MIR Cleaner - an autonomous mapping and vacuum platform

João Santos, 76912[1]

[1]Department of Electronics, Telecommunications and Informatics, University of Aveiro

July 2022

1 Introduction

One of the key responsibilities of each and every person is keeping the environment around us clean.

With their cutting-edge technologies, robots are playing a significant role in modern society and improving the quality of human life. The floor-cleaning robots are useful in households where they can substitute humans.

Fundamentally, the robot vacuum cleaners have stood out for their abilities. Some currently available products are based on the basic obstacle avoidance (employing infrared or ultrasonic sensors), while others use laser mapping technologies.

Each cleaning system has unique benefits and drawbacks. For instance, some of the robots that use laser mapping are considerably quicker, more energy-efficient but also more expensive. In contrast, due to their random cleaning, obstacle avoidance-based vacuum robots require more time and energy but are less expensive.

In this assignment, a new approach was implemented based on exploration and planning techniques, based on the MIR100 mobile base.

2 Installation

The base framework for this assignment was the Robot Operating System (ROS) Noetic Ninjemys and so it should be already installed. Additionally, Gazebo must also be installed. Rviz is also needed so the cleaning agent can be started and it also is helpful for visualization.

On top of the previous software, some ROS packages are also needed that have not been developed by the author. They are:

- *mir_gazebo*

- *mir_naviagtion*
- *ira_laser_tools*
- *move_base*
- *gmapping*
- *explore_lite*
- *amcl*
- *map_server*
- *costmap_2d*

To install them, use the following bash script:

```
#!/bin/bash
sudo apt install ros-noetic-mir-gazebo
sudo apt install ros-noetic-mir-navigation
sudo apt install ros-noetic-ira-laser-tools
sudo apt install ros-noetic-move-base
sudo apt install ros-noetic-gmapping
sudo apt install ros-noetic-explore-lite
sudo apt install ros-noetic-amcl
sudo apt install ros-noetic-map-server
sudo apt install ros-noetic-costmap-2d
```

The final external package needed is rm_mir_cleaner, that must be inside the *src* folder of the ROS workspace, can be obtained by:

```
git clone git@github.com:nunolau/rm_mir_cleaner.git
```

Then, just add this package[1] to the mentioned *src* folder and build the workspace using *catkin build*.

3 Usage

The developed package is composed by multiple launch files. The base launch file (which is requested for both the exploration and cleaning agents) is *bringup.launch*.

This launch file is responsible for launching two fundamental nodes: laserscan_multi_merger and move_base. laserscan_multi_merger is a node from ira_laser_tools that listens from both laser scanners of the MIR100 robot and publishes a combined laser scan message.

The move_base node deals with the navigation stack of the robot, namely, the local and global costmaps and planners.

Four arguments can be set:

[1] `https://github.com/joao-pm-santos96/rm_cleaner_76912`

rviz Set to *true* to launch a pre-configured Rviz instance. Default is *false*.

unpause Set to *true* to unpause Gazebo. Default is *false*.

map Load pre-defined maps. Set *0* for *mir_empty_world* with *rm_mir_cleaner* simple model, or *1* for *mir_maze_world* from *mir_gazebo*. Default is *-1* (no map).

gui used together with **map**. If *true*, Gazebo GUI is launched. Default is *false*.

The **map** argument will launch the described maps. Yet, other maps may be launched externally and used with this package. Is this is the case, keep this argument to the default value.

3.1 Exploring agent

To perform the exploration of a unknown environment, the *explore.launch* file is used. This launch file launches *gmapping* and *explore_lite*.

To start the exploring agent in the default map 1 and with Rviz, run (in separate terminals)

```
roslaunch rm_cleaner_76912 bringup.launch gui:=false map:=1
    unpause:=true rviz:=true
roslaunch rm_cleaner_76912 explore.launch
```

and then you should visualize something like Fig. 1.

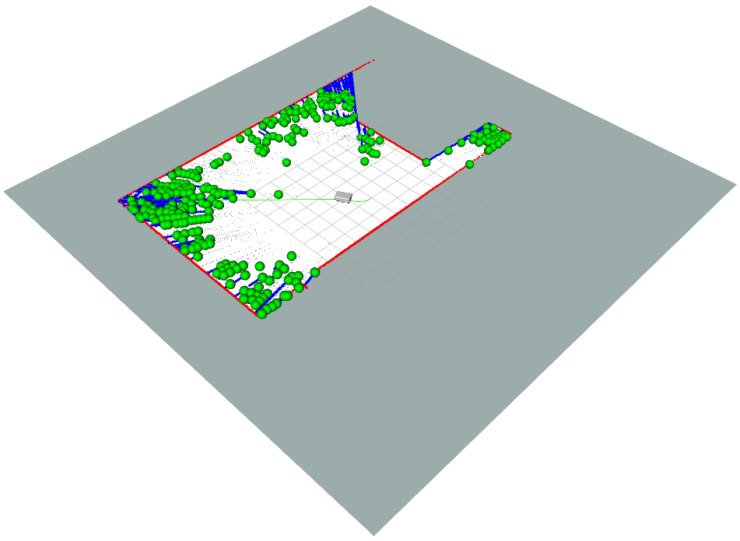

Figure 1: Exploration agent on default map 1.

