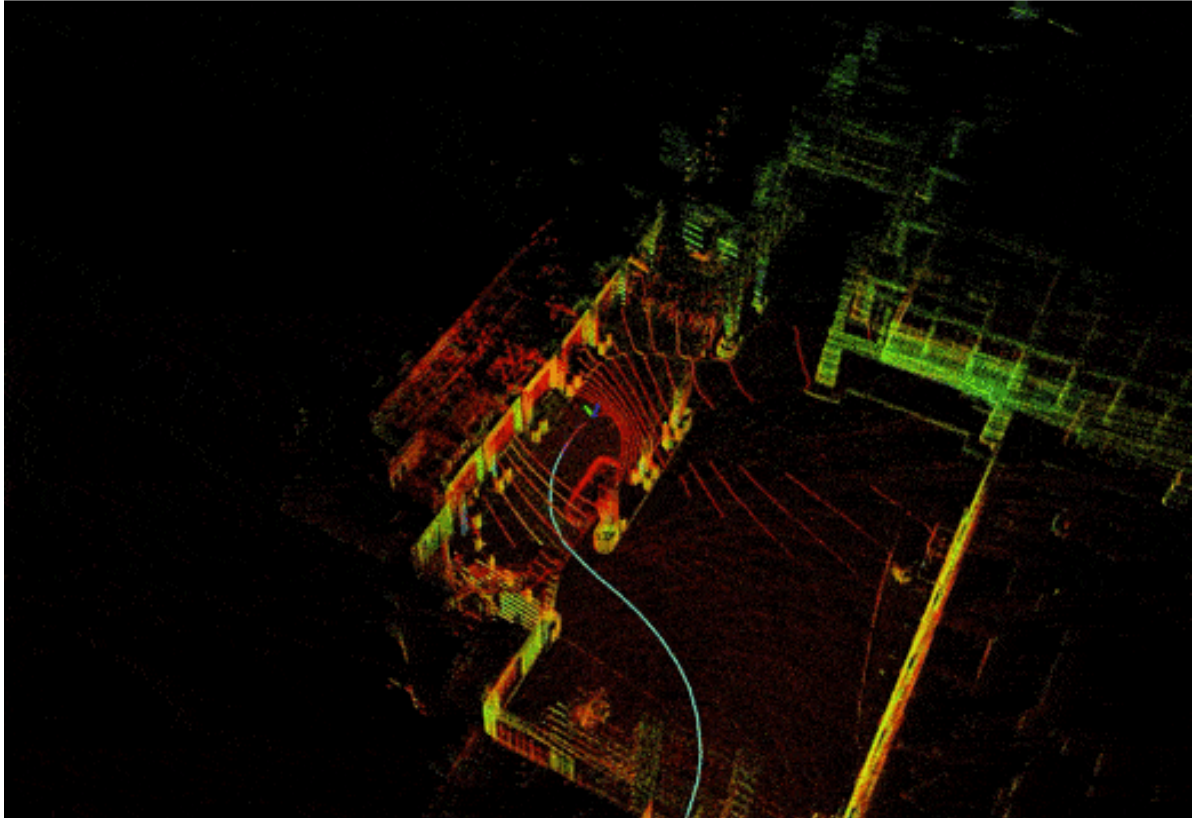


LIO-SAM

A real-time lidar-inertial odometry package. We strongly recommend the users read this document thoroughly and test the package with the provided dataset first. A video of the demonstration of the method can be found on [YouTube](#).



