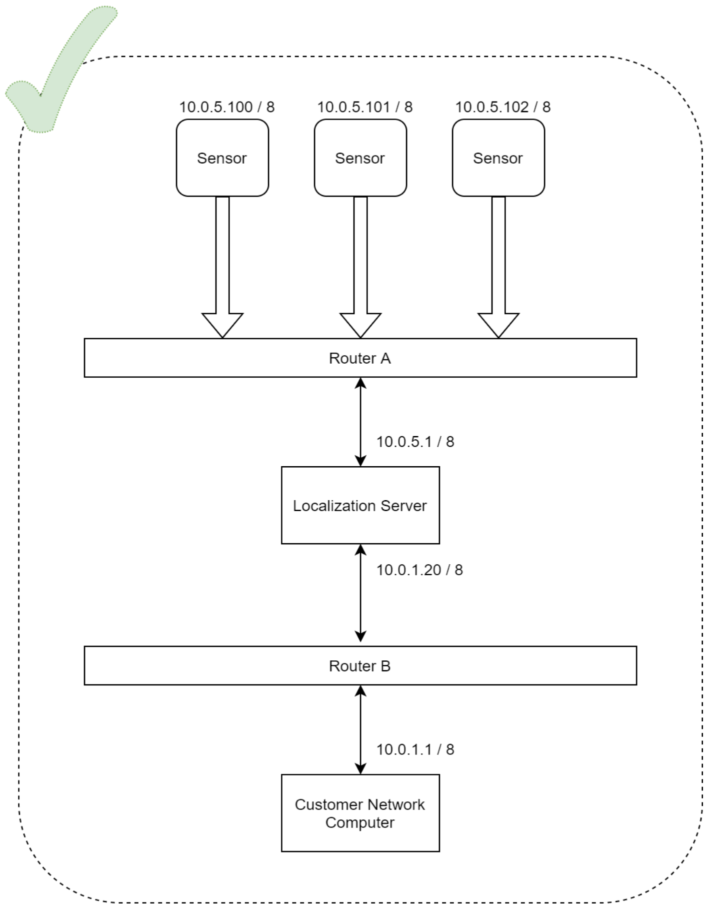
If you see flashing point clouds in the SENSR software, this can be caused by network traffic overload.

4.4.1 Large Installations Strategy

It is strongly recommended to use a static IP strategy when designing and operating large installations. All communications are IP based, including the Lidar sensors and Seoul Robotics cannot guarantee that the sensors will operate in a DHCP environment. A re-assignment of IP addresses due to a reset of the DHCP server, for example, may cause the entire sensor setup and calibration to be lost.

It is also recommended to separate the sensor network from the client network. This will ensure that the sensor traffic will not flood the client network and only the Master Server has access to the sensors.

In this example, the sensor network is on subnet 10.0.5.XXX with a mask of 255.255.255.0 and the client network is on 10.0.1.XXX with a mask of 255.255.255.0. No computer on the client network will be able to directly reach the sensors and vice-versa.



4.5 List of Supported LiDAR sensors

Manufacturer	LiDAR Name	Link
Velodyne	Puck Series	https://velodynelidar.com/products/
Velodyne	Alpha Prime	https://velodynelidar.com/products/alpha-prime/
Hesai	Pandar20A/B	Product web page is currently not available
Hesai	PandarXT series	https://www.hesaitech.com/en/PandarXT
Hesai	PandarQT	https://www.hesaitech.com/en/PandarQT
Hesai	Pandar64	https://www.hesaitech.com/en/Pandar64
Hesai	Pandar40 series	https://www.hesaitech.com/en/Pandar40
Ouster	OS series	https://ouster.com/products/
Innoviz	Innoviz Pro	Product web page is currently not available
Livox	MID series	https://www.livoxtech.com/mid-40-and-mid-100
Livox	Horizon	https://www.livoxtech.com/horizon
Livox	Tele-15	https://www.livoxtech.com/tele-15
Livox	Avia (SENSR 2.1)	https://www.livoxtech.com/avia
Cepton	Vista series	https://www.cepton.com/products.html
Baraja	Spectrum-Scan	https://www.baraja.com/product/
Quanergy	M8	https://quanergy.com/products/m8/
Waymo	Honeycomb	https://waymo.com/lidar

Seoul Robotics continually adds support for newly released LiDAR to the SENSR product line. If your preferred sensor type is not in this list, please contact us.

5. Launch

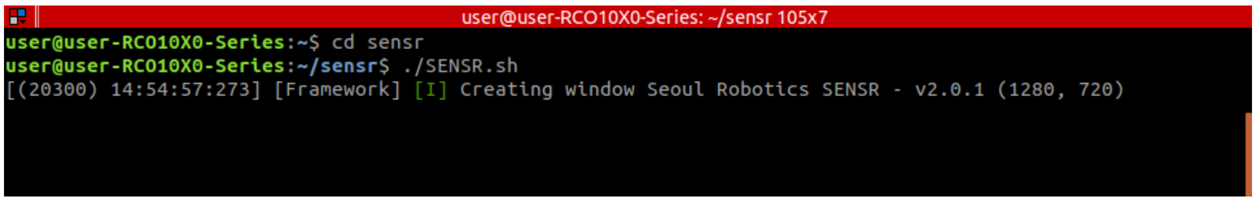
If you have received a Discovery kit, the SENSR software can be started by double-clicking on the desktop icon.

5.1 Manual Launch

To manually launch the application do:

1. Open Terminal in SENSR folder
   a. Use CTRL-ALT-T shortcut
   b. Navigate to the installation folder (default: `cd sensr`)
2. Type: `./SENSR.sh`

When you start SENSR for the very first time, you should see the following in the console window and the 'New Project' page will show.



```
user@user-RCO10X0-Series: ~/sensr 105x7
user@user-RC010X0-Series:~$ cd sensr
user@user-RC010X0-Series:~/sensr$ ./SENSR.sh
[(20300) 14:54:57:273] [Framework] [I] Creating window Seoul Robotics SENSR - v2.0.1 (1280, 720)
```

If you relaunch SENSR after having made a project, it will automatically load the last active project.

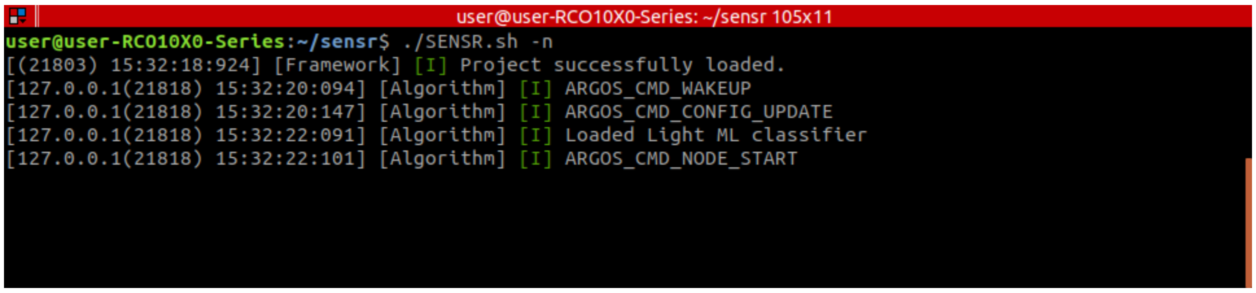
5.2 Launch SENSR with No-GUI Mode enabled

If you have finished setting up sensors, calibration, and parameters and no longer need to see the 3D View, you can launch SENSR with no-GUI mode enabled. SENSR will load your settings and process the LiDAR data. You can check the sensor and processing status with the command line interface and web interface.

1. Open a terminal and navigate to the SENSR folder
2. Type: `./SENSR.sh -n`

In this mode the software will run 'headless', meaning that it will run all the data processing and its data output will be active, but it will not visualize any graphics on the screen.

The terminal will show the following on a successful start.



```
user@user-RCO10X0-Series: ~/sensr 105x11
user@user-RC010X0-Series:~/sensr$ ./SENSR.sh -n
[(21803) 15:32:18:924] [Framework] [I] Project successfully loaded.
[127.0.0.1(21818) 15:32:20:094] [Algorithm] [I] ARGOS_CMD_WAKEUP
[127.0.0.1(21818) 15:32:20:147] [Algorithm] [I] ARGOS_CMD_CONFIG_UPDATE
[127.0.0.1(21818) 15:32:22:091] [Algorithm] [I] Loaded Light ML classifier
[127.0.0.1(21818) 15:32:22:101] [Algorithm] [I] ARGOS_CMD_NODE_START
```

Note that this mode will load and automatically start the last known active project. If no project exists, it will not start.

5.3 Making SENSR autostart.

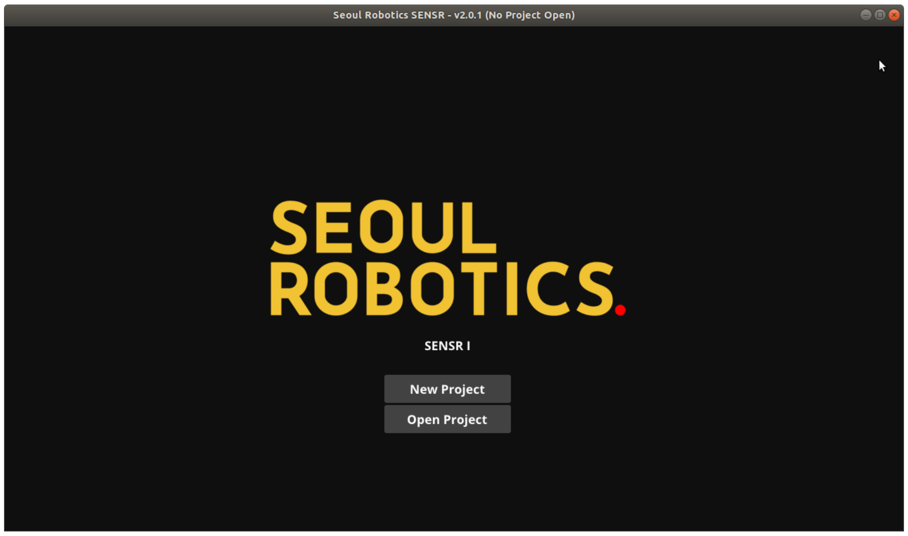
There are many ways to make a process autostart. This can be useful to automatically restart SENSR after a power outage for example.

A good explanation on how to add SENSR to the start-on-boot sequence can be found here:
https://askubuntu.com/questions/48321/how-do-i-start-applications-automatically-on-login

Make sure to include the '-n' option in the start command, so the GUI will not be launched and SENSR will automatically load and start the active project.

6. Projects

The first time you launch SENSR, the application will take you to the New Project screen. Here you can create a new project or load an existing one.



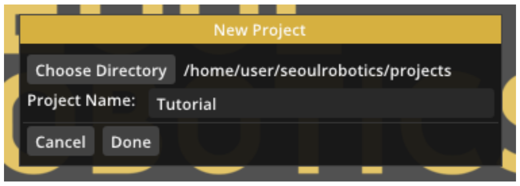
6.1 Projects

A Project in SENSR is the combination of all settings, configuration and calibration of a set of sensors. A group of sensors that is calibrated together to merge the sensor tracking information is commonly called a 'site' and is managed through a SENSR Project.

For example, if you are managing multiple road intersections but are not tracking vehicles between the intersections, each intersection is a 'site' and has its own project in SENSR.

6.2 Creating a Project

Click the *New Project* button under *File* in the top menu bar.



/home/user/seoulrobotics/projects

Project Name: Tutorial

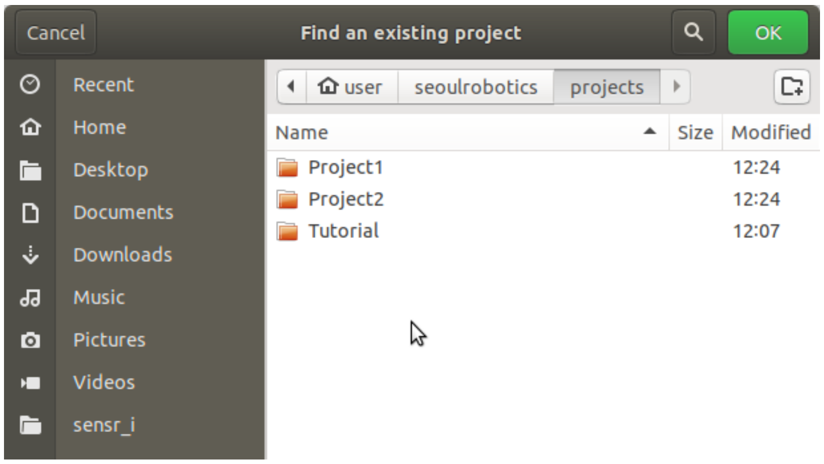
You have the option to choose a directory where the project will be stored. By default, each Project is stored in the '`seoulrobotics/projects`' folder, so they will remain in place when you upgrade the software.

If you decide to change the projects directory, do not place them in any directories that are protected by the operating system. This means '`/var`, `/bin`, `/lib`, `/root`' and most others outside of the '`/home`' directory. It is therefore recommended that you place them in the default location.

6.3 Loading a Project

To load a project, click the *Open Project* button under *File* in the top menu bar. You select a project by selecting the folder containing all the project files.

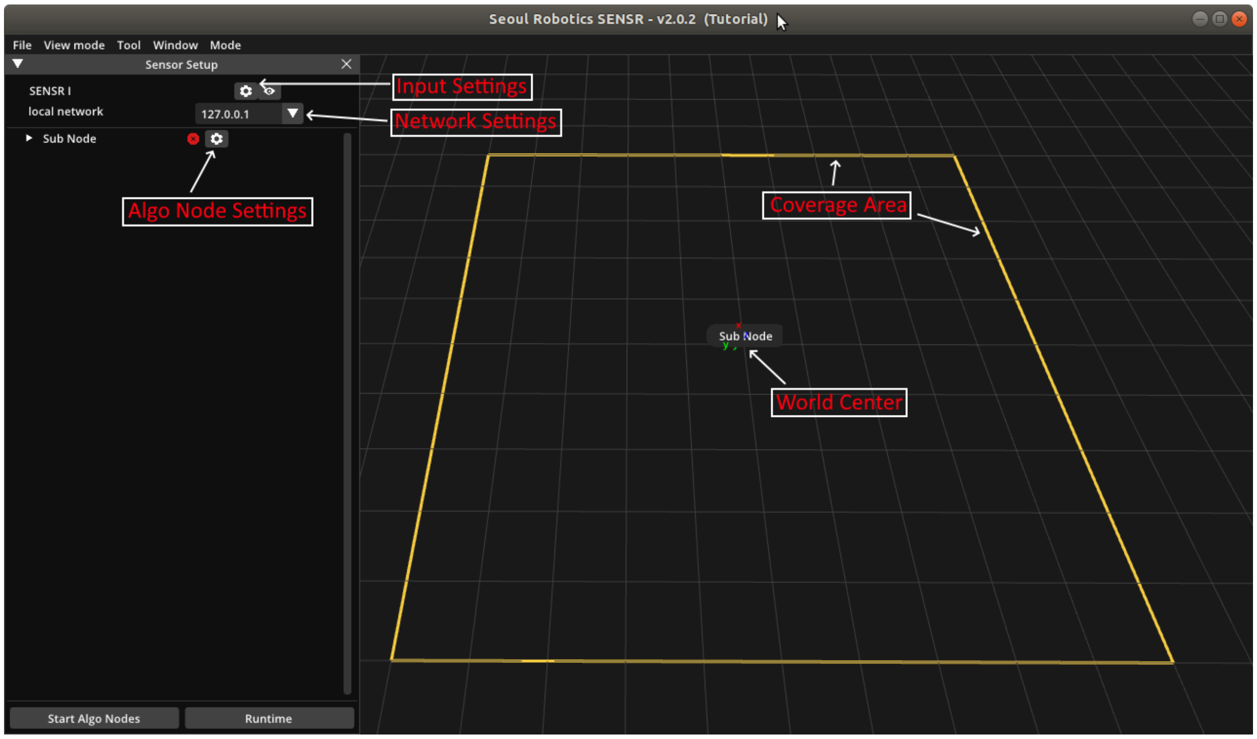
In the below example, the 'projects' folder holds three projects. Select the desired project (i.e. highlight the folder by left-clicking on it once) and then click OK.



Name	Size	Modified
Project1		12:24
Project2		12:24
Tutorial		12:07

6.4 Sensor Setup Screen

Selecting or loading a project will take you to the Sensor Setup Screen. In this screen you can modify the system setup and add, modify, or delete sensors.



6.4.1 Input Settings

SENSR is able to process all supported sensors directly through usage of the manufacturer supplied sensor drivers that are packaged with SENSR. This is the most straightforward way to process sensor data and therefore the recommended method to use in a live situation.

In addition, SENSR supports playing ROS (Robot Operating System) rosbag files with PointCloud2 topics. This can be useful if raw data was recorded as a rosbag and you want to play it back later to test in the SENSR system.

Note that ROS generates very large files and is very intensive to process. For this reason SENSR comes with its own more efficient internal replay system. Refer to Chapter10 for more information on the replay system.

6.4.2 Local Network Settings

In case the computer has multiple ethernet ports and network setups, you can choose the IP address that SENSR will use for the output data stream. All currently assigned IP addresses will be available to choose from.
If you run everything locally on a single computer, you can choose '127.0.0.1'.

