Chapter 1: Introduction to ROS

1.1 Overview of ROS

Robot Operating System (ROS) is an open-source middleware framework that provides tools and libraries for building robot applications. Rather than an operating system in the traditional sense, ROS offers standardized message passing, package management, and a rich ecosystem of drivers, algorithms, and simulation tools.

1.2 Versions and Evolution of ROS

- **ROS 1 (e.g., Kinetic, Melodic, Noetic)**: Launched in 2010; uses a centralized master node (roscore), catkin build system, and custom TCP/UDP transport.

- **ROS 2 (e.g., Ardent, Bouncy, Crystal, Dashing, Eloquent, Foxy, Galactic, Humble, Iron)**: First released in 2018; built on DDS (Data Distribution Service) middleware for decentralized, real-time-capable communication, and uses colcon for building.

1.3 Comparison Between ROS 1 and ROS 2

Aspect	ROS 1	ROS 2
Communication	Master-based, custom transport	DDS-based, peer-to-peer
Real-time	Limited support	Improved; can integrate with RTOS
Security	None	DDS Security plugins (authentication, encryption)
Multi-robot Discovery	Manual namespace/configuration	Automatic via DDS discovery
Build System	catkin	colcon
Supported Languages	C++, Python, Java (community)	C++, Python, Rust, others
Lifecycle Management	No native support	Managed nodes with lifecycles

1.4 Installation Instructions for ROS 2 Humble

Please follow the steps mentioned in the below web page to install ROS-Humble.

https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html

NOTE: The above installation steps in the above web page will only work with Ubuntu 22.04. If running a different version of Ubuntu or a different OS, please refer the ROS documentation.

Chapter 2: LIDAR Setup

2.1 Fundamentals of LiDAR Technology

LiDAR (Light Detection and Ranging) is a remote sensing method that uses laser pulses to measure distances to objects. By scanning across an environment and timing the return of each pulse, LiDAR generates precise, high-resolution 3D point clouds.

2.2 Working Principle of LiDAR

1. **Pulse Emission**: A laser diode emits short pulses of light.
2. **Time-of-Flight Measurement**: The sensor records the time interval between emission and reception of each pulse.
3. **Distance Calculation**: Distance = (speed of light) × (time-of-flight) / 2.
4. **Scanning Mechanism**: Rotating mirrors or spinning assemblies sweep the beam across a field of view to build a full 3D scan.

2.3 Specifications of Ouster OS1-32

Parameter	Value
Channels	32 vertical channels
Range	200 m (best reflection)
Accuracy	Min: ±0.5 cm Max: ±3 cm
Rotation Rate	10–20 Hz (adjustable)

Parameter	Value
Field of View (Vertical)	±21.2°
FOV (Horizontal)	360° rotating
Laser Wavelength	865 nm
Power Consumption	14-20 W, 16W Nominal