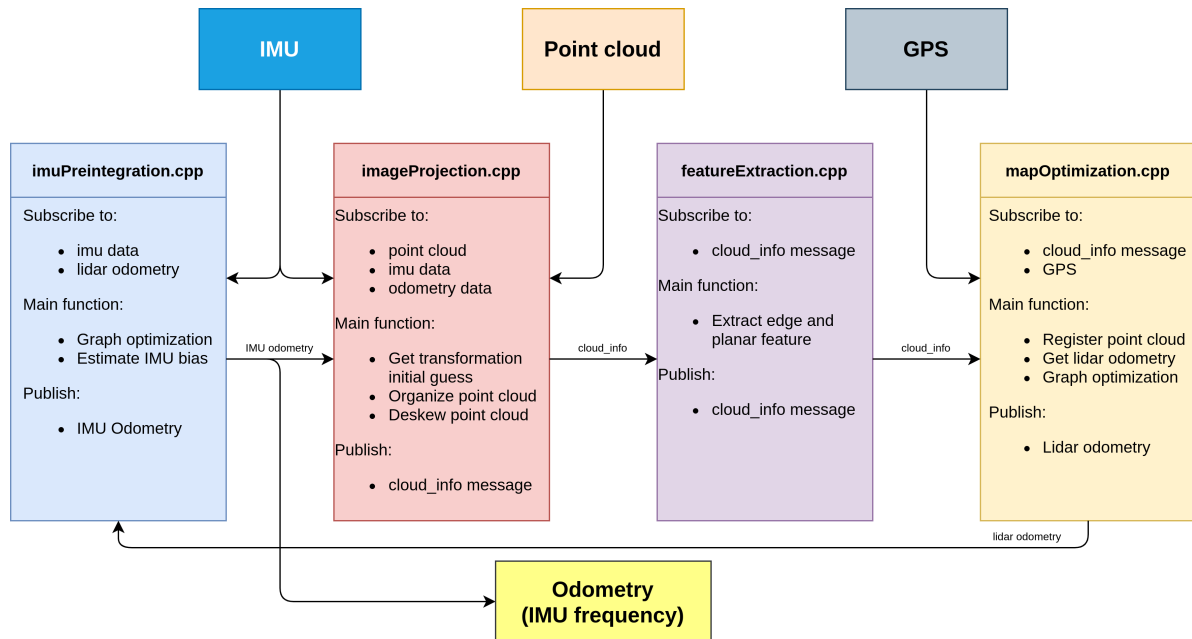


Menu

- [System architecture](#)
- [Package dependency](#)
- [Package install](#)
- [Prepare lidar data](#) (must read)
- [Prepare IMU data](#) (must read)
- [Sample datasets](#)
- [Run the package](#)
- [Other notes](#)

- [Issues](#)
- [Paper](#)
- [TODO](#)
- [Related Package](#)
- [Acknowledgement](#)

System architecture



We design a system that maintains two graphs and runs up to 10x faster than real-time.

- The factor graph in "mapOptimization.cpp" optimizes lidar odometry factor and GPS factor. This factor graph is maintained consistently throughout the whole test.
- The factor graph in "imuPreintegration.cpp" optimizes IMU and lidar odometry factor and estimates IMU bias. This factor graph is reset periodically and guarantees real-time odometry estimation at IMU frequency.

Dependency

This is the original ROS1 implementation of LIO-SAM. For a ROS2 implementation see branch `ros2`.

- [ROS](#) (tested with Kinetic and Melodic)

```

sudo apt-get install -y ros-kinetic-navigation
sudo apt-get install -y ros-kinetic-robot-localization
sudo apt-get install -y ros-kinetic-robot-state-publisher
  
```

- [gtsam](#) (Georgia Tech Smoothing and Mapping library)

```

sudo add-apt-repository ppa:borglab/gtsam-release-4.0
sudo apt update
sudo apt install libgtsam-dev libgtsam-unstable-dev
  
```

Install

Use the following commands to download and compile the package.

```
cd ~/catkin_ws/src
git clone https://github.com/TixiaoShan/LIO-SAM.git
cd ..
catkin_make
```

