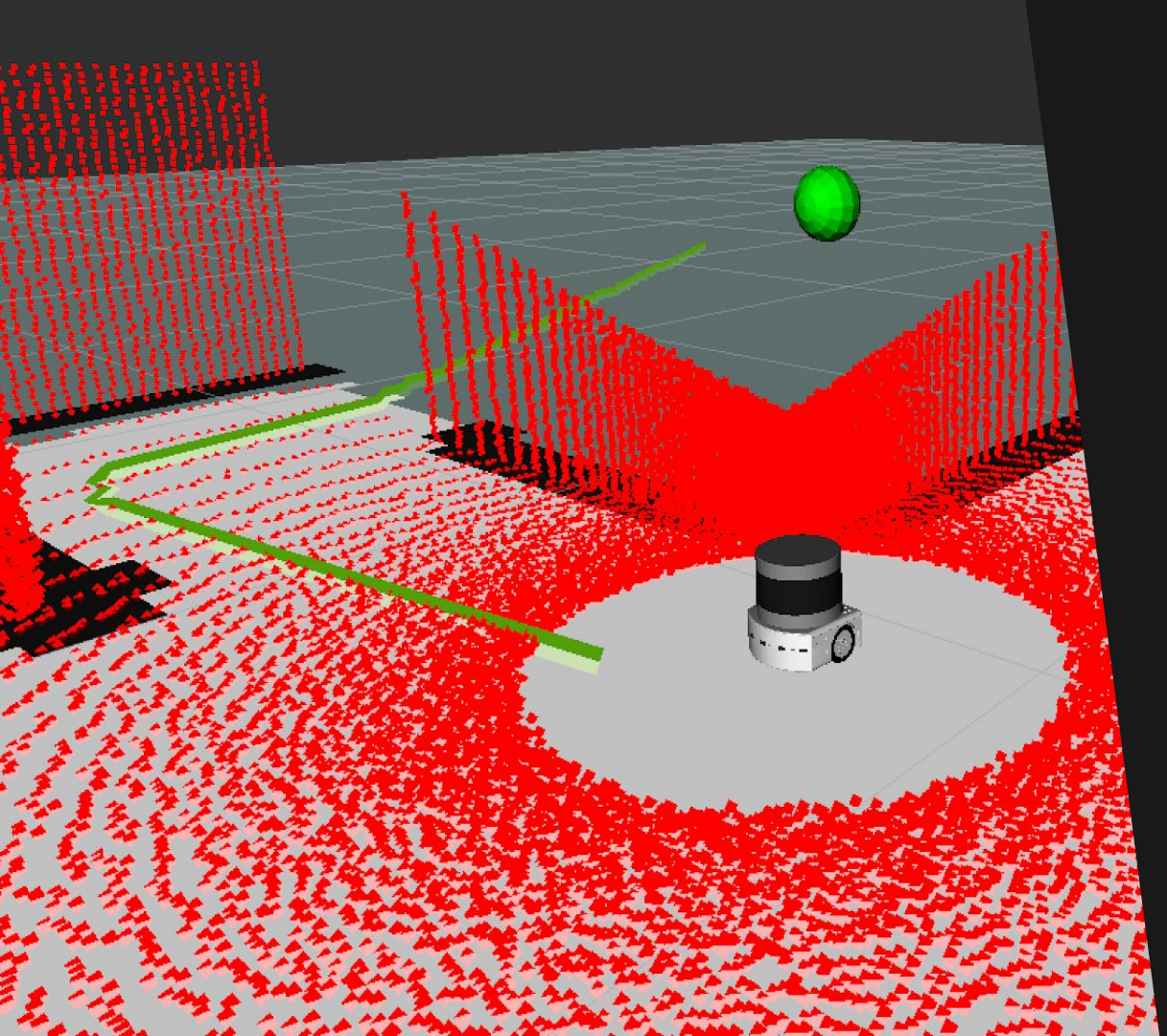


Robotics 2020

LIDAR Mapping + Path Planning

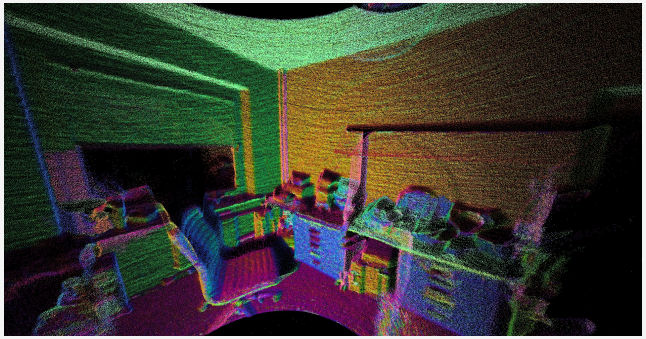
Marco Ferri
Simone Eandi
Nadia Younis

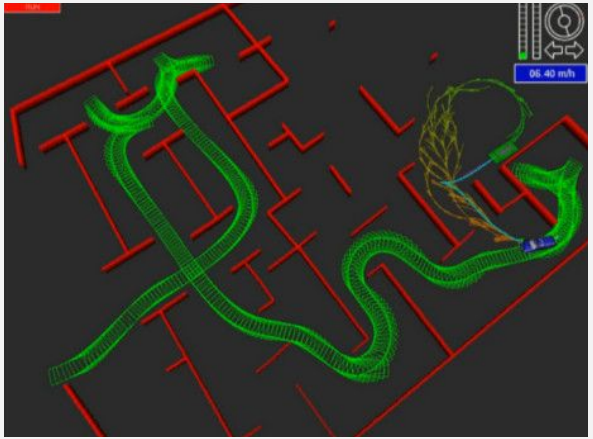
https://github.com/seandi/thymar

GOAL

Explore an unknown environment to search for a Target object.