2.4 System Setup of Velodyne VLP-32C

1. Connect the LiDAR sensor to the power supply and the ethernet cable of the LiDAR to the system.

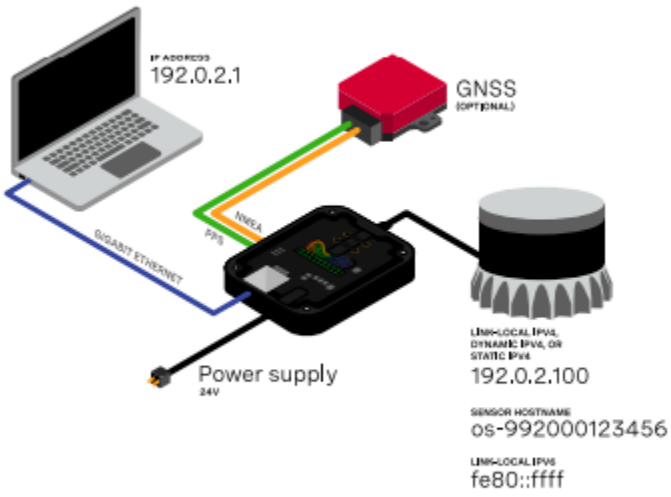
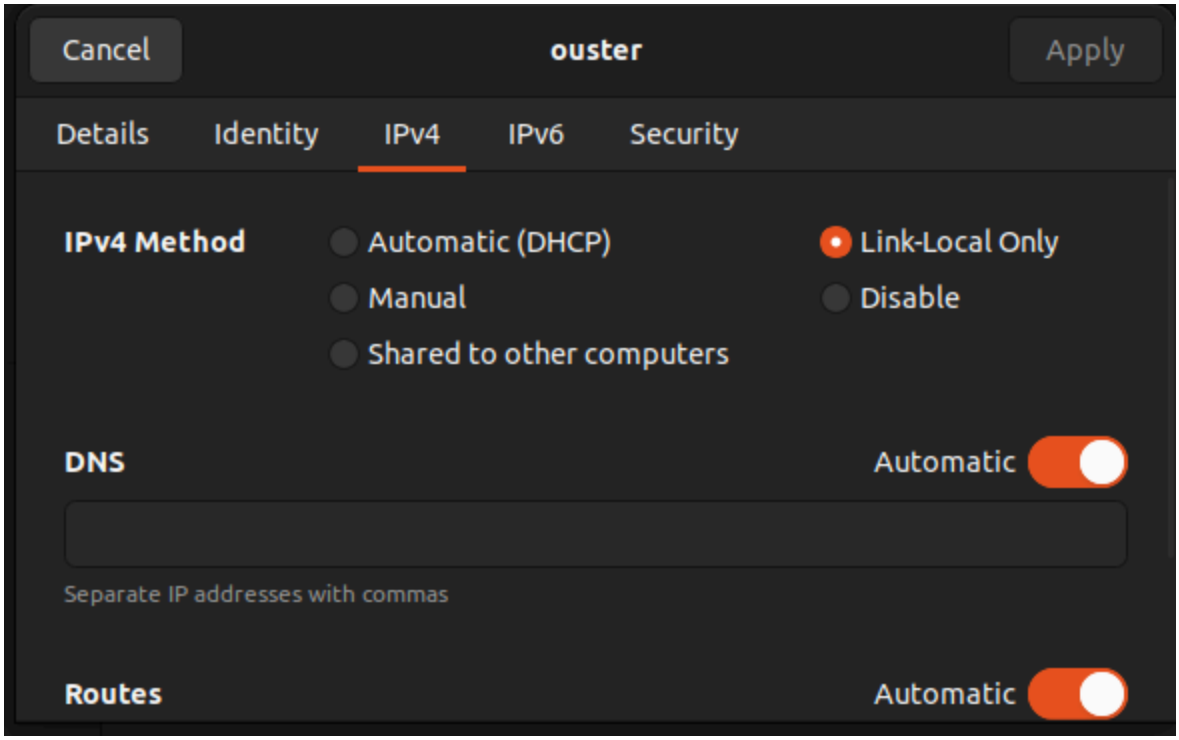


Figure 3.1: Network Configuration and Setup

---

2. Once the LiDAR is connected to the power supply and to the system via ethernet, open the network settings and configure the network manually like below.



3. Next open the internet browser and enter the hostname of the lidar like http://os-122427001080.local/. This should open the web interface of the LiDAR which can be used to configure the LiDAR settings. The hostname of the sensor is a 12 digit number and is imprinted on the top of the lidar sensor.

Chapter 3: Sensor Driver (ouster_driver)

In this chapter, we see how to launch the Ouster sensor driver in ROS2 to access and visualize the point cloud data generated by the LiDAR.

3.1 Sensor Driver Setup

- Install the required dependencies by running

```
sudo apt install -y        \
   build-essential         \
   libeigen3-dev           \
   libjsoncpp-dev          \
   libspdlog-dev           \
   libcurl4-openssl-dev    \
   cmake                   \
   python3-colcon-common-extensions


sudo apt install -y             \
   ros-$ROS_DISTRO-pcl-ros      \
   ros-$ROS_DISTRO-tf2-eigen    \
   ros-$ROS_DISTRO-rviz2
```

- Clone the sensor driver from github into the workspace using

```
mkdir -p ros2_ws/src && cd ros2_ws/src
git clone -b ros2 --recurse-submodules https://github.com/ouster-lidar/ouster-ros.git
```

- source /opt/ros/<ros-distro>/setup.bash # replace ros-distro with 'rolling', 'humble', 'iron', 'jazzy' or `kilted`

