Part 1 SLAM Map Construction Principles

1. Introduction to SLAM

SLAM (Simultaneous Localization and Mapping) refers to **instantaneous localization and mapping**.

Localization is to **locate the position of the body in a coordinate system**. The **origin and attitude** of the coordinate system can be obtained **from the first key-frame**, **an existing global map or road map point, or GPS**.

Mapping refers to the **creation of a map of the environment perceived by the robot**, and the **basic geometric elements of the map are points**. **The main role of the map is to localize and navigate**, **navigation can be split into guidance and navigation**, **guidance includes global planning and local planning**, **navigation is to control the robot movement after planning**.

2. Principle of SLAM Mapping

There are three main processes involved in SLAM mapping:

1. Preprocessing: **Optimize the point cloud raw data of the radar, eliminate some problematic data, or filter it.**

Using laser as a signal source, a pulsed laser is emitted by a laser, which hits the surrounding obstacles and causes scattering.

A part of the light wave will be reflected to the receiver of the LIDAR, and then calculated according to the principle of laser ranging, the distance from the LIDAR to the target point can be obtained.

About the point cloud: In common terms, the information about the surrounding environment acquired by LiDAR is called the point cloud. It is a reflection of the part of the robot's environment that is visible to the “eye”. The information it collects about an object is presented as a series of dispersed pieces of information with precise angles and distances.

2. Matching: The point cloud data of this current localized environment is matched by searching the corresponding position on the established map.

Usually, the laser SLAM system calculates the laser LIDAR's distance and attitude change relative to the motion by matching and comparing two pieces of point clouds at different moments, which also accomplishes the localization of the robot itself.

3. Map Fusion: stitching new rounds of data from LiDAR into the original map to finalize the map update.

3. Mapping Notes

1. When starting a mapping build, it's best for the robot to face a straight wall, or use a closed cardboard box instead, to allow the radar to sweep as many points as possible

2. Try to ensure the completeness of the map as much as possible, in the possible path of the robot, all the surrounding 360 ° area, need to be detected by the LIDAR, the purpose is to increase the completeness of the map.

3. In some large environments to build a map, it is best to let the robot first complete the building of the closed loop, and then go to the detailed scanning of the environment of all the small details.

4. Judgment of Mapping results

Finally, after the map has been constructed, the following points can be used to determine whether the results satisfy the requirements for navigation:

1. Whether the edges of obstacles in the map are clear;

2. Whether there are areas in the map that are inconsistent with the actual environment (e.g., the presence or absence of closed loops);

3. Whether there are gray areas in the map within the robot's area of operation (e.g., areas not scanned);

4. Whether there are obstacles in the map that will not be present during subsequent localization (e.g., moving obstacles);

5. Whether or not the map ensures that the robot has been detected anywhere in the active area within a 360-degree field of view.

Part 2 Gmapping Mapping Algorithm

1. Gmapping Description

Gmapping algorithm is based on **RBPF particle filter algorithm** which **separates the process of localization and mapping**. It will localize first, then perform mapping. Gmapping algorithm makes effective use of wheel odometer information, hence it is not strict with Lidar frequency.

As the scene enlarges, more particles are required. And each particle carries a map, which leads to overwhelming computation and storage occupation in building a large map. Therefore Gmapping is more applicable to map the small scene.

2. Gmapping Principle

Gmapping algorithm will estimate the Lidar pose firstly based on the previous map and motion model. Then it will compute the weight according to the sensor observation to resample and update the map of particle. It will execute these steps in cycle to complete mapping.

But RBPF particle filter algorithm has defects. **It requires ample particles**, and **resamples frequently.** Large amount of particles will result in great computation and storage occupation. And Gmapping has proposed two solutions for these two problems, **including improving proposal distribution and selective resampling.**

3. Start Lidar Mapping

3.1 Enable Service

1) Start JetAuto, and then connect it to NoMachine.

2) Open the command line terminal

3) Input command "**sudo systemctl stop start_app_node.service**" and press Enter to stop APP service.

4) Open a new terminal, and input command "**roslaunch jetauto_slam slam.launch slam_methods:=gmapping**" to enable mapping service. If no error is reported, modeling service starts successfully.

5) Open a new command line terminal, and input command "**roslaunch jetauto_slam rviz_slam.launch slam_methods:=gmapping**" to open the model viewing software.