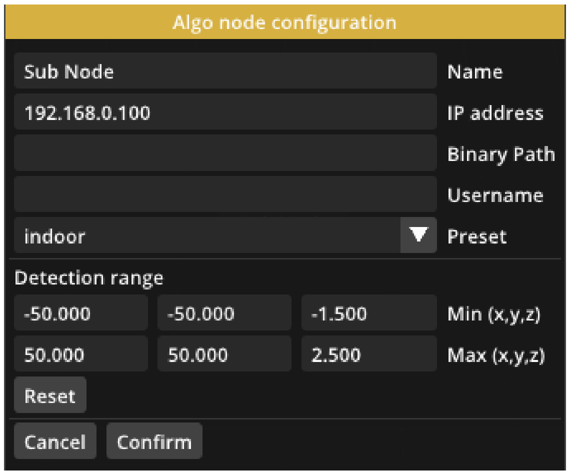
If you are running a remote Algo Node, make sure to set the local network settings to the IP address of the ethernet adapter of the Master Server that the Algo Node is connected to.

For example, if the ethernet adapter of the Master Server that the Algo Nodes are connected to has IP 10.0.0.50, make sure to use that number in your local network settings. Do **NOT** use '127.0.0.1' when running a remote Algo Node, you must select the specific IP address the Algo Node is connected to.

6.4.3 Algo Node Settings

An 'Algo Node' in SENSR is a process that performs the processing of the raw sensor data and transforms this into object tracking data. This data is sent to the master node, which will combine all incoming data and act as the front-end, monitoring and output server. The algo node can be either local (i.e. on the same machine as the master node) or remote.

To change the configuration of an algo node, click the gear icon next to the algo node in the sensor set-up panel. This will open up a window to configure the algo node.



Algo node configuration

Sub Node	Name
192.168.0.100	IP address
	Binary Path
	Username
indoor	Preset

Detection range

-50.000	-50.000	-1.500	Min (x,y,z)
50.000	50.000	2.500	Max (x,y,z)

Name

You can change the name of an algo node to something easily recognizable. The name will be displayed as a floating label in the 3D viewport.

IP Address

To connect to a remote algo node, change the IP address of the node to the IP address of the remote machine.. Make sure the network settings and firewall are configured correctly to allow the two computers to communicate. The ports being used for internal communication between Master and Algo are 5055 and 5056.

If you use only a single computer, you must enter 'localhost' in the IP address field. You will notice that the 'Binary Path' and 'Username' fields will disappear as these are no longer necessary.

Binary Path

The path of the algo node executable is located in the remote machine. The default path is: `/home/<user>/sensr/bin/` , where <user> is the user login name of the Algo Node.

Username

Username of the remote machine. Master node uses SSH to launch the algo node in the remote machine. Make sure you followed the SSH authentication steps in chapter 2.6.2.

Preset

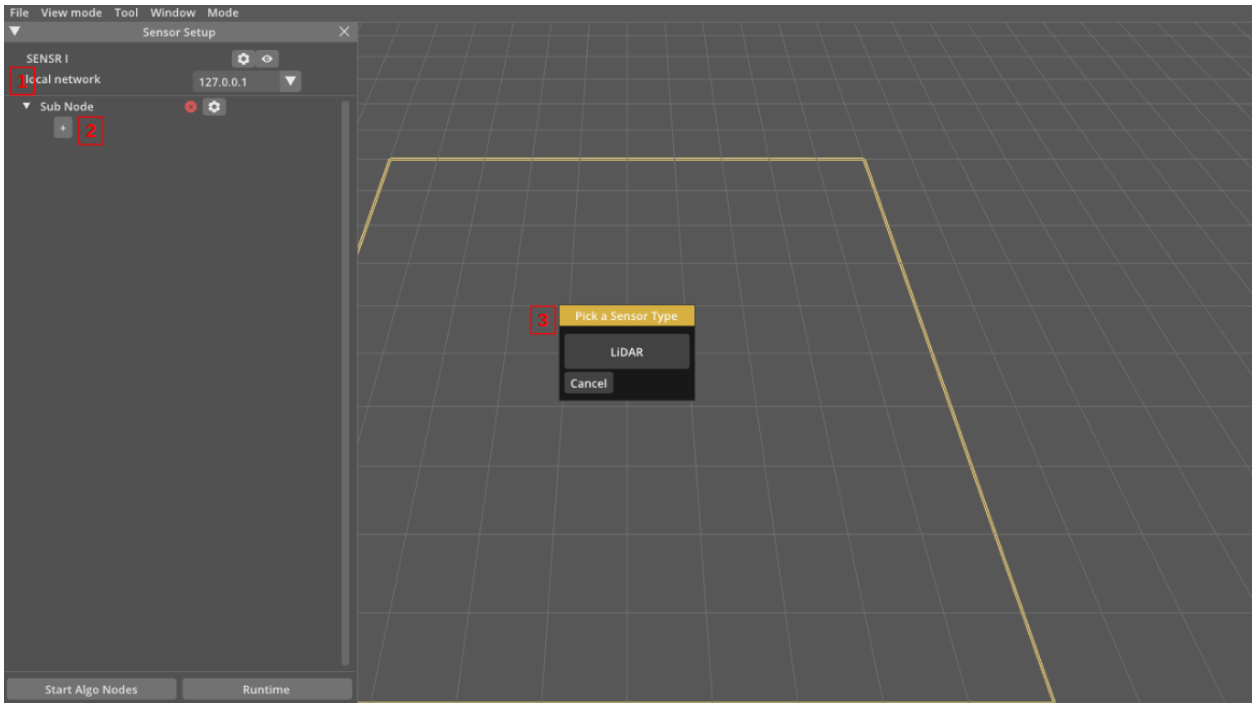
SENSR comes with a number of algorithm presets to fit a variety of situations. The two default presets are `Indoor` and `Outdoor` which work well in most use cases. The `Indoor` preset is optimized for a situation with level-ground, short range, and many objects moving around. The `Outdoor` preset is optimized for outdoor, medium-long range settings and can track larger objects such as trucks or buses.
Additional presets will be made available in the future.

Detection Range

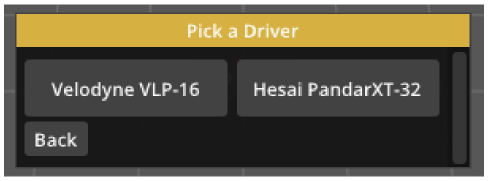
This range indicates the total coverage area that SENSR will process for that algo node and attempt to track objects in. The larger this space, the more computing power is required, so it is recommended to keep this space as small as needed. This will be explained further in the setup and calibration tutorial section.

6.4.4 Sensor Settings

For adding sensors on an algo node, you need to open up the Algo Node first by clicking the collapse arrow (1) and then click '+' button (2) for opening the sensor setting popup (3) as in the following picture.



Click the LiDAR button (3). Then, you will see the available sensor list. The contents of this list is determined by your license configuration. In this example, the license only contains VLP-16 and PandarXT-32 support.



And Each lidar has different arguments which are related to lidar connection. For example, VLP-16 has LiDAR IP and Port.

Detection Range

All lidar settings share the 'Detection Range' argument. This range limits the output range of the sensor. This can help with reducing noise from far away objects or limiting the range to conserve processing power. The detection range is specified in meters.

As an example of how to use this range: Say you have two sensors A and B that are spaced 50m apart from each other. An object that is 5m away from Sensor A will be clearly picked up by Sensor A, but a small error in calibration of Sensor B could see the same object in a different location, creating a ghost image. By reducing the Detection Range of Sensor B to 35m, any object that comes close to Sensor A will be ignored by B, ensuring that small errors in calibration do not cause problems.

As a general rule of thumb, for indoor applications with lots of obstructions, the *Detection Range* can often be reduced to as low as 25m for each sensor.

Intensity Filtering

This option sets a minimum intensity threshold that a Lidar point has to meet to be processed. This filtering can be useful to filter out noise from dust, exhaust fumes and falling leaves for example.
The correct setting will be different for each Lidar. There is no common way to calculate and report intensity, so the values can be very different between Lidar. Try to start low and increase the value to see the behavior change. As an example, for a Livox Horizon, a value as low as three (3) will filter out most dust and exhaust.

7. Calibration

When sensors connect to the software, the sensors become part of the SENSR world. This means that SENSR needs to understand where each sensor is in its world, so it can transform and merge the information into a single coordinate system. This process is called the Calibration Process.

This chapter will walk you step-by-step through the calibration process using the sample data that is available as part of the SENSR system.

If you wish to follow this tutorial, you will need to download the sample data. To do this, do the following steps:

- Open a terminal window (CTRL-ALT-T)
- Navigate to the installation folder (default: `cd sensr`)
- Type: `./download_sample.sh`

7.1 Setup and Calibration Tutorial

For this tutorial we will create a brand new project, run our pre-packaged sample data and calibrate the sensors.

7.1.1 Create the Tutorial Project

- Either in the first start screen or through the *File->New Project* menu, create a new project. We'll call it 'tutorial'.
    - Leave the directory as default to ensure the rest of the tutorial matches your configuration
    - Enter 'tutorial' as the *Project name*.
    - Click *Done*

- Click on the *Input Settings* gear icon (see image in Chapter 6.3). For this tutorial we will be using the provided sample ROS bag and we will configure the project for this.
    - Switch to *Ros Interface*.
    - Click *OK*

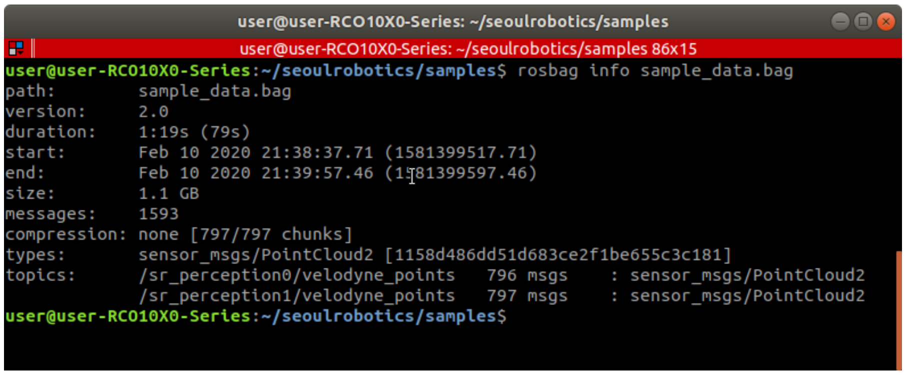
7.1.2 Set Up Sample Replay

- Click *Algo Node Settings* gear icon.
    - Fill in the full sample data path. If everything is installed in the default locations, the path will be */home/user/seoulrobotics/samples/sample_data.bag*
    - Click *Confirm*

7.1.3 Set up Sensors

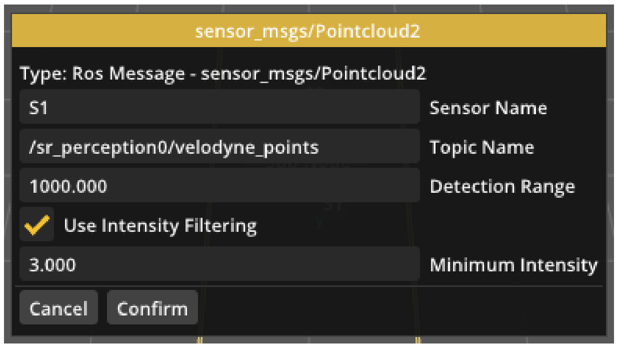
A ROS bag can contain a wide variety of information. SENSR only requires the LiDAR point cloud data to function. To find out what data a ROSbag contains, you can query the bag file for its contents.

- Open a new terminal (CTRL-ALT-T)
- Navigate to the location of the bag file.
    - Type: `cd seoulrobotics/samples`
    - Type: `rosbag info sample_data.bag`
        - If you get the error message: *command 'rosbag' not found*, type the following command before trying again:
          `source /opt/ros/melodic/setup.bash`
    - You should see the following:



```
user@user-RCO10X0-Series: ~/seoulrobotics/samples
user@user-RCO10X0-Series: ~/seoulrobotics/samples 86x15
user@user-RC010X0-Series:~/seoulrobotics/samples$ rosbag info sample_data.bag
path:        sample_data.bag
version:     2.0
duration:    1:19s (79s)
start:       Feb 10 2020 21:38:37.71 (1581399517.71)
end:         Feb 10 2020 21:39:57.46 (1581399597.46)
size:        1.1 GB
messages:    1593
compression: none [797/797 chunks]
types:       sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:      /sr_perception0/velodyne_points   796 msgs    : sensor_msgs/PointCloud2
             /sr_perception1/velodyne_points   797 msgs    : sensor_msgs/PointCloud2
user@user-RC010X0-Series:~/seoulrobotics/samples$
```

- ○ Any topic of type PointCloud2 can be used by SENSR
- In SENSR, double-click the *Sub Node* or single-click on the arrow to expand the *Sub Node* menu.
- Click the *+* icon below the *Sub Node*
    - Pick the message type: *sensor_msgs/PointCloud2*
        - Give the sensor a unique name. For example 'Sensor 1'
        - Copy the topic name from the rosbag info into the *Topic Name* field. In this example: */sr_perception0/velodyne_points*



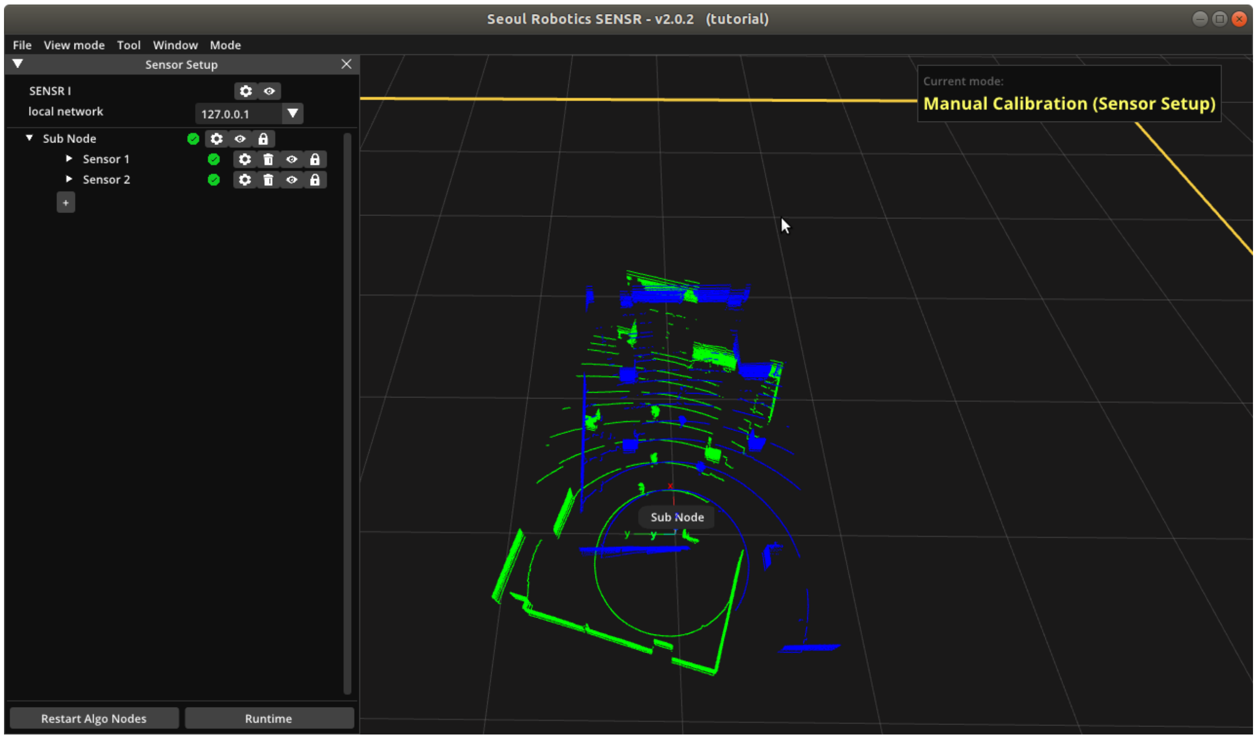
sensor_msgs/Pointcloud2

Type: Ros Message - sensor_msgs/Pointcloud2

S1	Sensor Name
/sr_perception0/velodyne_points	Topic Name
1000.000	Detection Range
✓ Use Intensity Filtering	
3.000	Minimum Intensity

        - Click *Confirm*
    - Repeat this for all PointCloud2 topics in the bag file.
    - You should have two sensors with unique names and topics associated with them. Make sure the second sensor has the topic name */sr_perception1/velodyne_points.*
- Click *Start Algo Nodes* to start the calibration process.

If everything went well, you should see the following on the screen.



Green check marks indicate that the Algo Node and Sensors are up and running and processing data correctly. Each sensor will generate a point cloud in its own color. As you can see the point clouds are displayed and both sensors are put in the default location.

Each node in the list has some buttons next to it. The functionality of each button is as follows:

- ⚙ Open up the settings window for the node.
- 🗑 Delete the node from your project.
- 👁 Toggle the visibility of the node. If visibility is disabled the node cannot be interacted with
- 🔒 Lock the node in place and make it non-interactable. Locking an algo node also automatically locks the child sensor nodes of that algo node.

The scenario of the replay is three people walking around a room that also has a number of static pillars in it. Two sensors are placed at opposite sides of the room near the ceiling.

In the 3D viewport, you can use the mouse to move the camera position and angle relative to the 3D world.

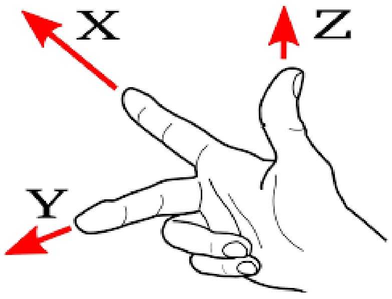
- To **move**, hold the middle mouse button and move the mouse. The WASD keys can also be used to move.
- To **rotate**, hold the right mouse button and move the mouse around.
- To **select**, click the left mouse button when hovered over a point cloud.