In the end, when the exploration ends, you may save the map by running

```
roslaunch rm_cleaner_76912 map_save.launch map_name:=<map_name
    >
```

which will save the current map to the default folder (the *maps* folder on the package root) with the <map_name>.yaml name.

3.2 Cleaning agent

Now that the environment is known, the agent is able to vacuum it. For that, *clean.launch* must be started.

This file has two arguments

map_file the path to the map the agent has already created. It must be the full path and end with the file extension (.yaml).

clean_goal the ratio of the environment to be cleaned, in the range of 0.0 to 1.0.

and will launch four nodes: *amcl*, *map_server*, *costmap_2d* and *cleaner_bot.py*.

Continuing the previous example, run (in separate terminals)

```
roslaunch rm_cleaner_76912 bringup.launch gui:=false map:=1
    unpause:=true rviz:=true
roslaunch rm_cleaner_76912 clean.launch map_file:=/home/<
    user_name>/rm_cleaner_ws/src/rm_cleaner_76912/maps/<
    map_name>.yaml clean_goal:=<goal>
```

after which, something similar to Fig. 2 should appear.

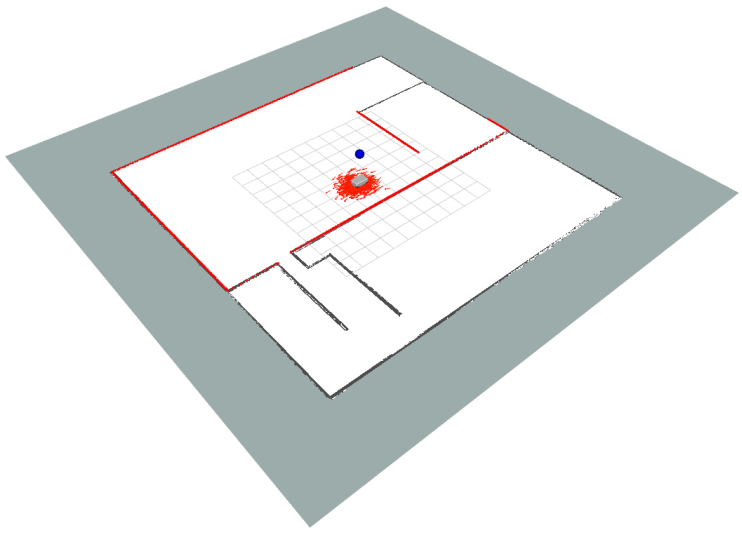

Figure 2: Cleaning agent on default map 1.

To start the cleaning behaviour, press the blue marker (see Fig. 2) on Rviz. This will instruct the robot to perform the computed path until it cleans at least the desired ratio or the path is concluded (in which case, it is assumed that all the accessible space has been cleaned).

Notice that, because *amcl* is being used for the cleaning agent, a start pose is required. By default, *amcl* expects the robot to be at the origin of the map, and within a 5×5 m^2 area. If this is not the case, please provide an approximated pose through the Rviz GUI or a published message.

4 Methodology

The adopted methodology consists in two distinct agents. One that will explore the environment, and another that will vacuum it.

Their implementations are discussed in the following sections.

4.1 Exploration and map building

The exploration agent is based is two well-known packages: *gmapping* and *explore_lite*. Combined, they are able to autonomously navigate through unknown spaces, maintaining a probabilistic localization within it and recording both the free and occupied spaces already encountered.

4.1.1 SLAM

The *gmapping* [1, 2] package provides laser-based Simultaneous Localization and Mapping (SLAM), enabling the creation of a 2D occupancy grid based on the laser data and pose of the robot.

This package implements a Rao-Blackwellized particle filter as an effective mean to solve the SLAM problem. This method makes use of a particle filter where each particle carries its own map of the surrounding area.

4.1.2 Exploration

In parallel, the *explore_lite* [3] package provides greedy frontier-based exploration, which will instruct (through the *move_base* node) the robot to greedily explore the environment until no frontiers can be found. This behaviour is illustrated on Fig. 3.