3.2 Start Mapping

1) Open a new terminal, and input command "**roslaunch jetauto_peripherals teleop_key_control.launch**" and press Enter to enable the keyboard control service.

If you receive the following hint, the keyboard control service is enabled successfully.

2) Control the robot to move around to map by pressing the corresponding keys.

3.3 Save the Map

1) Open a new terminal and input the command "**roscd jetauto_slam/maps**" and press Enter to enter the folder where the map is stored.

2) Input command "**rosrun map_server map_saver -f map_01 map:=/jetauto_1/map**" and press Enter to store the map.

"**map_01**" in the command is the name of the map, and you can rename it. If the following prompts occur, the map is kept successfully

3) If you want to stop running the program, you can press "**Ctrl+C**".

Gmapping Mapping Algorithm Launch file analysis

1. Introduction to the documents

1.1 File path

The main documents involved are listed below.

slam.launch:**Selecting a mapping method**
(location: /ros_ws/src/jetauto_slam/launch/slam.launch)

slam_base.launch:**Basic topic settings of the mapping method and startup of the mapping function** (location: /ros_ws/src/jetauto_slam/launch/include/slam_base.launch)

gmapping.launch:**Specific topic settings and parameter settings for the mapping method**
(location: jetauto_slam/launch/include/gmapping.launch)

1.2 Framework

1.3 Main Contents

According to the structure of the file, it is mainly analyzed by **Selecting a mapping method**, **Basic topic settings of the mapping method and startup of the mapping function**, **Specific topic settings and parameter settings for the mapping method,**and the specific syntax can be referred to the “ROS Basic Course”.

1.3.1 Selection of mapping method

```
<?xml version="1.0"?>
<launch>
  <!--建图方法选择-->
  <arg name="slam_methods" default="gmapping" doc="slam type
    [gmapping, cartographer, hector, karto, frontier, explore, rrt_exploration, rtabmap]"/>

  <arg name="gmapping"       default="gmapping"/>
  <arg name="cartographer"   default="cartographer"/>
  <arg name="hector"         default="hector"/>
  <arg name="karto"          default="karto"/>
  <arg name="frontier"       default="frontier"/>
  <arg name="explore"        default="explore"/>
  <arg name="rrt_exploration" default="rrt_exploration"/>
  <arg name="rtabmap"        default="rtabmap"/>

  <arg name="sim"            default="false"/>

  <include file="$(find jetauto_slam)/launch/include/jetauto_robot.launch">
    <arg name="sim"               value="$(arg sim)"/>
    <arg name="robot_name"   value="$(arg robot_name)"/>
    <arg name="master_name"   value="$(arg master_name)"/>
  </include>
  <include file="$(find jetauto_slam)/launch/include/slam_base.launch">
    <arg name="sim"            value="$(arg sim)"/>
    <arg name="slam_methods"    value="$(arg slam_methods)"/>
  </include>
</launch>
```

Before building a diagram, you need to select the diagram building method, as shown above:

slam_methods Indicates the method to build the map, the default value is gmapping, and the optional methods to build the map are:
Mapping manually: **gmapping**、**cartographer**、**hector**、**karto**

Autonomous mapping: **frontier**、**explore**、**rrt_exploration**

3D mapping: **rtabmap**

If you use the gmapping method to build a map, the command behavior in the terminal when executing this launch file: **“roslaunch jetauto_slam slam.launch slam_methods:=gmapping”**

1.3.2 Startup of the map building function and configuration of related topics

```
<?xml version="1.0"?>
<launch>
  <!--建图方法选择-->
  <arg name="slam_methods" default="gmapping" doc="slam type
    [gmapping, cartographer, hector, karto, frontier, explore, rrt_exploration, rtabmap]"/>

  <arg name="gmapping"        default="gmapping"/>
  <arg name="cartographer"    default="cartographer"/>
  <arg name="hector"          default="hector"/>
  <arg name="karto"           default="karto"/>
  <arg name="frontier"        default="frontier"/>
  <arg name="explore"         default="explore"/>
  <arg name="rrt_exploration" default="rrt_exploration"/>
  <arg name="rtabmap"         default="rtabmap"/>

  <arg name="sim"             default="false"/>

  <include file="$(find jetauto_slam)/launch/include/jetauto_robot.launch">
    <arg name="sim"               value="$(arg sim)"/>
    <arg name="robot_name"   value="$(arg robot_name)"/>
    <arg name="master_name"   value="$(arg master_name)"/>
  </include>
  <include file="$(find jetauto_slam)/launch/include/slam_base.launch">
    <arg name="sim"             value="$(arg sim)"/>
    <arg name="slam_methods"    value="$(arg slam_methods)"/>
  </include>
</launch>
```

After selecting the slam_methods method, you need to start the mapping function via slam_base.launch and set the topic name.