You can also select specific sensors by clicking on their name in the list.

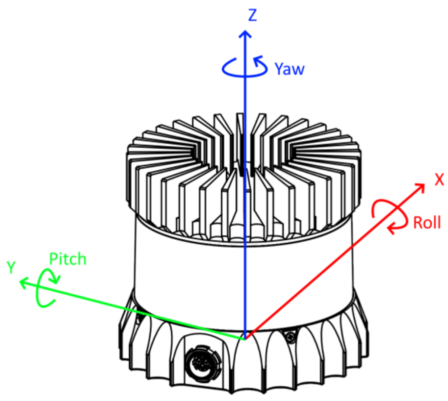
7.1.4 A word about coordinate systems

SENSR uses a right-handed coordinate system. The ground plane aligns along the X-Y plane and Z being altitude.

There are six values that have to be solved during the calibration phase. First is the location (X, Y and Z) and second is the rotation (Pitch, Roll, Yaw).



For the most common type of LiDAR, a rotating lidar, the lidar will rotate around the Z axis. The connector on the LiDAR is, in most cases, opposite to the zero angle, which is the rotation angle that is considered the start of a new rotation in the sensor. In the image this zero aligns along the X axis.



For this tutorial, and for calibration in general the following is defined:

- Rotation around Z = Yaw
- Rotation around X = Roll
- Rotation around Y = Pitch

7.1.5 Start the Calibration Process

The two main goals to achieve during the calibration process are:

1. Align the ground plane of the sensor to the ground plane of the SENSR software.
2. Align each sensor relative to each other sensor.

It is important to know that, by default, nothing below the ground plane is being tracked, so correct ground alignment is critical to achieve good tracking performance and full coverage..

Calibration can be done many different ways. This tutorial describes one of the methods that can be applied to small and large installations alike. In this tutorial we are working with only two sensors, but in full deployments it could be 20 or more. At the start of a new calibration, SENSR places all sensors at the zero location (0,0,0). This can be confusing to work with, so it is recommended to start with a clean slate and calibrate sensors one at a time.

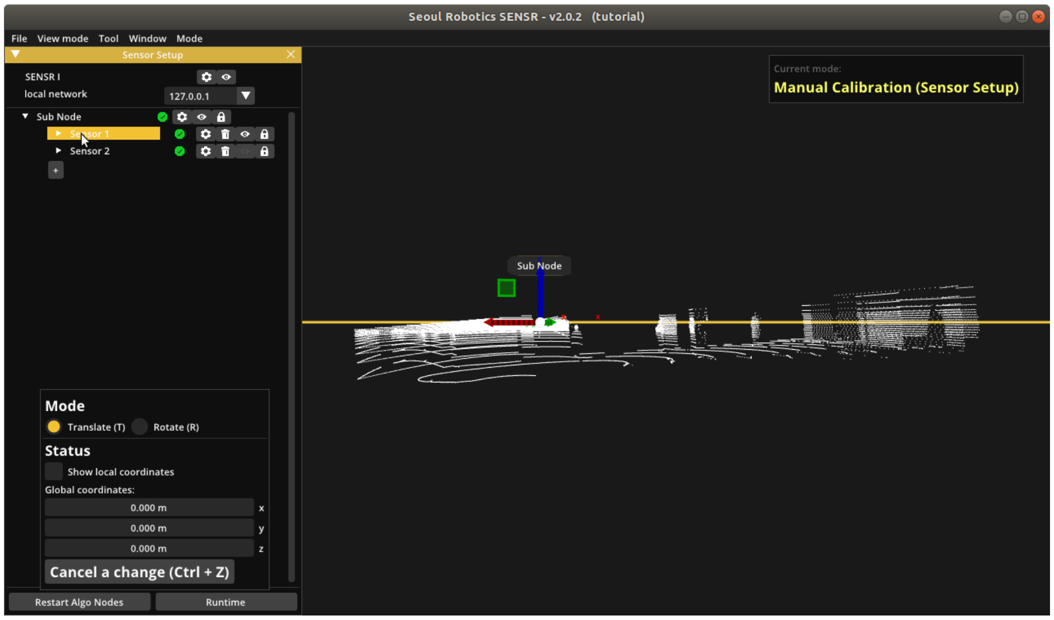
- Go to each sensor in the list and click on the eye icon. This will make each sensor invisible.
- Now make only Sensor 1 visible.
- Select *Sensor 1* in the list by clicking on the name.

Now only Sensor 1 should be visible and selected. You can tell whether a sensor is actively selected by the point cloud belonging to that sensor turning white.
As mentioned before, we need to resolve six values (X, Y, Z, Pitch, Roll, Yaw). For the very first sensor, it is best to start with ground alignment, this will solve three values all at once (Z, Pitch and Roll).

Before we do this:

- Go to the 3D viewport, hold the right mouse button and rotate the view until the camera is level to the ground.

This is when you cannot rotate any further and the yellow rectangle turns into a line. You are now in side-view. The yellow line indicates the ground level of the SENSR world. Below this level nothing will be tracked.



You can now clearly see that the majority of the point cloud is below ground level. This is because the Z of a new sensor is set to zero, so SENSR assumes that the lidar is on the ground.

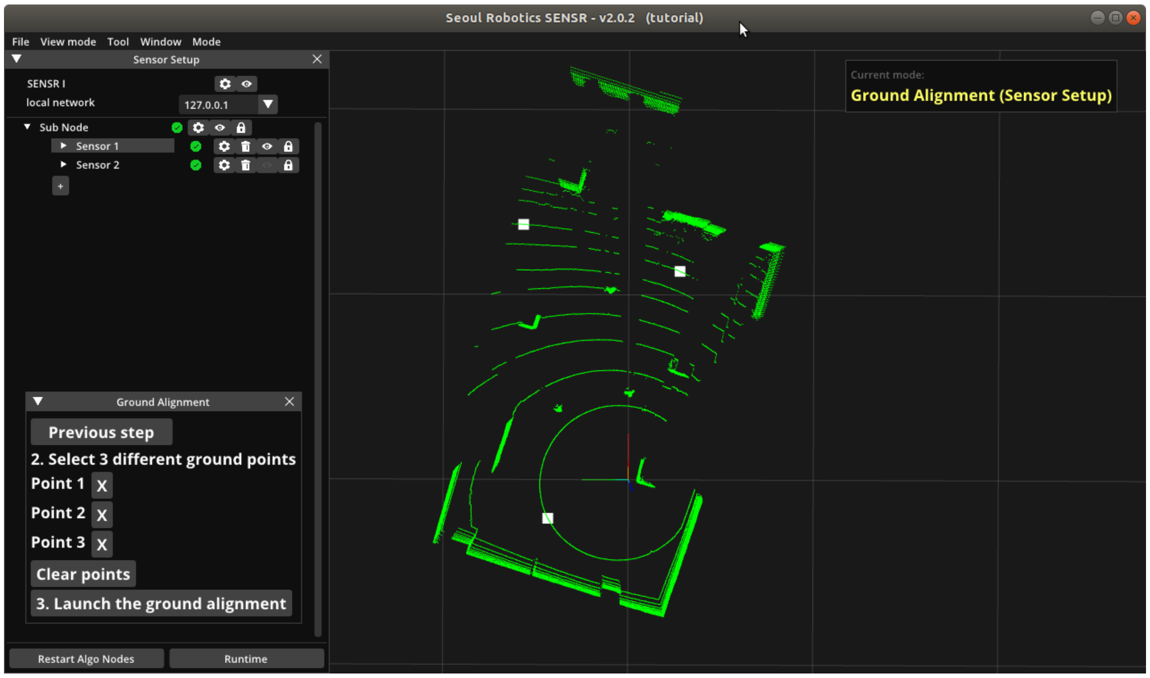
7.1.6 Ground Alignment

Our first goal is to raise the Z to the actual installed height of the sensor and the ground level. Fortunately, SENSR comes with a very handy tool for ground alignment that works perfectly for flat ground. This tool is called *Ground Alignment* and can be found in the *Tool* menu or through the **F4** shortcut.

- Select the Ground Alignment tool
- Select any point in the visible sensor to start the ground alignment for that sensor
- Rotate the point cloud to an angle that you have a good view
- Select three points that:
    - You know are on ground level
    - Are spread far apart

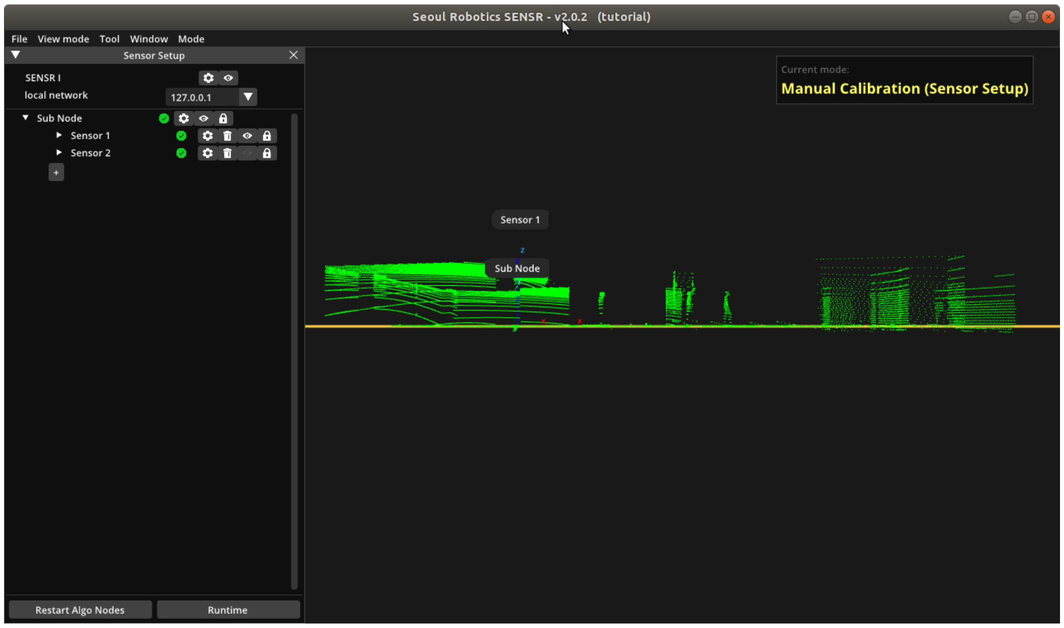
- And ideally make a wide triangle

In this image the three white markers indicate the three selected points.



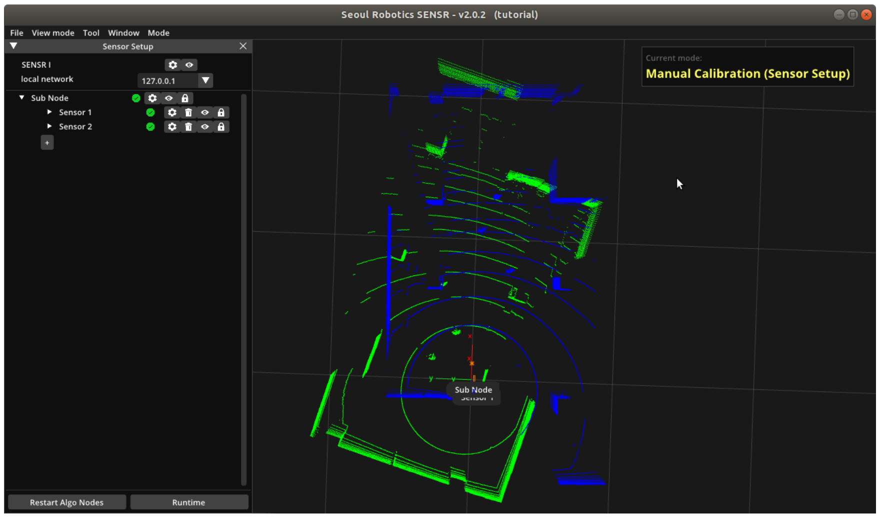
Now click the *Launch the ground alignment* button

You will see the point cloud shift a little. What happened is that SENSR took those three points and aligned them to the ground plane. Go back to the side view and you will notice the difference.



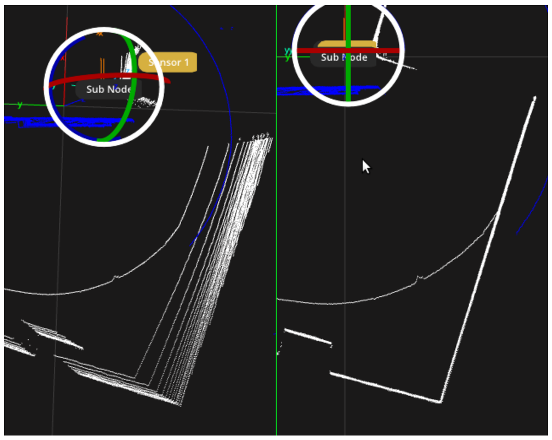
The ground in the point cloud is now clearly aligned to the ground in SENSR. You will notice that the label 'Sensor 1' has appeared in its new location. If you zoom in to the 'Sensor 1' location you will see that the little axis indicator has moved to a new location, indicating the exact location of Sensor 1 relative to the world.

Repeat these steps for Sensor 2. You can make Sensor 1 invisible first then make Sensor 2 visible for a clean view of the point cloud. Once Sensor 2 is ground aligned, the calibration should look like this.



7.1.7 Moving and Rotating Sensors

Now the values Z, Pitch and Roll have been solved for both sensors. That leaves X, Y and Yaw.

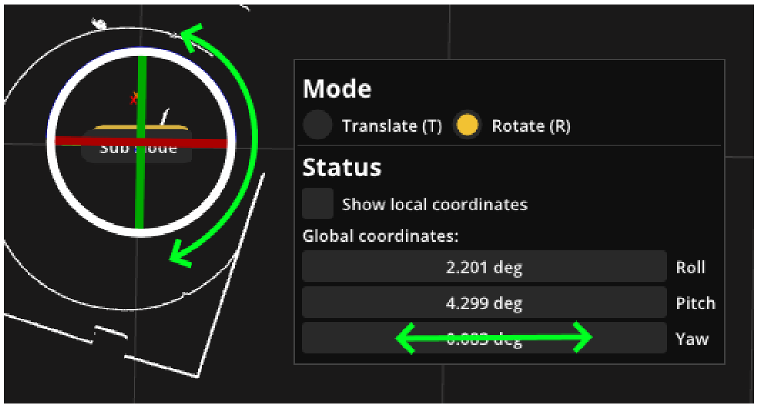


To solve these, for indoor scenarios it is recommended to switch from **perspective** to **orthographic (top-view)** mode. This switches the system to a 2D top-down view and will remove the effect of perspective from the point cloud data and make it much easier to match up walls and corners.

Just like the ground alignment, it is recommended to start with the first sensor and get it aligned properly. To do this:

- Make Sensor 2 invisible
- Select Sensor 1

You will see a *Mode* box pop-up on the screen. This box allows you to move the sensor and rotate it. We are going to fix Yaw first, so select the *Rotate* option or use the '**R**' keyboard shortcut



Mode

Translate (T) [✓ Rotate (R)]

Status

Show local coordinates

Global coordinates:

2.201 deg	Roll
4.299 deg	Pitch
0.083 deg	Yaw

There are a couple of ways to rotate.

1. You can use the 3D widget on the left and drag it.
2. You can hold the mouse on the Yaw field and drag it left to right. If you hold Shift while dragging, it will rotate faster.

3. You can double-click or CTRL-click on the Yaw field to enter a value manually.

- Rotate Sensor 1 so the walls line up with the grid. The Yaw value should be around the 16.5 value for this tutorial.
- Sensor 1 is in place. To prevent any accidental movement of Sensor 1 you can click the *Lock* icon next to the sensor to lock it in place.
- Now make Sensor 2 visible.

It looks like Sensor 2 is already rotated correctly, but remember that Sensor 2 is at the opposite side of the room. This means that it actually has to rotate 180 degrees to align correctly to Sensor 1.

This is why it is important to, when you have a large coverage area with many sensors, have a floorplan ready as a reference with the rough installation locations of each sensor, so you have an idea of where each sensor should be relative to the others.

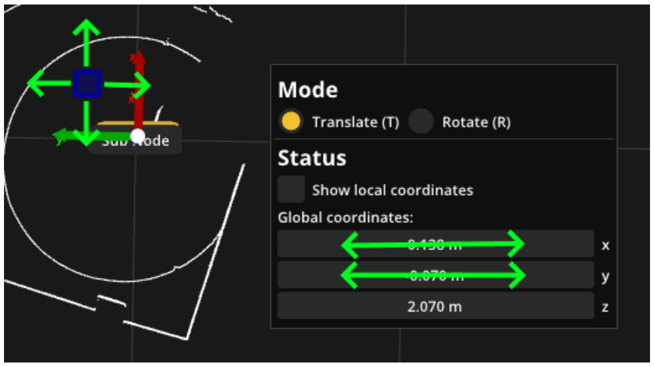
- Take Sensor 2 and rotate it by 180 degrees. In this case it is fastest to double-click the *Yaw* field and enter the value 180.