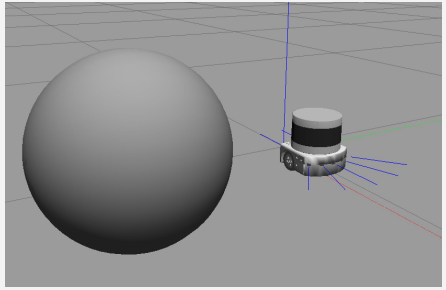
- Sense using a **LIDAR**
- Build the **Map** of the environment
- Identify a **Target**
- **Reach** the Target
- **Explore** unknown areas





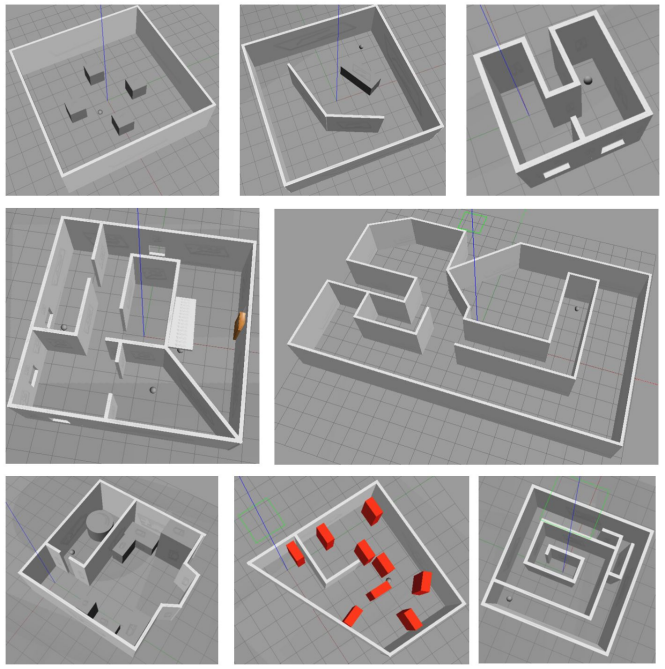
ASSUMPTIONS

- **Localization is given**
  We use the Mighty Thymio *ground truth* Odometry.
- The Target is **static**, its **shape is very simple** and already known.
- The environment is relatively simple with a **smooth floor** and easily **distinguishable obstacles** (which must be clearly different from the Target).



Robot and Target (30cm sphere)

Available Worlds



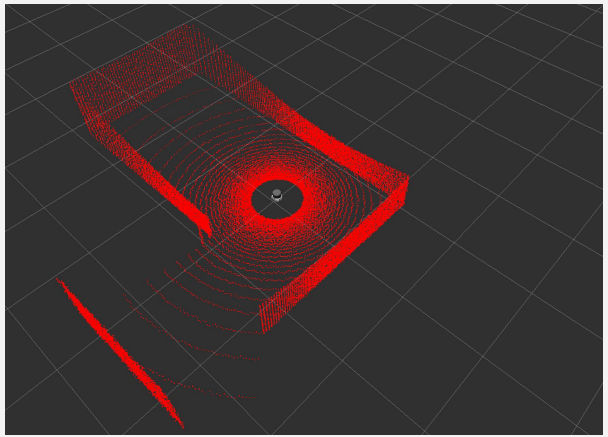
01

LIDAR MAPPING

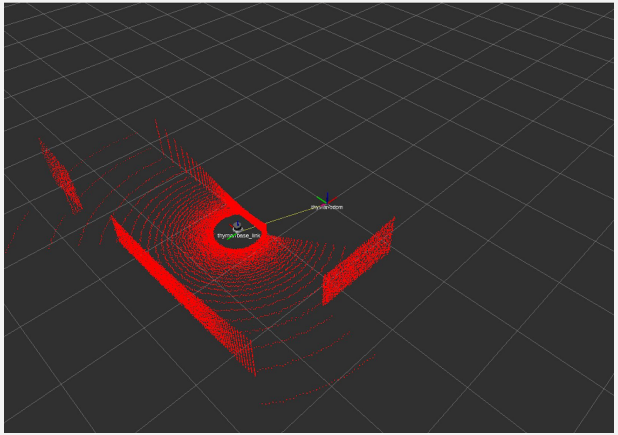
LIDAR VIEW

LIDAR readings change over time.