Frontiers are regions on the border between open space and unexplored space. The robot can view into unmapped space from any frontier and add the new discoveries to its map, which may include some new frontiers. This way, the agent can create a map of all locations that are able to be reached in the environment by exploring each frontier or determining that it is inaccessible.

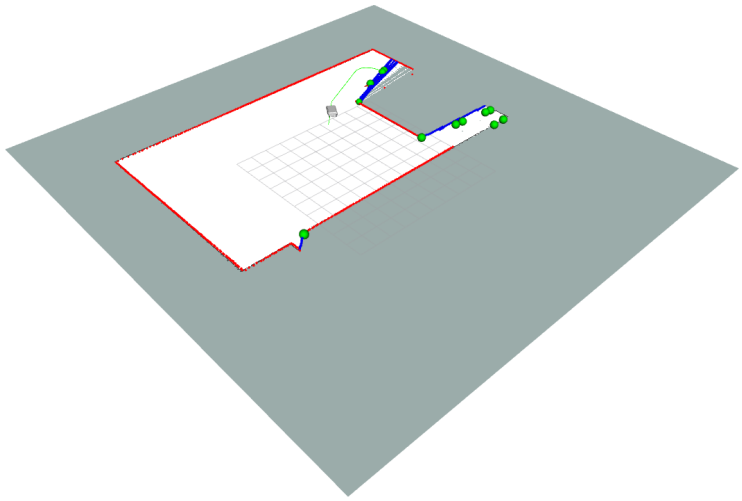

Figure 3: Exploration step. The robot will follow the path (green line) to try to see through the next frontier (blue lines).

4.2 Cleaning

The cleaning agent has, *a priori*, the knowledge of the map it is required to clean. This is an advantage, allowing the agent to plan the path to transverse. The challenge here is then to compute a full coverage path, i.e., a path that allows the clean bar (mounted in front of the MIR100) to be moved across all space.

4.2.1 Vacuum area

The agent is requested to clean (i.e. transverse) a given ratio of all free space in the known map. To compute this ratio, the agent must know the space that it is free to visit and the space that it has already vacuumed.

The information about the the free space is encoded in the provided map and, thus, trivial to obtain.

On the other hand, there are some challenges on tracking the cleaned area. For example, if the robot visits the same area multiple times, the ratio of cleaned space must not change. The implemented solution is based on an extra occupancy grid (not managed by the *move_base* node) that, at the start, marks all cells as unknown. The trick is that the cells in this occupancy grid (from now on, called cleaned area map) are marked as free if and only if they are run across by the cleaning bar. This behaviour is shown in Fig. 4.

With this information, the agent is able to compute both the free and cleaned areas and then, the ratio of the known area cleaned.

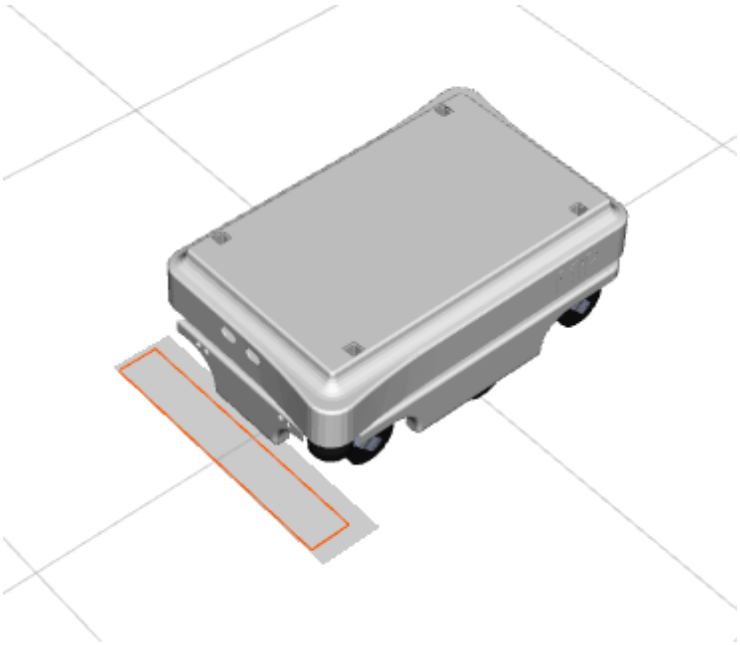

Figure 4: Detailed demonstration of the virtual clean bar (in red) and the area marked as cleaned (in gray).