As in the above figure, <sim> indicates whether to use simulation or not, here we refer to the parameter sim above, the default is not to start the node simulation false, <slam_methods> that is, the method of building the map, according to the method of building the map, it is set to “gmapping”, and <robot_name> indicates the name of the node of the robot.

For details on the contents of the slam_base.launch file, please refer to “slam_base.launch File Analysis”.

1.3.3 Startup of the map building function and configuration of related topics

```
<group if="$(eval slam_methods == 'gmapping')">
        <include file="$(find jetauto_slam)/launch/include/$(arg slam_methods).launch">
          <arg name="scan"       value="$(arg scan_topic)"/>
          <arg name="base_frame"  value="$(arg base_frame)"/>
          <arg name="odom_frame"  value="$(arg odom_frame)"/>
          <arg name="map_frame"   value="$(arg map_frame)"/>
        </include>
</group>
```

In the slam_base.launch file launched at the same time, as shown in the figure above, it contains the launch file of the corresponding mapping method, according to the mapping method gmapping, then the mapping method parameter configuration file should note the gmapping.launch file (location: jetauto_slam/launch/include/gmapping.launch). include/gmapping.launch).

Topic Parameter Configuration

```
  <!-- Arguments -->
  <arg name="scan"       default="scan"/>
  <arg name="base_frame"  default="base_footprint"/>
  <arg name="odom_frame"  default="odom"/>
  <arg name="map_frame"   default="map"/>
```

The above figure shows some topic parameter configurations for the mapping method:

<scan> for LIDAR scan topics.

<base_frame> The name of the robot polar coordinate system topic, set to base_footprint.

<odom_frame> Name of the odometer topic, set to odom.

<map_frame> Map topic name, set to map.

This can be viewed via the **rostopic list**

Basic Parameter Configuration File

```
  <!-- Gmapping -->
  <node pkg="gmapping" type="slam_gmapping" name="jetauto_slam_gmapping"
output="screen">
      <param name="base_frame"    value="$(arg base_frame)"/>
      <param name="odom_frame"    value="$(arg odom_frame)"/>
      <param name="map_frame"     value="$(arg map_frame)"/>
      <remap from="/scan"        to="$(arg scan)"/>
      <rosparam command="load"    file="$(find jetauto_slam)/config/gmapping_params.yaml" />
  </node>
```

In addition to the basic topics passed into the gmapping function package, there is a configuration file, gmapping_params.yaml (the configuration file for the basic parameters, the path can be localized to: jetauto_slam/config/gmapping_params.yaml):

```
map_update_interval: 0.1
maxUrange: 5.0
maxRange: 12.0
sigma: 0.05
kernelSize: 1
lstep: 0.05
astep: 0.05
iterations: 1
lsigma: 0.075
ogain: 3.0
lskip: 0
minimumScore: 50
```

Above is a snippet of a portion of the gmapping_params.yaml parameter file, the parameters to note are:

Name	Affect
map_update_interval	Map update speed, in seconds (s), generally the smaller the setting, the greater the amount of computation required.
maxUrange	Detection of the maximum usable range, i.e. the range that the beam can reach
maxRange	Sensor maximum range. If there are no obstacles within the sensor's distance range it should be shown as free space on the map
sigma	Matching method for endpoints
lstep	Translation optimization step
astep	Rotation optimization step
lsigma	Laser standard deviation of scanning match probability
minimumScore	Avoid using limited distance laser scanners in large space areas to get a good match

```
srr: 0.01
srt: 0.02
str: 0.01
stt: 0.02
```

Name	Affect
srr	Mileage error in translation as a function of translation (radians)
srt	Mileage error in translation as a function of rotation (radians)
str	Mileage error in rotation as a function of translation (angle)
stt	Mileage error in rotation as a function of rotation (angle)

Part 3 Hector Mapping Algorithm to Part 9 RRT Exploration Mapping

TBD