

1.怎么安装基线算法？

基线算法是在 rrt_exploration 功能包的基础上进行改进的, 所以我们需要首先安装 rrt_exploration 功能包。这里主要参考以下网址：

http://wiki.ros.org/rrt_exploration/Tutorials 使用命令行安装 gmapping、navigation、python 组件等依赖。

安装依赖：

```
$ sudo apt-get install ros-melodic-gmapping ros-melodic-navigation  
$ ros-melodic-kobuki ros-melodic-kobuki-core ros-melodic-kobuki-gazebo  
$ sudo apt-get install python-opencv python-numpy python-scikits-learn
```

这里注意 melodic 的 kobuki 需要使用源码安装， 安装方法 1 如下：

安装依赖：

```
$ sudo apt-get install ros-melodic-kobuki-*  
$ sudo apt-get install ros-melodic-ecl-streams  
$ sudo apt-get install libusb-dev  
$ sudo apt-get install libspnav-dev  
$ sudo apt-get install ros-melodic-joystick-drivers  
$ sudo apt-get install bluetooth  
$ sudo apt-get install libbluetooth-dev  
$ sudo apt-get install libcwiiid-dev
```

新建工作空间:

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

```
$ catkin_init_workspace
```

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

添加工作空间到 bashrc 文件：

```
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

启用环境变量：

```
$ source ~/.bashrc
```

<https://github.com/qingchen-bi/kobuki-package.git> 下载 kobuki_package,

解压到 catkin_ws/src 目录下，编译工作空间。

接下来我们再安装 rrt_exploration 和 rrt_exploration_tutorials 两个功能包，我们只需要下载两个功能包到刚刚安装 kobuki 的工作空间下即可：

```
$ cd ~/catkin_ws/src/
```

```
$ git clone https://github.com/hasauino/rrt\_exploration.git
```

```
$ git clone https://github.com/hasauino/rrt\_exploration\_tutorials.git
```

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

安装过程中可能出现的错误解决方法见附录。如果安装不上，请参考附录安装方法 2。

2. 怎么使用基线算法？

参考网址：

http://wiki.ros.org/rrt_exploration/Tutorials/singleRobot 我们就可以运行基线

算法进行单机器人探索：

