3D Laser SLAM Development Guide

introduction

3D laser SLAM algorithm

This set of programs uses the `LIO-SAM` algorithm as the lidar SLAM algorithm when building maps, which tightly couples the lidar data and the `IMU` data fed back by the robot dog itself, and realizes the function of synchronous positioning and map creation. For further understanding of the `LIO-SAM` algorithm and development on this basis, please refer to original paper and [Github warehouse](https://github.com/TixiaoShan/LIO-SAM).

Positioning based on 3D map

When patrolling, the map built during map creation will be loaded, and the `NDT` algorithm will be used for 3D point cloud matching and positioning. This algorithm needs to provide approximate initial pose information (all values are 0 by default, so the initial position of the robot dog when starting the patrol The posture needs to be the same as the initial posture when the mapping is started), the `LIO-SAM` algorithm will not be run during patrol.

Path planning and obstacle avoidance

Path planning and obstacle avoidance use the `navigation` package of ROS, and use `teb_local_planner` as the local path planner.

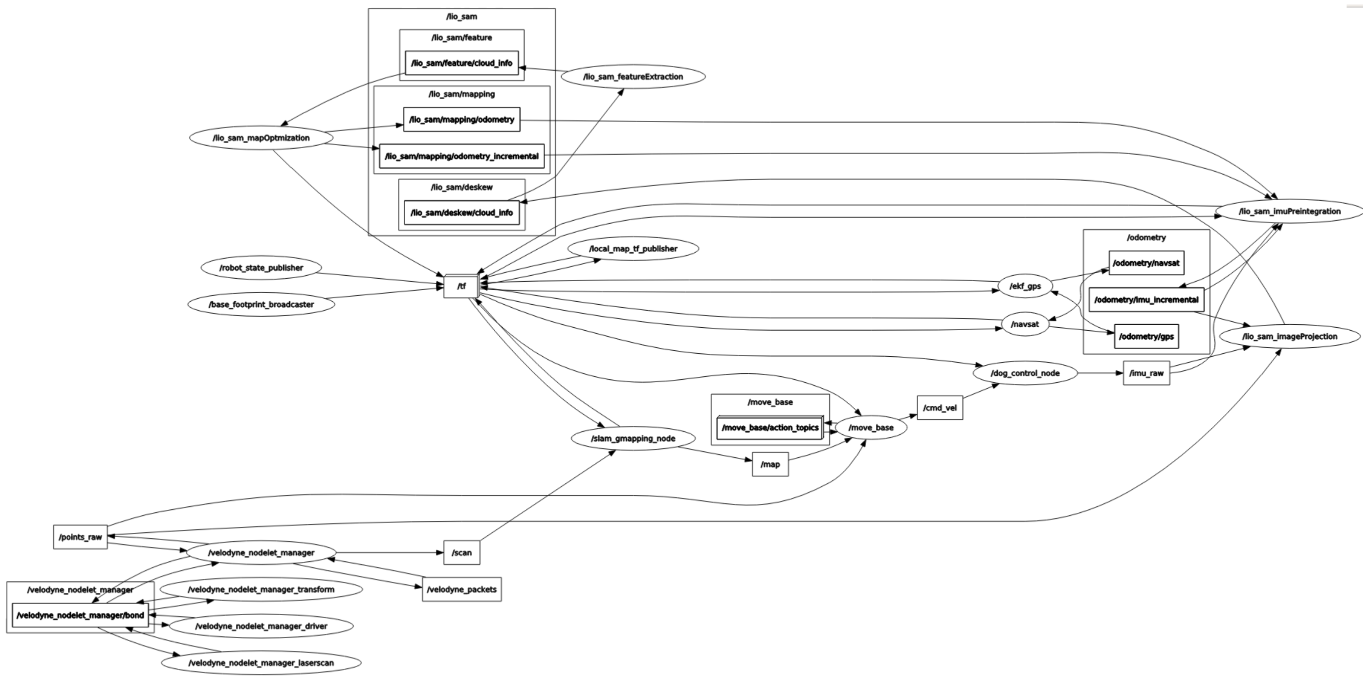
At the same time, `gmapping` is used to construct a 2D global map for global path planning (LIO-SAM also generates a global map and is used for positioning during patrols, but it is not suitable for global path planning in environments with height differences).

