MOBILE ROBOT PROGRAMMING PROJECT: VIRTUAL MIR EXPLORATION AND AUTONOMOUS MAPPING

Final report

Authors:
Weronika Zawadzka 12244068
Jerzy Pawlik 12242751
Loris Persico 12244072

Klagenfurt University
1.07.2023

1 Task description

The goal of this project is to program a robot to explore an unknown environment and generate a complete map of it. The robot provides odometry information and data from laser scanners, but has no map of the environment nor localization at the beginning. We are to solve the problem of SLAM - Simultanous Localization and Mapping.

1.1 Mobile robot

The mobile robot used in this project is MIR100 from Mobile Industrial Robots (see Figure 1).

Figure 1: MIR100

1.2 Other hardware

Gazebo, open-source 3D robotics simulator, was used for the simulation. Odometry and laser scanner sensors are used.

1.3 Environment

Ros Noetic and C++ are used.

1.4 Rules

Final assignment rules state to deliver a ROS package containing implementation of an exploration strategy and safe navigation, a design document and a final report. For solving the task using all of the available ROS packages is allowed except for the hector mapping, which would decrease the grade for a project.

2 The approach

2.1 Initial idea

For SLAM process, gmapping package was supposed to be used. Gmapping provides laser-based SLAM and so creates 2D occupancy grid from data collected by robot. However, exploration algorithm is needed so that we are able to cover all of the map. Our idea was to use the explore_lite ROS package for exploration. It uses greedy frontier-based exploration method. Explore_lite subscribes to a navmsgs/OccupancyGrid and mapmsgs/OccupancyGridUpdate messages to construct a map where it looks for the frontiers. It sends the movement commands to ROS navigation stack to navigate in an autonomous manner. Frontiers are the border places between explored and unexplored areas, where robot can go. The algorithm chooses the closest frontier and sets it as the destination. That is to say, when the node is running the robot will autonomously explore the environment as long as no frontiers can be found.
After the described process, we planned to save the map using map_server. With complete map saved, we can begin path planning stage. For localization we plan to use AMCL, probabilistic localization system for a robot moving in 2D. Then, navigation stack can be used, which outputs safe velocity commands that are sent to a mobile base.

2.2 Initial tests

As hector_mapping package is already provided in MIR100 robot package, we decided to use it for initial tests to get ourselves familiarized with the workings of the SLAM. We used it with explore_lite so that the process of creating the map was automatic. These algorithms were not the easiest to run and perform, and often we did not get satisfactionary results - sometimes robot would lose its estimated position and start the mapping process from a different point. That resulted in doubled miss-matched not finished maps, which of course we did not want. Finally however, we were able to run it and obtain the finished map.

2.3 Changes made

Unfortunately we were not able to make gmapping solution work. We were having the exact problems as when using hector mapping method. Therefore, as we were finally able to obtain the finished map with the previous method, and this one failed, we decided to switch to hector mapping for this project. Our idea for this project is then summarized in Figure 2.

