SWROB2
Exercise 5 - Point cloud operations
Mandatory

Submitted by:

Study nr	Name	Study line
202005180	Nicolaj Meldgaard Pedersen	E
202105443	Johannes Baagøe	E
201270449	Anders Sandø Østergaard	EP
201905293	Daniel F. Borch Olsen	E

Århus Universitet

April 3, 2024

Contents

1 Exercise: Extract range and angle from scan

1.1 Introduction

This section underscores the significance of extracting range and angle information from scanning data, a critical component in localization methodologies. Localization is foundational in robotics, enabling autonomous navigation and interaction with the environment. The TurtleBot platform, utilized both in educational settings and research, serves as a practical example for applying these concepts. This document delineates the approach for two distinct scenarios:

- Implementation on a physical TurtleBot robot.
- Application within a simulated environment.

1.2 Objective

The primary goal of this endeavor is to conceive and implement an algorithm that proficiently extracts range-angle coordinates from the scanning data acquired by the TurtleBot. The exercise entails the acquisition of range data, its subsequent processing to ascertain the distance and angle relative to a wall, and the corroboration of these computational findings with empirically measured values. The success of this algorithm is pivotal for the robot to achieve an accurate understanding of its spatial orientation, which is quintessential for effective navigation and task execution.

1.3 Methodology

This segment elucidates the systematic procedure adopted for data acquisition and subsequent processing.

1.3.1 Data Acquisition

The acquisition of range data constitutes the first step in our methodology. Utilizing the ROS framework, the TurtleBot's onboard LaserScan sensor gathers two-dimensional scan data, which are encapsulated in the form of a Laserscan (2D) message. The sensor's angular resolution and range precision are instrumental in determining the fidelity of the data captured.

For calibration and validation purposes, the TurtleBot is meticulously positioned at a predetermined distance and angle with respect to a well-defined wall. This setup ensures that the range data collected are grounded in a known reference frame, which is essential for the subsequent stages of data analysis.

Note: It would be beneficial to include a diagram here that visually represents the TurtleBot's positioning relative to the wall, showing the angle of incidence and the specific region of the wall being scanned. This would help clarify the setup and expected data collection geometry.

1.3.2 Sensor Calibration

Prior to the initiation of data acquisition, calibrating the sensor is paramount to ensure the precision of the measurements. Calibration encompasses the fine-tuning of sensor parameters to rectify any systematic errors, thus harmonizing the sensor's output with the established reference distance. This procedure is iterated until the deviation of the sensor's readings from the expected values is minimized.

Error Differentiation In the discourse on errors, it is essential to differentiate between systematic and random errors. Systematic errors are consistent and directional biases that can be corrected through calibration. In contrast, random errors manifest as unpredictable fluctuations that calibration cannot eliminate. The random errors are inherent in any measurement and can be mitigated through statistical methods such as averaging over multiple observations.

1.3.3 Algorithm Development

Detail the development of the algorithm for extracting range-angle coordinates. Discuss the implementation of line fitting or similar techniques to enhance robustness against irregularities on or near the wall.

1.4 Experimentation and Testing

1.4.1 Algorithm Testing

Explain how the algorithm is tested, including the procedure for driving the robot along a wall at a fixed distance and adjusting its driving angle to maintain this distance.

1.4.2 Optional Enhancements

Discuss optional methods for improving algorithm robustness, such as the implementation of k-means clustering or the Hough transform, to focus on fitting lines or planes to the most significant wall area while ignoring corners and other non-relevant features.

1.5 Results

Present the results of the algorithm testing, including comparisons between the extracted data and true measured values. Include any relevant data visualizations or statistical analyses.

1.6 Discussion

Analyze the performance of the developed algorithm, highlighting its strengths and limitations. Discuss any discrepancies between the extracted data and true values, and suggest possible explanations and improvements.

1.7 Conclusion

Summarize the findings of the exercise, emphasizing the importance of accurate range and angle data extraction in localization methods. Reflect on the potential applications of this work in robotics and future research directions.

1.8 Simulation of Turtlebot

In this section the konfiguration for the simulated Turtlebot will be shown.