```
$ roslaunch rrt_exploration_tutorials single_simulated_house.launch
```

```
$ roslaunch rrt_exploration single.launch
```

然后我们点击 rviz 中的 Publish Point 选项，依次在 rviz 的左上角、左下角、右下角、右上角定义机器人的初始探索区域范围，最后点击中心位置，定义全局探索 rrt 树的初始生长点，随后机器人便开始运行。运行过程中可能出现的错误解决方法见附录。



这里介绍如何把机器人替换为 husky：

首先 <https://github.com/qingchen-bi/husky-exploration.git> 下载 husky 机器人模型，并安装 husky 机器人模型在同一个工作空间；

然后 <https://github.com/qingchen-bi/husky-exploration-launch.git> 下载

single_simulated_exploration_husky.launch

move_baseSafe_teb_husky.launch

运行 single_simulated_exploration_husky.launch 文件和 single.launch

文件即可。

注意修改 single_simulated_exploration_husky.launch 包含的 move_baseSafe_teb_husky.launch 的路径，move_baseSafe_teb_husky.launch 中 TEB 配置文件的路径，或者换为自己计划使用的规划插件。

3.基线算法的原理简述

输入：此算法需要在rviz中利用Publish Point 定位四个点定义探索范围，定位最后一个点，定义全局rrt树生长初始位置。然后，rrt搜索树生长触碰已知和未知区域的分界点

输出：筛选出位于已知和未知区域的边界点作为探索目标点发布给机器人去探索，最终利用move_base 和 gmapping 前往探索目标点，并绘制未知区域的地图。

原理：基线算法是在rrt_exploration进行改进的，基线算法本质上是基于边界点的探索算法，只不过它是通过rrt算法生成随机树，使用随机树去触碰位于已知区域和未知区域之间的分界点，检测机器人即将探索的边界点，边界点检测器如图3-1所示：

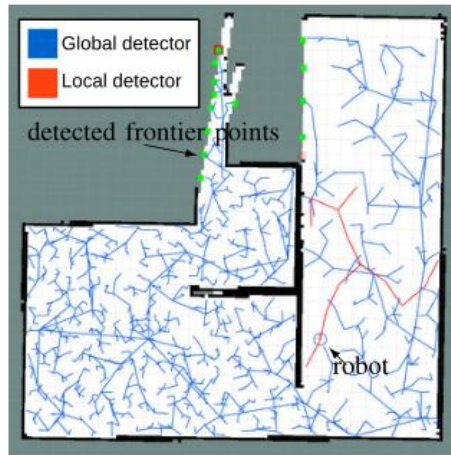


图3-1：边界点检测器

其中，局部边界点检测器检测边界点的过程如图3-2所示：

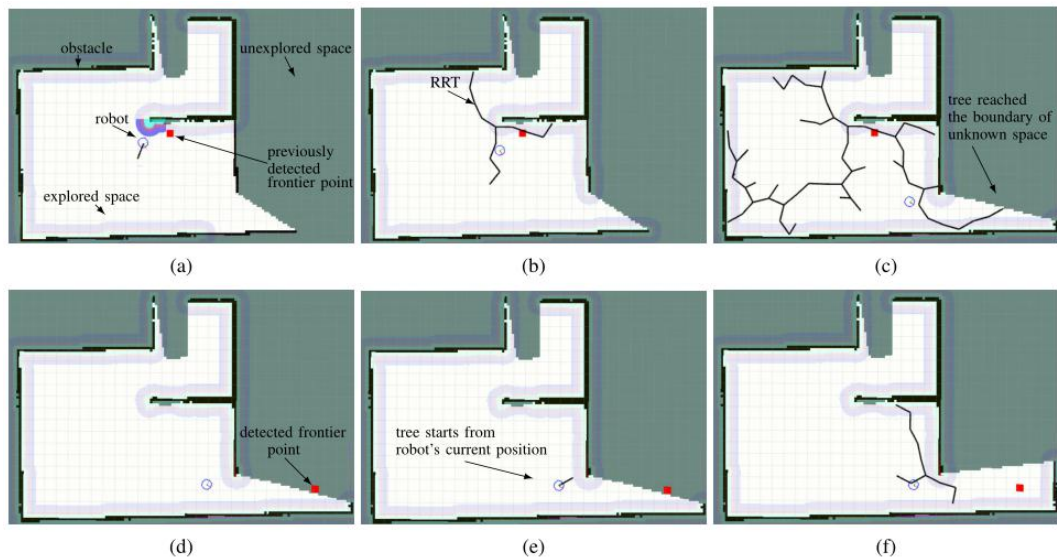


