MIR cleaner

Tiago Silva Pereira[98360]

Universidade de Aveiro, DETI, DEM

Abstract. Keywords: Mobile Robotics · Explore · Cleaning path.

1 Introduction

This project was made in the realm of the course Mobile Robotics. The objective of this project is to create two agents, that with the data that come from the sensors, have to map and clean the environment. We are going to use a simulation of the Mobile Inteligent Robots MiR 100, to complete this tasks.

This project can be foun the in the following link:
https://github.com/tiagospp55/RM-TP2

2 Environment

This project is based on ROS simulation and Gazebo. It includes a diferencial drive robot, that is controled by his linear and angular velocitys. In terms of sensors, we will have two sensors (the front and back sensor) but with a tool provided by the professor, we are going to merge them to one single sensor. We can see the environment in the Gazebo, but this environment is defines by walls that create entries in a way that we have physical obstacles.

3 Packages

In this project we used some packages, that are:

- iris_ua
- iris_lama_ros
- rm_mir_cleaner
- m-explorer
- path_coverage_ros

We are going to talk about the last two packages, because those are the ones that woren't provides by the professor.

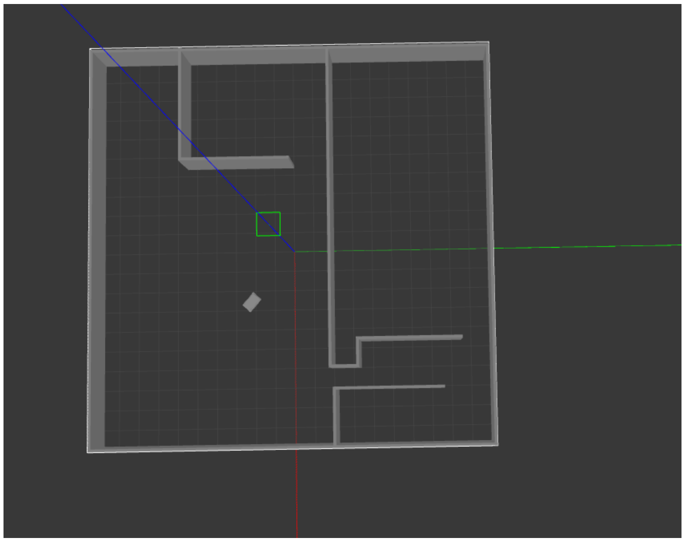


Fig. 1. Environment saw by Gazebo

3.1 m-explorer

This package was created by the Github User "hrnr" and originally was created to handle with problems that required multi robot exploration, but we are going to use the base of this program, to control a unique robot, using the explore package.

This package provide us a greedy frontier exploration model. When the main node is running, the robot is going to explore its environment until there is no frontiers to analyse. The movement command are send to move_base node. We then, use the package explore_lite that is a package made for exploration, that does note create a costmap, makint it easier and efficient to use than other methods.

3.2 path_coverage_ros

This package was created by the Github user Humpelstilzchen and was created for applications like cleaning and mowing agents, meaning that the agent need's to fully cover the environment area. The area to cover in our project is given by a Subscriver that reads the metadata presented in the map, that is given by the m-explorer package. The the original package has the disavantage that the area that the agent is going to cover is allways needed to be previously selected using the "Publish Point". So we changed the way that the algorithm received this points. The rest of the algorithm we maintained basically the same, dividing the selected area into cells using a algorithm that resembles the output of the Boustrophedon Cellular Decomposition, so that each cell is covered with simple back and forth movements.

4 Agent 1

In this section we are going to create a agent that can map and save any environment were the agent is present. For that we started by starting the main nodes and for developing facility, we concatenated all in the same launch file.

We are going to analyse the agent1 launch file, and explain line by line, what the code is doing.

```
<include
file="$(find mir_gazebo)/launch/mir_maze_world.launch">
</include>
<include
file="$(find rm_mir_cleaner)/launch/move_base.launch">
</include>
```

The first line is responsable for the launch of the Gazebo world and the launch of the robot. In this case this is the launch of the maze world, that is the more complex map that the professor provided to us. The second line launches the move_base that is the node responsale for the movement of the robot.

```
<include
file="$(find rm_mir_cleaner)/launch/laserscan_multi_merger.launch">
</include>
```

As it was asked, we need to merge the front and back laser scans of our robot, combinig them in two one scan. We had to change the laserscan_topics to the combination of the topic /b_scan and /f_scan and the new topic is called /scan_multi.