Prepare lidar data

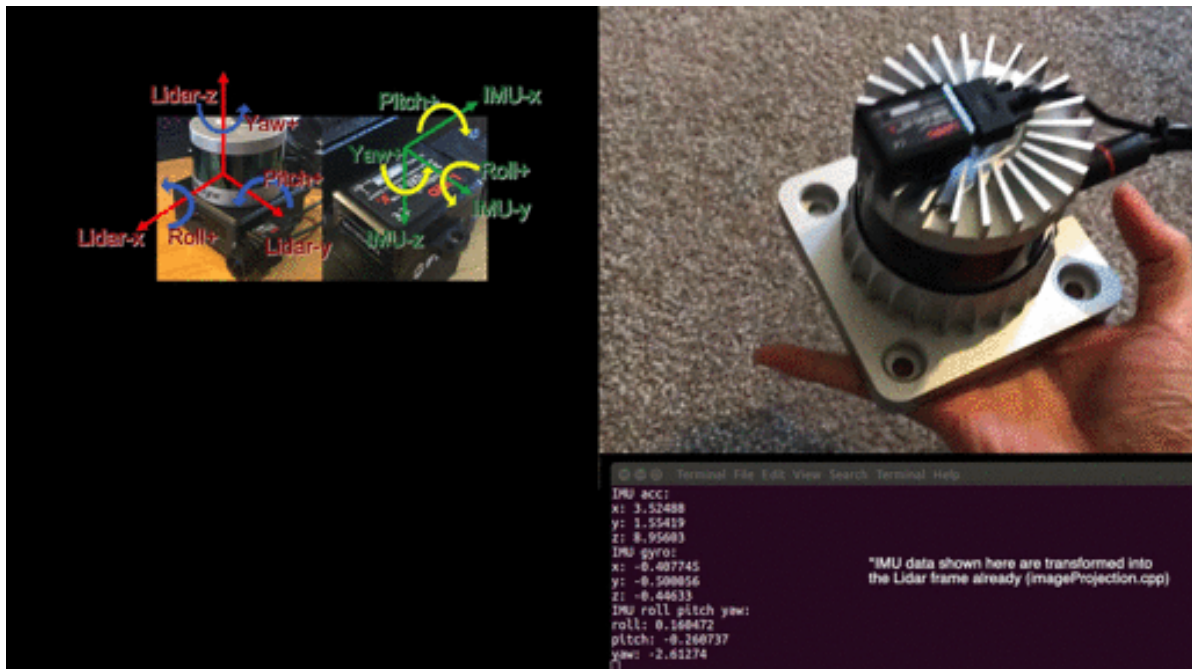
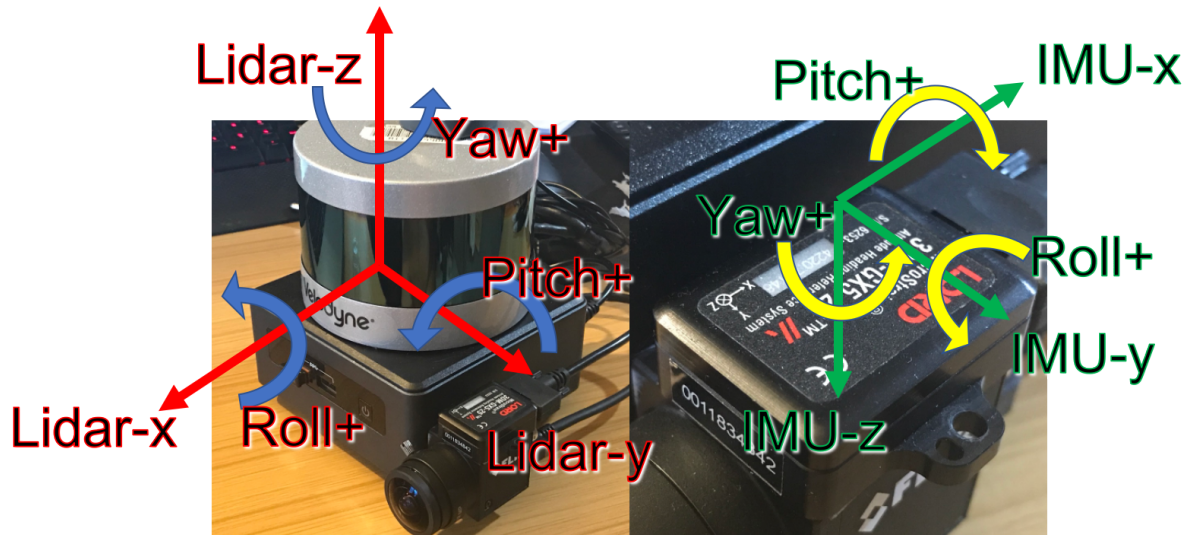
The user needs to prepare the point cloud data in the correct format for cloud deskewing, which is mainly done in "imageProjection.cpp". The two requirements are:

- **Provide point time stamp.** LIO-SAM uses IMU data to perform point cloud deskew. Thus, the relative point time in a scan needs to be known. The up-to-date Velodyne ROS driver should output this information directly. Here, we assume the point time channel is called "time." The definition of the point type is located at the top of the "imageProjection.cpp." "deskewPoint()" function utilizes this relative time to obtain the transformation of this point relative to the beginning of the scan. When the lidar rotates at 10Hz, the timestamp of a point should vary between 0 and 0.1 seconds. If you are using other lidar sensors, you may need to change the name of this time channel and make sure that it is the relative time in a scan.
- **Provide point ring number.** LIO-SAM uses this information to organize the point correctly in a matrix. The ring number indicates which channel of the sensor that this point belongs to. The definition of the point type is located at the top of "imageProjection.cpp." The up-to-date Velodyne ROS driver should output this information directly. Again, if you are using other lidar sensors, you may need to rename this information. Note that only mechanical lidars are supported by the package currently.

Prepare IMU data

- **IMU requirement.** Like the original LOAM implementation, LIO-SAM only works with a 9-axis IMU, which gives roll, pitch, and yaw estimation. The roll and pitch estimation is mainly used to initialize the system at the correct attitude. The yaw estimation initializes the system at the right heading when using GPS data. Theoretically, an initialization procedure like VINS-Mono will enable LIO-SAM to work with a 6-axis IMU. The performance of the system largely depends on the quality of the IMU measurements. The higher the IMU data rate, the better the system accuracy. We use Microstrain 3DM-GX5-25, which outputs data at 500Hz. We recommend using an IMU that gives at least a 200Hz output rate. Note that the internal IMU of Ouster lidar is an 6-axis IMU.
- **IMU alignment.** LIO-SAM transforms IMU raw data from the IMU frame to the Lidar frame, which follows the ROS REP-105 convention (x - forward, y - left, z - upward). To make the system function properly, the correct extrinsic transformation needs to be provided in "params.yaml" file. **The reason why there are two extrinsics is that my IMU (Microstrain 3DM-GX5-25) acceleration and attitude have different coordinates. Depend on your IMU manufacturer, the two extrinsics for your IMU may or may not be the same.** Using our setup as an example:
 - we need to set the readings of x-z acceleration and gyro negative to transform the IMU data in the lidar frame, which is indicated by "extrinsicRot" in "params.yaml."
 - The transformation of attitude readings is slightly different. We rotate the attitude measurements by -90 degrees around "lidar-z" axis and get the corresponding roll, pitch, and yaw readings in the lidar frame. This transformation is indicated by "extrinsicRPY" in "params.yaml."

- **IMU debug.** It's strongly recommended that the user uncomment the debug lines in "imuHandler()" of "imageProjection.cpp" and test the output of the transformed IMU data. The user can rotate the sensor suite to check whether the readings correspond to the sensor's movement. A YouTube video that shows the corrected IMU data can be found [here](#) ([link to YouTube](#)).



Sample datasets

- Download some sample datasets to test the functionality of the package. The datasets below are configured to run using the default settings:
 - **Walking dataset:** [\[Google Drive\]](#)
 - **Park dataset:** [\[Google Drive\]](#)
 - **Garden dataset:** [\[Google Drive\]](#)

- The datasets below need the parameters to be configured. In these datasets, the point cloud topic is "points_raw." The IMU topic is "imu_correct," which gives the IMU data in ROS REP105 standard. Because no IMU transformation is needed for this dataset, the following configurations need to be changed to run this dataset successfully:
 - The "imuTopic" parameter in "config/params.yaml" needs to be set to "imu_correct".
 - The "extrinsicRot" and "extrinsicRPY" in "config/params.yaml" needs to be set as identity matrices.
 - **Rotation dataset:** [\[Google Drive\]](#)
 - **Campus dataset (large):** [\[Google Drive\]](#)
 - **Campus dataset (small):** [\[Google Drive\]](#)
- Ouster (OS1-128) dataset. No extrinsics need to be changed for this dataset if you are using the default settings. Please follow the Ouster notes below to configure the package to run with Ouster data. A video of the dataset can be found on [YouTube](#):
 - **Rooftop dataset:** [\[Google Drive\]](#)
- Livox Horizon dataset. Please refer to the following notes section for parameter changes.
 - **Livox Horizon:** [\[Google Drive\]](#)
- KITTI dataset. The extrinsics can be found in the Notes KITTI section below. To generate more bags using other KITTI raw data, you can use the python script provided in "config/doc/kitti2bag".
 - **2011_09_30_drive_0028:** [\[Google Drive\]](#)