各软件包解读

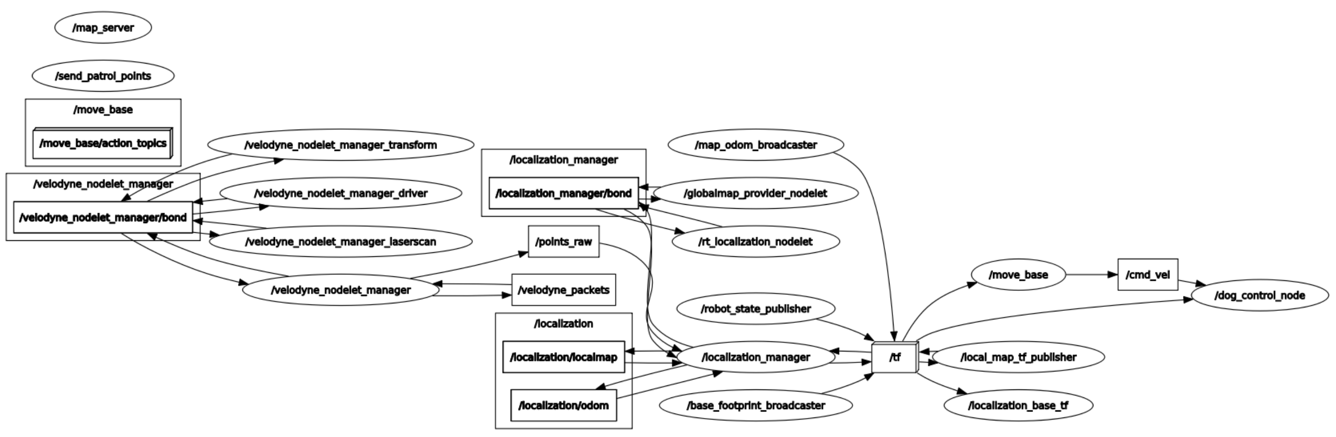
Package	Function
gmapping	Construct a 2D global map for global path planning. Unlike the source program, the source code has been modified. The odom of gmapping uses the odom of lio-sam.
lio_sam	SLAM algorithm, the parameters are modified under config/params.yaml, the source code has been modified
navigation	Call the launch file of move_base and the parameter configuration. The robot's motion performance largely depends on the configuration here
ndt_localization	Positioning algorithm during patrol
start	The launch file to start the task, and some small programs that play the role of "glue", such as the release of patrol points
unitree_legged_msgs	The message type required to read the data returned by the robot dog
unitree_legged_real	The program needed to communicate with the bottom layer of the robot dog
velodyne	Start the lidar driver

Run the node graph

ROS node graph when building the graph：


/lio_sam
/lio_sam/feature
/lio_sam/feature/cloud_info
/lio_sam_featureExtraction
/lio_sam/mapping
/lio_sam/mapping/odometry
/lio_sam_mapOptmization
/lio_sam/mapping/odometry_incremental
/lio_sam/deskew
/lio_sam/deskew/cloud_info
/lio_sam_imuPreintegration
/odometry
/local_map_tf_publisher
/robot_state_publisher
/tf
/odometry/navsat
/ekf_gps
/base_footprint_broadcaster
/odometry/imu_incremental
/navsat
/odometry/gps
/lio_sam_imageProjection
/dog_control_node
/imu_raw
/cmd_vel
/move_base
/move_base/action_topics
/move_base
/slam_gmapping_node
/map
/points_raw
/velodyne_nodelet_manager
/scan
/velodyne_nodelet_manager
/velodyne_nodelet_manager_transform
/velodyne_packets
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager_driver
/velodyne_nodelet_manager_laserscan

ROS node graph during patrol：



Platforms and sensors

The robot platform that this software is adapted to is: `Unitree Go1`.

The sensor used by this software is: `Vledyne VLP-16` Lidar.

Installation dependencies (Optional)

Note that on the shipped version of Go1's NX, all dependencies have been configured by default, and users do not need to configure them by themselves.

