Lidar mapping

The following workspace contains the function packages in the entire ydlidar_ws. If you need to transplant it to your own development board, you need to copy all the function packages to the src of the workspace for compilation, and install the relevant environment.

Note: This course uses Rosmaster-X3Plus as an example. Users need to modify it according to their own motion model.

Different from the handheld lidar mapping content, this mapping adds odom data, so if you use your own motion model, you also need to have odom data.

Function package path: ~ydlidar_ws/src/yahboomcar_nav

This section focuses on several commonly used mapping algorithms: gampping、hector、karto、cartographer.

1. Start mapping

Input following command:

```
roslaunch yahboomcar_nav laser_bringup.launch
roslaunch yahboomcar_nav yahboomcar_map.launch map_type:=gmapping
```

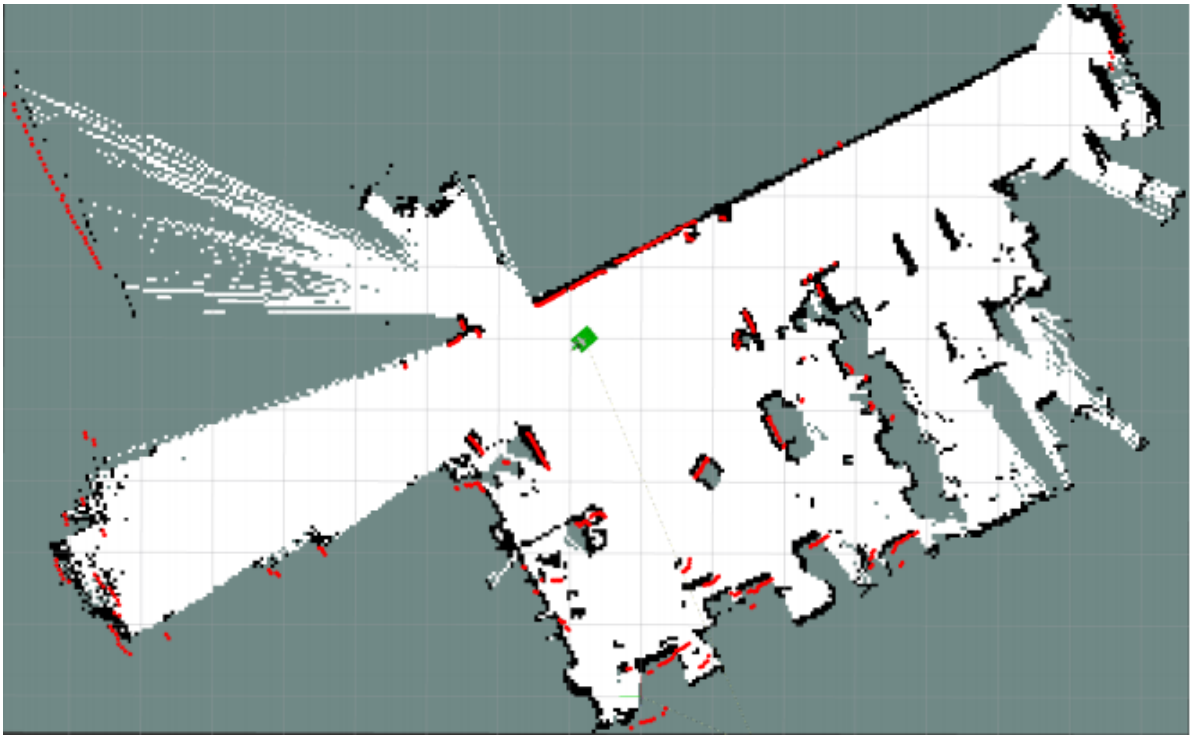
Note: When building a map, the slower the speed, the better the effect (note that the rotation speed should be slower). If the speed is too fast, the effect will be poor.