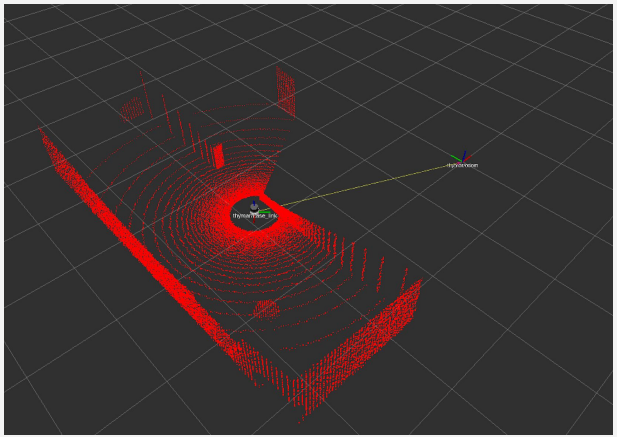
For handling them, we **keep track of robot's position** with respect to the initial pose.



step 0



step 30

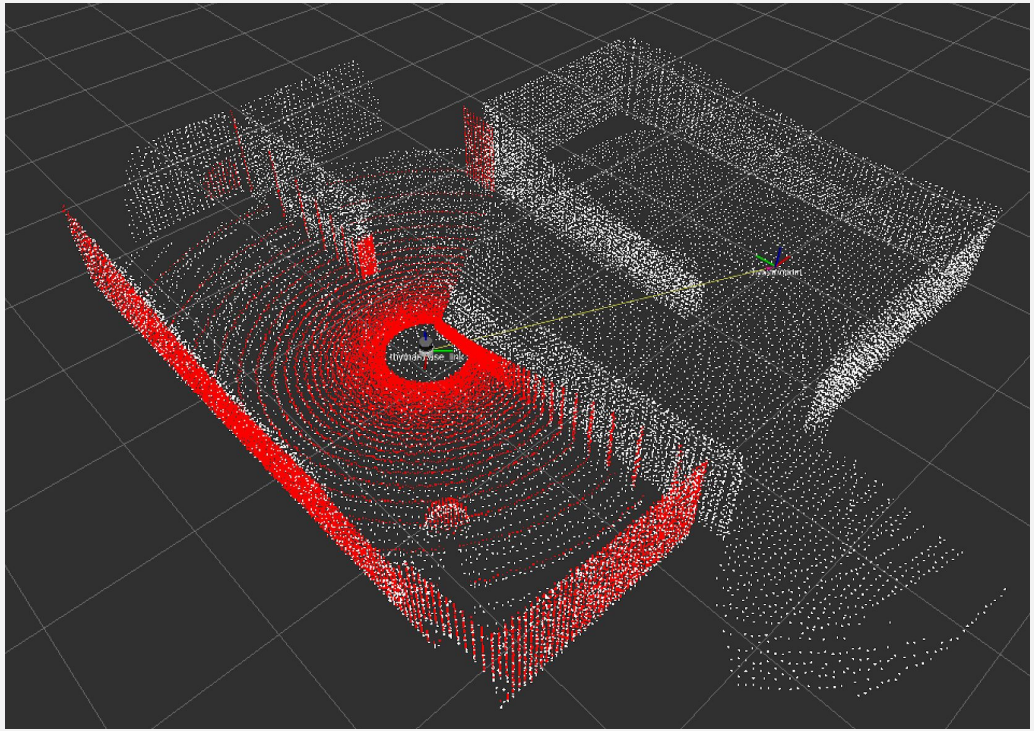


step 100

3D MAP

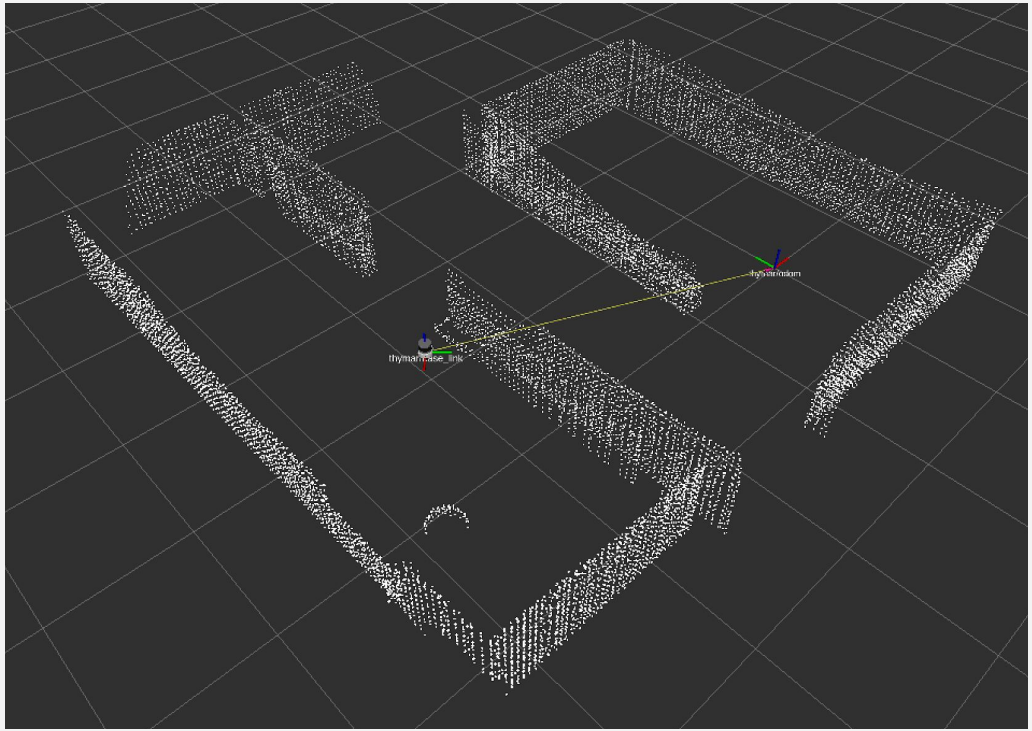
Merging LIDAR point clouds over time, we can build the 3D map of the environment.