4.2.2 Localization

In contrast to the exploration agent, SLAM is not required. The localization process is done through a probabilistic localization system, the *amcl* package. It employs the adaptive Monte Carlo localization technique, which compares the pose of the robot to a known map using a particle filter (see Fig. 5).

4.2.3 Path planning

As previously mentioned, knowing the map is a big advantage for the agent, allowing it to plan a path. The implemented strategy consists in a *zig-zag* motion. For this, the agent computes a set of poses that then are sent to the *move_base* node which, in turn, will move the robot.

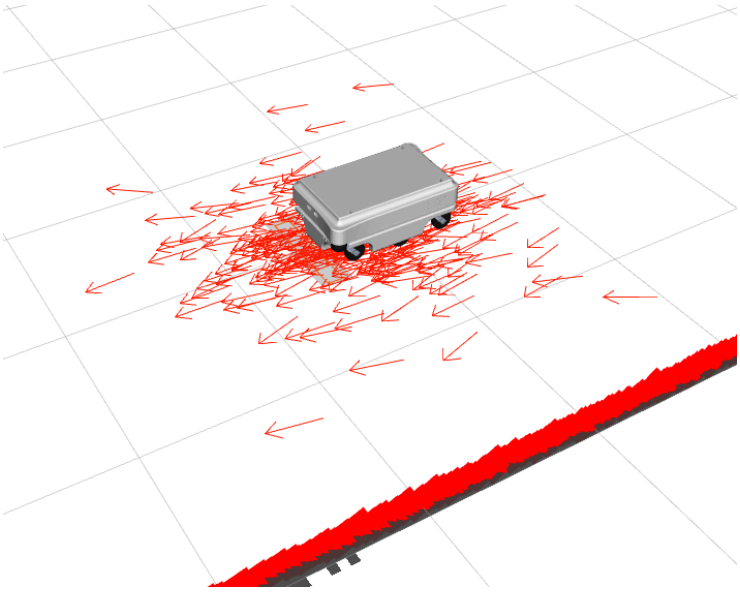

Figure 5: Set of possible initial poses computed by *amcl*.

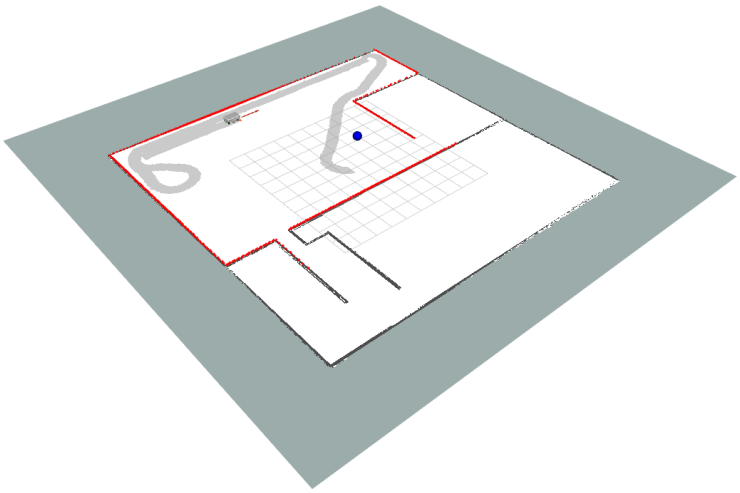

Figure 6: Example step of the cleaning path (already covered 7.56%).

5 Results

The main take away from this assignment was the successful implementation of an autonomous exploring and cleaning agents. In this section, the results regard the harder map scenario (argument **map**:=1).

In Fig. 7 the output map is shown. It is visible that the majority of the walls are completely reconstructed, apart from the upper left corner where some gaps exist. Yet, these gaps are not an issue to the current implementation given that they are severely smaller than the robot.