- Parameter map_type: mapping algorithm, you can choose gmapping, hector, karto, cartographer, the default is gmapping

Keyboard control robot:

```
roslaunch yahboomcar_ctrl yahboom_keyboard.launch
```

Make the robot cover the area to be mapped and the map should be as closed as possible.



2. Save map

几种建图算法保存地图的方式是不同的，

- cartographer: Input following command

  ```
  bash ~/ydlidar_ws/src/yahboomcar_nav/maps/carto_map.sh
  ```

- gampping、hector、karto：Input following command

  ```
  bash ~/ydlidar_ws/src/yahboomcar_nav/maps/map.sh
  ```

The map will be saved to the ~/ydlidar_ws/src/yahboomcar_nav/map folder, a pgm image and a yaml file.

3. Mapping algorithm

3.1 gmapping

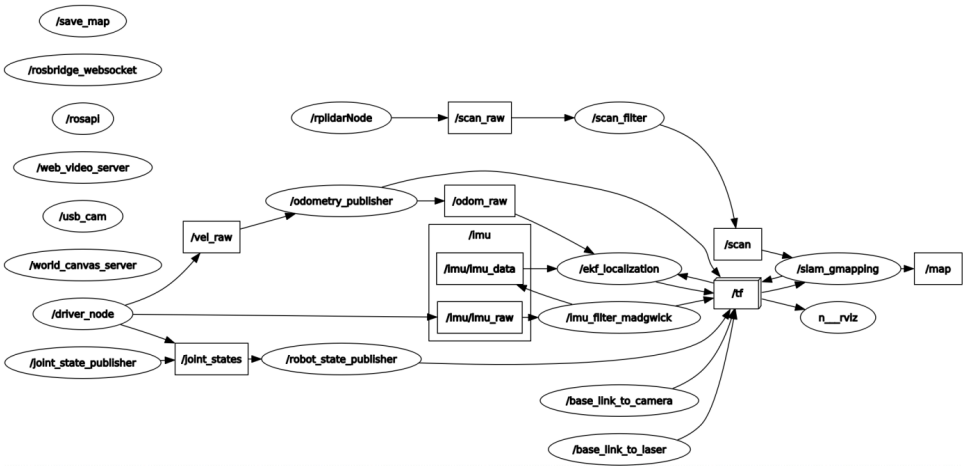
1）Topics and services

Subscribe to topics	**Type**	**Description**
tf	tf/tfMessage	Used for conversion between lidar coordinate system, base coordinate system and odometer coordinate system
scan	sensor_msgs/LaserScan	Lidar scan data
Topic	**Type**	**Description**
map_metadata	nav_msgs/MapMetaData	Publish map Meta data

Subscribe to topics	Type	Description
/map	nav_msgs/OccupancyGrid	Publish map raster data
~entropy	std_msgs/Float64	Release the estimation of robot pose distribution entropy
Service	**Type**	**Description**
dynamic_map	nav_msgs/GetMap	Get map data