This operation is done by applying some noise reduction.



OBSTACLES

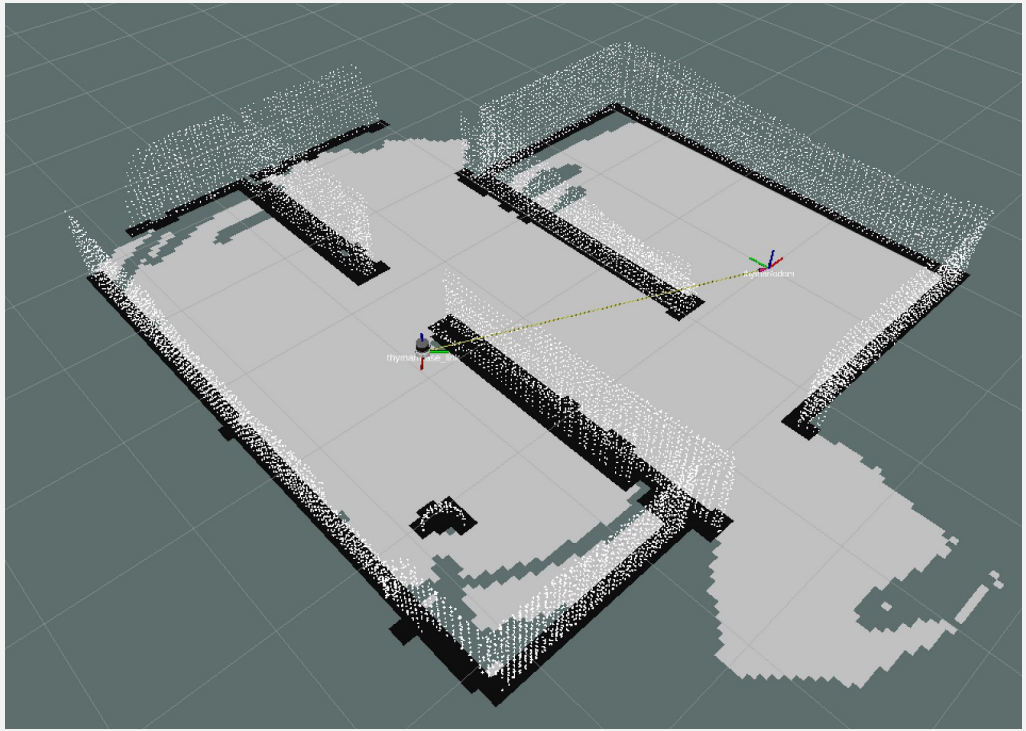
Filtering on 3D map **heights**, obstacles can be separated from the terrain.



2D MAP

Obstacles are used for creating the 2D map of the environment.

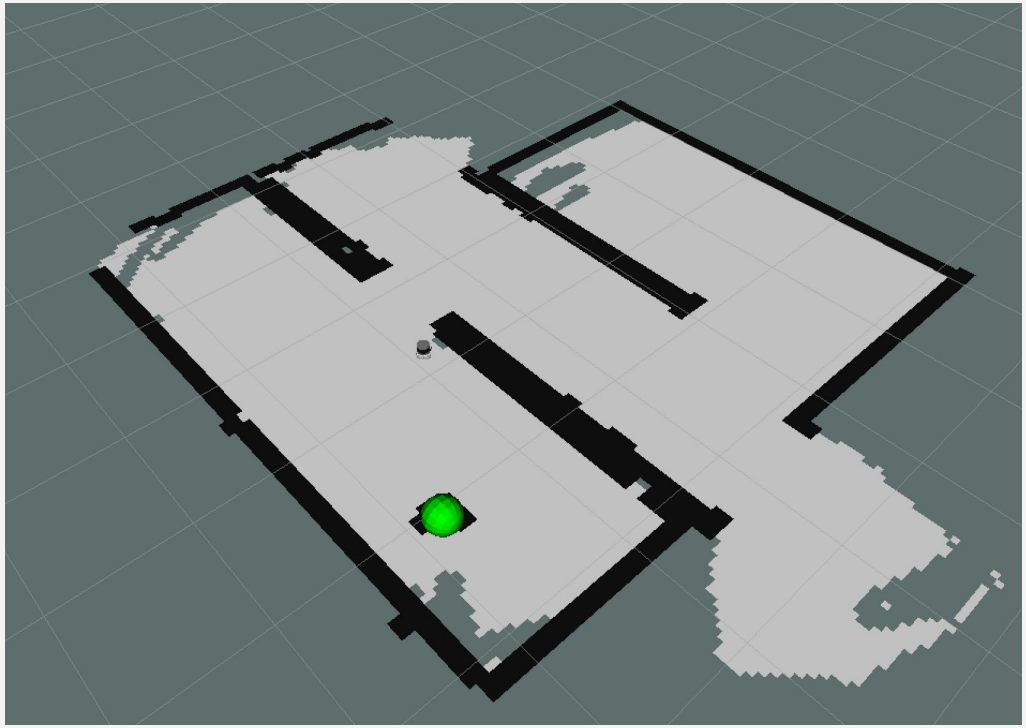
The map **discretize the available space** with a resolution of 5cm and keeps track of **obstacles**, **known** and **unknown** areas.



