Week 10: Setup for 3D Pointcloud Map Making with Ouster OS2, KVH1750 IMU, and LIO-SAM. Making a Test Map Indoors

We will now set up for making a 3D pointcloud map that we will run on Autoware for autonomous driving. The easiest, most robust algorithm found for making maps has been LIO-SAM. Previously it was available only on ROS1 but we are fortunate to have a ROS2 version as of November 2025. We will use the MKZ's Ouster OS2 and Novatel KVH1750 IMU for making 3D pointcloud maps.

References:

- LIO-SAM ROS2 Branch (Github)
- Making Pointcloud Maps (Autoware)
- LIO-SAM Reference (Autoware)
- SWRI's Novatel GNSS/IMU Drivers for ROS2 (Github)
- Ouster OS2 Official ROS2 Drivers (Github)

Downloads:

- Full Novatel, Ouster, and LIO-SAM ROS2 Package

LIO-SAM (Lidar Inertial Odometry via Smoothing and Mapping) is versatile and robust because it only requires a LiDAR, 9-Axis IMU, and GNSS (optional). The program does not make the map real-time while driving around with the LiDAR and IMU activated. LIO-SAM post processes the data and makes the map, so setting up the Ouster OS2 and Novatel KVH1750 9-Axis IMU beforehand to work with LIO-SAM is crucial.

Install the SWRI Novatel Drivers, Ouster OS2 Drivers, and LIO-SAM ROS2 package given in the **downloads** section above. There were many issues and debugging needed for smooth and resolute mapping, the drivers in the provided package have been configured to work with LIO-SAM which will be elaborated below:

LIO-SAM ROS2 repository requires Ouster OS2 data collection with timestamp mode "TIME_FROM_PTP_1588", however this does not align with the SWRI Novatel KVH IMU's timestamp mode, so it has been set to ROSTIME.

The SWRI Novatel KVH IMU driver has been modified to output the /imu topic to transfer data with BEST_EFFORT QoS setting. With default settings it will record data with RELIABLE QoS.

Figure 1: Novatel GPS Node modified to publish /imu topic with BEST_EFFORT QoS


```
<launch>

  <arg name="ouster_ns" default="ouster"
    description="Override the default namespace of all ouster nodes"/>
  <arg name="sensor_hostname"
    description="hostname or IP in dotted decimal form of the sensor"/>
  <arg name="udp_dest" default=""
    description="hostname or IP where the sensor will send data packets"/>
  <arg name="lidar_port" default="0"
    description="port to which the sensor should send lidar data"/>
  <arg name="imu_port" default="0"
    description="port to which the sensor should send imu data"/>
  <arg name="udp_profile_lidar" default=""
    description="lidar packet profile; possible values: {
    LEGACY,
    RNG19_RFL8_SIG16_NIR16,
    RNG15_RFL8_NIR8,
    RNG19_RFL8_SIG16_NIR16_DUAL,
    FUSA_RNG15_RFL8_NIR8_DUAL
    }"/>
  <arg name="lidar_mode" default="512x10"
    description="resolution and rate; possible values: {
    512x10,
    512x20,
    1024x10,
    1024x20,
    2048x10,
    4096x5
    }"/>
  <arg name="timestamp_mode" default="TIME_FROM_ROS_TIME"
    description="method used to timestamp measurements; possible values: {
    TIME_FROM_INTERNAL_OSC,
    TIME_FROM_SYNC_PULSE_IN,
    TIME_FROM_PTP_1588,
    TIME_FROM_ROS_TIME
    }"/>
  <arg name="ptp_utc_tai_offset" default="-37.0"
    description="UTC/TAI offset in seconds to apply when using TIME_FROM_PTP_1588"/>
  <arg name="metadata" default=""
    description="path to write metadata file when receiving sensor data"/>
  <arg name="viz" default="true"
    description="whether to run a rviz"/>
  <arg name="rviz_config" default="$(find-pkg-share ouster_ros)/config/viz.rviz"
    description="optional rviz config file"/>

  <arg name="sensor_frame" default="os_sensor"
    description="sets name of choice for the sensor_frame tf frame, value can not be empty"/>
  <arg name="lidar_frame" default="os_lidar"
    description="sets name of choice for the os_lidar tf frame, value can not be empty"/>
  <arg name="imu_frame" default="os_imu"
      description="sets name of choice for the os_imu tf frame, value can not be empty"/>
  <arg name="point_cloud_frame" default=""
    description="which frame to be used when publishing PointCloud2 or LaserScan messages.
    Choose between the value of sensor_frame or lidar_frame, leaving this value empty
```

Figure 2: Ouster OS2 launch file “sensor.launch.xml” set up for timestamping using ROSTIME