View node

```
rqt_graph
```

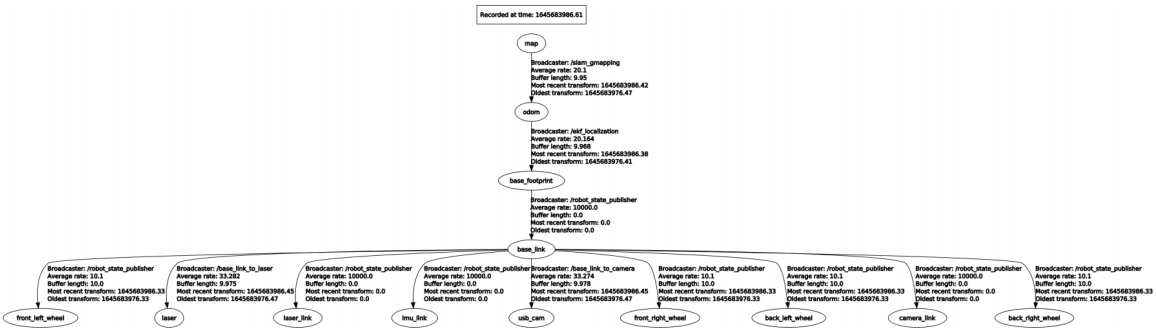


2) TF Transforms

Required tf Transforms TF	Description
laser-->base_link	sually a fixed value, broadcast periodically by a robot_state_publisher, or a `tf` static_transform_publisher.
base_link-->odom	usually provided by the odometry system (e.g., the driver for the mobile base)
Provided tf Transforms	**Description**
map-->odom	the current estimate of the robot's pose within the map frame

View TF tree

```
rosrun rqt_tf_tree rqt_tf_tree
```



3) Website:

Gmapping：http://wiki.ros.org/gmapping/

3.2 hector

1) Introduction

Features: hector_slam does not need to subscribe to the odometer/odom message, uses the Gauss Newton method, and directly uses the lidar to estimate the odometer information. However, when the robot speed is faster, it will cause deviations in the mapping effect, and the requirements for sensors are high.

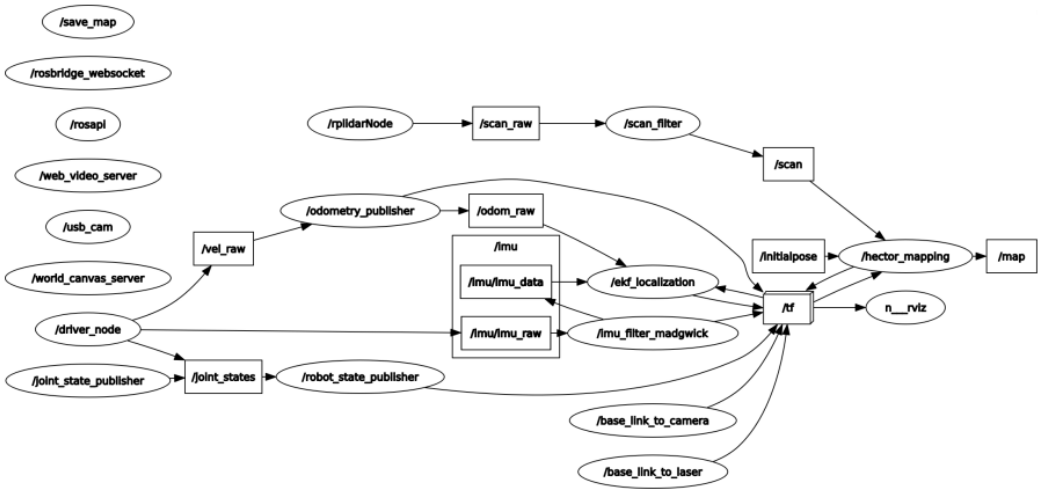
2) Topics and services

Topic subscription	**Type**	**Description**
scan	sensor_msgs/LaserScan	The laser scan used by the SLAM system.
syscommand	std_msgs/String	System command. If the string equals "reset" the map and robot pose are reset to their inital state.
Published Topics	**Type**	**Description**
map_metadata	nav_msgs/MapMetaData	Get the map data from this topic
map	nav_msgs/OccupancyGrid	Get the map data from this topic

Topic subscription	Type	Description
slam_out_pose	geometry_msgs/PoseStamped	The estimated robot pose without covariance
poseupdate	geometry_msgs /PoseWithCovarianceStamped	The estimated robot pose with an gaussian estimate of uncertainty
Services	**Type**	**Description**
dynamic_map	nav_msgs/GetMap	Call this service to get the map data.
reset_map	std_srvs/Trigger	Call this service to reset the map, and hector will start a whole new map from scratch. Notice that this doesn't restart the robot's pose, and it will restart from the last recorded pose.
pause_mapping	std_srvs/SetBool	Call this service to stop/start processing laser scans.
restart_mapping_with_new_pose	hector_mapping/ResetMapping	Call this service to reset the map, the robot's pose, and resume mapping (if paused)