TARGET

Point clouds are also used for **fitting the model of a sphere** (15 cm radius) in the available space.

If the Target is found, a **marker** is published for better visualization.

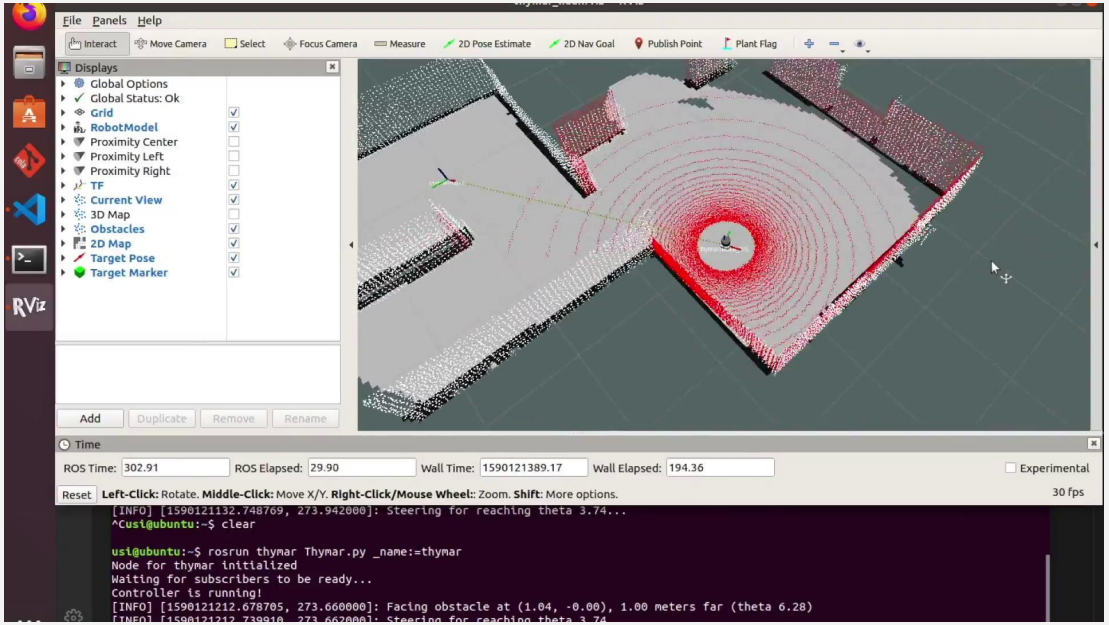


02

TARGET REACHING

RANDOM EXPLORATION

Current robot's pose in the 2D map can be used for reacting when **facing an obstacle** in order to compute the **best way to turn around** in search of a free path.

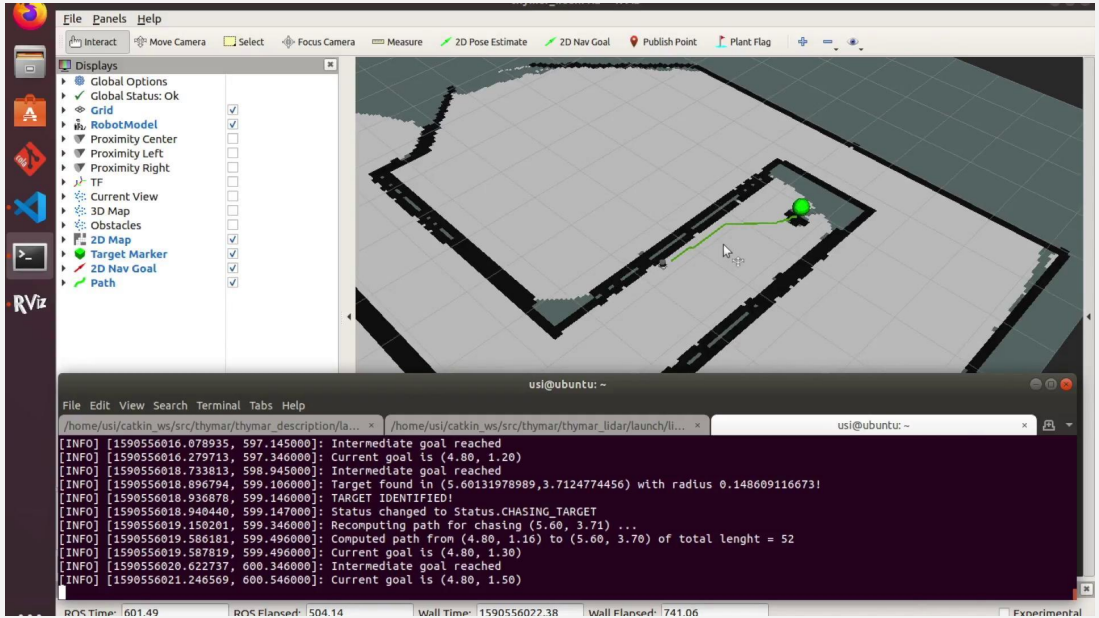


The robot keeps going until no obstacle is 1 meter directly in front of it. If so, the robot decide to turn left or right by evaluating 45 degrees straight in the both directions, so it starts to try different orientations until a good one is found.