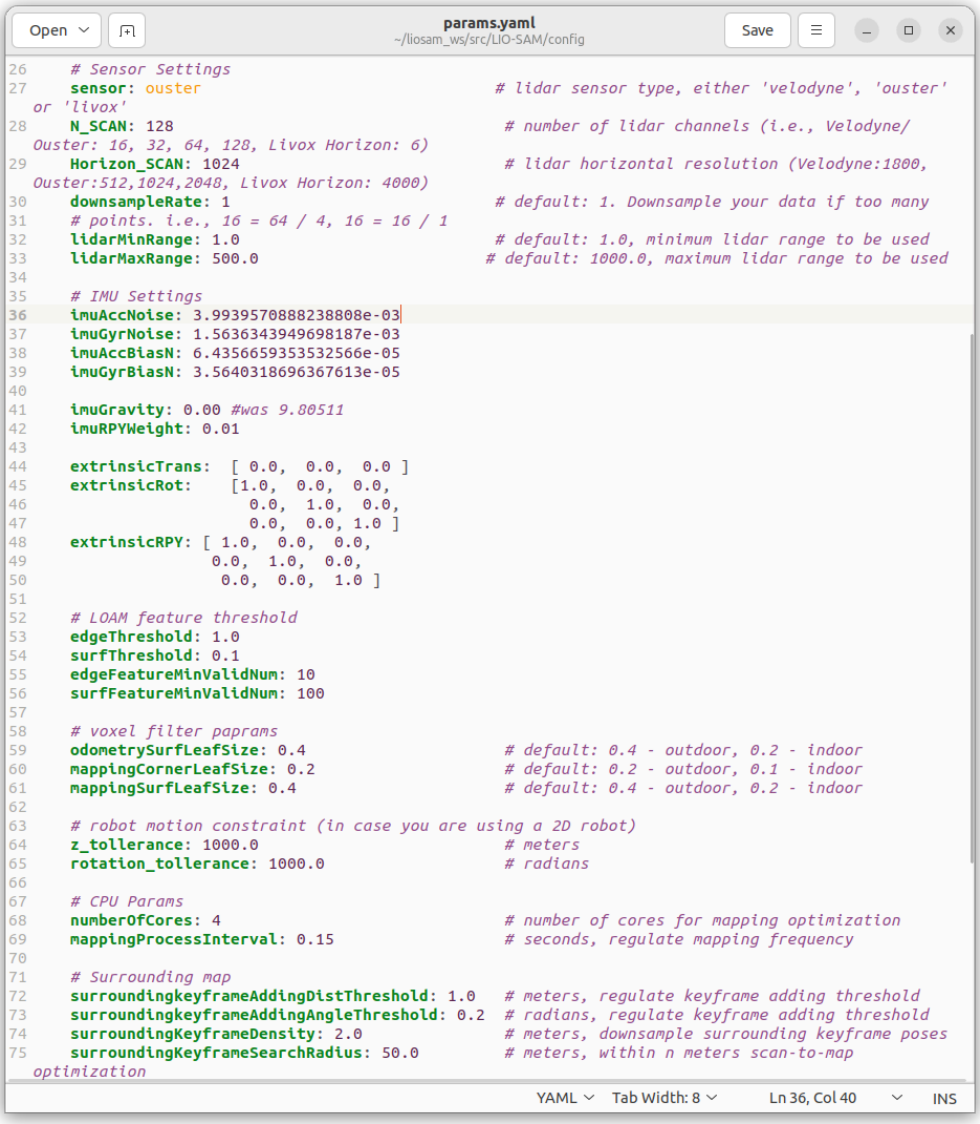

Figure 3: LIO-SAM params.yaml configuration file set up for the UNLV MKZ

Creating a Test Map Inside Work Area

Before taking the car out, we will create a map at the workstation where the car is parked. The procedure, which will be further detailed below, is a follows: launch Ouster OS2, launch Novatel KVH1750 IMU, record ROS bag file with topics /imu and /ouster/points, and finally run LIO-SAM using the recorded bag file to create the map.

1. Prepare workspaces for Ouster OS2, Novatel, and LIO-SAM
    a. Ensure that the OS2 drivers from a different workspace is not being used. It is recommended to have three workspaces for this, one for the Ouster, one for the Novatel and one for LIO-SAM. Source the workspaces automatically by adding source lines in bashrc (refer to previous exercises)

2. Launch the Ouster OS2, IMU, and record a ROSBAG file with the LiDAR and IMU data.
    a. The Novatel should be connected to the computer via USB and the Ouster via Ethernet. Launch the two drivers using the following commands:

```
# In Terminal 1 Launch Ouster (note we disable RVIZ launch, this is optional but will help reduce packet drops)
ros2 launch ouster_ros sensor.launch.xml  viz:=false sensor_hostname:=os-992315000023.local

# In Terminal 2 Launch the Novatel GPS and IMU Unit
ros2 launch novatel_gps_driver tester_for_usb.launch.py

# In Terminal 3 record a ROS Bag that captures the topics /imu and /ouster/points. The bag file can get quite
# large (a few gigabytes so it is recommended to store in a drive where there is ample space
ros2 bag record -o bagName /ouster/points /imu       # the highlighted bagName can be changed
```

The terminal/workflow should look like the following figure

Figure 3: Three terminals with commands loaded up for LiDAR and IMU data ROSBAG recording

We are just testing that we have our workflow set up correctly and that LIO-SAM will work, so record this stationary vehicle ROSBAG file for a minute or so.