Run the package

1. Run the launch file:

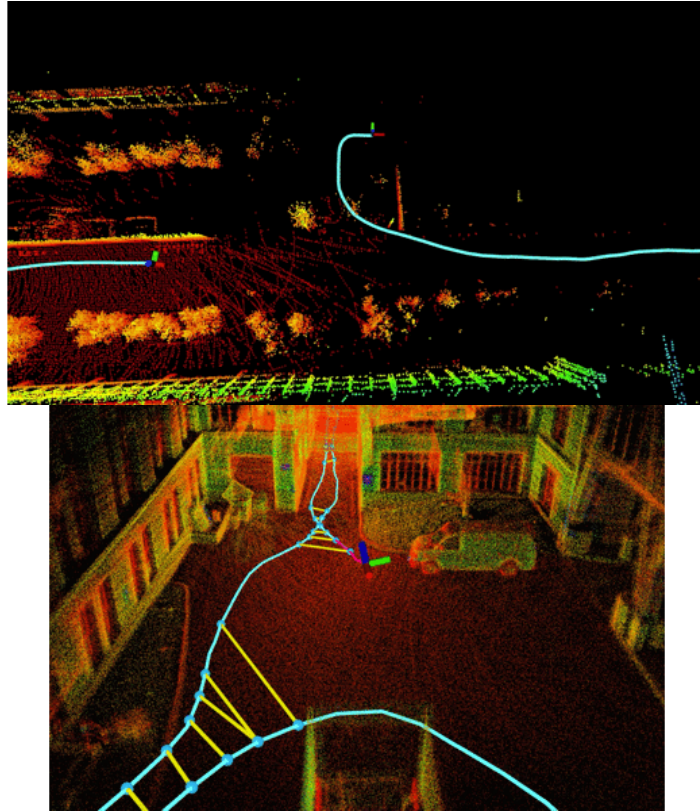
```
roslaunch lio_sam run.launch
```

2. Play existing bag files:

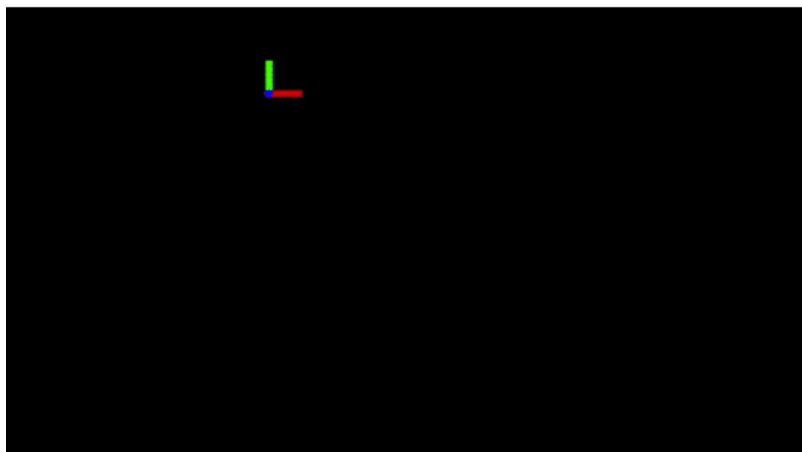
```
rosbag play your-bag.bag -r 3
```

Other notes

- **Loop closure:** The loop function here gives an example of proof of concept. It is directly adapted from LeGO-LOAM loop closure. For more advanced loop closure implementation, please refer to [ScanContext](#). Set the "loopClosureEnableFlag" in "params.yaml" to "true" to test the loop closure function. In Rviz, uncheck "Map (cloud)" and check "Map (global)". This is because the visualized map - "Map (cloud)" - is simply a stack of point clouds in Rviz. Their position will not be updated after pose correction. The loop closure function here is simply adapted from LeGO-LOAM, which is an ICP-based method. Because ICP runs pretty slow, it is suggested that the playback speed is set to be "-r 1". You can try the Garden dataset for testing.