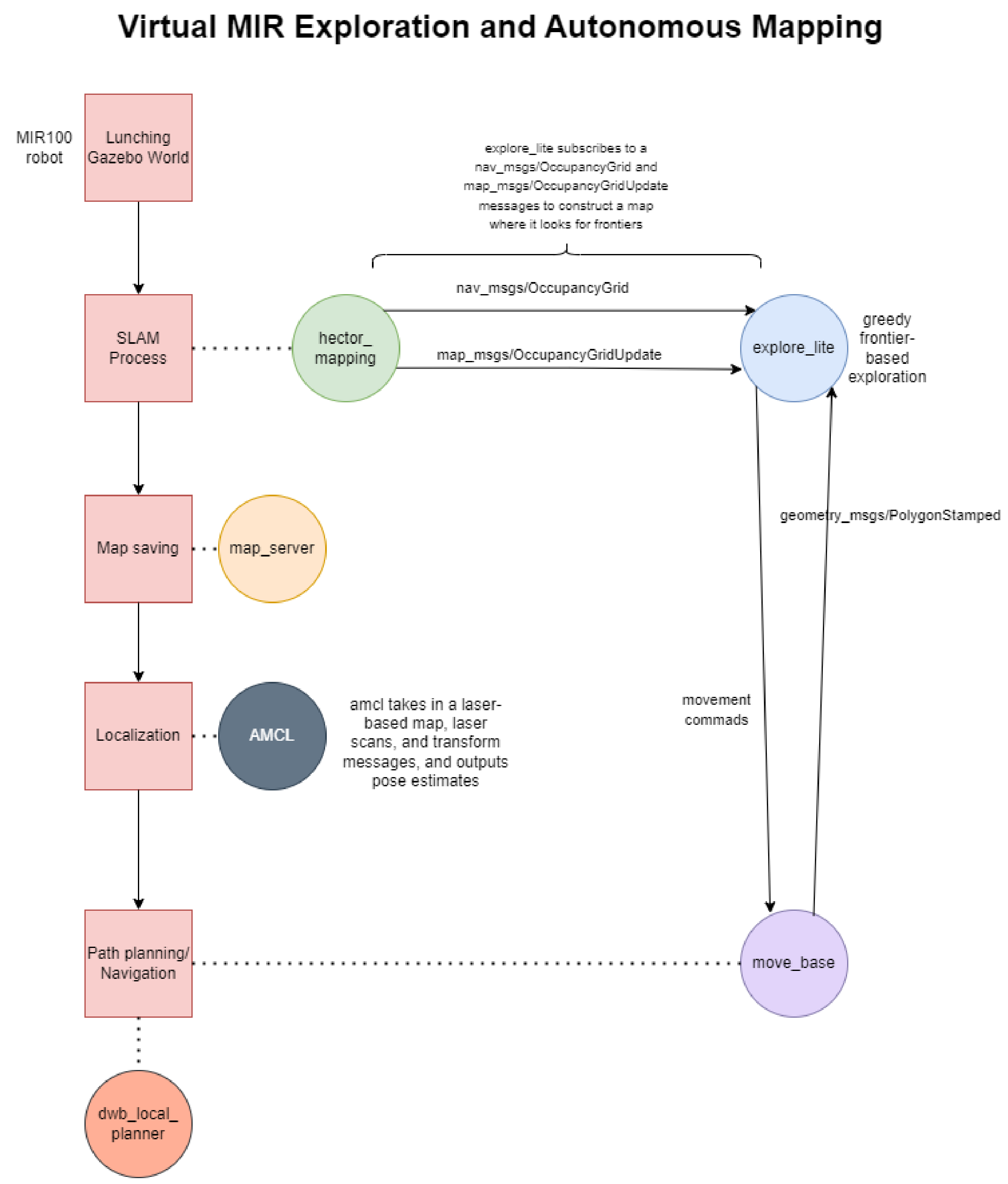
Virtual MIR Exploration and Autonomous Mapping
MIR100 robot
Lunching Gazebo World
explore_lite subscribes to a nav_msgs/OccupancyGrid and map_msgs/OccupancyGridUpdate messages to construct a map where it looks for frontiers
nav_msgs/OccupancyGrid
SLAM Process
hector_ mapping
map_msgs/OccupancyGridUpdate
explore_lite
greedy frontier-based exploration
geometry_msgs/PolygonStamped
Map saving
map_server
Localization
AMCL
amcl takes in a laser-based map, laser scans, and transform messages, and outputs pose estimates
movement commads
Path planning/ Navigation
move_base
dwb_local_ planner

Figure 2: Overview of the approach

2.4 Packages in use

- gmapping
- explore_lite
- navigation stack
- map_server
- move_base
- amcl
- dwb_local_planner

3 Results

3.1 Mapping

We created a package named mir_project inside our mir_robot directory. Inside it we have two folders: maps and launch. In the launch folder we submit 2 launch files, one called create_map.launch for SLAM process. The other one is for localization and navigation, and is called navigation.launch. To our package we provide a README file with all instructions of running the code. In the maps folder, we placed the best map we achieved with hector_mapping method, as visible in the Figure below.

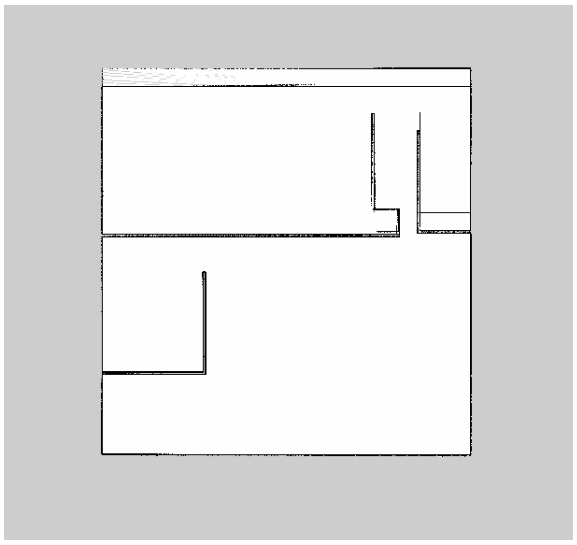

Figure 3: Created map

3.2 Localization and navigation

Here we used already provided amcl, move_base and dwb local planner packages. We loaded the map obtained in the previous step and then started the process of localization and navigation. The workings of this step is also not of best quality, sometimes the amcl pose estimation spread and robot moves in a oscillating manner. We checked it also with the map already provided by original MIR100 package, but the same issue appeared. That is to say, sometimes robot had problems to accurately define its position, but in general, it was able to reach desired destination.