The final step is to move Sensor 2 so all the walls and pillars match Sensor 1. To do this:

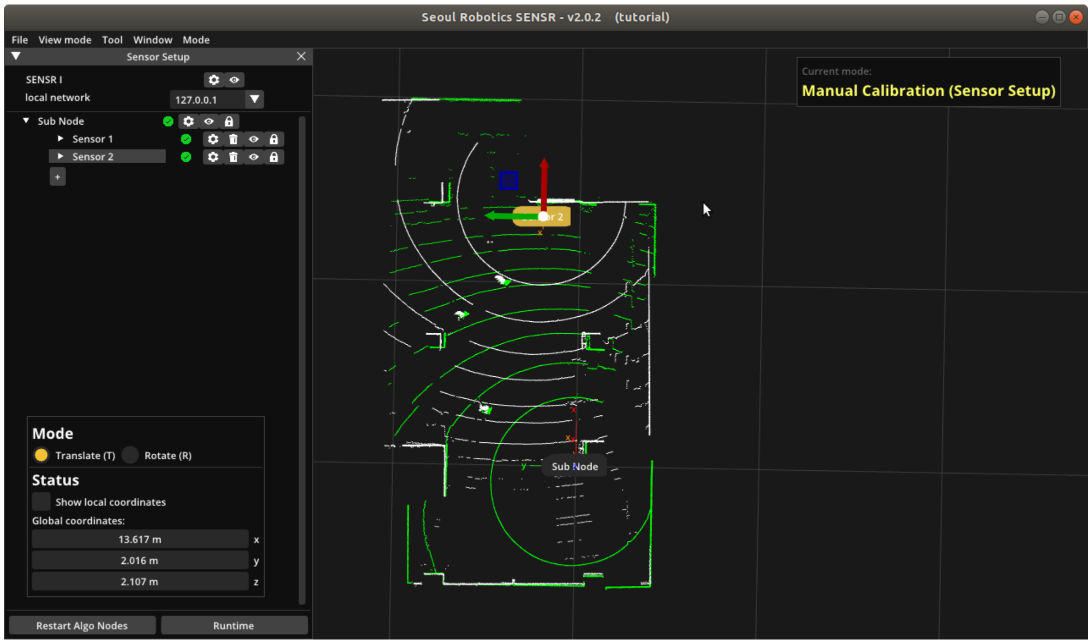
- Switch to the *Translate* mode or press the '**T**' keyboard shortcut.
- Move Sensor 2 so it matches Sensor 1

You have three options to move the sensor:

1. Drag the blue rectangle in the 3D widget to move the sensor along the ground without affecting the Z value
2. Hold and drag the mouse in the X and Y value boxed. Hold shift to move faster.
3. Double-click or CTRL-click in X and Y to enter a manual value.

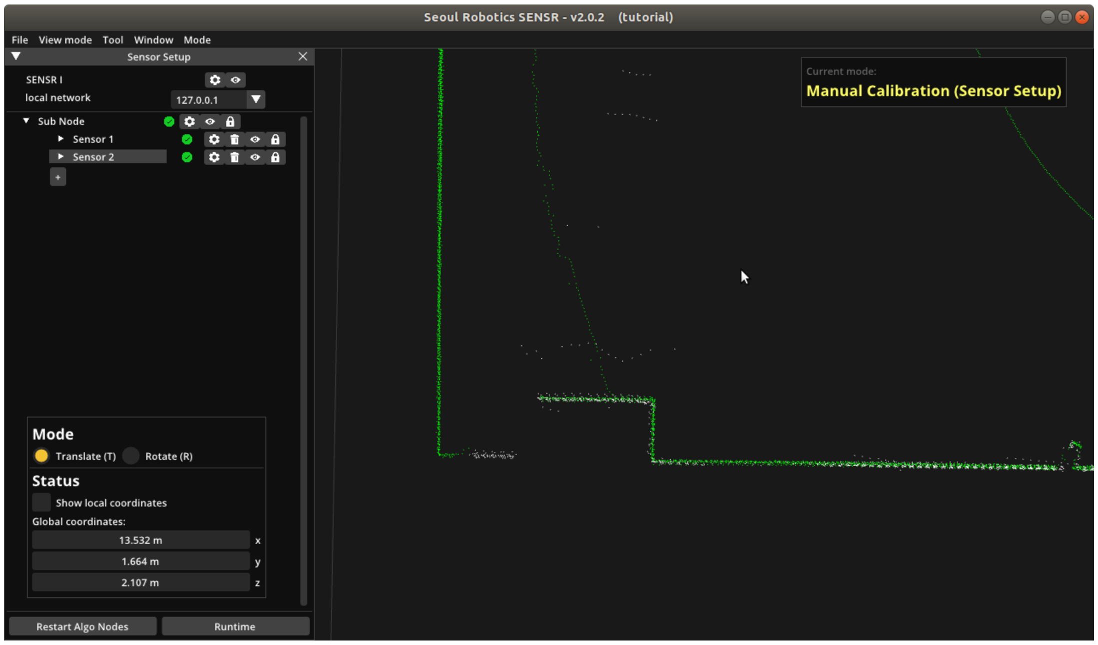


In this tutorial the X and Y value of Sensor 2 should be 13.6 and 2.0 respectively.



The final step is tweaking the location and the rotation so all walls and corners match up as close as possible. In general, for rooms like this, it is best to get the Yaw dialed in perfectly and then move the unit to the correct location. Long straight walls can be used to tweak yaw and hard corners are best to use for the final location.

In the below image Sensor 1 is shown in green and the active Sensor 2 in white. Once you have achieved perfect overlap, you can lock down the calibration of these two sensors.



The calibration of Sensor 1 and Sensor 2 is now complete. If your installation has more sensors, you can make them visible one at a time and continue the calibration process.

You can switch back to perspective mode and look at the calibration from different angles to make sure everything looks correct.

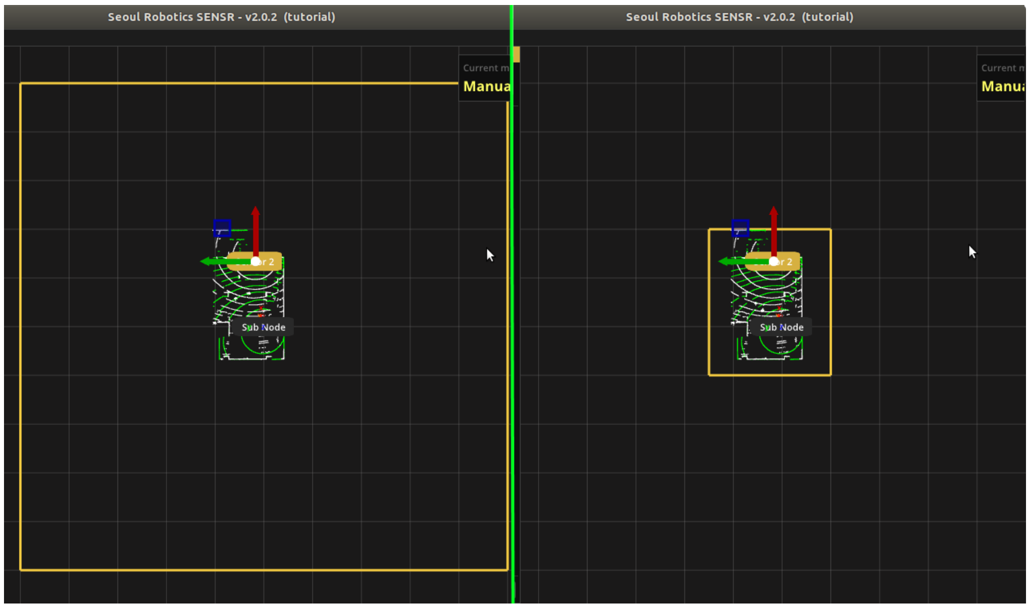
7.1.8 Optimizing the World Size

This step is optional, but it is useful to understand how this works so you are able to optimize computing resources and maximize tracking performance.

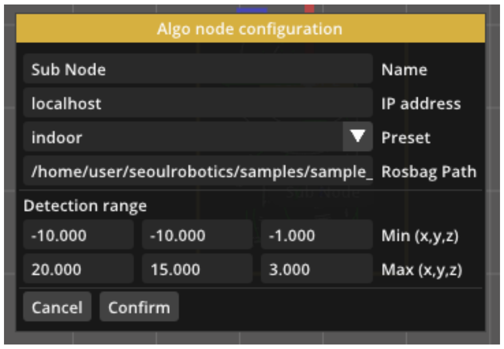
In Chapter 6.3.3 we discussed the *Algo Node Settings*. One of these settings is the *Detection Range*[1]. This is the total world size that SENSR is processing. The default for this world size is 100x100x5m in space. For this tutorial dataset, we are only tracking people in a single room, so the world size is much greater than the actual observable area. We can reduce processing and memory resources by reducing the world size.

[1] Note that both Algo Nodes and Sensors have Detection Ranges. A Sensor Detection Range applies to only one sensor. The Algo Node Detection Range applies to all sensors equally.

On the left is the default world size of 100x100m, on the right is the optimized world size of 20x35m, an improvement of 93%


Seoul Robotics SENSR - v2.0.2 (tutorial)
Seoul Robotics SENSR - v2.0.2 (tutorial)
Sub Node
Sub Node

- Click on the *Algo Node* settings (gear icon) and change the *Detection Range* values to the following:



Algo node configuration

Sub Node			Name
localhost			IP address
indoor			Preset
/home/user/seoulrobotics/samples/sample_			Rosbag Path
Detection range			
-10.000	-10.000	-1.000	Min (x,y,z)
20.000	15.000	3.000	Max (x,y,z)

The calibration process is complete. The calibration will automatically be saved when you exit the application. Saving can also be done manually at any time during the calibration by pressing *File > Save* or using the CTRL+S hotkey.

You can now launch runtime to check the tracking results, or continue the manual and create zones first.

8. Zone Setup

SENSR has built-in zone management. Zones can be used to trigger entry/exit, calculate occupancy or areas and to eliminate noise.

You can access the zone setup screen using the *Mode > Zone Setup* menu item. When you create a new zone

8.1 Event Zones

There are two types of zones:

8.1.1 Event Zones

An *Event Zone* is a special area that reacts to objects entering and exiting. The data output stream (see Chapter 13) will generate a message whenever an object enters or exits an event zone. Additional triggers are available (see Chapter 8.2.3)

8.1.2 Exclusion Zones

Exclusion zones are used to exclude an area from being processed. Any object that enters an exclusion zone is no longer tracked and will not be part of the output stream. This can be done for a variety of reasons.

Privacy

It is possible that in the total coverage area not all areas are allowed to be tracked, but the sensors have line of sight.

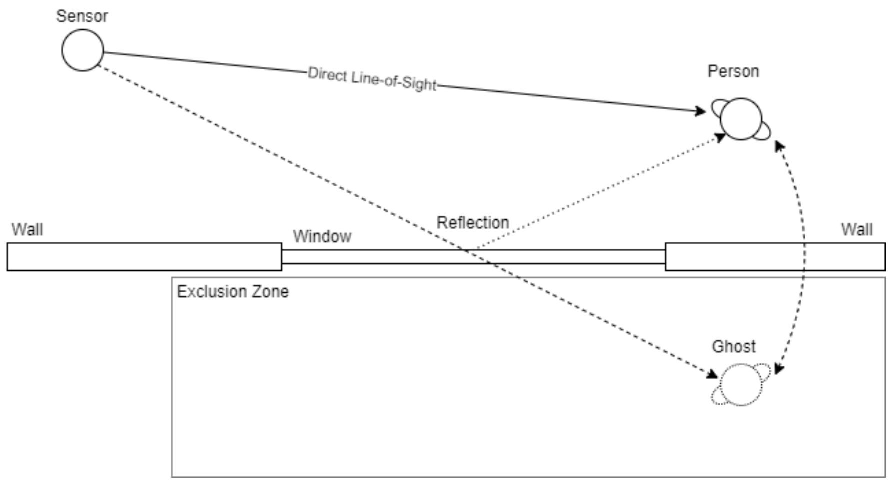
Noise

It is possible for certain seemingly static objects to create movement. Vegetation in particular can move in the wind and create false positives. If a coverage area has a known trouble spot, a manually created exclusion zone around this object can eliminate false positives in this area.

Reflections

The 'Li' in LiDAR stands for 'Light'. Light reflects off of shiny surfaces such as mirrors and glass. This has the potential to create *Ghost Images*.

For example:
A sensor and a person are in the same room with a large window. The sensor has a direct line-of-sight to the person, creating an object. The sensor also has an indirect line-of-sight through a reflection off the window. The sensor is not able to differentiate a reflection from a direct line-of-sight and will provide the SENSR system with ghost points on the other side of the window.



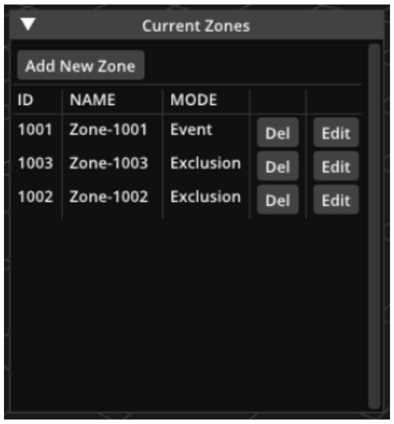
The solution to this is to create an exclusion zone on the other side of the window, so any objects will be ignored.

Performance

Another important reason to create exclusion zones is to reduce the amount of processing that is required. If your sensors have line-of-sight to an area, but you are not interested in tracking in that location, an exclusion zone will skip any processing that would otherwise take place inside the zone.

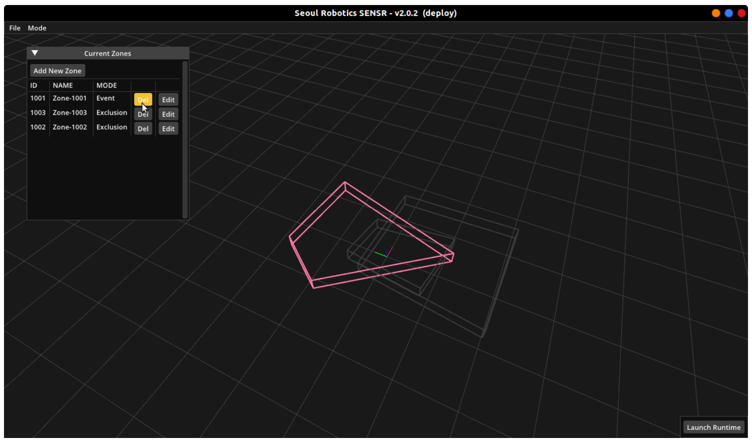
8.1 Zone List View

When you enter *Zone Mode*, you will see a window with all existing zones listed.



ID	NAME	MODE		
1001	Zone-1001	Event	Del	Edit
1003	Zone-1003	Exclusion	Del	Edit
1002	Zone-1002	Exclusion	Del	Edit

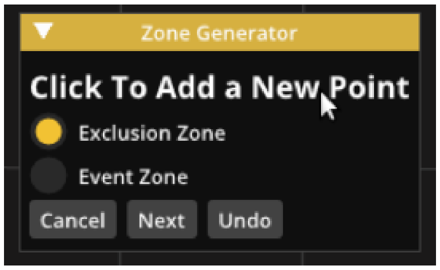
In the window, you can see a list of zones on the 'Current Zones' widget. If you hover your mouse over the *Del* or *Edit* buttons, that zone is highlighted in the 3D view with a brighter color.



ID	NAME	MODE		
1001	Zone-1001	Event	Del	Edit
1003	Zone-1003	Exclusion	Del	Edit
1002	Zone-1002	Exclusion	Del	Edit

8.2 Zone Generator

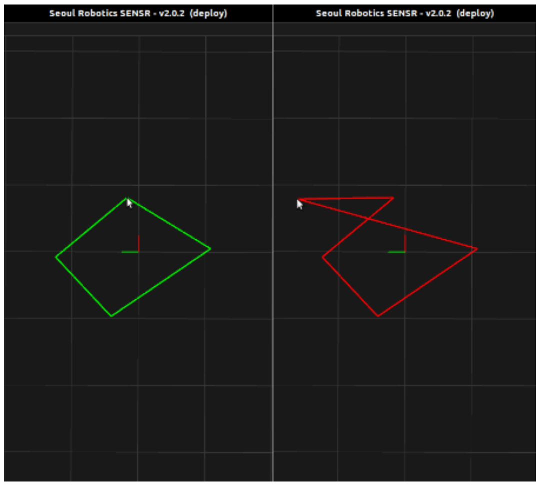
Clicking 'Add New Zone', camera view changes to top view and Zone setup tool will load the zone generator so that you can create a new zone.



8.2.1 Drawing a Zone

After 'Add New Zone' is clicked, the 3D viewport changes to a 2D top-down view of the site. You can now select the type of zone you want to add.

To start creating the zone, click on the location of the corners of a zone in the 3D viewport. A **left-click** will add a new point and a **right-click** will remove the most recent point.


Seoul Robotics SENSR - v2.0.2 (deploy)
Seoul Robotics SENSR - v2.0.2 (deploy)

To cancel the creation of a zone during the point selection process, click the *Cancel* button.

There are two points that you have to keep in mind while drawing the zone.

