7.hector mapping algorithm

7.hector mapping algorithm
- 7.1 Introduction
- 7.2 Use
  - 7.2.1 Start
  - 7.2.2 Control the robot
  - 7.2.3 Map save
- 7.3 Topics and Services
- 7.4 configuration parameters
- 7.5 TF transformation

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

map_server： https://wiki.ros.org/map_server

7.1 Introduction

Features: hector_slam does not need to subscribe to odometer/odom messages, uses the Gauss-Newton method, and directly estimates odometer information using lidar. However, when the speed of the robot is high, slippage will occur, resulting in deviations in the mapping effect and high requirements for sensors. When building a map, set the rotation speed of the cart as low as possible.

There is no odom coordinate system usage method, taken from Wiki.

2. Use without odom frame

If you do not require the use of a odom frame (for example because your platform does not provide any usable odometry) you can directly publish a transformation from map to base_link:

```
<param name="pub_map_odom_transform" value="true"/>
<param name="map_frame" value="map" />
<param name="base_frame" value="base_frame" />
<param name="odom_frame" value="base_frame" />
```

7.2 Use

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

According to different models, you only need to set the purchased model in [.bashrc], X1(ordinary four-wheel drive) X3(Mike wheel) X3plus(Mike wheel mechanical arm) R2(Ackerman differential) and so on. Section takes X3 as an example

Open the [.bashrc] file

```
sudo vim .bashrc
```

Find the [ROBOT_TYPE] parameter and modify the corresponding model

```
export  ROBOT_TYPE=X3    # ROBOT_TYPE: X1 X3 X3plus R2 X7
```

7.2.1 Start

Start command(robot side), for the convenience of operation, this section takes [mono + laser + yahboomcar] as an example.

```
roslaunch  yahboomcar_nav  laser_bringup.launch            # laser + yahboomcar
roslaunch  yahboomcar_nav  laser_usb_bringup.launch        # mono + laser +
yahboomcar
roslaunch  yahboomcar_nav  laser_astrapro_bringup.launch   # Astra + laser +
yahboomcar
```

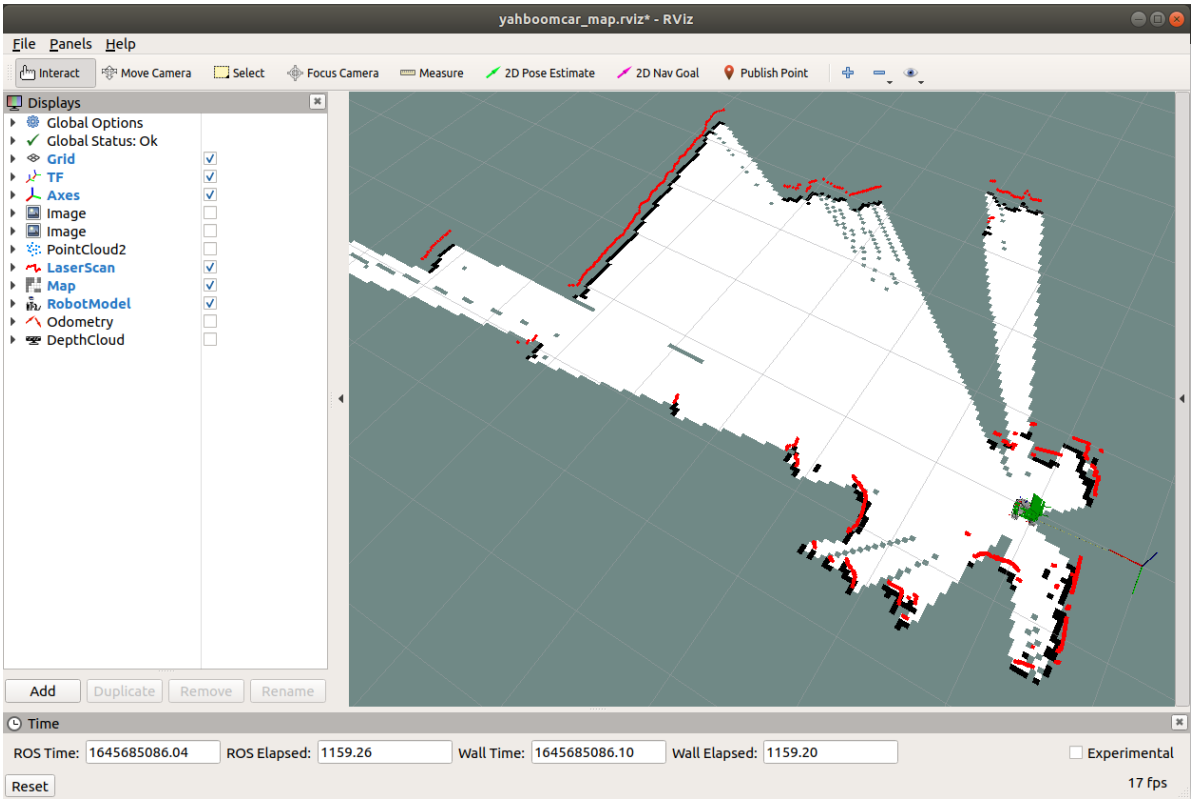
Mapping command(robot side)

```
roslaunch  yahboomcar_nav  yahboomcar_map.launch  use_rviz:=false
map_type:=hector
```