NOTE: obstacles point cloud is not updated while steering

PATH PLANNING with A*

After the Target has been found, a Path Planning algorithm is used to lead the robot towards it. The Path is followed pose-by-pose by the robot.



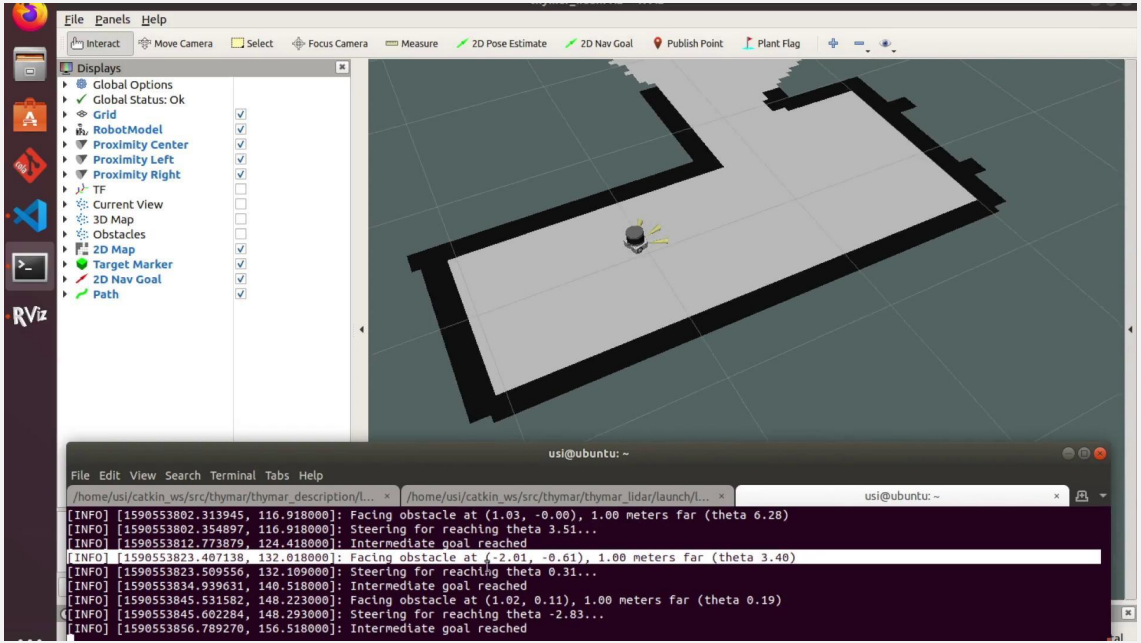
Exploration of the environment stops and switch to Path Tracking as soon as the Target is identified.

03

IMPROVING EXPLORATION

RANDOM IS NOT ALWAYS GOOD

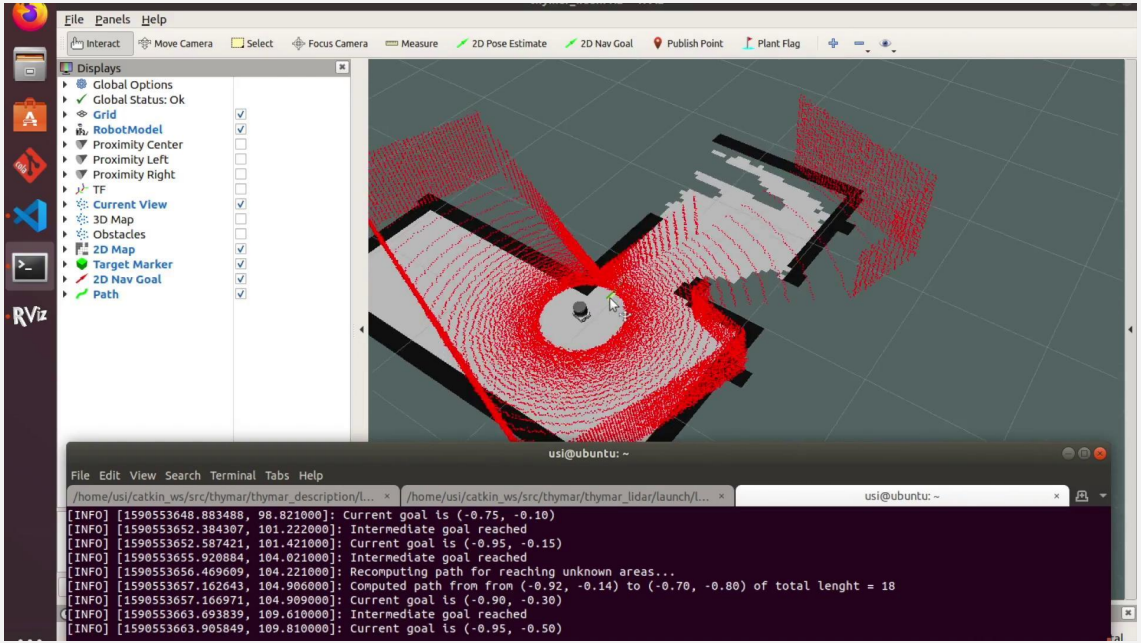
- Not efficient
- Can generate loops



The robot gets stuck taking the same decisions over and over. Remember that this exploration is not completely “random”, but instead based on some collision avoidance policies.