- Then run the below commands to build the driver

```
cd ros2_ws
colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release
```

3.3 Sensor Driver Launch:

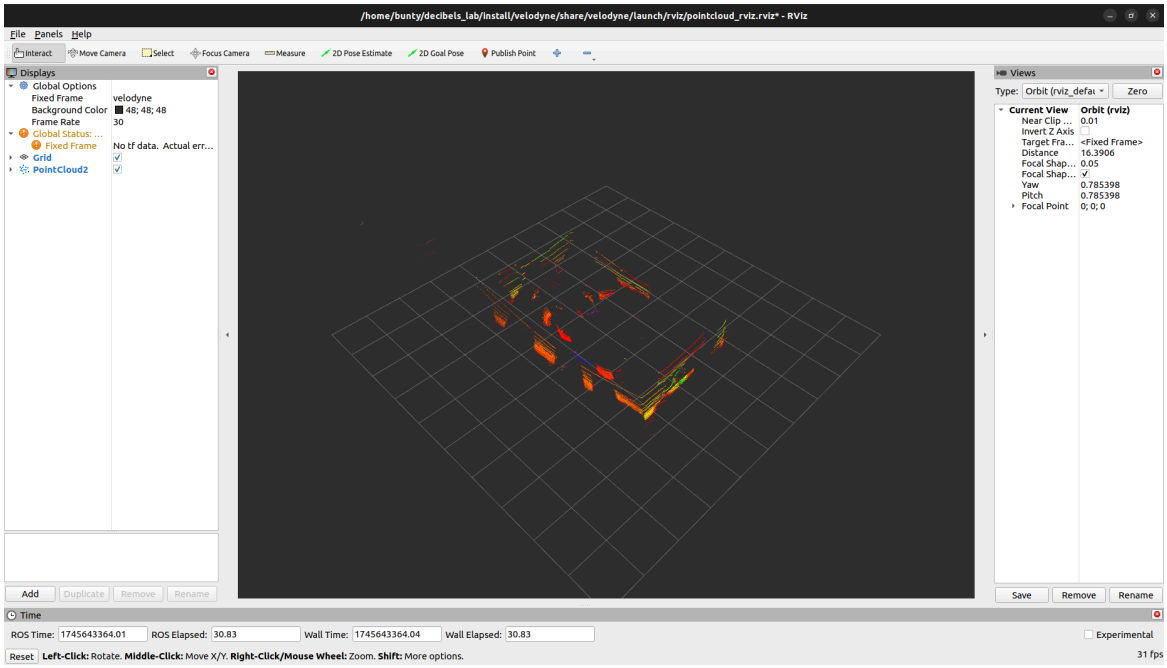
To launch the sensor:

1. Open a terminal and navigate to the already built ROS2 workspace.

2. Source the ROS2 workspace by entering the command

   source install/setup.bash
3. Open the sensor.launch.xml file in the below path and change the sensor hostname to os-122427000031.local (set the hostname of the sensor you are connecting)
   ros2_ws/install/ouster_ros/share/ouster_ros/launch/sensor.launch.xml
4. Run the below command on the terminal, which will launch the ouster_ros_driver which will launch the sensor driver and opens an RViz to visualize the pointcloud data.

   ros2 launch ouster_ros sensor.launch.xml



3.4 Field of View (FOV) Configuration

What is Field of View (FOV)?

The Field of View (FOV) of the LiDAR sensor defines the angular range over which it can detect and measure objects. It described how much of the surrounding environment the LiDAR can “see” at once.

Types of FOV in LiDAR
LiDAR has two main types of FOV:

1. **Horizontal FOV (HFOV)**
    - This is the angular span from left to right.
    - Example: 360deg(full rotation) or 90deg(partial forward scan)
2. **Vertical FOV (VFOV)**
    - This is the angular span from top to bottom.
    - Example: +-22.5deg, or 45deg vertical coverage.

