ouster-lidar / **ouster-ros** Public

Official ROS drivers for Ouster sensors (OS0, OS1, OS2, OSDome)

ouster.com

View license

173 stars 202 forks Branches Tags Activity

Star Notifications

Code Issues 64 Pull requests 21 Discussions Actions Projects Wiki Security Insights

ros2 16 Branches 22 Tags Go to file Go to file Code

This branch is 106 commits ahead of and 70 commits behind `master`. #499

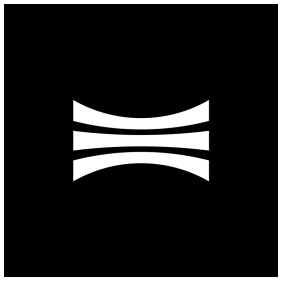
Erumeldor Add OpenCV to library dependencys (#477) a46cda1 · 2 weeks ago

.github	Port the pointcloud mask feature to ROS2 (#...	5 months ago
docs	Update published TF transforms time with s...	3 years ago
ouster-ros	Add OpenCV to library dependencys (#477)	2 weeks ago
ouster-sensor-msgs	SW-6906: publish sensor telemetry in ouster...	10 months ago
.gitmodules	move ouster-sdk to a subfolder of ouster-ros	2 years ago
CHANGELOG.rst	Drop whole archive linkage (#489)	2 months ago
Dockerfile	Port the pointcloud mask feature to ROS2 (#...	5 months ago
LICENSE	SW-5459: add a parameter for utc/tai time o...	2 years ago
README.md	Port the pointcloud mask feature to ROS2 (#...	5 months ago

README License

Official ROS driver for Ouster sensors

ROS1 (melodic/noetic) | ROS2 (rolling/humble/iron/jazzy/kilted) | ROS2 (galactic/foxy)



ROS Version	Build Status (Linux)
ROS1 (melodic/noetic)	ouster-ros passing

ROS Version	Build Status (Linux)
ROS2 (rolling/humble/iron/jazzy/kilted)	ouster-ros passing
ROS2 (galactic/foxy)	ouster-ros passing

Overview

This ROS package provide support for all Ouster sensors with FW v2.0 or later targeting ros2 distros. Upon launch the driver will configure and connect to the selected sensor device, once connected the driver will handle incoming IMU and lidar packets, decode lidar frames and publish corresponding ROS messages on the topics of `/ouster/imu` and `/ouster/points`. In the case the used sensor supports dual return and it was configured to use this capability, then another topic will published under the name `/ouster/points2` which corresponds to the second point cloud.

Supported Devices

The driver supports the following list of Ouster sensors:

- OS0
- OS1
- OS2
- OSDome

You can obtain detailed specs sheet about the sensors and obtain updated FW through the website downloads section.

Requirements

This branch is only intended for use with **Rolling**, **Humble**, **Iron**, **Jazzy** and **Kilted** ROS 2 distros. Please refer to ROS 2 online documentation on how to setup ROS on your machine before proceeding with the remainder of this guide.

> **Note**
> If you have *rosdep* tool installed on your system you can then use the following command to get all required dependencies:
> `rosdep install --from-paths $OUSTER_ROS_PATH -y --ignore-src`

Linux

In addition to the base ROS installation, the following ROS packages are required:

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

where `$ROS_DISTRO` can be either `rolling`, `humble`, `iron`, `jazzy` or `kilted`.

> **Note**
> Installing `ros-$ROS_DISTRO-rviz` package is optional in case you didn't need to visualize the point cloud using rviz but remember to always set `viz` launch arg to `false`.

The following packages are also required

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

> **Note**
> You may choose a different *ssl* backend for the *curl* library such as `libcurl4-gnutls-dev` or `libcurl4-nss-dev`

> **Note**
> To use the PCAP replay mode you need to have `libpcap-dev` installed

Windows

TBD

Mac

TBD

Getting Started

To build the driver using ROS2 you need to clone the project into the `src` folder of a ros2 workspace as shown below:

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

Next to compile the driver you need to source the ROS environemt into the active termainl:

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

Finally, invoke `colcon build` command from within the catkin workspace as shown below:

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

> **Note**
> Specifying `Release` as the build type is important to have a reasonable performance of the driver.

> **Note**
> For ROS2 we recommend using **CycloneDDS** over **FastDDS**, through out Galactic, Foxy, Humble distros.
> **FastDDS** is usually the default ros middleware on most platforms, please follow the Guide to learn how to enable **CycloneDDS** on your platform.
> The **Zeonh** ROS middleware is now available for use with ouster-ros driver from Humble and afterwards (exlcuding Iron).
> Zenoh have received great feedback from the ROS community but that's in general, we don't have a comparative analysis against the other middlewares, to enable Zenoh:
>
> ```
> sudo apt install ros-${ROS_DISTRO}-rmw-zenoh-cpp
> export RMW_IMPLEMENTATION=rmw_zenoh_cpp
> ```
>
> then follow the instructions here and rebuild the ouster-ros driver (clean build).