PATH PLANNING with DIJKSTRA

To optimize the exploration and also the target searching, the robot can be set up to **explore areas still not known**. For doing so, Dijkstra is used to plan a Path towards the **nearest unknown cell** in the map.

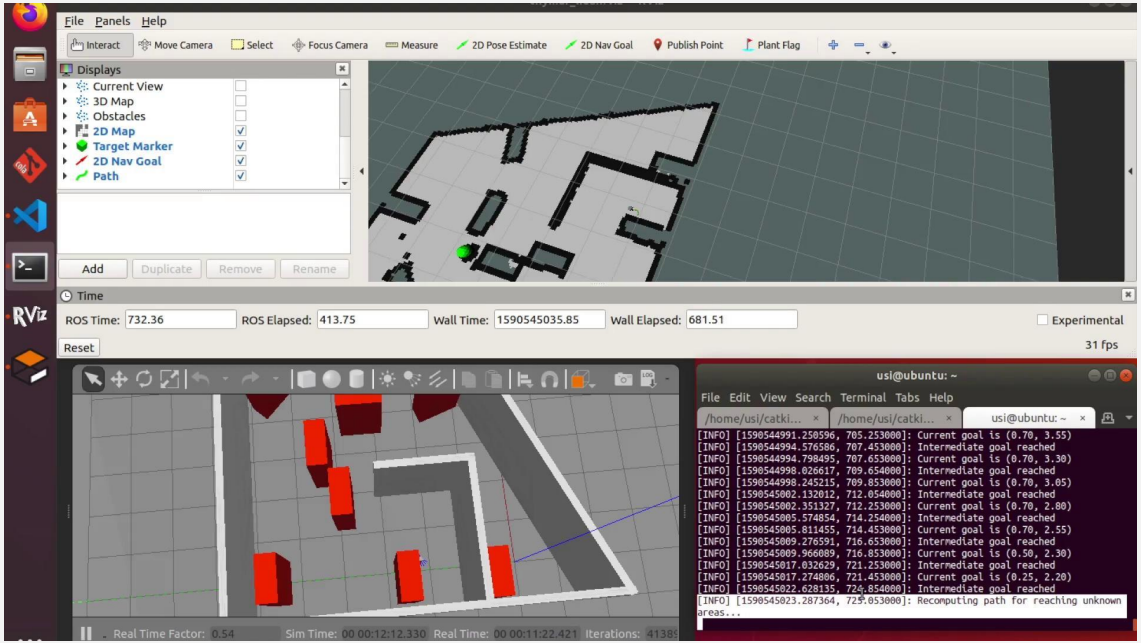


Now, the robot immediately finds the way to the Target.

NOTE: point clouds readings are updates once per second for improving simulation performances

MAP COVERAGE

The previous technique can be used for exploring all the available environment **until no unknown areas are reachable**: this is a map coverage task.



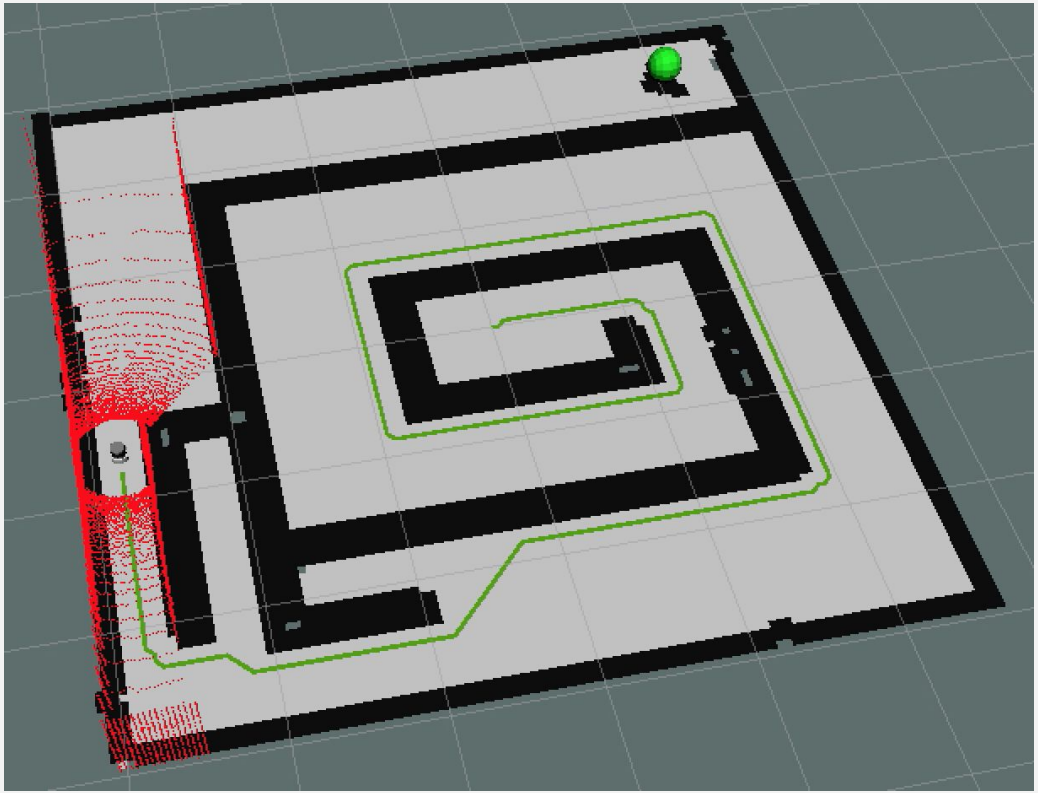
Objective of the robot in the video is to explore all the environment, actually ignoring the Target if found.