ROS

Official installation steps reference：

- http://wiki.ros.org/melodic/Installation/Ubuntu

The installation steps using domestic sources are as follows：

```
sudo sh -c '. /etc/lsb-release && echo "deb
http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ $DISTRIB_CODENAME main" >
/etc/apt/sources.list.d/ros-latest.list'

sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

sudo apt update

sudo apt install ros-melodic-desktop-full

echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc

source ~/.bashrc

sudo apt install python-rosdep python-rosinstall python-rosinstall-generator
python-wstool build-essential

sudo apt install python-rosdep
sudo rosdep init

rosdep update
```

gtsam-4.0.2

The official website of GTSAM is here:

- GTSAM

Note:

- here we need to install the specified version of `gtsam-4.0.2`, which is a prerequisite of LIOSAM.
- The newest version of gtsam, such as 4.1 will cause compiling problem `/usr/bin/ld: cannot find -lBoost::timer`.

Steps to install gtsam-4.0.2 are as follows:

```
wget -O gtsam.zip https://github.com/borglab/gtsam/archive/4.0.2.zip

unzip gtsam.zip

cd gtsam-4.0.2/

mkdir build && cd build

cmake -DGTSAM_BUILD_WITH_MARCH_NATIVE=OFF -DGTSAM_USE_SYSTEM_EIGEN=ON ..

make -j4

sudo make install
```

unitree_legged_sdk

Download and compile：

```
cd ~/Unitree/sdk

git clone https://github.com/unitreerobotics/unitree_legged_sdk.git

cd unitree_legged_sdk

mkdir build

cd build

cmake ..
```

```
make
```

Configure environment variables：

```
echo "export UNITREE_LEGGED_SDK_PATH=~/Unitree/sdk/unitree_legged_sdk" >>
~/.bashrc
```

controller-manager

```
sudo apt install ros-melodic-controller-manager
```

libpcap-dev

```
sudo apt install libpcap-dev
```

openslam_gmapping

```
sudo apt install ros-melodic-openslam-gmapping
```

Optional

Increase swap space

The default memory of `miniPC` is only 4G, which is a bit small for compilation, so you need to increase the swap space in order to quickly pass the compilation. The `swap` space will be deleted after each restart, you can use `free -h` to check

The method of adding 16GB of swap space is as follows:

```
sudo dd if=/dev/zero of=/swapfile bs=64M count=256 status=progress

sudo chmod 600 /swapfile

sudo mkswap /swapfile

sudo swapon /swapfile

free -h
```

Compile

Place the folder `patroldog_ws` under the path `~/Unitree/autostart`:

compile：

```
cd patroldog_ws

catkin_make
```

Compilation issues that may be encountered

Conflict of PCL and OpenCV

If the following errors occur during compilation, it may be due to the conflict between PCL and OpenCV:

```
error: field ‘param_k_’ has incomplete type ‘flann::SearchParams’
```

Refer to the following link for the solution：

- PCL-OpenCV conflict resolution
- https://github.com/strands-project/strands_3d_mapping/issues/67

Optional

The robot dog NX operating environment has been configured intact.

running

The operation needs to run on the NX of the robot dog, and its ip address is: `192.168.123.15`.

The code of this software is located in the path `~/Unitree/autostart/patroldog_ws`.

The following operations need to enter the folder first:

```
cd ~/Unitree/autostart/patroldog_ws
```

build a map

Start the mapping task：

```
sudo su

roslaunch start build_map.launch map_name:=my_map_name
```

Start RVIZ visualization：

```
rosrun rviz rviz -d ./src/lio_sam/launch/include/config/rviz.rviz
```

When building a map, you can use the `X` key on the controller to set patrol points. The patrol points are saved in the `txt` file under the folder `patroldog_ws/src/start/maps/gmapping`. The three values in a row are: `x`, `y`, `yaw` (angle), `time` (residence time)

Patrol

Start patrol mission:

```
sudo su

roslaunch start start_patrol.launch map_name:=my_map_name
```

Start RVIZ：

```
rosrun rviz rviz -d ./src/ndt_localization/prm_localization/rviz/result.rviz
```

Notice: -When patrolling, you need to start the patrol task at the starting position of the map (otherwise it will not be able to locate accurately), and the robot dog will patrol in turn according to the patrol points set during the map creation.