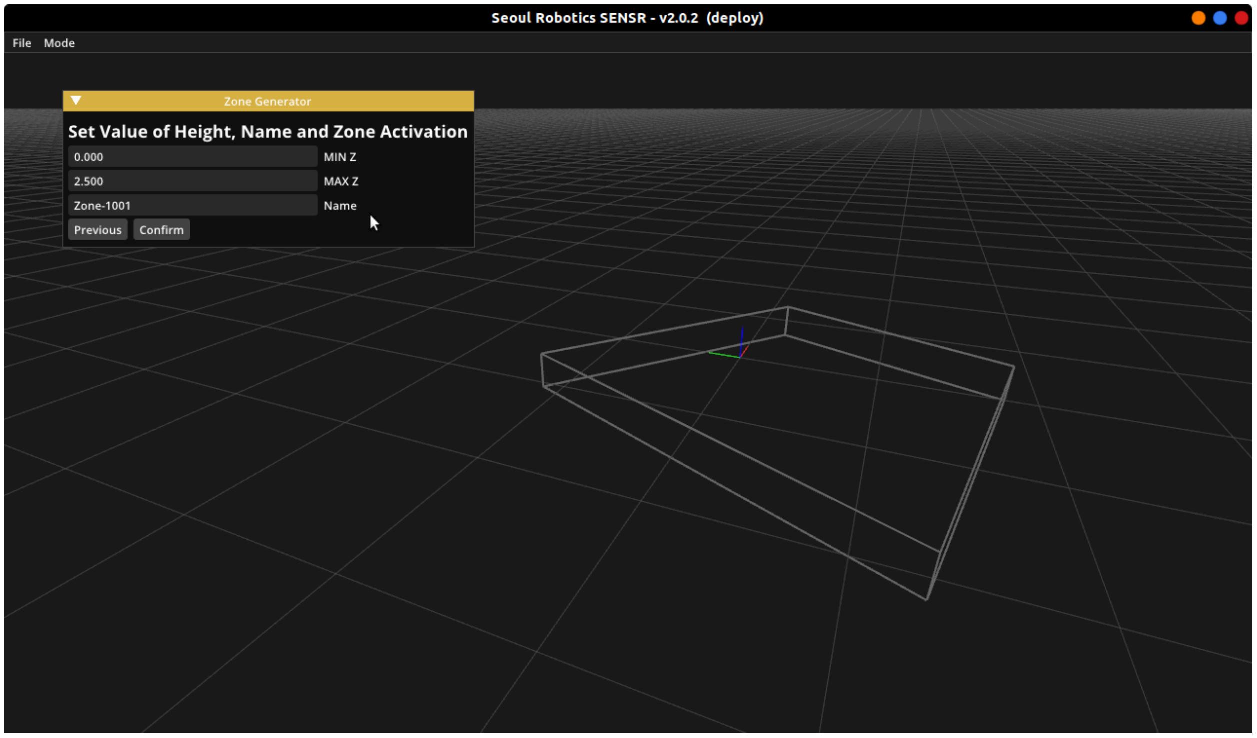
- A zone should have at least 3 points.
- A zone can not have self-intersection.

The zone can assume three colors during the zone creation process

Red: The zone does not satisfy the above conditions and cannot yet be completed. Add more points or right-click to remove a self-intersection.
Gray: The zone can be completed or more points can be added.
Green: The current location of the cursor will complete the zone on a left-click.

8.2.2 Setting Height and Name

After drawing the zone, a new window will show, allowing you to set the name and height of the new zone.



Set Value of Height, Name and Zone Activation

0.000	MIN Z
2.500	MAX Z
Zone-1001	Name

MIN Z

This is the height in meters of the bottom of the zone as measured from the ground level. This allows a zone to be created above ground level.

If you create a zone with a minimum height of non-zero, make sure to check the 'Allow Floating Object' option in the settings (see Chapter 11). If the zone has a *MIN Z* of greater than zero, the bottom of the object must be above the *MIN Z* value for the object to be considered in the zone.

MAX Z

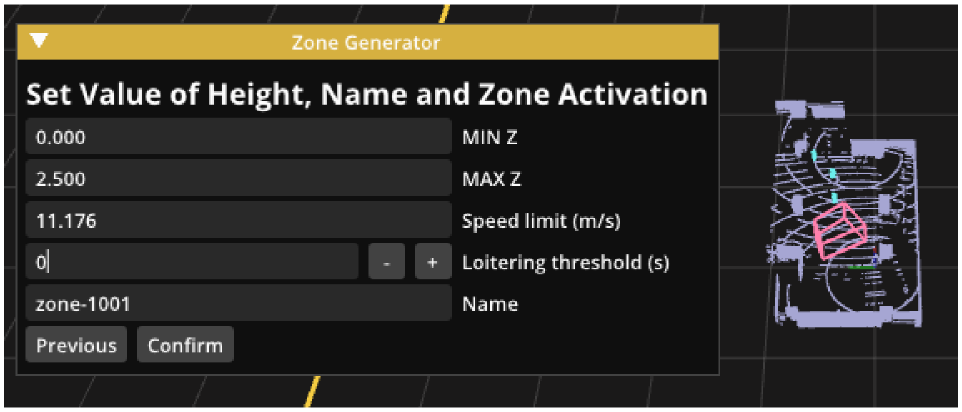
This is the height in meters of the top of the zone as measured from ground level.

Name

The name is a label that you can create for your zone. This name is part of the output stream (see Chapter 13). A default name is created using the format 'Zone-{zone_id}'. You can change it to your liking and it does not have to be a unique name.

8.2.3 Event Zones Additional Trigger Settings

SENSR Event Zones come with a number of more advanced triggers that can be configured.



Set Value of Height, Name and Zone Activation

0.000	MIN Z
2.500	MAX Z
11.176	Speed limit (m/s)
0	Loitering threshold (s)
zone-1001	Name

Speed Limit

This number, in meters per second, will trigger an 'exceed speed' event message in the output stream when an object that is inside the zone boundaries exceeds the speed threshold.

Loitering Threshold

Any object that is inside the zone boundaries will be tracked how long it has been within the boundaries. When the time exceeds the threshold a 'loitering' message will be generated. If the object exits and re-enters the zone, the elapsed time will be reset on exit.

For an object to be considered in a zone, the following criteria must be met

- Both X and Y at the center of the bounding box at ground level must be inside the zone
- The entire height Z of an object must be inside the zone (both top and bottom of the bounding box)

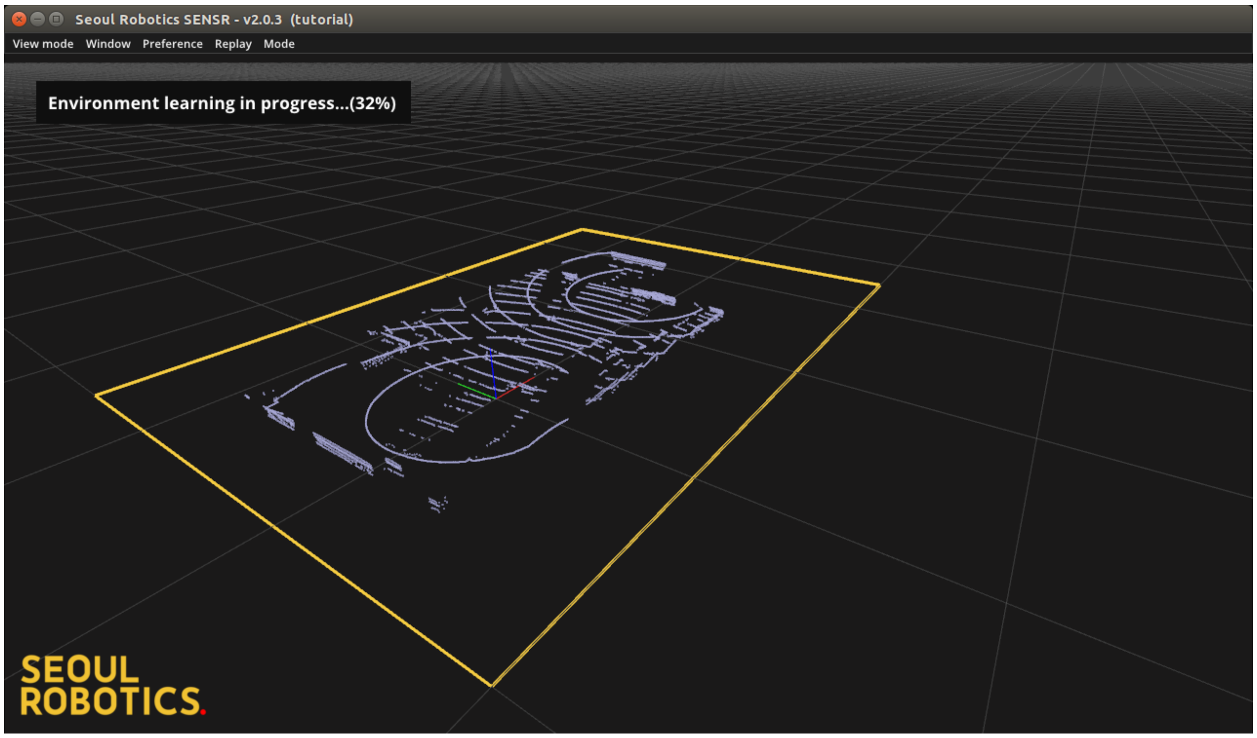
9. Runtime & Object Tracking

When the calibration is ready, click *'Runtime'* to enter runtime mode. Object tracking is only active in runtime mode.

When you click runtime, parts of the point cloud may temporarily disappear as the SENSR software enters the *Environment Learning* phase.

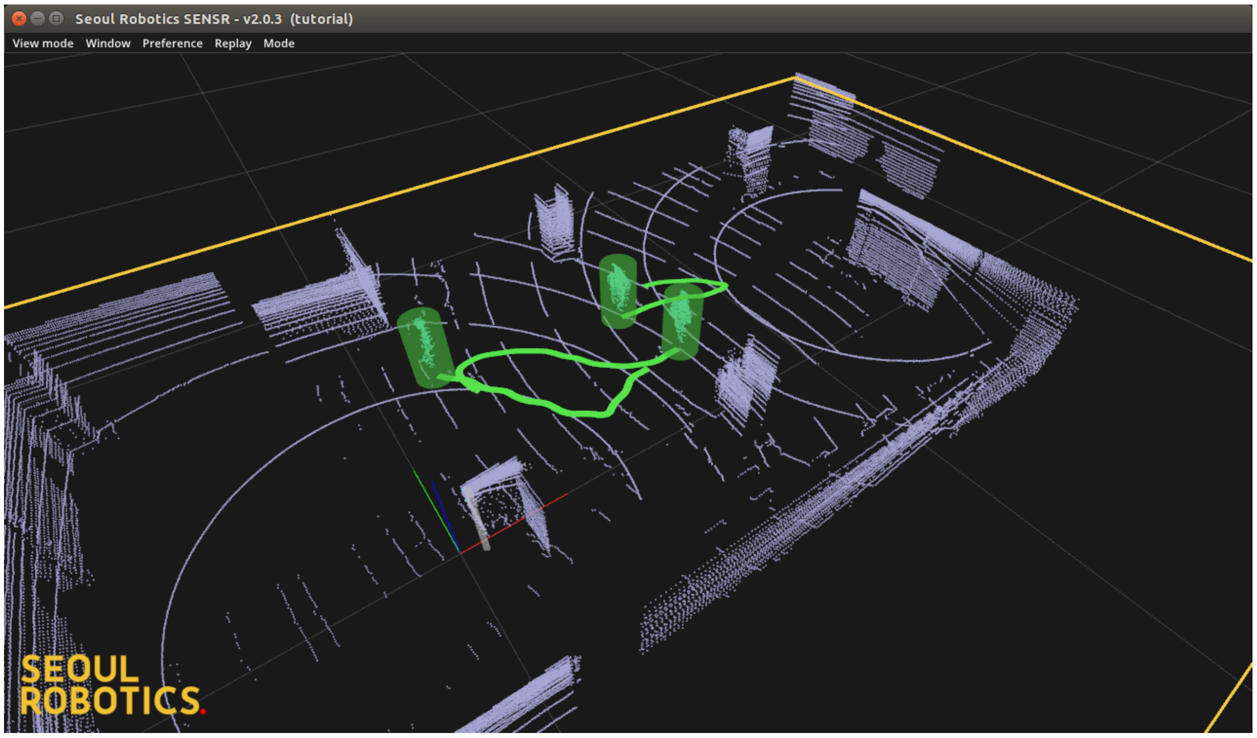
9.1 Environment Learning

SENSR runs the initial environment learning before starting object detection. The Environment Learning phase will learn ground and any static objects in the scene. The duration of this phase depends on the number of sensors, the total coverage area and some of the settings, like resolution.


Seoul Robotics SENSR - v2.0.3 (tutorial)
View mode
Window
Preference
Replay
Mode
Environment learning in progress...(32%)
SEOUL ROBOTICS.

9.2 Runtime

Once the learning phase completes, SENSR will switch into Runtime mode. In this mode any objects will be tracked and the output streams become active.



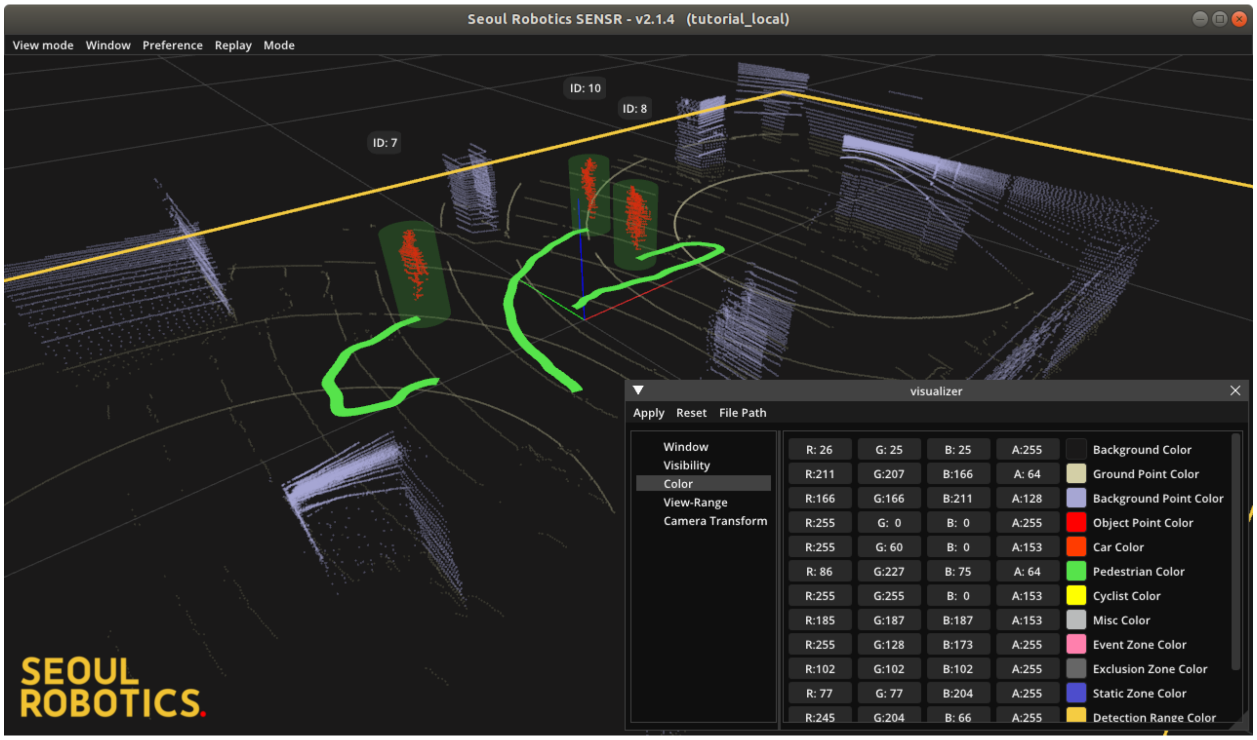
The above image shows the 3D viewport with points and objects in the default colors. The walls, ground and static objects are considered **background** and are both shown in light blue. Any movement that is detected is considered **foreground** and is shown in green.

Important: The foreground and background update at different rates in the visualizer. This is done by SENSR to preserve network and processing bandwidth. You will notice that tracked objects move smoothly through the world while the shadows they generate update at a slower rate. This is normal. The default update rate for foreground objects is 100ms (10 fps) and the default update rate for background is 1s (1 fps).

This can be changed in the settings

It is possible to change the colors of various points and objects in the 3D viewport. To do this, select the *Preference* menu, select *Visualizer* and this will bring up a new screen. You can also use the **F11** keyboard shortcut.

In the window, select the *Color* page. This page is organized in a table with the rows representing the types of items that can be changed and the columns are the color elements (Red, Green, Blue and Alpha/Transparency)



R: 26	G: 25	B: 25	A:255	⬛	Background Color
R:211	G:207	B:166	A: 64	⬜	Ground Point Color
R:166	G:166	B:211	A:128	🟪	Background Point Color
R:255	G: 0	B: 0	A:255	🟥	Object Point Color
R:255	G: 60	B: 0	A:153	🟧	Car Color
R: 86	G:227	B: 75	A: 64	🟩	Pedestrian Color
R:255	G:255	B: 0	A:153	🟨	Cyclist Color
R:185	G:187	B:187	A:153	⬜	Misc Color
R:255	G:128	B:173	A:255	🟪	Event Zone Color
R:102	G:102	B:102	A:255	⬛	Exclusion Zone Color
R: 77	G: 77	B:204	A:255	🟦	Static Zone Color
R:245	G:204	B: 66	A:255	🟨	Detection Range Color

In the above example, the ground is made yellow to show the difference between ground and other background elements. The foreground points are marked in bright red.

Changing colors can help understand tracking performance problems such as making sure that moving objects are correctly seen as foreground and that the ground level is set correctly.

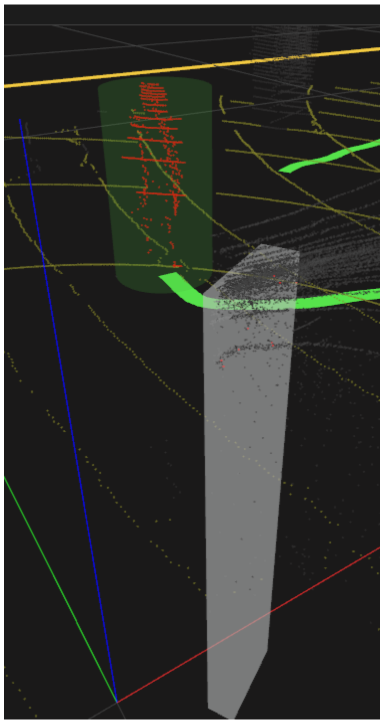
9.2.1 Object Tracking

When an object is detected and tracked, it will be covered in either a cylinder or rectangular bounding box, depending on the classification of the object. The color of the box or cylinder indicates the classification of the object.