04

EXPERIMENTS

OBSTACLE SAFE DISTANCE

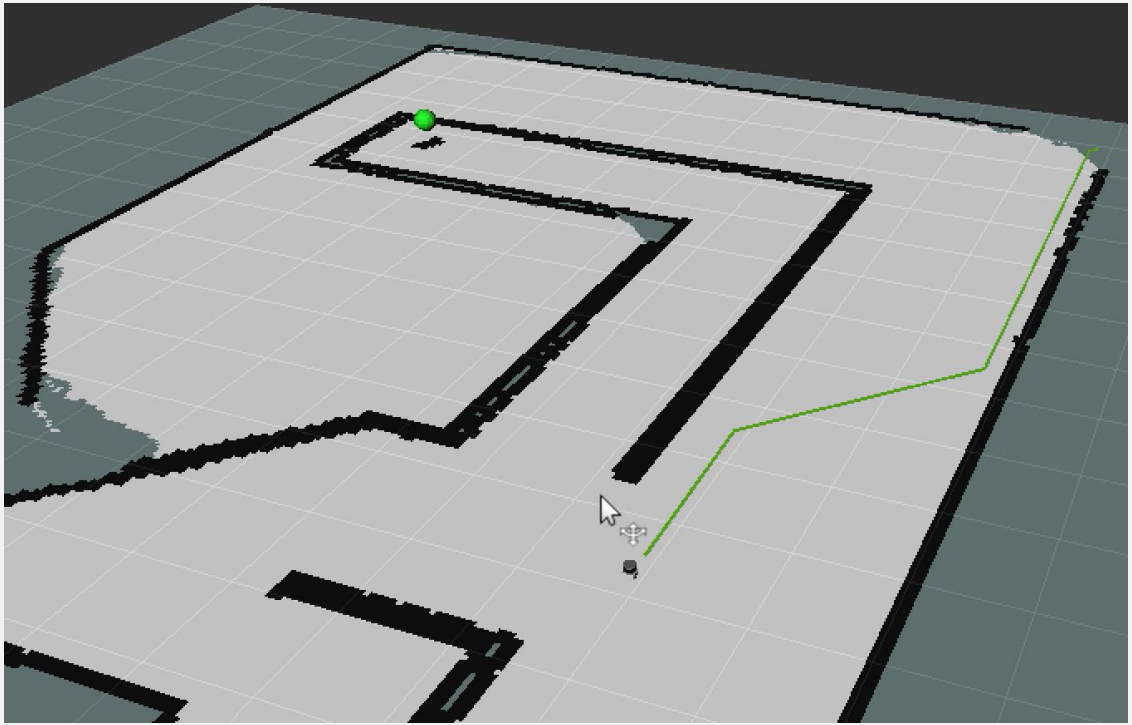
Computing traversable Paths implies to consider obstacles larger than they are. This “expansion amount” allows the robot to properly avoid obstacles while still passing through narrow passages.



The robot passes through a narrow passage while going back to its initial position, after having found the Target.

PATH UPDATE

Paths are updated periodically to find better ways of exploring. However, **following very long paths** means that most of the area has been already explored, so the **recomputation frequency can be lowered** very much.



The robot follows a long path in areas mostly already explored, which does not require any updating.

PLANNING THROUGH UNKNOWN AREAS

Our A* algorithm also allows to take into consideration the possibility of traversing unknown areas for reaching a given pose.

In this case, Path updates are very frequent so that the robot can **recompute trajectories as soon as an obstacle is found**. The video shows the robot reaching the given pose avoiding obstacles while discovering the map and finally coming back to the initial position.

FUTURE WORKS

- Improve Target recognition
- Work with non-static Target
- Implement Localization

THANK YOU FOR THE ATTENTION

Marco Ferri
Simone Eandi
Nadia Younis

Source code:
https://github.com/seandi/thymar

Other videos:
https://drive.google.com/drive/folders/1wQm72YXUPujmo2mkTtnnDbkYrLpn9Erk?usp=sharing

References:
ROS Robot Operating System
Gazebo Simulation Environment
RViz 3D visualization for ROS
Point Cloud C++ Library

Index