1.8.1 Simulation environment

Figure 1 is showing Gazebo and Turtlebot started up from the terminal by using following command `roslaunch turtlebot3_gazebo turtlebot3_house.launch` The Turtle-

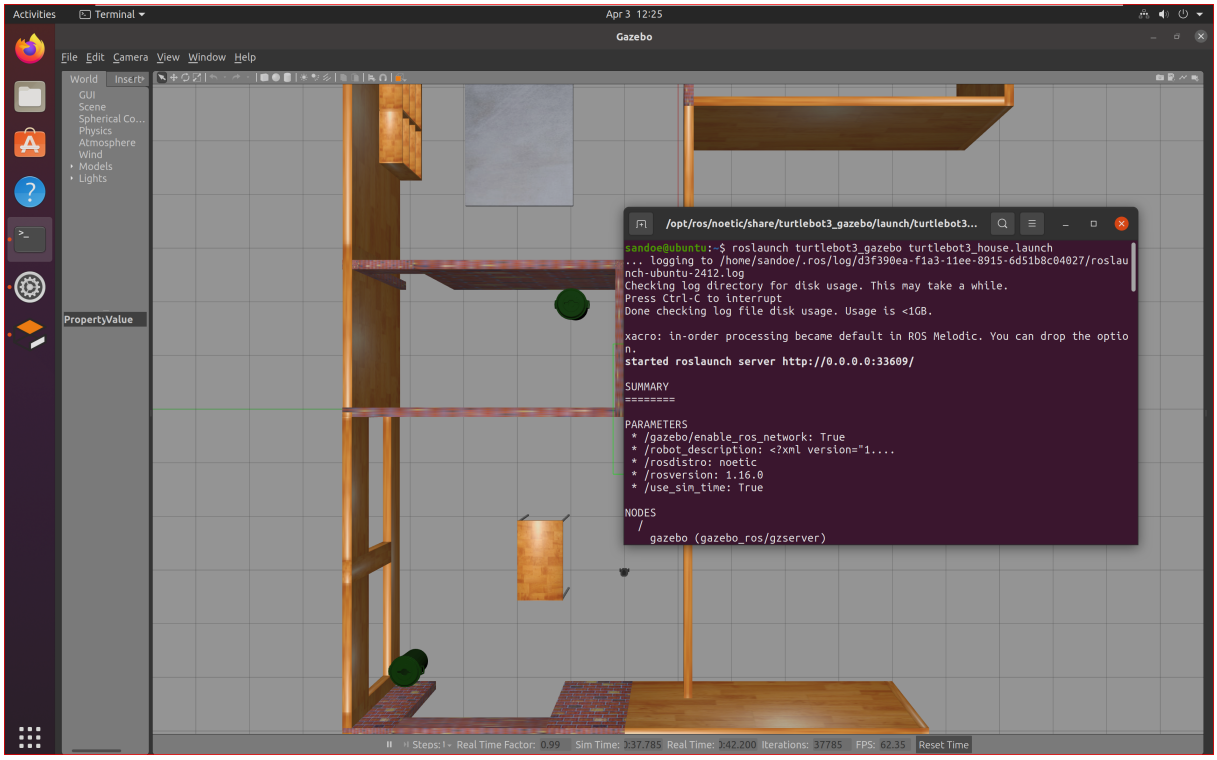

Figure 1: Simulation environment in Ubuntu (Fosscal), Gazebo V11 and Turtlebot3

bot model is missing it's camera and therefore we need to terminate the session and navigate to following folder `catkin_ws` for then be using the command `source devel/setup.bash`. We relaunch the simulation by following command:
`roslaunch turtlebot3_gazeboturtlebot3_house.launch`.

To verify that the camera and Lidar sensor is working then we are going to be running `rostopic list` followed up by `rostopic echo /scan` (Lidar) and `rostopic echo /camera/image_raw`. By looking at the data stream then we can verify that the configuration is done and our environment running.