The default classification colors are:

- Grey: Unknown/Miscellaneous
- Green: Pedestrian
- Yellow: Bicycle/Motorcycle
- Red: Car or larger vehicle

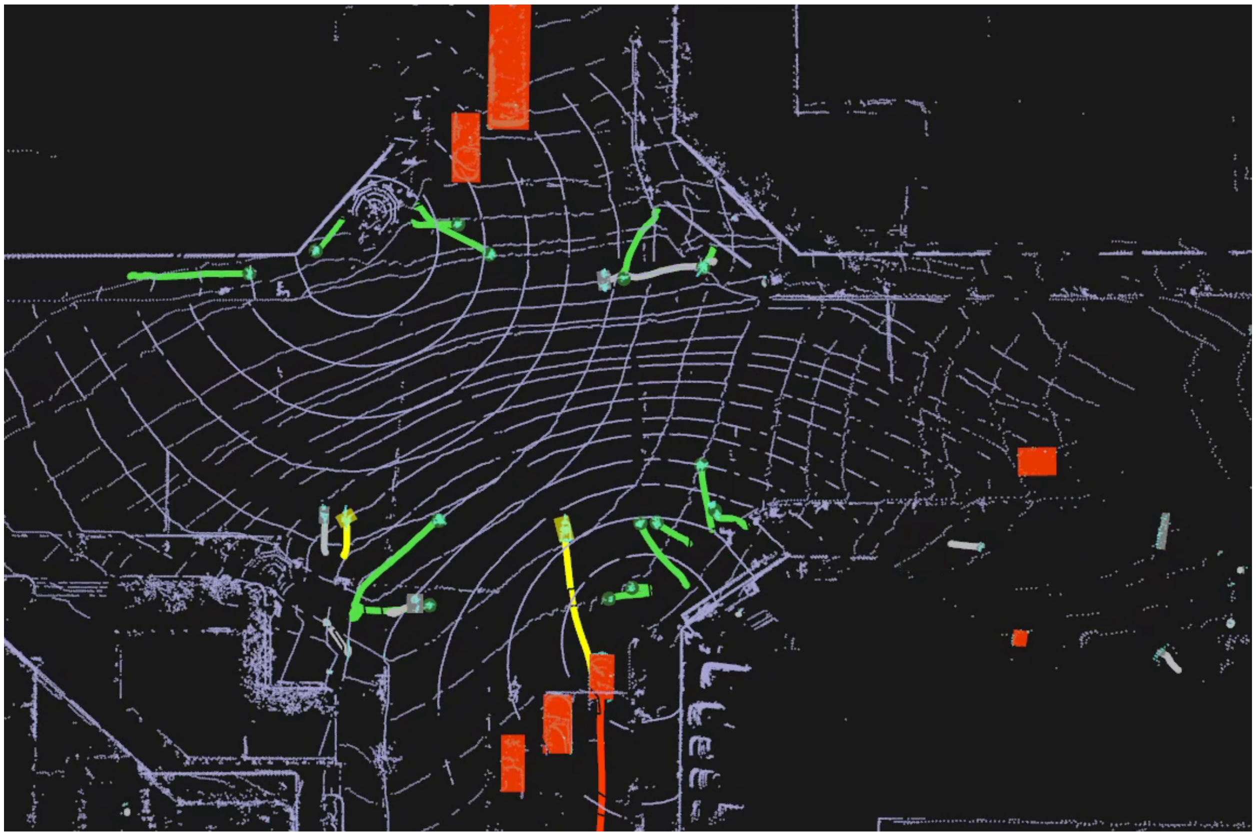
Each tracked object gets a unique ID number that stays with the object until tracking is lost. If the object reappears within (default) 1 second (see *Drifting Period* setting) in the expected location based on previous direction and velocity, the reacquired object will regain its previously assigned ID. If an object is lost for longer it will permanently lose its ID. This means that, for projects that require advanced tracking through a large space while maintaining the ObjectID throughout the entire journey, SENSR requires continuous coverage, taking into account obstacles that could break the line of sight.



In the example on the right a person is detected and tracked as indicated by the green cylinder around the object. There is also a large miscellaneous object in the scene. It is important to note that the majority of noise that is generated by sensors and/or the result of reflections in the scene will be classified as miscellaneous objects. You can choose to ignore these objects by making them invisible. To do this go to: *Preferences→Visualizer→Visibility* and disable the *Draw Misc-Objects* option

It is recommended, for best tracking performance, to surround the targets with sensors. This will help the SENSR system detect and maintain the size and shape of objects as they pass through the field of view.

In the below example , two sensors are placed on diagonally opposite sides of an intersection, allowing a small vehicle to be tracked and classified next to a large vehicle.



9.3 Menu Items

The following menu items are available in Runtime Mode.

9.3.1 View Mode

This menu lists the various view modes that are available.

- *Perspective* - the default view mode with 3D perspective enabled. You have full viewpoint freedom in this mode.
- *Ortho (Top View)* - a top-down view mode with perspective disabled, creating a 2D view. Perfect for using as a top-down floorplan
- *Ortho (Side View)* - a side view mode with perspective disabled, a good mode to check if the ground is level to the grid and to correct any height errors during calibration.
- *Isometric* - a three-quarters view with camera controls locked so only rotation and zoom are available.
- *Reset* - resets the camera viewpoint to the default for the selected view mode.

9.3.2 Window

Two additional types of windows are available.

- *Object* - this window gives additional information about objects in the scene. You can click on an object in the 3D viewport or in the list to display this information.
- *Secondary Window* - this allows you to open a secondary viewport with its own *View Mode* and camera angle.

9.3.3 Preferences

These are the settings that are available during runtime mode. For details about each setting, see Chapter 11.

9.3.4 Replay

This option allows you to record the data or replay a previously recorded data file. For more information, see Chapter 10.

9.3.4 Mode

This will allow you to switch to different modes and add, modify or delete sensors and zones.

- *Sensor Setup* - this will take you back to the sensor settings and calibration screen (see Chapter 6.3).
- *Zone Setup* - this will take you to Zone Setup mode (see Chapter 8)

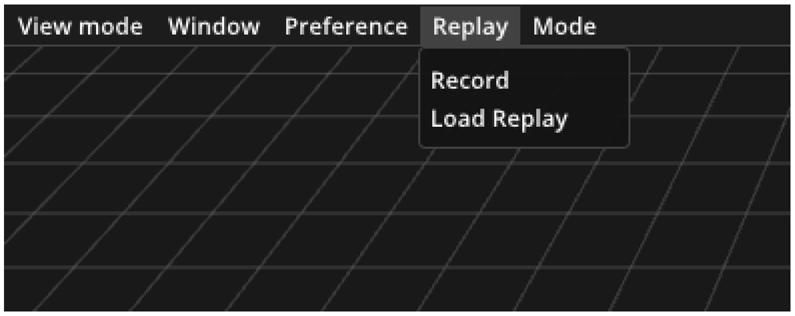
Important: When you enter Sensor or Zone setup mode, you leave Runtime mode. Object tracking will be disabled until you re-enter Runtime mode.

10. Replay Editor

SENSR includes a **replay system** that allows users to record the output of SENSR and replay it at a later time. The replay system is split into two modes:

- **Record:** Record the output data from SENSR and save to a replay file.
- **Replay:** Play a replay file and show the output in the visualizer.

To select either of these modes, you must first be in runtime mode. From the top menu bar, select **Replay** and then either **Record** or **Load Replay**.



10.1 Recording

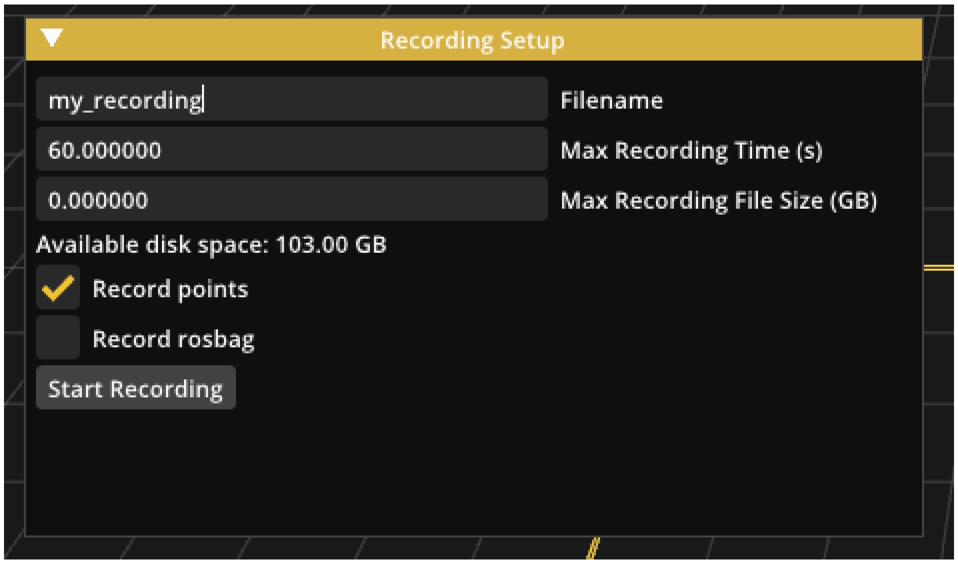
After clicking **Record** in the **Replay** drop-down menu the recording set-up window opens up. Before launching the recording mode the necessary settings need to be configured. These are

- **Filename:** The name of the file the recording is saved to. If the name does not have an extension (e.g. *my_recording* instead of *my_recording.tar.gz*), the extension *tar.gz* will automatically be added to the filename.
- **Max Recording Time:** This number specifies the maximum allowable recording time. Once SENSR has been recording for this amount of time, it automatically stops the recording and saves it to file. The recording can be manually stopped at any point before that, but this parameter is intended to limit the recording time in case there is limited space on the hard drive or if the user wants to record for a specific amount of time.

- **Max Recording File Size:** This number specifies the maximum allowable file size of the recording in gigabytes. Once the recording file size has reached this limit, the recording is automatically stopped and saved to file.

Additionally, the recording set-up includes a few options that can be toggled on/off. Leaving all of these options turned off records only the merged output of SENSR. The additional options include:

- **Record points:** Enabling this saves the point cloud to the replay file so it can be visualized during replay. Note that this drastically increases the recording file size so it is recommended that this option is turned off in the case of limited hard disk space or if you plan on recording for a long time. Recording the point cloud can be useful in case of tracking performance issues to share with the Seoul Robotics team.
- **Record rosbag:** Enabling this records a raw rosbag file inside of the algo node and saves it locally. All ros topics of type *sensor_msgs/PointCloud2* belonging to the algo node will be subscribed to. This rosbag file cannot be used for the replay mode at the moment, but is saved in the respective algo node. Note that rosbags get very big very quickly!


Recording Setup

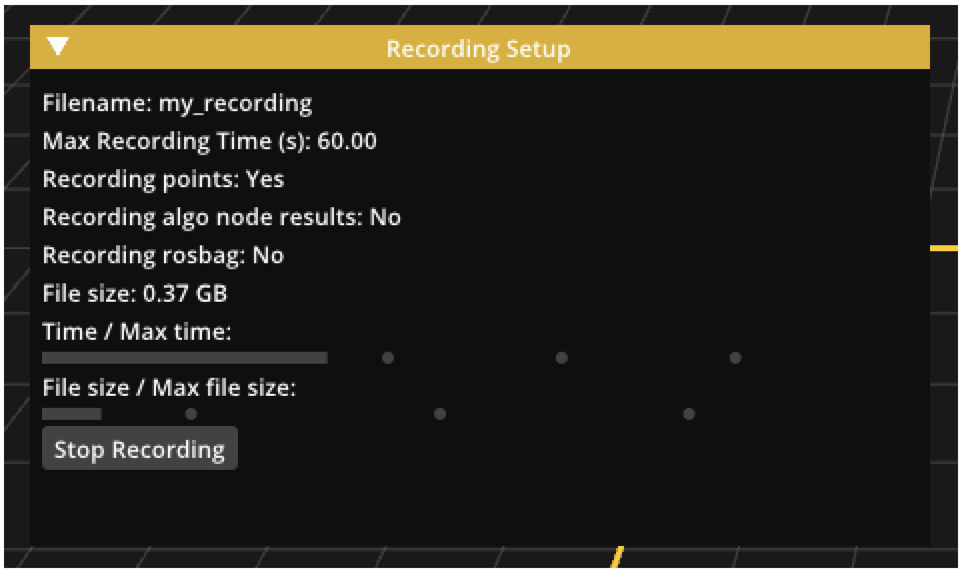
my_recording	Filename
60.000000	Max Recording Time (s)
0.000000	Max Recording File Size (GB)

Available disk space: 103.00 GB

✓ Record points

Record rosbag

When all the settings have been properly configured, the **Start Recording** button can be clicked to launch the recording mode. The recording window now shows information about the recording and progress bars for the **Max Recording Time** and **Max Recording File Size**. Pressing the **Stop Recording** button at any time will stop the recording and save to file. The replay system also keeps track of the available disk space and automatically turns off the recording if there is not enough space remaining. On finishing the recording, the replay file is saved in the replay subfolder of the project folder. The replay files there can be deleted to free up space if they are not needed anymore.

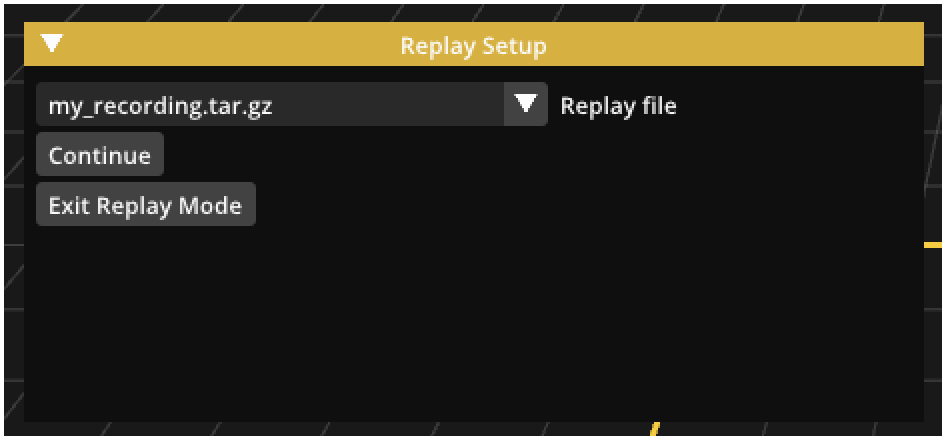


Filename: my_recording
Max Recording Time (s): 60.00
Recording points: Yes
Recording algo node results: No
Recording rosbag: No
File size: 0.37 GB
Time / Max time:
File size / Max file size:

10.2 Replaying

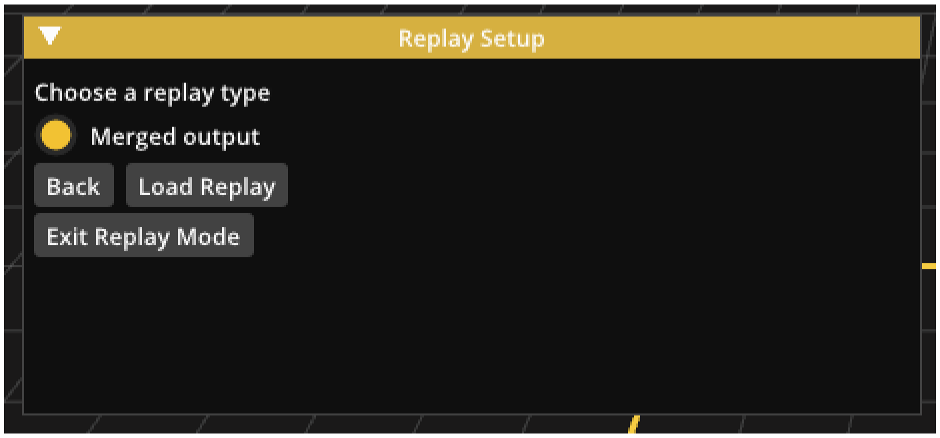
Once a replay file has been recorded it can be replayed at any time by pressing the **Load Replay** button in the **Replay** drop-down menu and launching the replay mode.

This opens up a window to choose the replay file from a list. The list is generated from the replay files saved in your project folder (inside *[project_folder]/replays/*). Choose a file from the list and press the **Continue** button.

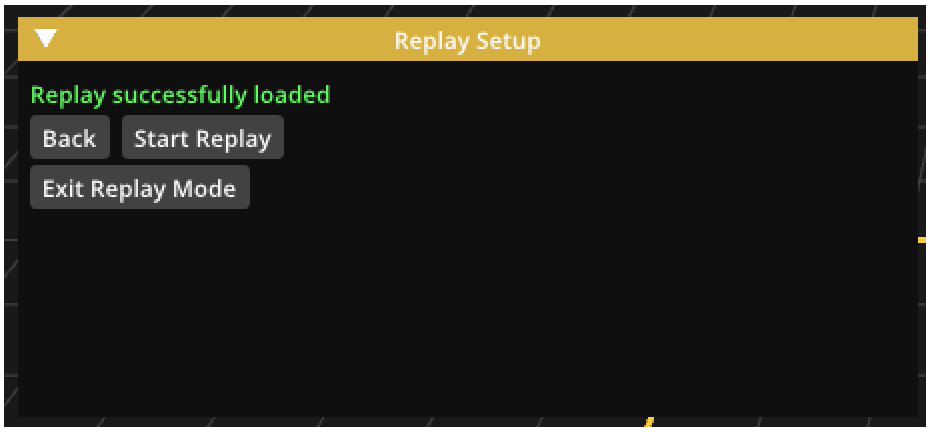


The next step is to choose a replay type. The file is verified to check which replay types are available and any type that is not present is greyed out. Note that in this release there is only one available replay type (*Merged output*).

