_graph resulting from the navigation stack

UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Autonomous and Mobile Robotics

Final Project Report

Professors:
Gianluca Palli

Students:
Filippo Guarda

Academic year 2021/2022

Abstract

These projects are aimed to acquire an understanding of ROS and autonomous robotics theory trough an hands-on approach using a Turtlebot3 as a proof of concept.
The first task concerns the generation of a map trough the use of explore_lite and SLAM.
In the second task it is required to navigate the robot trough different rooms of the map, while the robot must simulate the task of irradiating the environment with uv light, reaching a predetermined value of irradiation in all rooms.

Contents

Chapter 1

Map Generation

In order to generate a map of the surrounding environment trough SLAM, the robot must move trough the room while recording the Lidar messages, in conjuction with the robot's odometry. In the case of the Turtlebot3, this is done by launching the navigation stack along with the SLAM stack.
There is a problem tough, this method only works trough teleoperation, this means the robot must be manually moved trough the environment, which is not a feasible mean of obtaining a large map.

1.1 Explore Lite

The package `explore_lite` provides greedy frontier exploration, and will keep exploring until all the frontiers are found. This makes it a perfect solution for our task.
One problem of `explore_lite` is that it both needs SLAM and a `move_base` node, this means we need four nodes to be running simultaneously:

- `turtlebot3_big_house_gazebo`
- `turtlebot3_slam`
- `turtlebot3_slam`
- `explore_lite`

We accomplish this by creating a launch file, `map_generation.launch` for ease of use.
Roslaunch-ing the file starts the four nodes along with a **roscore**, allowing for fast and easy environment mapping.

Figure 1.1 – explore_lite in action

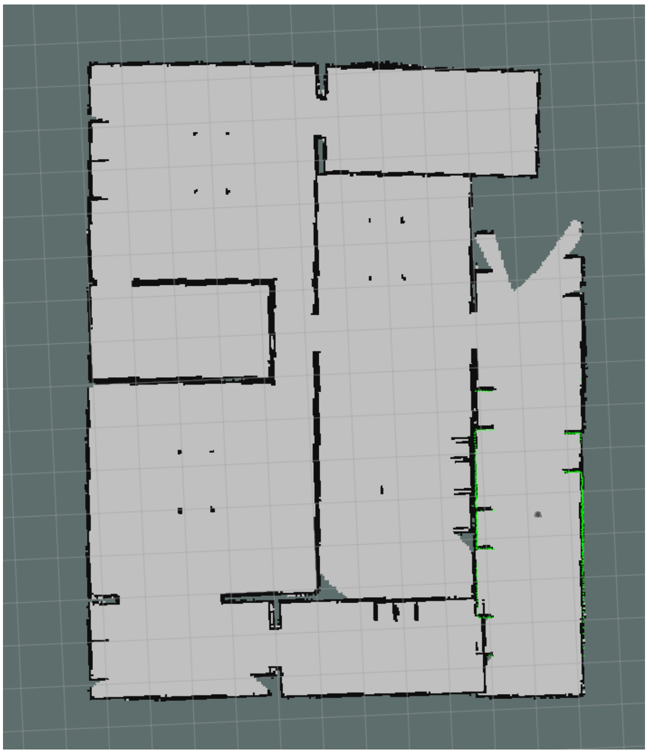
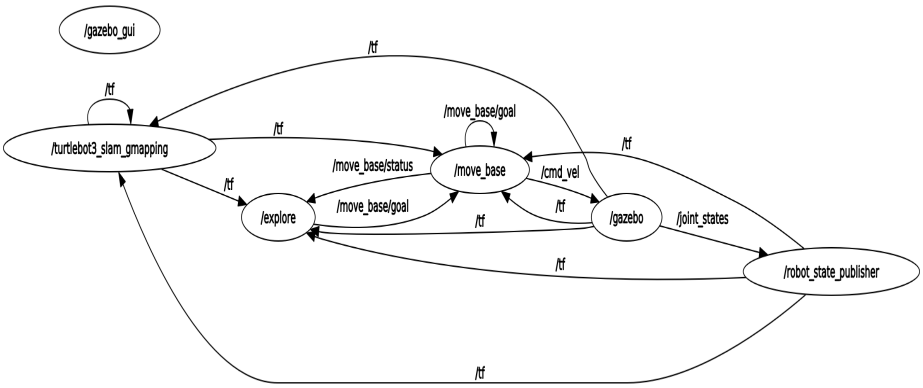


Figure 1.2 – The rqt_graph resulting from the launch file



Chapter 2

Navigation

The navigation task is subdivided in three different parts:
first, the Turtlebot must localize itself in the map without knowing a priori its position, then it must navigate to the center of a list of defined rooms. Lastly it has to start disinfecting the environments, while generating an irradiation map and ensuring all point in a room receive an irradiation of 10 w/m^2.

2.1 Localization

In order to localize itself, the Turtlebot uses the AMCL feature included in the libraries, while also moving itself in a bug-like algorithm without a set target. This ensures the robot explores enough map features to allow the AMCL to position itself precisely.
If by chance the position is incorrects, it's possible to spread again the AMCL particles trough a terminal command, restarting the process.

2.2 Center room navigation

After the localization ends, the localization node sends a message to the navigation node, then it shuts down. When the navigation node receives the message, it then enters a for cycle where room coordinates are taken from a text file, interpreted, and then sent as goal to the texttttmove_base node. This node is responsible for navigating the robot trough the various targets set by the navigation node.