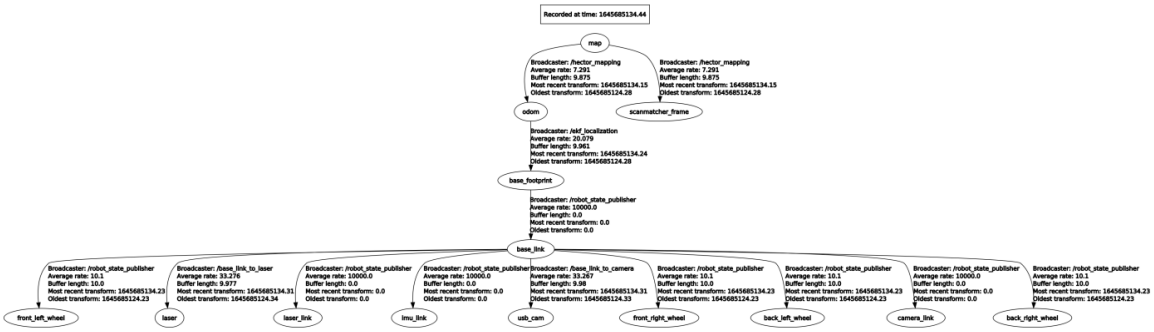
View nodes

```
rqt_graph
```



View TF tree

```
rosrun rqt_tf_tree rqt_tf_tree
```



3) TF Transforms

Required tf Transforms	**Description**
laser-->base_link	usually a fixed value, broadcast periodically by a robot_state_publisher, or a `tf` static_transform_publisher.
Provided tf Transforms	**Description**
map-->odom	the current estimate of the robot's pose within the map frame (only provided if parameter "pub_map_odom_transform" is true).

4) Website:

hector_slam：http://wiki.ros.org/hector_slam

hector_slam/Tutorials：http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot

hector_mapping：http://wiki.ros.org/hector_mapping

3.3 karto

1) Introduction

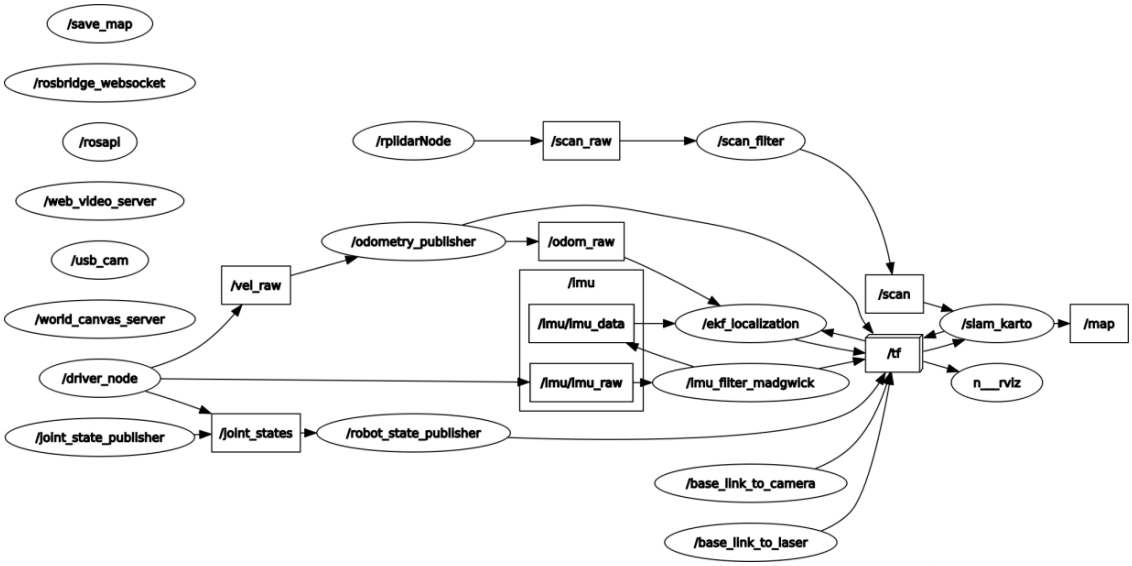
Karto is a 2D laser SLAM solution, which is based on a sparse graph optimization method with closed loop detection. Karto uses the spa (karto_slam) or g2o (nav2d) optimization library, and the front-end and back-end uses a single-threaded process. It uses odom to predict the initial position.