FOV coverage example:

Model	Horizontal FOV	Vertical FOV	Channels(Beams)
Ouster OS1-32	360deg	42.4deg(+21.2 to -21.2)	32
Ouster OS0-32	360deg	90deg(+45 to -45)	32

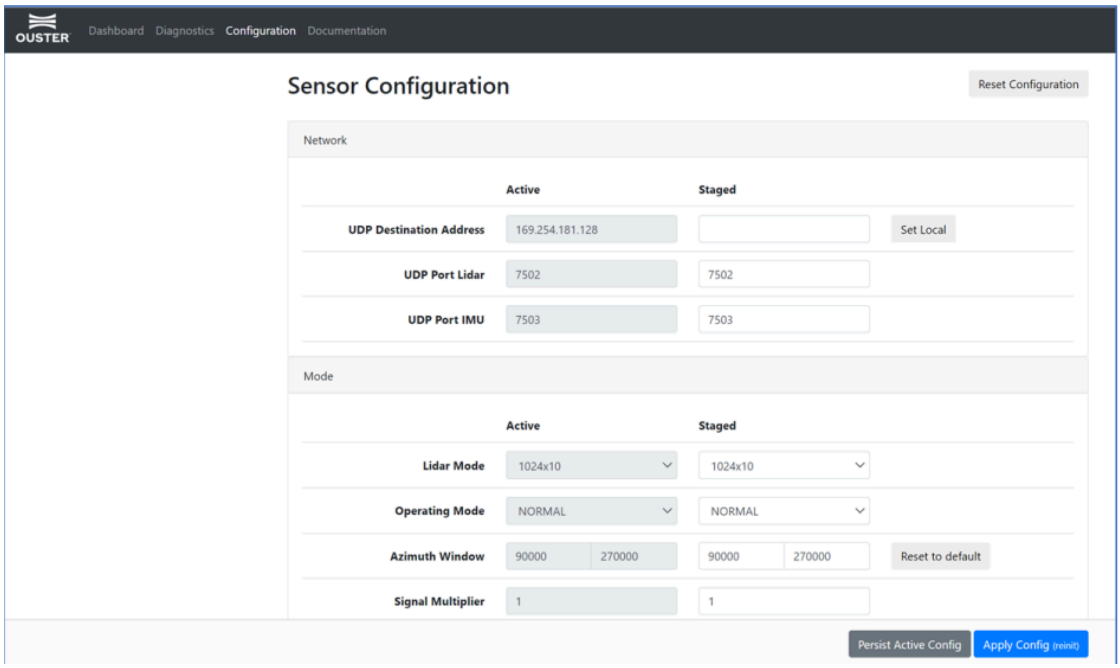
This tells us how much area the LiDAR can scan both around and above/below the sensor.

Why is FOV Important?
Coverage: A wider FOV allows the LiDAR to capture more of the surroundings in a single scan.
Obstacle Detection: A larger vertical FOV helps in detecting overhead or low-lying objects.
Sensor Placement: FOV helps decide how and where to mount the LiDAR on a robot or vehicle.

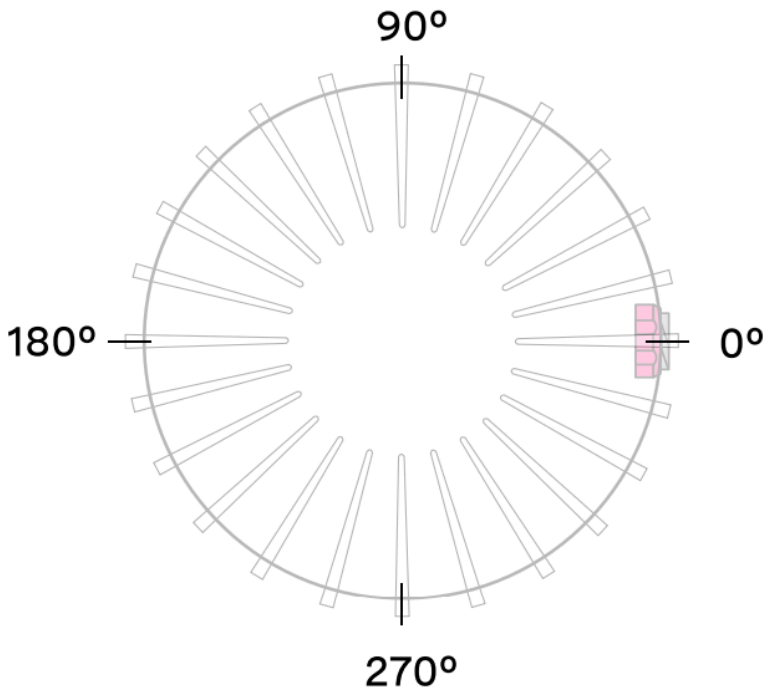
Settings to change to modify the Horizontal FOV

