ROBT502 – ROBOT PERCEPTION AND VISION

Laboratory Report 5

Z. Kuangaliyev, A. Amangeldi, K. Kantoreyeva

March, 2024

1. Introduction

This laboratory work involves using TurtleBot3 robot within the Gazebo simulation environment to understand and implement robot navigation and mapping. Our tasks were to setup the simulation environment, navigate robot through labyrinth and tune the navigation parameters to observe the performance of control algorithm.

2. Setup and Preparation

The initial step required creating a workspace for the simulation files. This involved navigating to the '**catkin_ws/src**' directory in the terminal and cloning the '**turtlebot3_simulations**' package from its Git repository After we have compiled workspace, we were ready to run our simulation.

3. Simulation and Navigation

A dedicated folder was created within the home directory to store the map files. Running '**ROSmap.py**' script we had generated a map, where TutrtleBot3 would navigate. Now we had to configure '**.yaml**' extension file, such that:

- The resolution is set to '**0.050000**' which makes each pixel in the map image represents a 5 cm x 5 cm square in the real world.
- Origin is shifted by **[-1, -1, 0]** vector.
- Negate parameter is set to '**false**' which means that white areas of the map are free and black areas are occupied (obstacles).
- Occupied threshold parameter is set to '**0.65**' making any pixel with a value above 65% towards being fully black is considered an obstacle.
- Free threshold parameter is set to '**0.196**' which suggests that any pixel with a value below 19.6% towards black is considered space may be used for navigation.

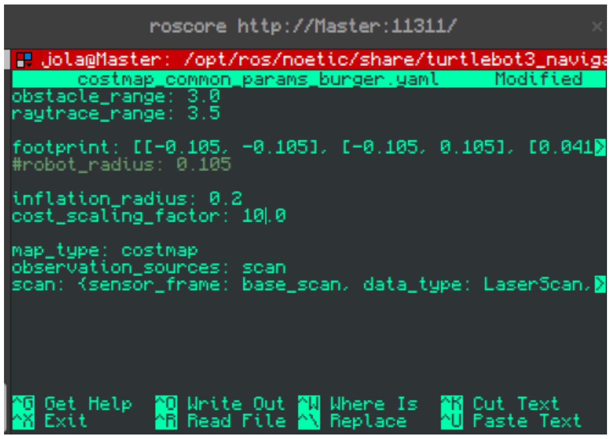
With the map created and configured, the next step involved launching the simulation environment and initiating the navigation process.

The Gazebo simulation environment was launched with the TurtleBot3 in a predefined maze environment using the '**roslaunch turtlebot3_gazebo labyrinth.launch**' command. To start the navigation we had to run '**roslaunch turtlebot3_navigation turtlebot3_navigation.launch**

map_file:=$HOME/(FULLPATHTOMAP)/map.yaml' command which links the TurtleBot3 navigation stack with the previously created and configured map. Now we had to set the robots' pose in the anywhere in the labyrinth and define the goal pose it must reach (see Video 1).

4. Tuning Navigation

Throughout the navigation, careful attention was paid to the responsiveness of the TurtleBot3 to the navigation commands and how well it adhered to the path dictated by the map. We had noted discrepancies between expected and actual navigation paths, and made adjustments as necessary to improve performance. The tuned parameters are shown in the figures below (Fig 1. and Fig. 2).



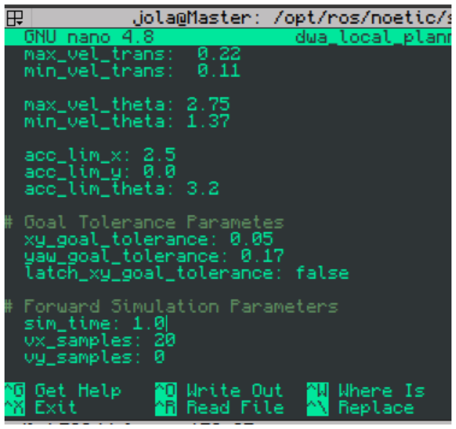
```
roscore http://Master:11311/
jola@Master: /opt/ros/noetic/share/turtlebot3_naviga
      costmap_common_params_burger.yaml       Modified
obstacle_range: 3.0
raytrace_range: 3.5

footprint: [[-0.105, -0.105], [-0.105, 0.105], [0.041>
#robot_radius: 0.105

inflation_radius: 0.2
cost_scaling_factor: 10.0

map_type: costmap
observation_sources: scan
scan: {sensor_frame: base_scan, data_type: LaserScan,>
```

Fig. 1. Navigation parameters



```
jola@Master: /opt/ros/noetic/s
  GNU nano 4.8                 dwa_local_plann
  max_vel_trans:  0.22
  min_vel_trans:  0.11

  max_vel_theta: 2.75
  min_vel_theta: 1.37

  acc_lim_x: 2.5
  acc_lim_y: 0.0
  acc_lim_theta: 3.2

# Goal Tolerance Parametes
  xy_goal_tolerance: 0.05
  yaw_goal_tolerance: 0.17
  latch_xy_goal_tolerance: false

# Forward Simulation Parameters
  sim_time: 1.0
  vx_samples: 20
  vy_samples: 0
```

Fig. 2. Navigation parameters

After setting everything up, we tested the navigation by placing the robot at the beginning of the labyrinth and setting the goal for it to navigate through to the end (see Video 2).

5. Comparing Default and Tuned Navigation

Comparing the two navigation stacks we have figured out the following:

- The default settings resulted in a somewhat indirect path with unnecessary turns and circles. While effective at avoiding collisions, the robot occasionally took overly cautious routes, leading to increased completion time.
- The tuned parameters resulted in a more direct and efficient path through the maze. The robot was able to make less unnecessary turns and optimize its route more effectively. Moreover, tuning allowed the robot to navigate closer to obstacles without compromising safety, thereby shortening the path length.

6. Videos

Video 1: link1

Video 2: link2