- [use_rviz] parameter: whether to enable rviz visualization.
- [map_type] parameter: set the mapping algorithm [hector].

Open the visual interface(virtual machine side)

```
roslaunch yahboomcar_nav view_map.launch
```



The gap at the back of the robot is due to the occlusion of the installation position of the display screen, so a certain range of Lidar data is shielded. The shielding range can be adjusted or not shielded according to the actual situation. For details, see [01. Lidar Basic Course].

7.2.2 Control the robot

- The keyboard controls the movement of the robot

```
rosrun  teleop_twist_keyboard teleop_twist_keyboard  # .py system     integration
roslaunch  yahboomcar_ctrl  yahboom_keyboard.launch         # custom
```

- The handle controls the movement of the robot

Make the robot walk all over the area to be built, and the map is as closed as possible.

There may be some scattered points during the mapping process. If the mapping environment is well closed and regular, the movement is slow, and the scattering phenomenon is much smaller.

7.2.3 Map save

```
rosrun map_server map_saver -f ~/yahboomcar_ws/src/yahboomcar_nav/maps/my_map
# The first way
bash ~/yahboomcar_ws/src/yahboomcar_nav/maps/map.sh
# second way
```

The map will be saved to ~/yahboomcar_ws/src/yahboomcar_nav/maps/ folder, a pgm image, a yaml file.

map.yaml

```
image :  map.pgm
resolution :  0.05
origin : [-15.4,-12.2,0.0]
negate :  0
occupied_thresh :  0.65
free_thresh:  0.196
```

Parameter parsing:

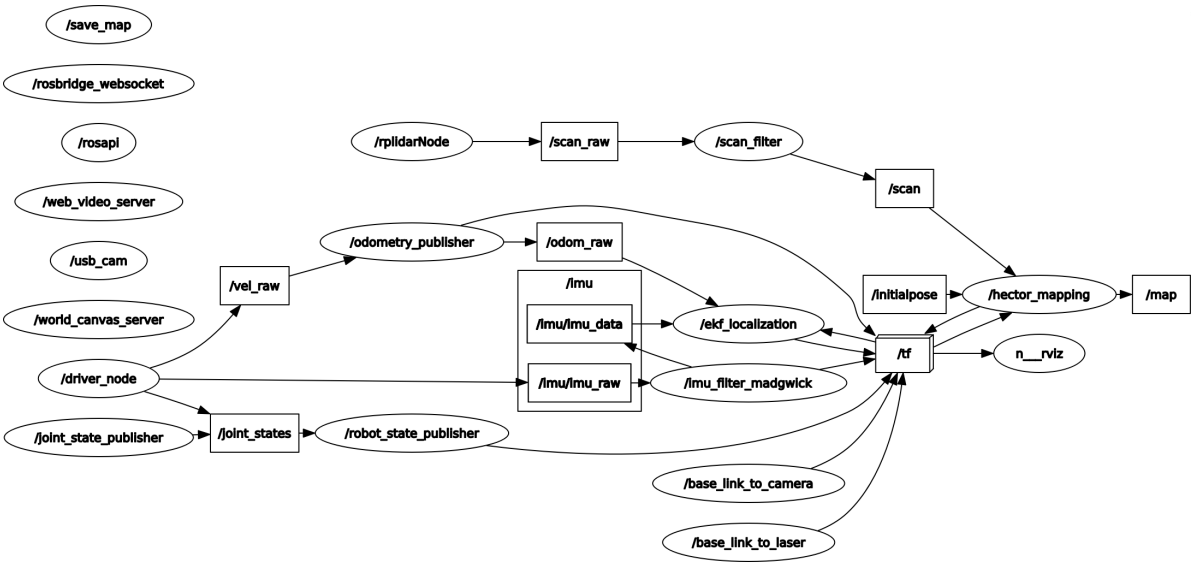
- image: The path of the map file, which can be an absolute path or a relative path
- resolution: the resolution of the map, m/pixel
- origin: The 2D pose(x, y, yaw) of the lower left corner of the map, where yaw is rotated counterclockwise(yaw=0 means no rotation). Many parts of the system currently ignore the yaw value.
- negate: whether to reverse the meaning of white/black, free/occupy(the interpretation of the threshold is not affected)
- occupied_thresh: Pixels with an occupancy probability greater than this threshold will be considered fully occupied.
- free_thresh: Pixels with an occupancy probability less than this threshold will be considered completely free.