Once the build succeeds, you must source the *install* folder of your ros2 workspace to add launch commands to your environment:

```
source ros2_ws/install/setup.bash
```

Usage

Launching Nodes

The package supports three modes of interaction, you can connect to a *live sensor*, *replay* a recorded bag or *record* a new bag file using the corresponding launch files. Recently, we have added a new mode that supports multicast. The commands are listed below, for convenience we do provide both launch file formats (xml and python) but the python format is the preferred method:

Sensor Mode

To connect to a live sensor you use the following launch file

```
ros2 launch ouster_ros sensor.launch.xml    \
    sensor_hostname:=<sensor host name>
```

The equivalent python launch file is:

```
ros2 launch ouster_ros driver.launch.py    \
    params_file:=<path to params yaml file>
```

If you don't pass a `params_file` then the file located at `ouster/config/driver_params.yaml` will be used. Note that in the params you can start with default options for everything except the `sensor_hostname` param which you should adjust to match the hostname or ip address of the Ouster sensor you are trying to connect to.

comptability mode If you are migrating from https://github.com/ros-drivers/ros2_ouster_drivers to the official ouster drivers we supply you with a file `driver_launch.py` which provides users with same topic name and accepts the same parameter file `community_driver_config.yaml`. Please note that this is provided for backward compatibilty it may not be maintained in the future, so it would be better to update to the new format `driver_params.yaml` which offers the same options and more.

Recording Mode

> Note As of package version 8.1, specifiying metadata file is optional since the introduction of the metadata topic

```
ros2 launch ouster_ros record.launch.xml    \
    sensor_hostname:=<sensor host name>     \
    bag_file:=<optional bag file name>      \
    metadata:=<json file name>              # optional
```

Replay Mode

> Note As of package version 8.1, specifiying metadata file is optional if the bag file being replayed already contains the metadata topic

```
ros2 launch ouster_ros replay.launch.xml    \
    bag_file:=<path to rosbag file>         \
    metadata:=<json file name>              # optional if bag file has /metadata topic
```

PCAP Replay Mode

> Note To use this feature you need to compile the driver with `BUILD_PCAP` option enabled

```
ros2 launch ouster_ros replay_pcap.launch.xml   \
    pcap_file:=<path to ouster pcap file>       \
    metadata:=<json file name>              # required
```

Multicast Mode (experimental)

The multicast launch mode supports configuring the sensor to broadcast lidar packets from the same sensor (live) to multiple active clients. You initiate this mode by using `sensor_mtp.launch.xml` file to start the node. You will need to specify a valid multicast group for the **udp_dest** argument which the sensor is going to broadcast data to it. You will also need to set **mtp_main** argument to **true**, this is need to configure the sensor with the specified **udp_dest** and any other sensor settings. You can control on which ip (IP4 only) you wish to receive the data on this machine from the multicast group using the **mtp_dest** argument as follows:

```
roslaunch ouster_ros sensor_mtp.launch.xml  \
    sensor_hostname:=<sensor host name>     \
    udp_dest:=<multicast group ip (ipv4)>   \
    mtp_main:=true                          \
    mtp_dest:=<client ip to receive data>   # mtp_dest is optional
```

Using a different machine that belongs to the same netwok subnet, you can start another instance of the client to start receiving sensor messages through the multicast group as shown below (note that **mtp_main** is set to **false**):

```
roslaunch ouster_ros sensor_mtp.launch.xml  \
    sensor_hostname:=<sensor host name>     \
    udp_dest:=<multicast group ip (ipv4)>   \
    mtp_main:=false                         \
    mtp_dest:=<client ip to receive data>   # mtp_dest is optional
```

> **Note:** In both cases the **mtp_dest** is optional and if left unset the client will utilize the first available interface.

Invoking Services

To execute any of the following service, first you need to open a new terminal and source the ros2 workspace again by running the command `source ros2_ws/install/setup.bash`

GetMetadata

To get metadata while connected to a live sensor or during a replay session invoke the following command:

```
ros2 service call /ouster/get_metadata ouster_srvs/srv/GetMetadata
```

GetConfig

To get the current config of a live sensor, invoke the command:

```
ros2 service call /ouster/get_config ouster_srvs/srv/GetConfig
```

SetConfig

To change config via a file while connected to a live sensor, invoke the command:

```
ros2 service call /ouster/set_config ouster_srvs/srv/SetConfig \
    "{config_file: 'some_config.json'}"
```

Reset

To reset the new reset service, execute the following commnad:

```
ros2 service call /ouster/reset std_srvs/srv/Empty
```

When this service is invoked the client should stop streaming, dispose current connection, reset the sensor and reconnect again.

> **Note** Changing settings is not yet fully support during a reset operation (more on this)

Driver Parameters

The driver has several parameters that allow you to customize its behavior, all of these parameters are defined with the `driver_params.yaml` file found under `config` folder. The only required parameter is `sensor_hostname` which sets the sensor hostname or ip that you want to connect to through ouster-ros driver.

Other notable parameters include:

- **point_type**: This parameter allows to customize the point cloud that the driver produces through its `/ouster/points` topics. Choose one of the following values:
  - `original` : This uses the original point representation `ouster_ros::Point` of the ouster-ros driver.
  - `native` : directly maps all fields as published by the sensor to an equivalent point cloud representation with the additon of ring and timestamp fields.
  - `xyz` : the simplest point type, only has {x, y, z}
  - `xyzi` : same as xyz point type but adds intensity (signal) field. this type is not compatible with the low data profile.
  - `xyzir` : same as xyzi type but adds ring (channel) field. this type is same as Velodyne point cloud type this type is not compatible with the low data profile.

This is not a comprehenisve list of all the parameters that the driver supports for more detailed list please refer to the `config/driver_params.yaml` file.

For further detailed instructions about the driver refer to the main guide

License

Releases 5

ouster_ros2 v0.13.2 Latest

on Oct 8, 2024

+ 4 releases

Contributors 15

Languages

C++ 96.8% CMake 2.1% Other 1.1%