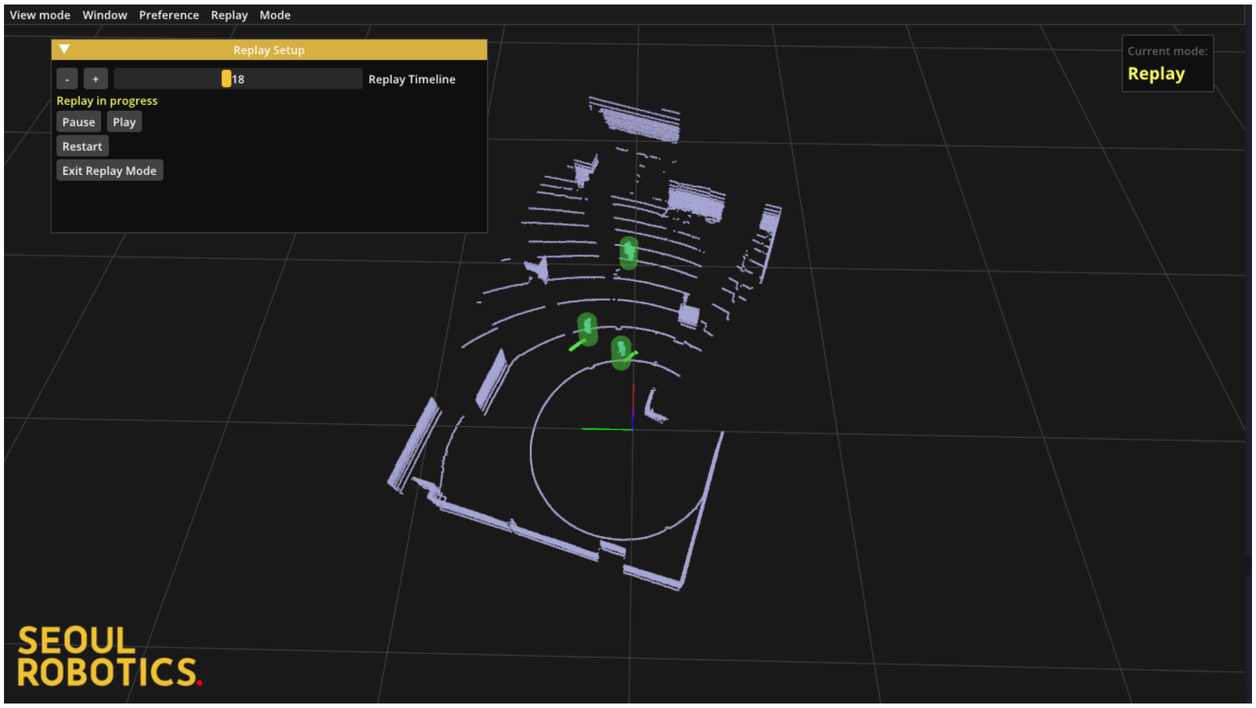
- **Merged output:** The output from SENSR is replayed from file without running any additional algorithm. This visualized result should be nearly identical to the result displayed during recording. This mode is what users will most frequently want.



Once the type is selected the **Load Replay** button can be pressed. This loads the replay, verifies that the replay file is not corrupted, and ensures the version of the replay file is compatible with the current application version. Once the loading is complete, the **Start Replay** button can be pressed.



The replay will start playing and can be paused and restarted from the beginning. The timeline can be interacted with to move forward or backward in the replay. At any point, the replay mode can be excited by pressing the **Exit Replay Mode** button.



11. Settings

11.1 Common setting

Common setting can be found in **Preference > Algorithm (common)**

Location	Name	Description	Unit	Default
Common > Output	Publish Point-Cloud	The level of publishing point cloud to output port. This setting will greatly increase the bandwidth between the algo nodes and the master node! 0: Nothing published 1: Only object point published 2: Both object & background points published	int	2
	Point-Cloud Bandwidth Reduction	This decreases the communication burden by reducing pointcloud data size. This option affects the update method which sequentially updates each lidar is in calibration mode and downsample background and ground points in runtime mode.	boolean	false
	Result-Integrator Update-Frequency	Frequency with which new results from algo nodes are integrated. A lower value results in a more rapid integration of new algo node results but might increase CPU-Usage.	seconds	0.025
	Downsampling Resolution	Set resolution of downsampling method of pointcloud bandwidth reduction. Value can be between 0 to 1.	Float	0.3
	Point-Cloud Update Frequency	Time between updates of background point cloud information from algo nodes to the master node. Updating frequently can be resource-intensive..	seconds	1.0
	Sensor Update Frequency	Time between updates of sensor information from algo nodes to the master node.	seconds	0.1
	Trigger Zone Events with MISC Objects	Publish Zone Event Messages, triggered from MISC labeled objects.	boolean	False

	Output Sending Type Selection	Method for sending and updating new results. 0 - Instant Sending: Publish result whenever a new one is available. 1 - Fixed-Period Sending: Publish the most recent result periodically. The published result might already have been published in a previous frame. 2 - Fixed-Period with Max-Delay: Publish the most recent result periodically. If there is no new result, the sending is postponed until a new result is received or a maximum delay time has passed.	int	1
	Update Frequency	Publish frequency for output sending type 1 - Fixed-Period Sending and type 2 - Fixed-Period with Max-Delay	seconds	0.1
	Max Delay	Maximum delay of output sending for the output sending type 2 - Fixed-Period with Max-Delay	seconds	0.05

11.2 Master setting

Master setting can be found in **Preference > Algorithm (master)**

Location	Name	Description	Unit	Default	Range
Output-Merger > Tracking History	Tracking History Length	The number of history positions to keep.	int	100	[0, +Inf.]
	History Smoothing Level	Trajectory smoothing level	int	3	[0, 3]
Output-Merger > Algo-Node Result Timeouts	Algo-Node Object Result-Timeout	If a point cloud receival time of an result from an algo node and the latest receival time differ by more than this timeout, the algo node result is removed from result aggregation. For the point result, the point-cloud update frequency is added to this timeout. For the sensor statuses, the sensor update frequency is added to this timeout.	seconds	0.2	[0.1, +Inf.]
	Algo-Node Point Result Timeout		seconds	0.2	[0.1, +Inf.]
	Algo-Node Sensor Status Timeout		seconds	0.2	[0.1, +Inf.]
Output-Merger > Trajectory Prediction	Number of Prediction Steps	Number of prediction steps	int	10	[0, +Inf.]
	Prediction step Time-Horizon	Time step between two predicted points	seconds	0.1	[0, +Inf.]

	Max. Acceleration Filter Threshold	Maximum acceleration for prediction	m/s^2	15.0	[0, +Inf.]
Output-Merger > Multi Algo-Node Merger	Overlap-Region Map Resolution	Resolution of overlap region map	meters	5.0	[0.1, +Inf.]
	Max-Box-Distance to Split	Max distance to split over merged objects	meters	0.75	[0, +Inf.]
	Grow-Size x-Tolerance	x-size increasing tolerance to control objects association	meters	2.0	[0, +Inf.]
	Grow-Size y-Tolerance	y-size increasing tolerance to control objects association	meters	1.5	[0, +Inf.]
	Grow-Size Rate Tolerance	size ratio tolerance to control objects association		1.2	[0, +Inf.]
	Grow-Size Addition-Tolerance	size addition tolerance to control objects association	meters	0.5	[0, +Inf.]
	Association Distance Threshold	Distance threshold to associate objects in two nodes	meters	2.0	[0, +Inf.]

11.3 Algorithm setting

To change algorithm settings you can open the **algorithm** window (with the **Preference** > **Algorithm (*ip_address*)** in the menu bar). A description of the parameter specification and a more detail of how to tune the parameter can be seen below. Be advised to change algorithm parameters with caution, as wrong parameters can negatively affect the performance.

Location	Field	Name	Description	Unit	Range
Parameters > Object		Allow Floating-Object	If True, min. z of the object is set to be the min. z-value of the object bounding box instead of ground height.	True/ False	
		Box Orientation Resolution	Object bounding box orientation resolution.	degrees	[3.75, 15.0]
Parameters > Sanity-Checks	Tracked Object	Max-Dimensions (W, L)	The maximum object size which can be tracked.	meters	[0, +Inf.]
Parameters > Tracking		Min. Points for Tracking	The minimum number of points of an object to be tracked.	-	
		Validation Period	Period of checking validity in the early stage of tracking. This time determines length of the VALIDATION period, or how quickly a	seconds	[0, +Inf.]

			new object will be tracked and classified		
		Invalidation Period	Period of short term prediction when tracking is lost while in the VALIDATION period. A longer period will allow for objects that are at the edge of detection range to be tracked, but will be more likely to introduce false alarms.	seconds	[0, +Inf.]
		Drifting Period	Period of short term location prediction when tracking is lost in TRACKING status. A longer period can help with obstructions, but can lead to object ID switching in busy environments	seconds	[0, +Inf.]
Parameters > Prediction		Time Horizon	Number of predicted points		
		Time Step	Prediction time step	seconds	[0, +inf.]
Pipeline > Ground Detector	Elevation Ground Detector	Max. Ground Height	Maximum height for ground segmentation.	meters	[-0.5, +Inf.]
Pipeline > Background Detector	Backgrou nd Detector	Apply	If True, use the background detector.	True/ False	
		Accumulation Frame Count	Number of accumulated frames to detect background points .	-	[1, 20]
		Time to Initialize Background	Initiated period to learn background points.	seconds	[0, +Inf.]
		Time to Become Background	The period for static objects becomes background.	seconds	[1, +Inf.]
		Time to Become Foreground	The period for a background object becomes foreground when it starts moving.	seconds	[1, +Inf.]
		Background Reset Rate	Changing rate from foreground to background when no point is detected		[0, +Inf.]
		Use Multi-lidar Background Fusion	Fuse background information from multi-lidar	True/Fal se	
		Resolutions (Range, Azimuth, Elevation)	Range (distance), Azimuth angle and Elevation angle resolution.	meters, degree, degree	[0, +Inf]
Pipeline > Clusterer		x-Resolution	Grid resolution in x-axis.	meters	(0, +Inf.]
		y-Resolution	Grid resolution in y-axis.	meters	(0, +Inf]
		Cell Point Threshold	Minimum points per cell for clustering		[1, +Inf.]
		Cluster Point Threshold	Minimum points per cluster to detect		
Pipeline > Tracker	Hybrid-Tr acker	Apply	If True, use Hybrid-Tracker	True/ False	
		Association-Dist ance	Distance threshold to associate objects in two consecutive frames.	meters	[0.01, +Inf]

		Min. Object Radius	Minimum object radius to be tracked.	meters	[0, +Inf]
		Max. Hole-Size	Maximum distance of objects which can be merged into one object.	meters	[0, +Inf]
		Merge Object Level	Object merging level to decide how easily small closed objects can be merged into one object.	-	[0, 10]
		Object-Size Shrinking-Rate	Shrinking rate of tracked object size. In case of occlusion, the size of a tracked object will be learnt and kept with this shrinking rate.	-	[0, 1]
		Use Object Splitter	Apply splitter check to over merger objects	True	
		Splitting Distance Threshold	Distance threshold to split objects	metters	[0, +Inf.]
		Splitting Min. Trajectory Distance	Minimum trajectory distance to split objects	metters	[0, +Inf.]
		Splitting Min. Object Lifetime	Minimum lifetime of objects to apply splitter check	seconds	[0, +Inf.]
	Human-Tr acker	Apply	If True, use Human-Tracker	True/ False	
		Association-Dist ance	Distance threshold to associate objects in two consecutive frames.	meters	[0.01, +Inf]
		Min. Object Radius	Minimum object radius to be tracked.	meters	[0, +Inf]
		Merge/Split Size Mul. Threshold	Size ratio threshold that allows to merge/split		[0, +Inf]
		Merge/Split Size Add. Threshold	Size increasing/decreasing threshold that allows to merge/split	meters	[0, +Inf]
		Min. Time for Splitting	Minimum lifetime of objects to apply merger/ splitter check	seconds	[0, +Inf]
Pipeline > Classifier	Target Classifica tion Classes	CAR/PED/CYC/ MISC	The classes the output of the classifier will include. E.g. if only PED(estrian) and MISC(ellaneous) are chosen all the objects will be classified as either PED or MISC.	True/ False	
	Size Classifier for Big Objects	Apply	If True, use the size classifier for big objects. If used, all objects which have size bigger than **Min Length, Width, Height** will be classified as one of **Include Classes**.	True/ False	
		Min Length, Width, Height	Min length, width, height to classify as big objects.	meters	[0, +Inf]
		Include Classes	The classes the output of the **size classifier for big objects** will include.	True/ False	
	Size Classifier for Small	Apply	If True, use the size classifier for small objects. If used, all objects which have size in the **Length Range, Width Range, Height**	True/ False	

	Objects		**Range** will be classified as one of **Include Classes**. If an object is classified as a big object, it won't be classified as a small object even though it's in the small object range.		
		Length Range/ Width Range/ Height Range	The range of length, width, height to classify as small objects.	meters	[0, +Inf]
		Include Classes	If True, the classes the output of the **size classifier for big objects** will include.	True/ False	
	Velocity Classifier	Apply	If True, use the velocity classifier.	True/ False	
		Non-MISC Min Velocity	Minimum velocity which object will not be classified as a MISC.	meters	[0, +Inf]
		Non-MISC Min Displacement	Minimum displacement which object will not be classified as a MISC.	meters	[0, +Inf]
		PED Max Velocity	Minimum velocity which object will not be classified as a PED.	km/h	[0, +Inf]
	ML Classifier	Apply	If True, use the machine learning classifier. If applied, it will be used only after using size classifier and velocity classifier. E.g, After size classifier and velocity classifier, the object is classified as either PED or CYC, ML classifier will decide whether it's a PED or a CYC.	True/ False	
		Min Num Points	Minimum number of points to use ML classifier.		[0, +Inf]
		Max Num Objects To Classify	Maximum number of objects can be classified by ML classier.		[0, +Inf]

11.4 More on algorithm setting

11.4.1 Detection range

It is recommended to change some parameters to better fit each specific use case. As mentioned in previous sections, *detection range* is one of the important parameters that should be tuned carefully to save computational resources, reduce noise and false detections (prefered to 6.3.4, 7.1.8). If you want to change *detection range* while in runtime mode, go back to *sensor setup mode* (with the **Mode** > **Sensor Setup** in the menu bar) and click the *Algo Node Settings* gear icon.

11.4.2 Tracked Object Max-Dimensions

Tracked Object Max-Dimensions (W, L) need to be bigger than the maximum size of objects you want to track and classify in the scene but don't set it too big compared to that size.

11.4.3 Background Detector and Zone

Background detector is used to automatically set *exclusion zones* (called *background zones*) to ignore static objects in the detection pipeline. If an object stays still for more than *Time to Become background* seconds, the *background detector* will set a *background zone* around that object. In consequence, that object will become background and won't be tracked. If a background object moves away from its background zone, it needs *Time to Become Foreground* seconds so that the zone is removed. Since lidar is noisy and some objects can vibrate (vegetations, flags, ...), the *background detector* can't completely remove all the static objects. It's recommended to use *exclusion zones* (prefered to 8.1.2) for the region which you don't want to cover.

If you also want tracked objects which can be static for longtime (e.g. track cars in the parking lot), you could disable *Background Detector* and use only *exclusion zones* instead.

11.4.4 Tracker

Human-tracker is more optimized for human only tracking applications. If lidar is used in the scene which doesn't have cars and cyclists (e.g. indoor), *human-tracker* is recommended. If lidar is used outdoors (which have cars/cyclists), *Hybrid-Tracker* would perform better.