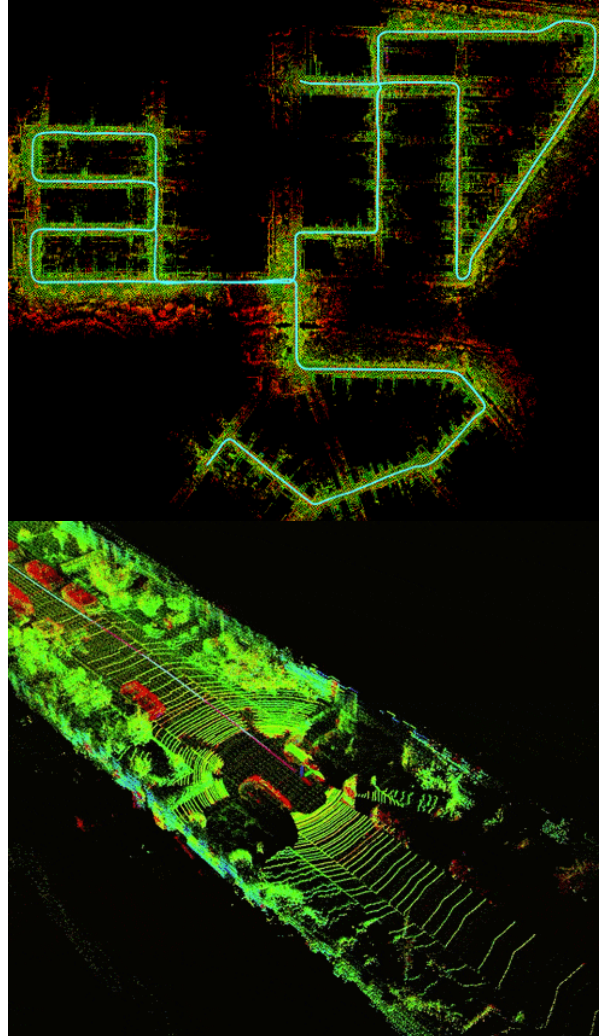


- **Using GPS:** The park dataset is provided for testing LIO-SAM with GPS data. This dataset is gathered by [Yewei Huang](#). To enable the GPS function, change "gpsTopic" in "params.yaml" to "odometry/gps". In Rviz, uncheck "Map (cloud)" and check "Map (global)". Also check "Odom GPS", which visualizes the GPS odometry. "gpsCovThreshold" can be adjusted to filter bad GPS readings. "poseCovThreshold" can be used to adjust the frequency of adding GPS factor to the graph. For example, you will notice the trajectory is constantly corrected by GPS when you set "poseCovThreshold" to 1.0. Because of the heavy iSAM optimization, it's recommended that the playback speed is "-r 1".