3. After a minute or so, terminate the bag file recording by pressing CTRL+C in the respective terminal. Make sure it is saved in a known location.

4. Run LIO-SAM and create the pointcloud map.

Now that we have a bag file containing the LiDAR and IMU data, we can run LIO-SAM to create a pointcloud map.

Run the following commands to launch LIO-SAM and create a pointcloud map. Launching LIO-SAM will open up an RVIZ window where you can visualize the process. The map making will begin when you play the bag file:

```
# Terminal 1 - Run LIO-SAM
ros2 launch lio_sam run.launch.py

# Terminal 2 - After RVIZ has launched and LIO-SAM is waiting on a ROSBAG file, play the ROSBAG file
ros2 bag play bagName
```

Once the above two commands are ran, it should look like the following:

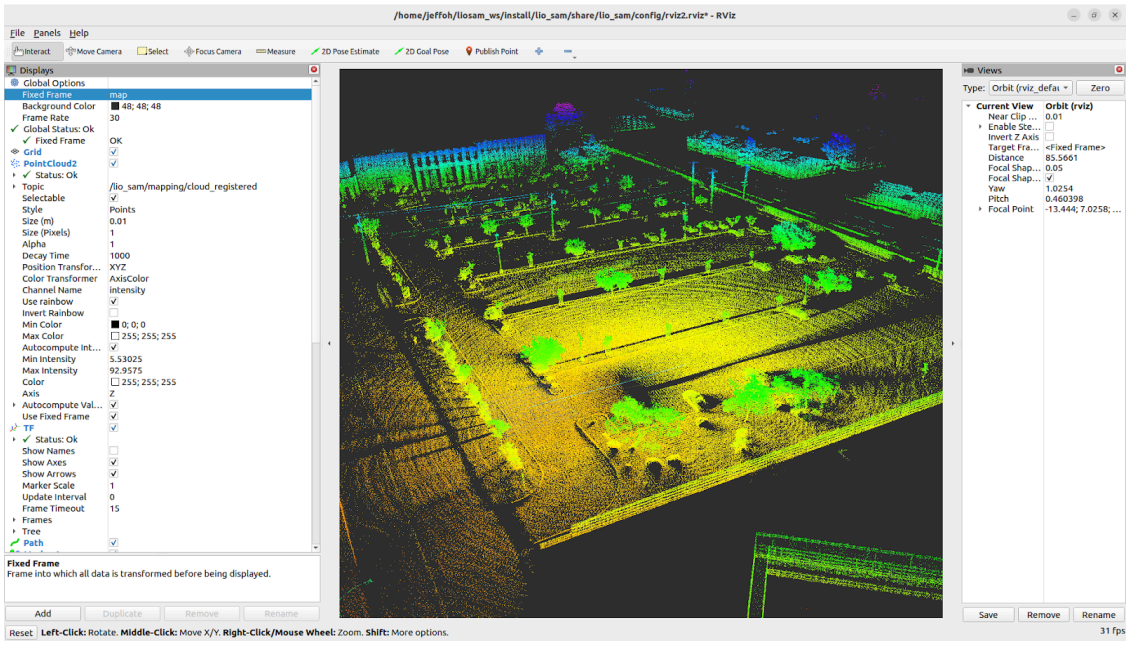

Figure 4: LIO-SAM creating a pointcloud map from a ROSBAG file

If there are issues with significant drifting, consult the LIO-SAM documentation on Github and diagnose the issue. It will likely be a small error with the transformation matrix, gravity setting etc. LIO-SAM has been proven to work well with the KVH and Ouster OS2 combination. The map file should be clean and there should not be pointclouds drifting away. If this occurs, diagnose the issue prior to making a map outdoors as it would be futile.

LIO-SAM will run as long as the ROSBAG file is running. Once a sufficient map has been created, run the following ROS service call in a new terminal to save the map to your desired destination:

```
ros2 service call /lio_sam/save_map lio_sam/srv/SaveMap "{resolution: 0.2, destination: /Downloads/service_LOAM}"  # change directory and downsampling resolution as needed
```

Downsampling is required as Autoware's upper limit of map size is 300 MB. LIO-SAM has a convenient downsampling feature built in which can be declared when using the service call. See the highlighted parameters above. There should be a few .PCD files in the saved directory. The file with the biggest size will be the main map that we will use later to vectorize lanes.