2.3 Room disinfection

In order to ensure total irradiation of the rooms to be over the required amounts, it was necessary to write a ray tracer, which is responsible for creating an irradiation map that is layered over the one generated by the SLAM.
This is done by iterating over the SLAM map in search of obstacles, and preventing the irradiation update on pixels that would not be reached by light.
Irradiation is defined as follows:

$$E(x, y, k) = \sum_{i=0}^{k} \frac{P_I \Delta_t}{(x - p_x(i\Delta t)^2) + (y - p_y(i\Delta t)^2)}; \tag{2.1}$$

The irradiation algorithm works by defining a new submap for every room, this submap can then be iterated over while also taking into account obstacles and the robot position.
In order not to send unreachable goals, while in the submap iterator, smaller circle iterator are used. Every index becomes the center of a circle iterator that checks if there are obstacles or the robot nearby; if one of these is found, that position is considered unreachable and as such it is not sent to the following algorithm.
To generate the next goal, the position of the lowest irradiated point in the submap is sent from the `uv_map` node to the `sanitizer_navigation` node; this node then sends the goal to the `move_base` node, which in turn is responsible for the actual movement of the turtlebot. The process is repeated until the required irradiation is reached, then `sanitizer_navigation` reads the next room coordinates from the text file, navigates to the center of that room. `Sanitizer_navigation` also sends the room coordinates to `uv_map`, which generates a new submap, and so on.

Figure 2.1 – viable submap of a room

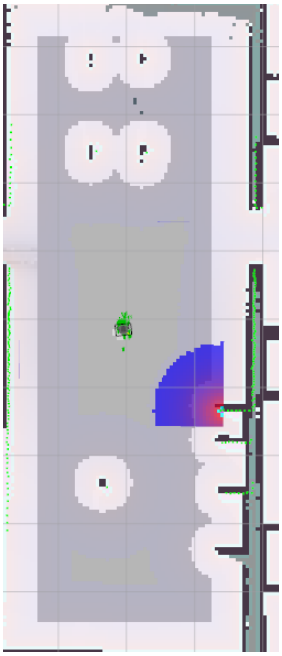
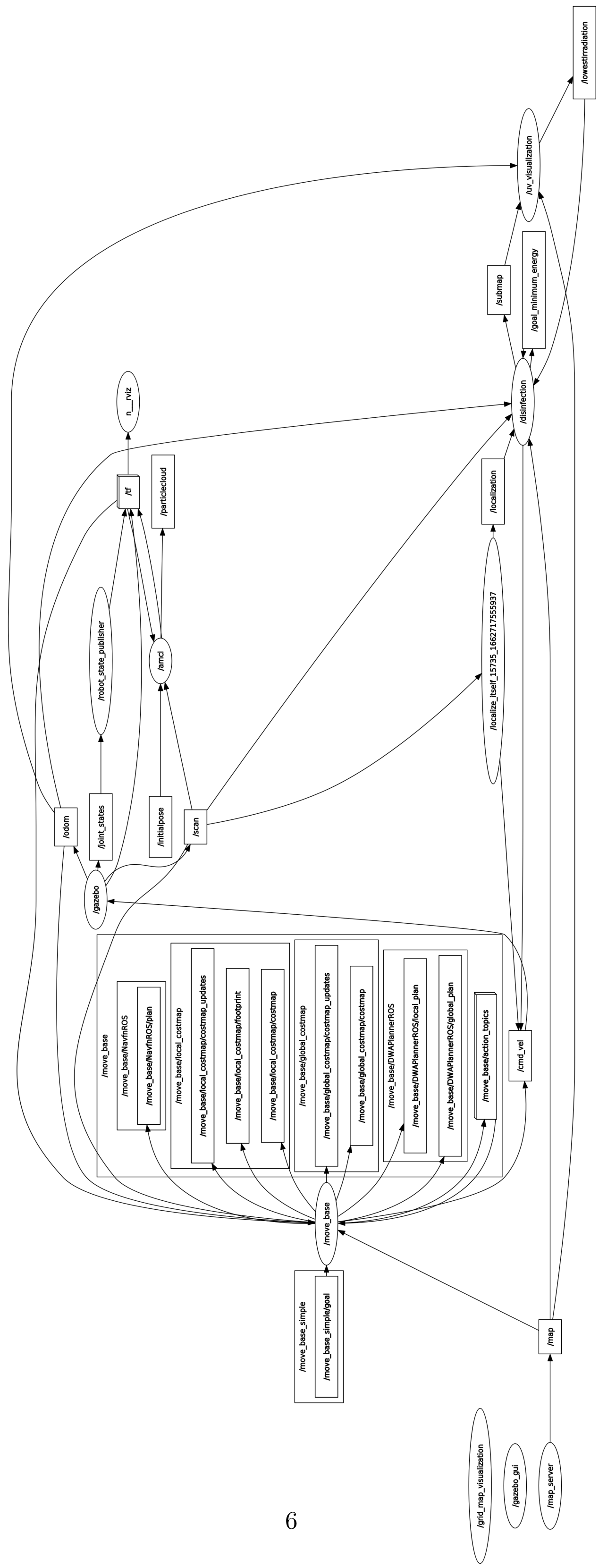


Figure 2.2 – uv_map of the rooms

Figure 2.3 – The rqt_graph resulting from the navigation stack



Conclusions

The objective of this work was to develop an understanding of distributed Autonomous and Mobile Robotics. This has been achieved by successfully completing a project that navigates and sanitizes and environment.