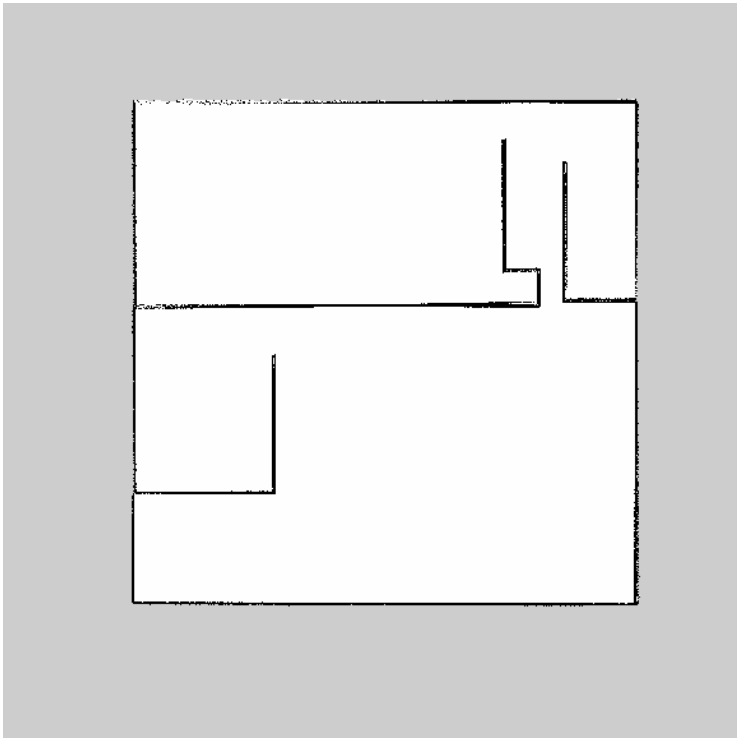

Figure 7: Output map autonomously created by the agent.

Using the previous map, the agent performs a cleaning pattern as shown on Fig. 8. In this scenario, 80% of all free (an potentially reachable) space was vacuumed.

A video of the behaviour is also available[2].

5.1 Future work

In a future development of this work, a better path planning methodology should be adopted. Namely, there are two aspects that can be greatly improved.

[2] `https://youtu.be/nQXJgW4vZOM`

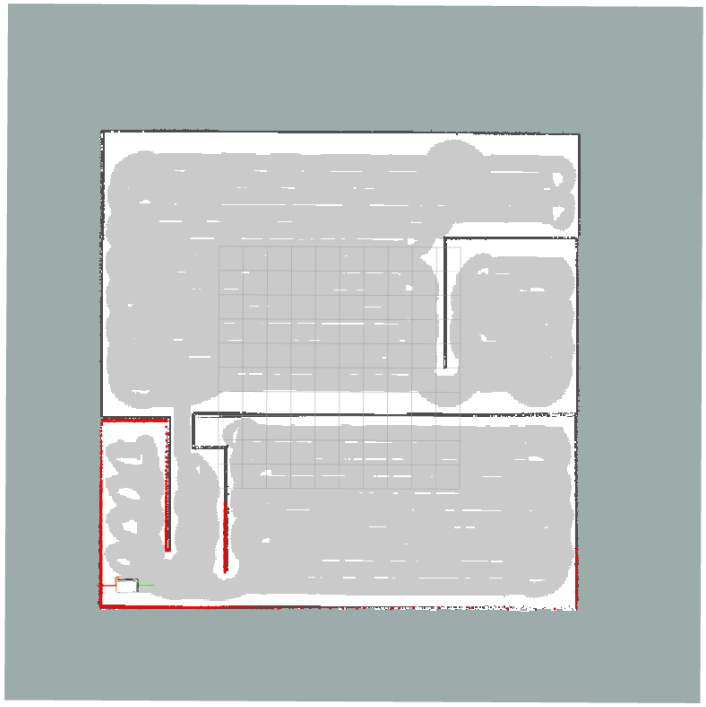

Figure 8: Area autonomously cleaned by the agent (80% of all free space).

First, in the implemented agent, the next goal pose is sent to the *move_base* node when the last one is reached. Yet, the last one is only considered as reached when the robot stops within a certain tolerance. The stop of the motion creates an uneven motion. A better approach would be to implement a set of ROS planners (a global and a local) as plugins. In fact, a full path global planner already exists[3] but none of the available local planners would follow it.

Secondly, the agent tries to move from one side of the map to another, in the *zig-zag* motion, regardless if there is a wall in the middle or not, is the global planner can produce a path to the pose, then it will move there. This may not be the most efficient solution for a vacuum robot. A better solution would be to perform a cell decomposition on the known map and only then, apply this motion within each cell. This would create a more efficient cleaning pattern and would avoid the motion of the robot across the same path, multiple times.

[3]see `http://wiki.ros.org/full_coverage_path_planner`

6 Conclusion

The implemented agents reach both proposed goals, to explore unknown environments and to coverage them using the map, autonomously. This is, by itself, a success. Nonetheless some improvements could be made to to both the motion of the agent and the cleaning behaviour.

References

[1] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling". In: *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE. 2005, pp. 2432–2437.

[2] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters". In: *IEEE transactions on Robotics* 23.1 (2007), pp. 34–46.

[3] Jiří Hörner. *Map-merging for multi-robot system*. Bachelor's thesis. Prague, 2016. URL: `https://is.cuni.cz/webapps/zzp/detail/174125/`.