2) Topics and Services

Subscribed Topics	**Type**	**Description**
scan	sensor_msgs/LaserScan	Transforms necessary to relate frames for laser, base, and odometry
tf	tf/tfMessage	Laser scans to create the map from
Published Topics	**Type**	**Description**
map_metadata	nav_msgs/MapMetaData	Get the metadata of the map data (resolution, width, height, ...)
map	nav_msgs/OccupancyGrid	Get the map data from this topic, which is latched, and updated periodically
visualization_marker_array	visualisation_msgs / MarkerArray	Get the pose graph from this topic, which is updated periodically
Published Topics	**Type**	**Description**
dynamic_map	nav_msgs/GetMap	Call this service to get the map data

View node

```
rqt_graph
```

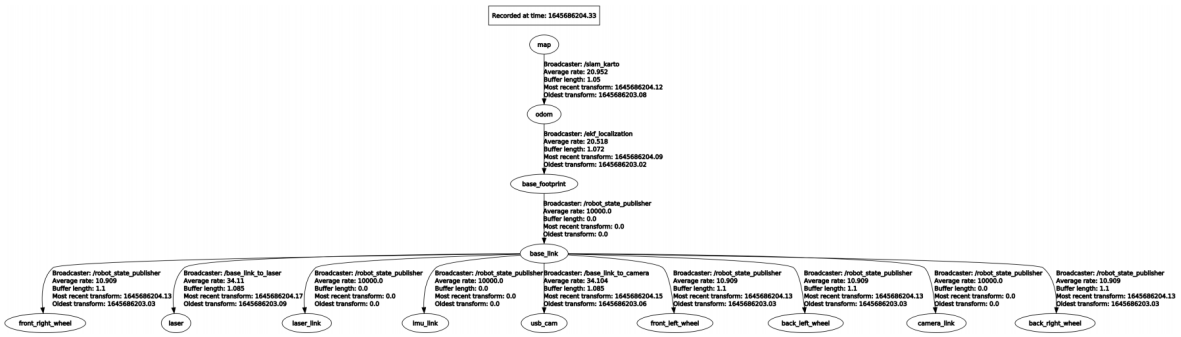


3) TF Transforms

Required tf Transforms	**Description**
laser-->base_link	usually a fixed value, broadcast periodically by a robot_state_publisher, or a `tf` static_transform_publisher.
base_link-->odom	usually provided by the odometry system (e.g., the driver for the mobile base)
Provided tf TransformsTF	**Description**
map-->odom	the current estimate of the robot's pose within the map frame

View TF tree

```
rosrun rqt_tf_tree rqt_tf_tree
```



4) Website:

karto：http://wiki.ros.org/slam_karto

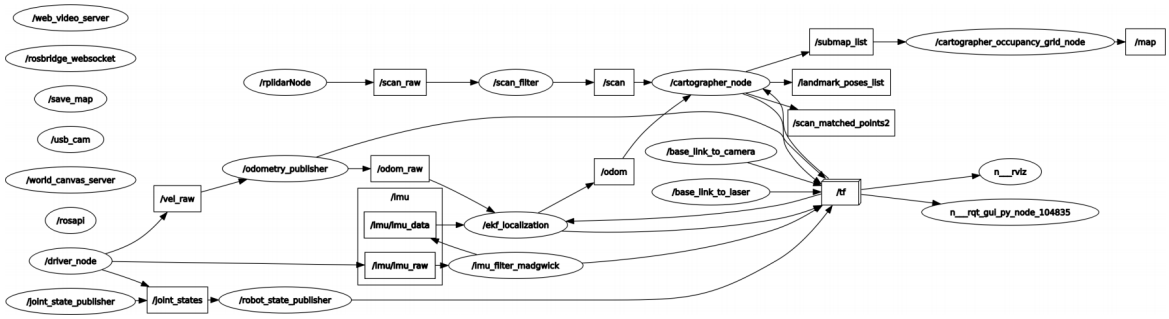
3.4 cartographer

1) Introduction

Cartographer is a 2D and 3D SLAM (simultaneous localization and mapping) library supported by a ROS system open sourced by Google. A graph-building algorithm based on graph optimization (multi-threaded back-end optimization, problem optimization built by cere). Data from multiple sensors (for example, LIDAR, IMU, and camera) can be combined to calculate the position of the sensor simultaneously and map the environment around the sensor.

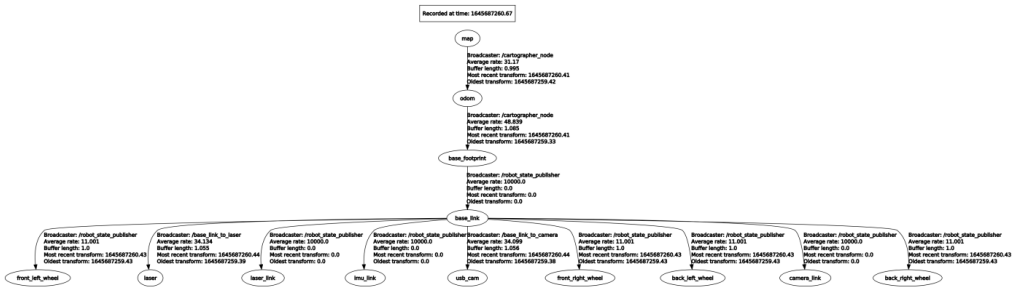
2) View node

```
rqt_graph
```



3) View TF transformation

```
rosrun rqt_tf_tree rqt_tf_tree
```



4) Parameters lua file

Parameters	Description
map_frame	Map coordinate system
tracking_frame	Convert all sensor data to this coordinate system
published_frame	The map points to coordinate system the
odom_frame	If true, the tf tree is map->odom->footprint; if false, the tf tree is map->footprint
provide_odom_frame	If true, the local, non-loop-closed, continuous pose will be published as odom_frame in map_frame
publish_frame_projected_to_2d	If enabled, the published pose will restrict 2D poses
use_odometry	Whether to use the odometer, if you use it, you must have odom tf
use_nav_sat	Whether to use gps
use_landmarks	Whether to use landmark
num_laser_scans	Whether to use single-line laser data
num_multi_echo_laser_scans	Whether to use multi_echo_laser_scans data
num_subdivisions_per_laser_scan	1 frame of data is divided into several processings, in generally it is 1
num_point_clouds	Whether to use point cloud data
lookup_transform_timeout_sec	Find the timeout of tf
submap_publish_period_sec	Time interval for publishing submap (seconds)
pose_publish_period_sec	The time interval for posting pose, when the value is 5e-3, it is 200HZ
trajectory_publish_period_sec	The time interval for publishing trajectory markers (trajectory nodes), the value is 30e-3 to 30ms
rangefinder_sampling_ratio	Fixed sampling frequency of lidar messages
odometry_sampling_ratio	Fixed sampling frequency of odometer messages

Parameters	Description
fixed_frame_pose_sampling_ratio	Fixed sampling frequency of fixed coordinate system messages
imu_sampling_ratio	Fixed sampling frequency of IMU messages
landmarks_sampling_ratio	Fixed sampling frequency for road sign messages

5) Website

Cartographer：https://google-cartographer.readthedocs.io/en/latest/

Cartographer ROS：https://google-cartographer-ros.readthedocs.io/en/latest/