To modify the Ouster Lidar's horizontal FOV from without running the ROS2 driver,

From the Ouster Lidar Web Interface:



- Here we can change the Lidar's horizontal FOV using the Azimuth window param. In the fields given enter the angles for the ROI
- Once the above config is set, only data between the entered angles will be recorded and the data outside this range is discarded



LIDAR COORDINATE FRAME TOP-DOWN VIEW

3.5 Using ROSBAG2 for Data Recording and Playback

What is Rosbag2?

Rosbag2 is a command-line tool provided with ROS2 distributions that allows the recording of messages published to one or more ROS2 topics. The tool saves the data into a database file (usually in SQLite3 format), which can later be inspected or replayed to reproduce system behavior for debugging, analysis, or sharing with collaborators.

Installation

Most ROS2 distributions (e.g., Foxy, Humble, Iron) come with rosbag2 pre-installed. If it is missing, install it using the following command (replace `<distro>` with your ROS2 version):

```
sudo apt install ros-<distro>-rosbag2 ros-<distro>-ros2bag
```

Example:

```
sudo apt install ros-humble-rosbag2 ros-humble-ros2bag
```

Discovering and Selecting Topics

Before recording, list available topics and check their contents:

- List all active topics:

  ```
  ros2 topic list
  ```

- View messages on a specific topic:

  ```
  ros2 topic echo <topic_name>
  ```

- Choose topics with continuous data streams (e.g., sensor data, velocity commands).

Recording Data

Recording a Single Topic

To record data from one topic:

text
```
ros2 bag record <topic_name>
```

Example:

text
```
ros2 bag record /velodyne_points
```

Tip: Run this command from inside the directory where you want the bag file to be saved. The output will be stored there.

Recording Multiple Topics

You can record several topics by listing them:

text
```
ros2 bag record /topic1 /topic2 /topic3
```

Example:

text
```
ros2 bag record /velodyne_points /velodyne_packets /initialpose
/tf
```

Recording All Topics

To record every topic being published:

text
```
ros2 bag record -a
```

Customizing Output File Name

Set the output filename using `-o`:

text
```
ros2 bag record -o <bag_file_name> /topic1 /topic2
```

Example:

text
```
ros2 bag record -o velodyne_points /velodyne_points
```

When recording begins, you'll see INFO messages indicating active subscriptions, database creation, and recording status.

Enabling Compression (Humble+)

For reduced file size, compression can be enabled:

```
ros2 bag record -a --compression-mode file --compression-format
zstd
```

Viewing Bag Information

After recording, inspect your bag file details:

```
ros2 bag info <bag_folder_name>
```

This displays:

- Storage format (e.g., SQLite3)
- Duration of recording
- Recorded topics and types
- Message counts

Example:

```
ros2 bag info velodyne_points/
```

Note: To view individual messages, open the database file with a SQLite3 client (ROS2 does not provide a direct message viewer).

Playing Back Recorded Data

Replay recorded datasets:

```
ros2 bag play <bag_folder>
```

Options:

- Change playback rate:

```
ros2 bag play my_bag -r 2.0
```

- (Plays at twice normal speed)
- Loop playback:

```
ros2 bag play my_bag -l
```

- Replay selected topics only:

```
ros2 bag play my_bag --topic /topic1 /topic2
```

Getting Help

For detailed command options, use the `--help` flag:

```
ros2 bag record --help
ros2 bag play --help
ros2 bag info --help
```

Summary Table

Task	Command Example	Notes
List topics	`ros2 topic list`	Shows all active topics
Echo topic	`ros2 topic echo <topic_name>`	Inspects topic data
Record one topic	`ros2 bag record <topic_name>`	Data saved in current directory
Record multiple topics	`ros2 bag record /topic1 /topic2`	List multiple topics
Record all	`ros2 bag record -a`	Records everything
Filename output	`ros2 bag record -o <name> /topic`	Set custom output

Compression	`--compression-mode file --compression-format zstd`	Supported on Humble+
Bag info	`ros2 bag info <folder>`	Shows summary
Playback	`ros2 bag play <folder>`	Replay data
Playback options	`-r <rate>`, `-l`, `--topic`	Speed, loop, topic select
Help	`--help`	List available options

Rosbag2 makes data recording and playback simple, enabling robust experimentation and collaborative work in ROS2.