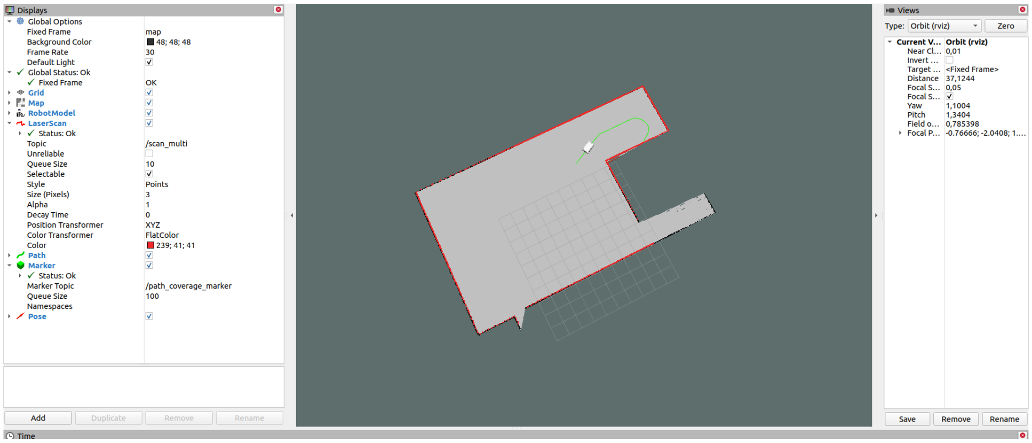
```
<node
pkg="rviz" type="rviz" name="rviz" args="–d $(find rm_mir_cleaner)/...
rviz/rviz.rviz"/>
```

This node is responsible for the launch of the rviz configuration. This was only made for facilitie of developning, so we don't need to launch each time that we run the rviz and select the topics that we are interested on.

```
<node pkg="gmapping" type="slam_gmapping" name="gmapping_node">
    <remap from="scan" to="scan_multi" />
    <param name="map_update_interval"  value="3.0"/>
    <param name="linearUpdate"  value="0.5"/>
</node>
```

This node is the definition of the algorighm that we are going to use to analyse the map. First of all, the chosen algorithm is the slam gmapping. THe data that comes to the scan topic is send automatically to the scan_multi. The maps is going to be updated each 3 seconds and he need to travel 0.5 units to update the map.

```
<include
file="$(find explore_lite)/launch/explore.launch">
</include>
```



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	☑
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	☑
Map	☑
RobotModel	☑
LaserScan	☑
✓ Status: Ok	
Topic	/scan_multi
Unreliable	□
Queue Size	10
Selectable	☑
Style	Points
Size (Pixels)	3
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	FlatColor
Color	🟥 239; 41; 41
Path	☑
Marker	☑
✓ Status: Ok	
Marker Topic	/path_coverage_marker
Queue Size	100
Namespaces	
Pose	☑

Type: Orbit (rviz)

Current V...	Orbit (rviz)
Near Cl...	0,01
Invert ...	□
Target ...	<Fixed Frame>
Distance	37,1244
Focal S...	0,05
Focal S...	☑
Yaw	1,1004
Pitch	1,3404
Field o...	0,785398
Focal P...	-0.76666; -2.0408; 1....

Fig. 2. Capturing of new data from the map

This line is going to launch the explore algorithm, that has we said before is going to control the move_base and navigate through the environment. This is tottally made by the m-explore library and it's connection with exploration_lite package.

```
<node
pkg="map_server" type="map_saver" name="map_saver_node" args="-f
$(find rm_mir_cleaner)/maps/mapa">
<remap from="/map" to="save_map"/>
</node>
```

Last of all, we are going to save the map to be later used for the agent 2. For this we created a node that is responsible for the publication of the map on a certain folder.

5 Agent 2

In this section we are going to talk about the agent 2 that needs to clean the entire maze, by covering the entire area of the maze, with the robot. As we said before we used a adapted library from Humpelstilzchen. The changes are made the with intuiton that we wanted to automatize the covering process. So, we are going to analyse the metadata given by the map made by the agent 1 and retrieve the corners of the map. This corners are then send to a function that is going to apply a algorithm that is going to create a "zigzag" path, having in consideration that the movement of the turning robot can make that this beats in the wall while rotating.

This part his working, because the agent cleans the full path, but it's not very efficient, because he doesn't folows the path, end to end between walls, but sometimes jumps corner to corner.