图3-2：局部边界点检测器检测边界点过程

如图 3-3 所示，基线算法主要分为五个模块：

1) Frontier Detector 部分：

采用基于 frontier（边界）的探索算法。为提高自主探索质量和效率，设计

了全局-局部的两层 frontier 检测框架。在运行 frontier 检测器时，需要在可视化界面设置 5 个点，前四个点构成搜索范围，最后一点是 RRT 搜索树的根节点。全局 frontier 检测器和局部 frontier 检测器根据机器人初始的位姿以及设定的探索边界进行边界点的检测。

检测 frontier 的方法为，通过生长的 RRT 搜索树去触碰已知区域与未知区域的分界线，触碰到分界线的这一节点被认为是边界点，将其作为后续筛选探索目标点的候选点。如图 3-1 所示，蓝色为全局 RRT 搜索树，红色为局部 RRT 搜索树，绿色的点为边界点。

全局 frontier 检测器指的是，以探索开始设置的初始点为根节点的全局 RRT 搜索树，全局 RRT 搜索树会随着探索进程一直生长，去触碰寻找 frontier；局部 frontier 检测器指的是，以每个机器人的位置为根节点的局部 RRT 搜索树，局部 RRT 搜索树会在机器人当前位置进行生长，每当触碰到新的 frontier 就会清空局部 RRT 搜索树，然后再以机器人当前新位置为新的根节点重新生长。

2) Map updates 部分：

机器人生成的地图和机器人位姿反馈给到全局 frontier 检测器和局部 frontier 检测器。

3) Filter 部分：

Frontier 检测器将检测到的边界点发送给过滤器 Filter。过滤器依据局部代价栅格地图进行聚类，聚类后的点发送给分配模块 Assigner。

4) Assigner 部分：

分配模块接收地图并通过计算每个机器人与每个候选点之间的探索增益来选出每个机器人下一个探索的目标点。探索增益的计算主要考虑以下两个条件：

导航成本:定义为机器人当前位置到达将要探索的候选目标位置的欧式距离。

信息增益:定义为给定的候选目标点周围一定区域内包含的未知区域的面积。

5) Robot 部分:

最后, 选出具有最大探索增益的目标点, 将选出的目标点发送给运动规划模块。该模块根据当前机器人位姿和实时的传感数据生成速度命令并发送给机器人, 以此来控制机器人向目标点运动。

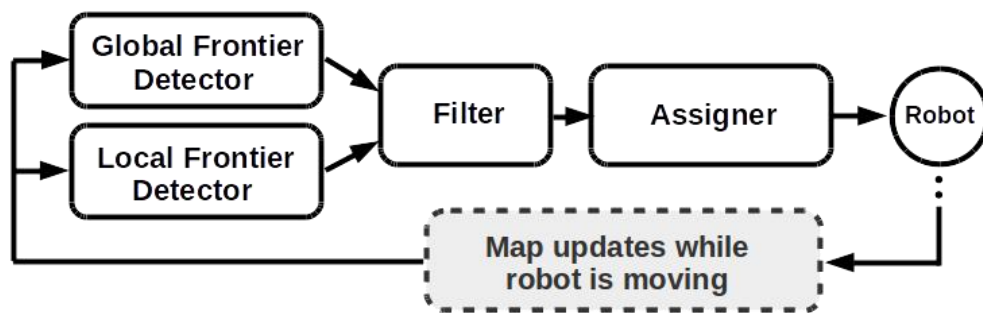


图 3-3. 机器人探索流程图

4.附录

1).源码安装 kobuki 方法 2:

我们参考以下网址:

<https://www.ncnynl.com/archives/201903/2884.html> 安装方法见下:

安装依赖:

```
$ sudo apt-get install ros-melodic-kobuki-*
```

```
$ sudo apt-get install ros-melodic-ecl-streams
```

```
$ sudo apt-get install libusb-dev
```

```
$ sudo apt-get install libspnav-dev
```

```
$ sudo apt-get install ros-melodic-joystick-drivers
```

```
$ sudo apt-get install bluetooth
```

```
$ sudo apt-get install libbluetooth-dev
```

```
$ sudo apt-get install libcwiiid-dev
```

新建工作空间，准备相关包：

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

```
$ git clone https://github.com/turtlebot/turtlebot_simulator
```

```
$ git clone https://github.com/turtlebot/turtlebot.git
```

```
$ git clone https://github.com/turtlebot/turtlebot_apps.git
```

```
$ git clone https://github.com/udacity/robot_pose_ekf
```

```
$ git clone https://github.com/ros-perception/depthimage_to_laserscan.git
```

```
$ git clone https://github.com/yujinrobot/kobuki_msgs.git
```

```
$ git clone https://github.com/yujinrobot/kobuki_desktop.git
```

```
$ cd kobuki_desktop/
```

```
$ rm -r kobuki_qtestsuite
```

```
$ git clone https://github.com/toeklk/orocos-bayesian-filtering.git
```

```
$ git clone https://github.com/turtlebot/turtlebot_msgs.git
```

```
$ git clone https://github.com/ros-drivers/joystick_drivers.git
```

复制 kobuki 和 yujin_ocs 依赖库到 catkin_ws/src 工作空间下：

```
$ mkdir -p ~/repos/
```



```
$ cd ~/repos/

$ git clone https://github.com/yujinrobot/kobuki.git

$ cp -r kobuki/* ~/catkin_ws/src/

$ git clone https://github.com/yujinrobot/yujin_ocs.git

$ cp -r yujin_ocs/yocs_cmd_vel_mux/ yujin_ocs/yocs_controllers

~/catkin_ws/src/
```

编译工作空间：

```
$ cd ~/catkin_ws

$ catkin_make
```

添加工作空间到 bashrc 文件：

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

启用环境变量：

```
source ~/.bashrc
```

注意，这里建议将刚刚下载的有关 kobuki 的功能包归置到一个文件夹下。

例如，新建 kobuki_package 文件夹，然后都移入 kobuki_package 文件夹。

2).可能出现的错误的解决办法：

安装 kobuki 时 catkin_make 报错：

Error at /usr/share/cmake-3.5/Modules/FindPkgConfig.cmake:解决方案

A required package was not found
Call Stack (most recent call first):

```
/usr/share/cmake-3.5/Modules/FindPkgConfig.cmake:532  
(_pkg_check_modules_internal)  
CMakeLists.txt:18 (pkg_check_modules)
```

这个时候定位到相关的 `cmakelist.txt` 文件，看一看需要什么软件包。

例如没有 `bfl`，那么 `sudo apt-get install ros-<版本>-bfl` （贝叶斯滤波器库）

安装 kobuki 时出现

***** 没有规则可制作目标 “/usr/local/lib/libboost_system.so.1.74.0”，由
“/home.....**

这个时候会发现需要 1.74.0 版本的 `boost .so` 文件，可以安装 `boost1.74.0`

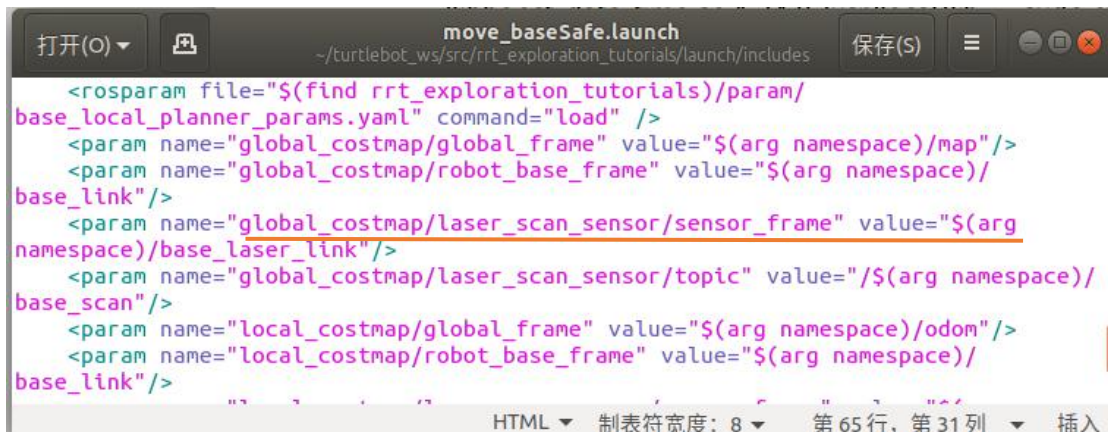
运行算法时报错：

TF Exception that should never happen for sensor frame:

/robot_1/base_laser_.....

参考下图划线部分修改

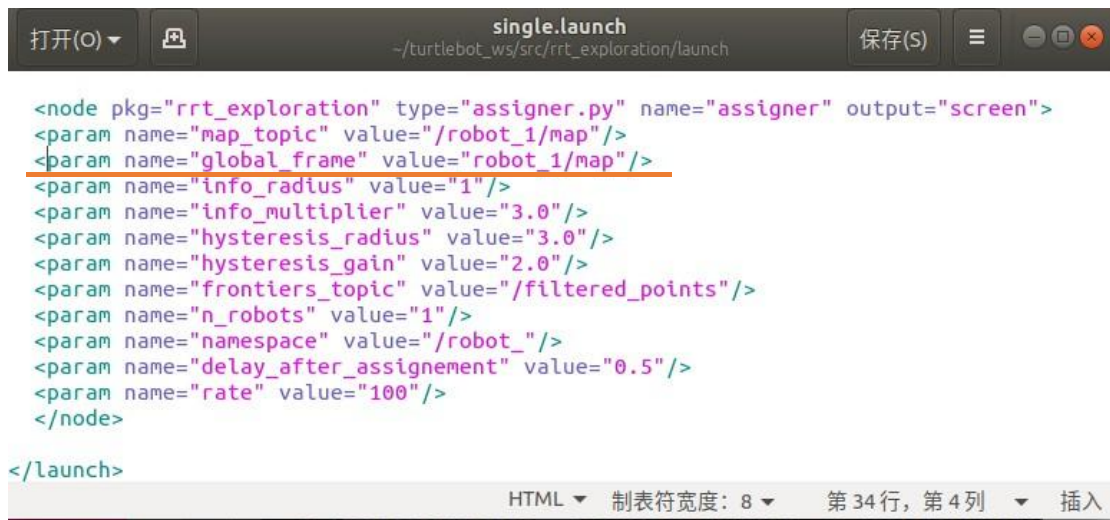
rrt_exploration_tutorials/launch/includes/move_baseSafe.launch