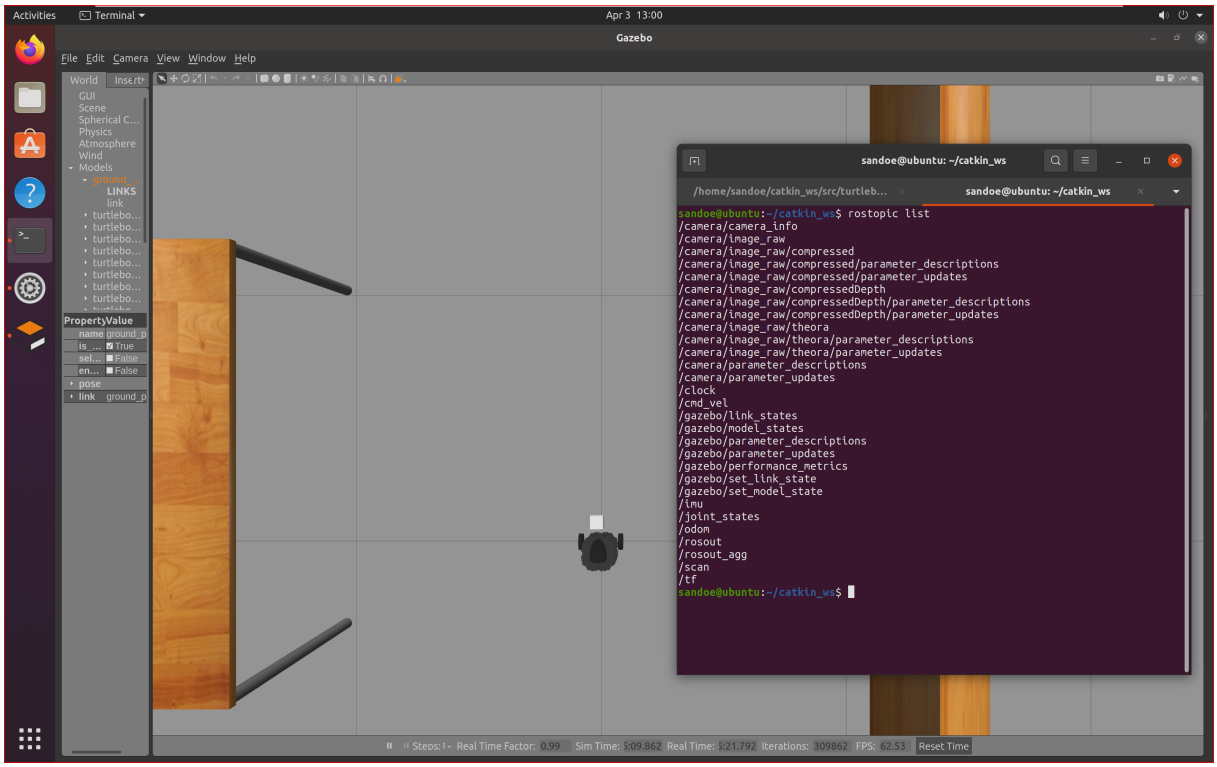

Figure 2: Showing rostopic list

1.8.2 Simulation of Turtlebot behavior

1.8.3 Turtlebot and PC konfiguration

1.9 Physical Turtlebot

1.9.1 Turtlebot and PC konfiguration

This section is for the physical Turtlebot and will describe how this task is going to be solved.

1.9.2 Connection to the TurtleBot3 from powershell

For making the connection to the turtlebot we are connecting to the WiFi

- **ssid:** turtlebot
- **password:** turtlebot3

from powershell type:

ssh ubuntu@192.168.72.251, password: turtlebot

1.9.3 Starting ROS on turtlebot from powershell

from powershell type

roscore

1.9.4 Connection to TurtleBot3 from Matlab

For setting the ros environment variable and setting the IP on the host (turtlebot):

```
setenv('ROS_MASTER_URI','http://192.168.72.251:11311')
```

For setting the IP on the local machine

```
setenv('ROS_IP','192.168.72.220')
```

The following command is closing existing connection to be ensure that when the user is connection the robot isn't connected to anyone else

```
rosshutdown();
```

This command will be doing the initialization of the connection between ROS and Matlab

```
rosinit('http://192.168.72.251:11311','NodeHost','
192.168.72.220');
```

Matlab script for init

```
setenv('ROS_MASTER_URI','http://192.168.72.251:11311')
setenv('ROS_IP','192.168.72.220')
rosshutdown();
rosinit('http://192.168.72.251:11311','NodeHost','
192.168.72.220');
```