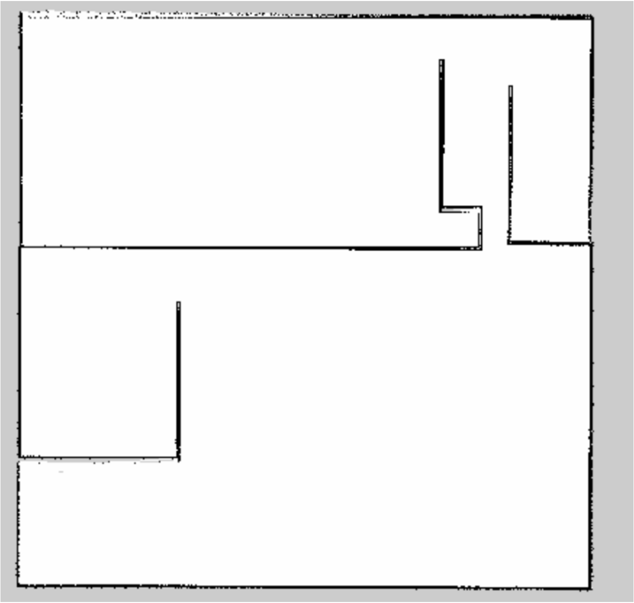


Fig. 3. Map captured by the agent

```
<include
 file="$(find mir_gazebo)/launch/mir_maze_world.launch">
</include>
<include
 file="$(find rm_mir_cleaner)/launch/move_base_clean.launch">
</include>
<include
 file="$(find mir_navigation)/launch/amcl.launch">
</include>
<include file="$(find rm_mir_cleaner)/launch/start_map.launch">
<arg name="map_file" value="$(find rm_mir_cleaner)/maps/mapa.yaml"/>
</include>
```

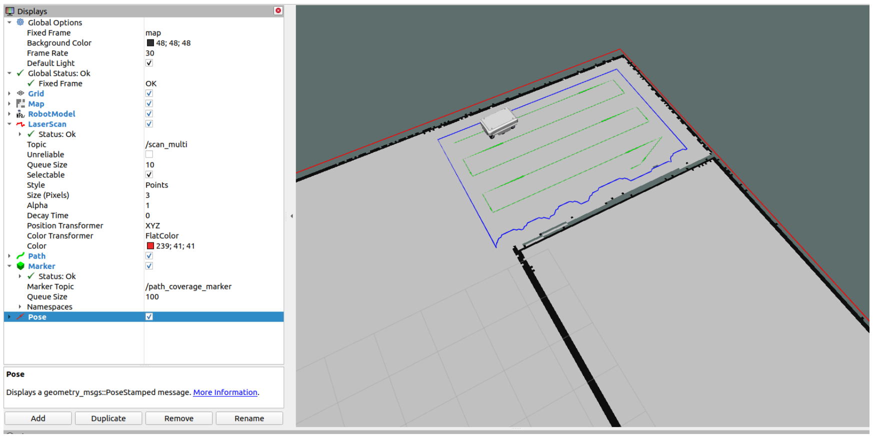
This four lines of code are responsable for the launch of the map and the robot in the gazebo environment. They also connect the node with the map from agent 1, that is going to be used has reference to the start of our agent.

```
 <node
 pkg="rviz" type="rviz" name="rviz" args="–d $(find rm_mir_cleaner)/...
 rviz/rviz.rviz"/>
```

Upload the rviz configuration, just like in the agent 1.

```
    <include file="$(find path_coverage)/launch/path_coverage.launch"> </include>
```

This is the launch of the responsable for the path coverage. This line launches the algorithm that creates the path needed for the cleaning task. This task needs of a previous defined map.



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	☑
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	☑
Map	☑
RobotModel	☑
LaserScan	☑
✓ Status: Ok	
Topic	/scan_multi
Unreliable	□
Queue Size	10
Selectable	☑
Style	Points
Size (Pixels)	3
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	FlatColor
Color	🟥 239; 41; 41
Path	☑
Marker	☑
✓ Status: Ok	
Marker Topic	/path_coverage_marker
Queue Size	100
Namespaces	
Pose	☑

Pose

Displays a geometry_msgs::PoseStamped message. More Information.

Fig. 4. Definition of the cleaning path followed in that corner

6 Conclusion

THis was a very interesting project because it improved our skills using the ROS framework and allowed to learn more about the planification of path and the scan of the environment. I think that the results are positive but it could a lot better, because this algorithm, although it's working, it's not that efficient and it's very slow.

References

1. https://github.com/hrnr/m-explore
2. https://gitlab.com/Humpelstilzchen/path_coverage_ros