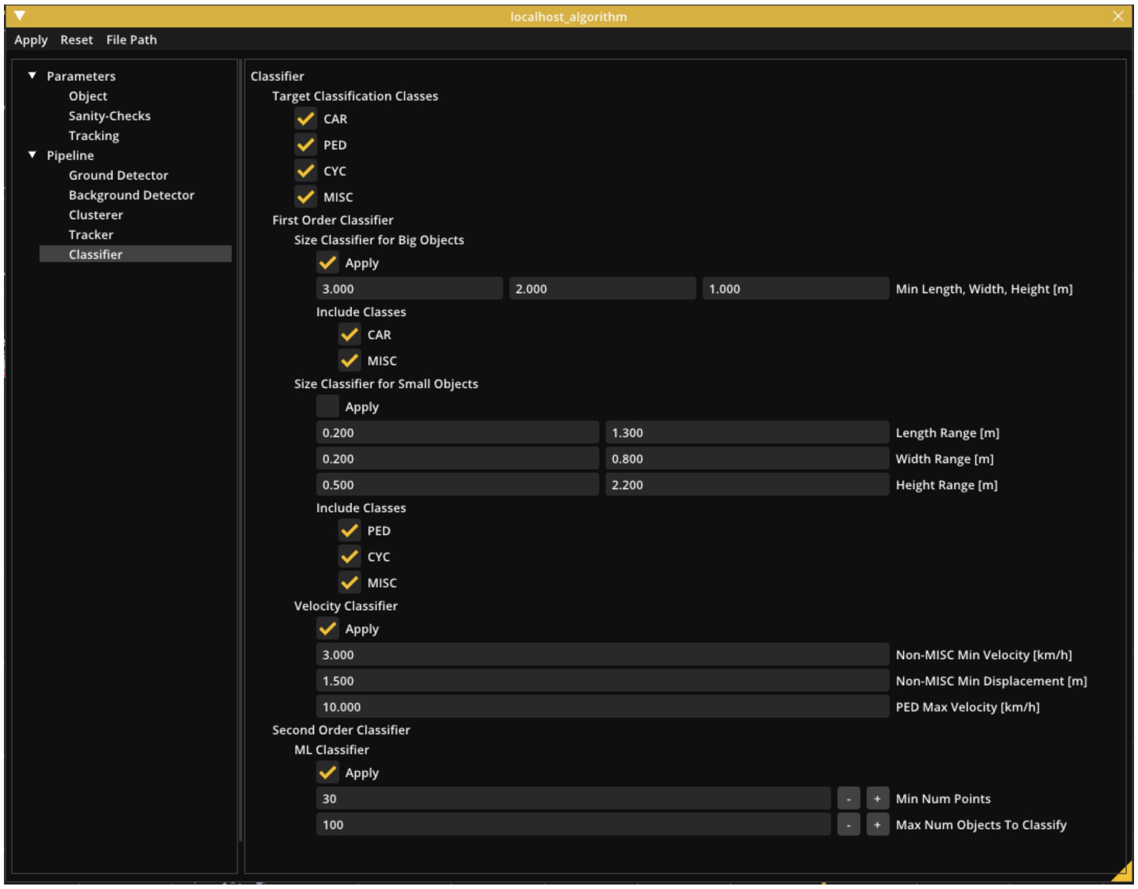
Since lidar point-clouds are sparse, a big car can break into multiple small objects. *Hybrid-Tracker* can merge those small objects together. The *Max. Hole-Size* and *Merge Object Level* parameters help to control this merging process. *Max. Hole-Size* is the maximum distance of those small objects which can be merged while *Merge Object Level* indicates how easy the merge process is. If lidars are used to track big objects (big trucks/containers) and there aren't many people, increasing *Max. Hole-Size* and *Merge Object Level* can help to detect the big object better. But if the scene is a crowded intersection with many pedestrians those values need to be kept small to avoid merging closed distance pedestrians to one object. *Merge Object Level* can have value in the range [0, 10]. We recommend using *Merge Object Level* 3-4 for crowded intersections and 7-8 for highways with many big trucks cases.

11.4.5 Classifier

Use *Target Classification Classes* can reduce false classification. E.g. If lidars are used on highways to detect cars/cyclists, it recommends choosing CAR, CYC, MISC as target classes. And if lidars are used indoors, choosing PED, MISC as target classes could perform better.

If only one target class is selected, then all objects are classified as the selected class without considering properties of the objects (i.e. the classification is skipped). In the case that no target class is selected, all objects are classified as MISC.

First order classifier (size based and velocity based) will be used first, and only if it can't decide which class the object belongs to the *second order classifier* (machine learning based) will be applied. Let's consider the following classifier setting and an object of size (4.5m, 2.7m, 1.5m).



Classifier
Target Classification Classes
✓ CAR
✓ PED
✓ CYC
✓ MISC
First Order Classifier
Size Classifier for Big Objects
✓ Apply
3.000 2.000 1.000 Min Length, Width, Height [m]
Include Classes
✓ CAR
✓ MISC
Size Classifier for Small Objects
Apply
0.200 1.300 Length Range [m]
0.200 0.800 Width Range [m]
0.500 2.200 Height Range [m]
Include Classes
✓ PED
✓ CYC
✓ MISC
Velocity Classifier
✓ Apply
3.000 Non-MISC Min Velocity [km/h]
1.500 Non-MISC Min Displacement [m]
10.000 PED Max Velocity [km/h]
Second Order Classifier
ML Classifier
✓ Apply
30 Min Num Points
100 Max Num Objects To Classify

The object is classified as either CAR or MISC by *Size Classifier for big Objects*. If it moves at a speed 20km/h, it is not a MISC or a PED by *Velocity Classifier*. So it will be classified as a CAR. In case it's not moving, *ML Classifier* will be used to determine if it's a CAR or a MISC.

11.4.6 Dealing with limited computational resources

If running with multiple lidars or using a computational-resource-limited machine, you can improve the speed of the algorithm by:

- Optimize the detection range (prefered to 11.3.1).
- Increase resolution values in **Pipeline > Clusterer**. Resolution 0.1 x 0.1 means that points in a grid of size 0.1 x 0.1 will be grouped in one object. Increasing those

values can help the clusterer run faster with the trade off that closed objects can be merged together.

- Increase resolution values in **Pipeline > Background Detector > Range, Azimuth, Elevation.** Increasing those values can help the Background Detector run faster with the trade off that objects which are closed to background can become background.

11.5 Rendering Config Editor

Rendering setting can be found in **Preference > Visualizer (or press F11)**

Location	Name	Description	Default
Window	Default Window-Width	Initial width of the application window.	1080
	Default Window-Height	Initial height of the application window.	720
Visibility	Object Points	Enable or disable showing points belonging to objects.	True
	Ground Point	Enable or disable showing ground points. (e.g. floor.)	True
	Background Points	Enable or disable showing background points. (e.g. wall or desk…)	True
	Grid	Enable or disable showing squared Grid.	True
	Grid Circular	Enable or disable showing circular Grid.	False
	Axes	Enable or disable showing the world XYZ coordinate axes.	True
	Objects	Enable or disable showing tracked objects. (e.g. pedestrian)	True
	Misc Objects	Enable or disable showing miscellaneous objects.	True
	Object Annotation	Enable or disable showing id on top of each tracked object.	False
	Object Trail	Enable or disable showing the object trail.	True
	Predicted Trajectory	Enable or disable showing the predicted trajectory of moving objects.	False
	Map Image	Enable or disable showing a map image.	True

	Detection Range	Enable or disable showing the detection range of each algo node.	True
	Points Outside of Detection Range	Enable or disable showing points outside of the detection range.	False
	Lidar Name	Enable or disable showing name of each Sensor.	False
	Algo Node Name	Enable or disable showing name of each Algo Node.	False
	Zone Name	Enable or disable showing name of each Zone.	False
Color	Background Color	Color of the background.	
	Ground Color	Color of the ground points.	
	Background Color	Color of the background points.	
	Object Color	Color of points belonging to objects.	
	Car Color	Color of cars.	
	Pedestrian Color	Color of pedestrians.	
	Cyclist Color	Color of cyclists.	
	Misc Color	Color of miscellaneous objects.	
	Event Zone Color	Color of event zones.	
	Exclusion Zone Color	Color of exclusion zones.	
	Detection Range Color	Color of the detection range outline.	
View Range	Min point height	Minimum height to show points (Unit: meters) The point lower than this value will not be shown even if you enabled Env. Points ON.	-10.0
	Max point height	Maximum height to show points (Unit: meters) The point higher than this value will not be shown even if you enabled Env. Points ON.	10.0
Camera Transform	Rotate Y	Rotation of the camera along the y-axis	180.0
	Rotate X	Rotation of the camera along the x-axis	35.0
	Zoom	Zoom of the camera	100.0
	Position	Position of the camera	(0,0,0)

12. REST API

SENSR can be remote-controlled by REST API. You can change the parameters, add and remove zones, and restart the pipeline with REST API.

12.1 Specification

Full documentation of REST API specification including usage examples is here: https://seoulrobotics.github.io/sensr-api-doc/2.1.4/.

13. Output

SENSR encodes data processing results with Protobuf and sends it through TCP using Websocket. With SENSR SDK, you can easily receive SENSR output in your C++ client. SENSR SDK code is here:
https://github.com/seoulrobotics/sensr_sdk/releases/tag/v2.1.7.

We also provide JavaScript and Python interfaces inside the SDK to easily work with SENSR output in those languages. For more information on how to use JavaScript and Python, please refer to the relevant readme in the SDK repository.

13.1 Communication Specification

Communication Protocol	TCP (Websocket)
Data Encoding	Protobuf 3
Output Port	5050
Protobuf Message	OutputMessage

Communication Protocol	TCP (Websocket)
Data Encoding	Protobuf 3
Output Port	5051
Protobuf Message	PointResult

13.2 Output Data Specification

The Protobuf data specification for all output messages can be found and downloaded from these links:
https://github.com/seoulrobotics/sensr_proto/blob/69097d7b6b7a5ccf0c0d499b11477aba4441b9ce/output.proto
https://github.com/seoulrobotics/sensr_proto/blob/69097d7b6b7a5ccf0c0d499b11477aba4441b9ce/point_cloud.proto

13.3 Streaming Data (Tcp and Websocket)

This is a live data stream that can be parsed and processed by external applications.

13.3.1 Output Message

This is a main message which includes all the output results during runtime of SENSR product. But SENSR does not always fill all the fields because of bandwidth issues. SENSR has update rates internally and each field of this message will be filled according to these rates.

Type: Continuous Stream (every frame)

Field	Field size	Data type
timestamp	1	TimeStamp
stream	0-1	Stream message
event	0-1	Event message
custom	0-1	Custom message

13.3.1.1 Stream Message

This message includes output results of SENSR. Each field will be filled according to the update rates of each field.

- objects : every frame
- zones : every 10 sec.
- system : every 1 sec.

Type: Continuous Stream

Field	Field size	Data type
objects	0-*	Object
zones	0-*	ZoneConfig
health	0-1	SystemHealth

13.3.1.1.1 Object

This is the new standard object tracking output. **points** and **histories** are optional fields. You can receive these optional fields by setting a specific configuration("common.output.publish_point_cloud").

Type: Continuous Stream (every frame)

Field	size	Data type	Note
id	1	int32	
label	1	Enumeration	0 = None 1 = Car 2 = Pedestrian 3 = Cyclist, 4 = Misc
confidence	1	float	0.0 - 1.0
bbox	1	BoundingBox	
velocity	1	Vector3 (float)	m/s + direction
tracking_status	1	Enumeration	0 = NONE 1 = VALIDATING 2 = INVALIDATING 3 = TRACKING 4 = DRIFTING 5 = EXPIRED (**VALIDATING** : *Checking validity in the early stage of tracking* **INVALIDATING** : *Short term prediction when tracking is lost in VALIDATING status* **TRACKING** : *Stable tracking* **DRIFTING**: *Short term prediction when tracking is lost in TRACKING status* **EXPIRED**: *Expired tracking*)
points	0-*	byte	Object shape (optional)
history	0-1	History	Object tracking history (optional)
prediction	0-1	Prediction	Object position prediction (Not Available)
zone_ids	0-*	int32	List of Zone ids this object currently occupies.

13.3.1.1.2 Zone Config

This is a low frequency message to communicate the zone shapes and locations for you who do not want to use the REST API.

Type: Continuous Stream (low frequency, 1 per 10 seconds or slower)

Field	size	Data type	Note
id	1	int32	Zone id
name	1	string	Zone name
pbox	1	PolygonBox	Array of X,Y,Z
type	1	Enumeration	0 = None 1 = Event 2 = Exclusion

13.3.1.2 Event Message

This message will be filled whenever SENSR detects events. There are 3 types of events. zone, losing, system. Before parsing each field, you need to check the size of each field. The size 0 means no event related to that field.

Type: Trigger on event

Field	Field size	Data type
zone	0-*	ZoneEvent
losing	0-*	LosingEvent
system	0-1	SystemHealth

13.3.1.2.1 Zone Event

This is an event triggered message. This message gets sent when an object interacts with a zone. Four interactions are: Enter Zone, Exit Zone, Loitering (more than X seconds in a zone) and Velocity (speed above X m/s in a zone)

Type: Trigger on event

Field	size	Data type	Note
timestamp	1	TimeStamp	Event trigger time.
id	1	int32	Zone id
type	1	Enumeration	0 = None 1 = Entry 2 = Exit 3 = Loitering 4 = Exceed speed
object	1	ZoneEvent_Object	Object information

13.3.1.2.1.1 Object in ZoneEvent

ZoneEvent saves an object of this type.

Field	size	Data type	Note
id	1	int32	
position	1	Vector3 (float)	Object location at time of trigger
heading	1	Float	Yaw (0.0-2.0Pi)
velocity	1	Vector3 (float)	m/s + direction (optional) Filled in case of overspeed events.

13.3.1.2.2 Losing Event

SENSR will send this message when an object is no longer detected.

Type: Trigger on event

Field	size	Data type	Note
timestamp	1	TimeStamp	
id	1	int32	Object id
position	1	Vector3 (float)	Last known location of object
heading	1	float	Yaw (0.0-2.0Pi)

13.3.1.3 Custom Message

This message includes additional information of SENSR.

Type: Continuous Stream

Field	size	Data type	Note
field_of_regard	0-*	PolygonBox	
bg_learning_progress	1	float	Background learning progress (0.0 - 1.0)

13.3.2 Point Result

This is an optional message. If a user wants to get raw point clouds, the user can use this. To get this, the user needs to enable a specific configuration("common.output.publish_point_cloud" = 2) in advance. **This message increases lots of burden in transfer bandwidth.**

Type: Continuous Stream (low frequency, 1 per 10 seconds or slower)

Field	size	Data type	Note
points	0-*	PointCloud	
uid	1	string	deprecated

1.3.3.2.1 PointCloud

Field	size	Data type	Note
type	1	Enumeration	- 0 = None - 1 = Ground - 2 = Background - 3 = Raw
id	1	string	Identifier of sensor Format: - In Calibration mode (algo uid + “#” + lidar topic) - In Runtime mode (meaningless)
points	*	byte	

13.3.3 System Health

This is part of the Health Monitoring System. This is both a continuous low frequency stream (acting as a heartbeat) and an event trigger stream

Type: Continuous Stream + Trigger on event

Field	size	Data type	Note
master	1	Enumeration	0 = None 1 = OK 2 = Storage Shortage 3 = SlowDown Error 4 = Internal Error
nodes	1-*	map<string, NodeHealth>	Each algo node health

13.3.3.1 Node Health

This is part of the System Health message.

Field	size	Data type	Note
status	1	Enumeration	0 = None 1 = OK 2 = ROS Error

			3 = Lost Connection
sensors	1-*	map<string, Enumeration>	0 = sensor dead 1 = sensor alive

13.3.4 Shared Data Types

13.3.4.1 Bounding Box

This is part of the Object message.

Field	size	Data type	Note
position	1	Vector3 (float)	In meters (1.0f = 1m)
size	1	Vector3 (float)	X: Relative to Yaw (longitudinal) Y: Relative to Yaw (lateral) Z: Height
yaw	1	Float	Heading (0.0-2.0Pi)

13.3.4.2 Polygon box

This is part of the Zone Config message.

Field	size	Data type	Note
points	3-*	Vector2 (float)	Shape of zone in 2D
min_z	1	float	Base point in z axis
max_z	1	float	Height of zone

13.3.4.3 History

This is part of the Object message.

Field	size	Data type	Note
states	0-*	History.State	List of history state

13.3.4.4 History.State

This is part of the Object message.

Field	size	Data type	Note
positions	1	Vector3 (float)	object’s tracked XYZ position.
timestamp	1	Timestamp	timestamp of the tracked XYZ position.

13.3.4.5 Prediction

This is part of the Object message.

Field	size	Data type	Note
positions	0-*	Vector3 (float)	List of predicted positions of an object

14. SDK

SENSR comes with a Software Development Kit (SDK) to make it easy to start working with our software and especially the output stream.

The latest release of the SENSR SDK can be found here:
https://github.com/seoulrobotics/sensr_sdk/releases

You can either download the zip file or git clone the repository

For instructions on how to install the SDK for SENSR 2.1, follow this link:
https://github.com/seoulrobotics/sensr_sdk/tree/v2.1.4

Follow the instructions in the link closely and do not forget to build and install protobuf for example.

The best and fastest way to get started with building software that listens to and parses the live data output stream is to build the console_output_sample project by following these steps:

- Start SENSR and run live data or the tutorial
  - Make sure to be in Runtime mode.
- Open up a terminal (CTRL-ALT-T shortcut)
- Navigate to the SDK installation folder (we recommend installing it at: `/home/<user>/sensr_sdk`
- Type: `./build_console_sample.sh`
  - Wait for the process to finish.
- Type: `./build_console/console_output_sample localhost object`

This will show the number of objects and ID for each object in the scene.
The code for this sample can be found at `sensr_sdk/samples/console_output/main.cpp`
You can use your favorite text editor to change the code and then run the steps above again to recompile the console_output_sample project with your changes.

15. Advanced Recording

If you contact Seoul Robotics you may be asked to do an in-depth recording for our engineers to be able to debug the system. The instructions for this are as follows:

1. Run SENSR and run your setup. Make sure the system is in runtime mode
2. Open a Terminal (CTRL-ALT-T)
3. Type: `source /opt/ros/melodic/setup.bash`
4. Type: `rosbag record -a`
    a. Optionally you can preset a duration like this:
    ```
    rosbag record -a --duration=5m
    ```
    This limits the recording to 5 minutes.

Please take a recording of at least 5 minutes. This will give the system enough time to settle down.

If you are recording a specific issue, please let us know roughly how many minutes into the video the problem occurs. If you can reproduce a problem manually, please allow for at least 60 seconds of recording before reproducing the issue.

Revision and Release History

Revision	Release date	Description
v2.0	2021-02-01	- First release
v2.1	2021-05-04	- Version 2.1
v2.1.8	2021-05-26	- Added new data fields to API

Contact Information

For non-urgent technical support, please contact us at:
support@seoulrobotics.org

For business related questions, please contact us at:
sales@seoulrobotics.org

For urgent requests, please contact your account manager or assigned technical support person directly.

Visit the Seoul Robotics website at: https://www.seoulrobotics.org/ for the latest news and product update information.

About Seoul Robotics.

Seoul Robotics is the pioneer in advanced 3D computer vision technology, and leading provider in Lidar solutions. With their base in Seoul, Ann Arbor, and Munich, Seoul Robotics serves international OEMs and government partners to democratize Lidar solutions.

Proud Partners