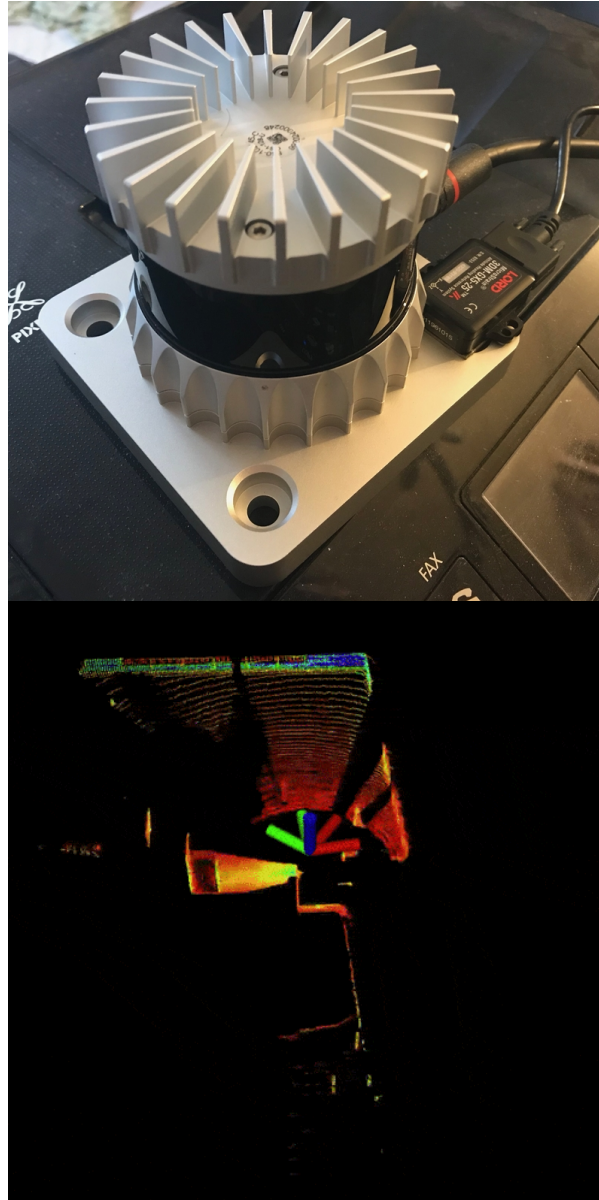
- **KITTI:** Since LIO-SAM needs a high-frequency IMU for function properly, we need to use KITTI raw data for testing. One problem remains unsolved is that the intrinsics of the IMU are unknown, which has a big impact on the accuracy of LIO-SAM. Download the provided sample data and make the following changes in "params.yaml":
 - extrinsicTrans: [-8.086759e-01, 3.195559e-01, -7.997231e-01]

- extrinsicRot: [9.999976e-01, 7.553071e-04, -2.035826e-03, -7.854027e-04, 9.998898e-01, -1.482298e-02, 2.024406e-03, 1.482454e-02, 9.998881e-01]
- extrinsicRPY: [9.999976e-01, 7.553071e-04, -2.035826e-03, -7.854027e-04, 9.998898e-01, -1.482298e-02, 2.024406e-03, 1.482454e-02, 9.998881e-01]
- N_SCAN: 64
- downsampleRate: 2 or 4
- loopClosureEnableFlag: true or false