Chapter 4: Lidar FOV Filter

Introduction

The **Lidar FOV Filter** package for ROS 2 enables users to restrict the effective horizontal field of view (FOV) of their LiDAR point cloud stream. This is useful for applications where only a specific angular range of the sensor is needed, such as front-facing obstacle detection or rear monitoring. By filtering point cloud data at the node level, the package helps reduce downstream computing requirements and tailor sensor output to your robot's operational needs.ros+1

Package Layout

The directory structure of the package is as follows:

text

```
lidar_fov_filter/
├── CMakeLists.txt
├── package.xml
├── setup.py
├── setup.cfg
├── resource/
```

```
|   └── lidar_fov_filter
├── lidar_fov_filter/
|   └── __init__.py
├── scripts/
|   └── lidar_fov_filter.py
├── launch/
|   └── lidar_fov_filter_launch.py
└── config/
    └── lidar_fov_filter_params.yaml
```

- `CMakeLists.txt`, `package.xml`, `setup.cfg`, `setup.py` are standard ROS 2 package files.
- `resource/lidar_fov_filter` is an empty marker file required by ROS 2 Python packages.
- `lidar_fov_filter/` contains the Python module code.
- `scripts/lidar_fov_filter.py` is the executable ROS 2 node.
- `launch/lidar_fov_filter_launch.py` allows easy node startup with parameter configuration.
- `config/lidar_fov_filter_params.yaml` stores the default node parameters.ros+1

What Does the Package Do?

The `lidar_fov_filter` package provides a ROS 2 node that:

- **Subscribes** to a PointCloud2 topic (from any LiDAR sensor).
- **Filters** points by calculating each point's horizontal angle (azimuth, using $$\text{atan2}(y, x)$$, keeping only those within a specified FOV centered at your chosen direction.
- **Publishes** the filtered point cloud on a new topic.
- Parameters (`view_direction`, `view_width`, `input_topic`, `output_topic`) allow configuration of the FOV region and topics.

This functionality is especially useful for robots that only require sensing within a certain angular sector (e.g., looking forward, backward, or in a narrow region), similar to options available for Velodyne or other advanced LiDAR packages.

How to Run the Package

Installation & Build

1. Place the package in your workspace's `src` folder:

```
cd ~/your_ros2_ws/src
git clone <repo-url> lidar_fov_filter
```

2. Or manually create and copy the package files to this structure.

3. Build the workspace:

```
cd ~/your_ros2_ws
colcon build --packages-select lidar_fov_filter
source install/setup.bash
```

Basic Usage

Launch the node with default parameters:

```
ros2 launch lidar_fov_filter lidar_fov_filter_launch.py
```

Customize parameters:
Edit `config/lidar_fov_filter_params.yaml` to adjust the FOV and topics:

```
lidar_fov_filter:
     ros__parameters:
     view_direction: 0.0            # Center FOV at 0 radians (front)
     view_width: 3.14159            # FOV width of π radians (180°)
     input_topic: "/ouster/points"
```

```
output_topic: "/ouster/points_filtered"
```

Or override from the command line:

```
ros2 run lidar_fov_filter lidar_fov_filter --ros-args -p view_direction:=0.0 -p view_width:=1.57
```

Launch File & Parameters

- The launch file (`launch/lidar_fov_filter_launch.py`) starts the node and loads the yaml parameter file.
- Parameter configuration allows easily changing the FOV direction and width for various sensing scenarios.

Example Scenarios

- **Front-facing FOV:** 0° to 180°

```
view_direction: 0.0
view_width: 3.14159
```

- **Rear-facing FOV:** -180° to 0°

```
view_direction: -3.124139361

view_width: 3.14159
```

- **Narrow FOV (-45° to 45°):**

```
view_direction: 0.0
view_width: 1.5708
```