7.3 Topics and Services

Topic subscription	type	describe
scan	sensor_msgs/LaserScan	Depth data from lidar scans
syscommand	std_msgs/String	system command. If the string is equal to "reset", the map and robot pose are reset to the initial state
Topic release	**type**	**describe**
map_metadata	nav_msgs/MapMetaData	Publish map Meta data
map	nav_msgs/OccupancyGrid	Publish map raster data
slam_out_pose	geometry_msgs/PoseStamped	Covariance-Free Robot Pose Estimation
poseupdate	geometry_msgs /PoseWithCovarianceStamped	Robot Pose Estimation with Gaussian Uncertainty Estimation
Serve	**type**	**describe**
dynamic_map	nav_msgs/GetMap	Get map data
reset_map	std_srvs/Trigger	Call this service to reset the map and the hector will create a brand new map from scratch. Note that this will not restart the robot's pose, 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 the map(if paused)

Node view

```
rqt_graph
```



7.4 configuration parameters

parameter	type	Defaults	describe
~base_frame	String	"base_link"	Robot base coordinate system for positioning and transformation of laser scan data
~map_frame	String	"map"	The coordinate system of the map
~odom_frame	string	"odom"	Odometer coordinate system(essentially the coordinate system pointed to by map)
~map_resolution	Double	0.025(m)	map resolution, edge length of grid cells
~map_size	Int	1024	the size of the map
~map_start_x	double	0.5	The origin of /map [0.0, 1.0] is relative to the grid map on the x-axis
~map_start_y	double	0.5	The origin of /map [0.0, 1.0] is relative to the grid map on the y-axis
~map_update_distance_thresh	double	0.4(m)	The threshold for map update, which is calculated from the first update on the map until the straight distance reaches the parameter value and is updated again
~map_update_angle_thresh	double	0.9(rad)	The threshold for map update, which is calculated from one update on the map until the rotation reaches this parameter value and then update again
~map_pub_period	double	2.0	map release cycle
~map_multi_res_levels	int	3	Map Multi-Resolution Grid Series
~update_factor_free	double	0.4	The map used to update free cells, the range is [0.0, 1.0]
~update_factor_occupied	double	0.9	The map used to update the occupied cells, the range is [0.0, 1.0]
~laser_min_dist	double	0.4(m)	The minimum distance of the laser scan point, the scan point less than this value will be ignored

parameter	type	Defaults	describe
~laser_max_dist	double	30.0(m)	The maximum distance of the laser scan point, the scan point less than this value will be ignored
~laser_z_min_value	double	-1.0(m)	The minimum height relative to the lidar, scan points below this value will be ignored
~laser_z_max_value	double	1.0(m)	Relative to the lidar's maximum height, scan points above this value will be ignored
~pub_map_odom_transform	bool	true	Whether to publish the coordinate conversion between map and odom
~output_timing	bool	false	Process the output timing information of each laser scan through ROS_INFO
~scan_subscrible_queue_size	int	5	Queue size for scan subscribers
~pub_map_scanmatch_transform	bool	true	Whether to publish the coordinate transformation between scanmatcher and map
~tf_map_scanmatch_transform_frame_name	String	"scanmatcher_frame"	Scanmatcher's coordinate system naming

7.5 TF transformation

Required TF Transform	**describe**
laser-->base_link	Usually a fixed value, the transformation between the lidar coordinate system and the base coordinate system is generally published by robot_state_publisher or static_transform_publisher
Published TF Transform	**describe**
map-->odom	The current estimate of the robot pose within the map frame(only provided if parameter "pub_map_odom_transform" is true).

View tf tree

```
rosrun rqt_tf_tree rqt_tf_tree
```