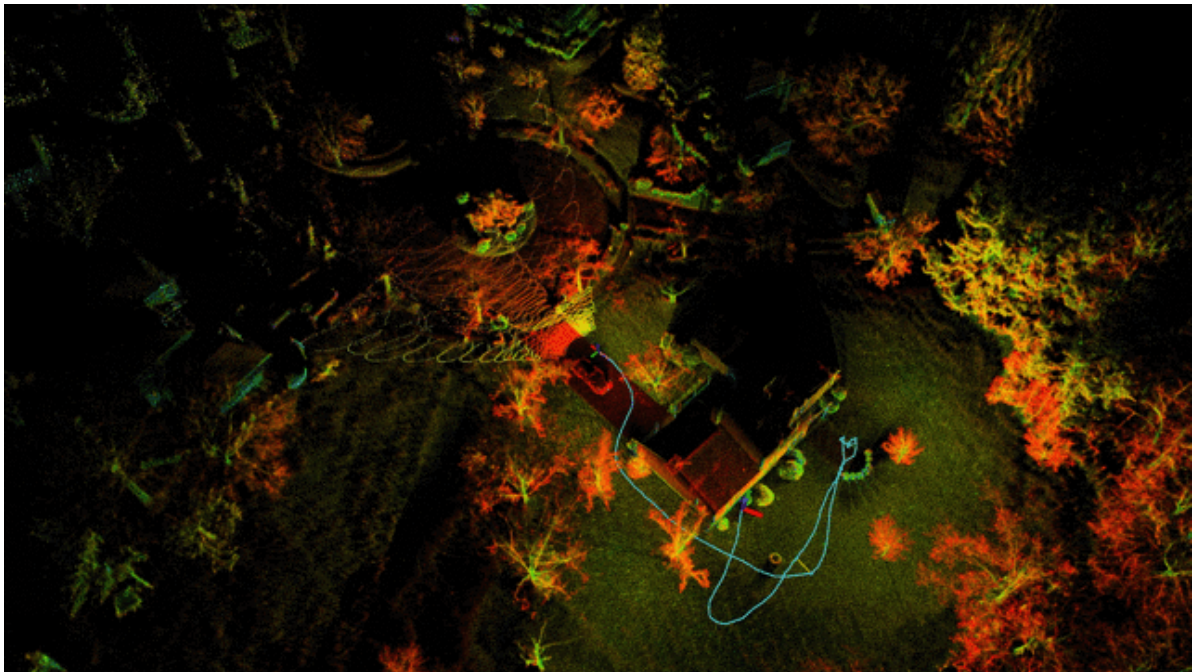


- **Ouster lidar:** To make LIO-SAM work with Ouster lidar, some preparations need to be done on hardware and software level.
 - Hardware:
 - Use an external IMU. LIO-SAM does not work with the internal 6-axis IMU of Ouster lidar. You need to attach a 9-axis IMU to the lidar and perform data-gathering.
 - Configure the driver. Change "timestamp_mode" in your Ouster launch file to "TIME_FROM_PTP_1588" so you can have ROS format timestamp for the point clouds.
 - Config:
 - Change "sensor" in "params.yaml" to "ouster".
 - Change "N_SCAN" and "Horizon_SCAN" in "params.yaml" according to your lidar, i.e., N_SCAN=128, Horizon_SCAN=1024.
 - Gen 1 and Gen 2 Ouster:

It seems that the point coordinate definition might be different in different generations. Please refer to [Issue #94](#) for debugging.



- **Livox Horizon lidar:** Please note that solid-state lidar hasn't been extensively tested with LIO-SAM yet. An external IMU is also used here rather than the internal one. The support for such lidars is based on minimal change of the codebase from mechanical lidars. A customized [livox_ros_driver](#) needs to be used to publish point cloud format that can be processed by LIO-SAM. Other SLAM solutions may offer better implementations. More studies and suggestions are welcome. Please change the following parameters to make LIO-SAM work with Livox Horizon lidar:
 - sensor: livox
 - N_SCAN: 6
 - Horizon_SCAN: 4000
 - edgeFeatureMinValidNum: 1
 - Use [livox_ros_driver](#) for data recording



Service

- `/lio_sam/save_map`
 - save map as a PCD file.

```
rosservice call [service] [resolution] [destination]
```

- Example:

```
$ rosservice call /lio_sam/save_map 0.2 "/Downloads/LOAM/"
```

Issues

- **Zigzag or jerking behavior:** if your lidar and IMU data formats are consistent with the requirement of LIO-SAM, this problem is likely caused by un-synced timestamp of lidar and IMU data.
- **Jumpping up and down:** if you start testing your bag file and the base_link starts to jump up and down immediately, it is likely your IMU extrinsics are wrong. For example, the gravity acceleration has negative value.
- **mapOptimization crash:** it is usually caused by GTSAM. Please install the GTSAM specified in the README.md. More similar issues can be found [here](#).
- **gps odometry unavailable:** it is generally caused due to unavailable transform between message frame_ids and robot frame_id (for example: transform should be available from "imu_frame_id" and "gps_frame_id" to "base_link" frame. Please read the Robot Localization documentation found [here](#).

Paper

Thank you for citing [LIO-SAM \(IROS-2020\)](#) if you use any of this code.

```
@inproceedings{liosam2020shan,  
  title={LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping},  
  author={Shan, Tixiao and Englot, Brendan and Meyers, Drew and Wang, Wei and Ratti, Carlo and Rus Daniela},  
  booktitle={IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)},  
  pages={5135-5142},  
  year={2020},  
  organization={IEEE}  
}
```

Part of the code is adapted from [LeGO-LOAM](#).

```
@inproceedings{legoam2018shan,  
  title={LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain},  
  author={Shan, Tixiao and Englot, Brendan},  
  booktitle={IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)},  
  pages={4758-4765},  
  year={2018},  
  organization={IEEE}  
}
```

TODO

☐ [Bug within imuPreintegration](#)

Related Package

- [Lidar-IMU calibration](#)
- [LIO-SAM with Scan Context](#)

Acknowledgement

- LIO-SAM is based on LOAM (J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time).