```
<rosparam file="$(find rrt_exploration_tutorials)/param/
base_local_planner_params.yaml" command="load" />
<param name="global_costmap/global_frame" value="$(arg namespace)/map"/>
<param name="global_costmap/robot_base_frame" value="$(arg namespace)/
base_link"/>
<param name="global_costmap/laser_scan_sensor/sensor_frame" value="$(arg
namespace)/base_laser_link"/>
<param name="global_costmap/laser_scan_sensor/topic" value="/$(arg namespace)/
base_scan"/>
<param name="local_costmap/global_frame" value="$(arg namespace)/odom"/>
<param name="local_costmap/robot_base_frame" value="$(arg namespace)/
base_link"/>
```

The goal pose passed to this planner must be in the robot_1/map frame. It is instead in the /robot_1/map frame.

参考下图划线部分修改 single.launch



```
single.launch
~/turtlebot_ws/src/rrt_exploration/launch

<node pkg="rrt_exploration" type="assigner.py" name="assigner" output="screen">
<param name="map_topic" value="/robot_1/map"/>
<param name="global_frame" value="robot_1/map"/>
<param name="info_radius" value="1"/>
<param name="info_multiplier" value="3.0"/>
<param name="hysteresis_radius" value="3.0"/>
<param name="hysteresis_gain" value="2.0"/>
<param name="frontiers_topic" value="/filtered_points"/>
<param name="n_robots" value="1"/>
<param name="namespace" value="/robot_1"/>
<param name="delay_after_assignment" value="0.5"/>
<param name="rate" value="100"/>
</node>

</launch>
```

另外，也可以在 rrt_exploration 文件夹中的 function.py 文件里 sendGoal()函数里面修改如下，这样就不需要修改 single.launch 文件了：

```
robot.goal.target_pose.header.frame_id = self.name[1:] + '/map'
```