3.3 ROS computation graph

The ROS computational graph for create_map process is visible in Figure below.

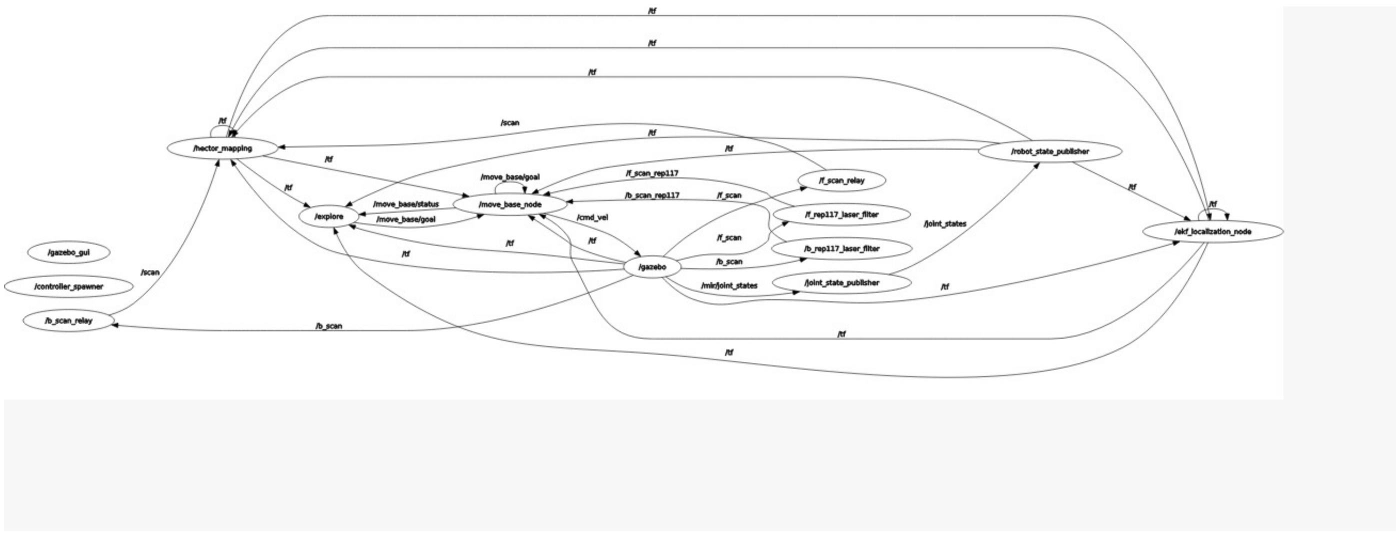

Figure 4: Nodes graph

4 Evaluation

4.1 Problems revisited

The biggest issues we had, as mentioned earlier, with process of creation of the map. We were trying to rerun the exploration multiple times, but every time in different moment the bugs or crashes were appearing. From what we have observed it was happening usually when the robot had to turn back, or calculate a new route. We were also trying to solve the bug by assigning more resources to virtual machine, changing the max step size in gazebo world physics parameters to make physics engine update the state of objects in the world more frequently and decreasing real time update rate and making the simulation slower to save resources. However none of it did succeed. What we also noticed, was that before a bug appeared there was a message on the console: "computeVelocityCommands exception: resulting plan has 0 poses in it" and later: "ERROR: failed to set start state", what indicates that move base is unable to find a valid path for the robot to follow. So we tried increased max planning retries and planner patience, which is a time after which planning the path is aborted. However again this didn't fix the bug completely. Moreover, we were able to make it work on one of our computers, so possibly computational power prevented us from achieving better results.

4.2 Map

Our result map is looking almost like the original one with a small difference on a top part. Original map can be seen below.

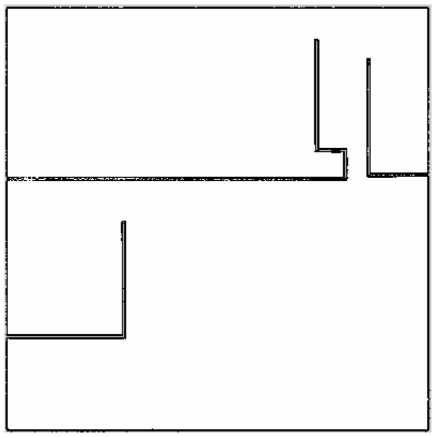

Figure 5: Original map

5 Conclusions

We were successfully able to create a map with simultaneous localization and mapping process and use created map for localization and navigation. We failed with using gmapping for the process even after hours spent on trying. However, this project was a big learning experience for us as we had no prior background with mobile robot programming.