MAKERERE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

ROBOTICS PROJECT FINAL REPORT

(MOBILE LOCALIZATION WITH OBSTACLE AVOIDANCE ROBOT)

By

Wasswa Goli Mark 2100707579 21/U/07579/EVE

***Abstract*—This document is a report which discusses my robotics final project of developing a mobile localization with obstacle avoidance robot. It is divided into three main chapters, Chapter I discussing: Datasets, Sensors, Sensing mechanism of this robot, Chapter II discusses: Actuators, Control and the Actuation mechanism of this robot and Chapter III discusses: the simulation of this robot.**

CHAPTER I

In recent years, there has been a growing demand for autonomous robots that can navigate complex environments while performing tasks such as localization and obstacle avoidance. This chapter describes **Datasets, Sensors, Sensing mechanism** and the development and evaluation of a mobile obstacle avoidance localization robot designed to navigate the environment autonomously, pinpoint its position, and avoid obstacles in real time.

Datasets, Sensors, Sensing mechanism

Datasets.

For this robot, a dataset containing information about the robot's environment, it’s internal state information and how the robot perceives and navigates in that environment is required for the proper navigation of this robot. A brief description of what the dataset contains is as follows:

Sensor data:

LiDar data:

The robot uses special sensors that emit laser beams to detect objects and measure distances. This helps the robot understand its surroundings and find obstacles.

IMU data:

Robots have sensors that detect movements such as acceleration, rotation, and direction changes. This information helps the robot find its way and avoid obstacles.

Ground truth labels:

Localization:

We need to know exactly where the robot is in the environment. This can be done using location coordinates (such as maps) or other location descriptions.

Environmental information:

Card/Map data:

A map of the environment helps the robot understand where it is and where obstacles are. Maps can display walls, doors, landmarks, and other important features.

Sensors

These refer instruments used to collect the dataset described in the first section.

In addition to the map, the robot is equipped with sensors that allow it to navigate through its environment. These sensors should be able to detect obstacles, measure distances, and determine the robot's location. The Lidar sensors are used in this robot;

Lidar sensors are used to create a detailed map of the robot's environment, and determine the distance between the robot and the obstacles. These sensors emit laser pulses and measure the time it takes for the pulses to bounce back. By combining these measurements, the robot can create a 3D map of its surroundings.

Sensing mechanism

The sensing mechanism of the robot involves both external and internal sensors. External sensors are those that are used to detect the environment, while internal sensors are those that are used to determine the robot's motion and location. The following sensing mechanisms is used;

1. **Reactive sensing:** Reactive sensing involves using external sensors to detect obstacles and adjust the robot's path in real-time. This approach is effective for navigating through simple environments but may not be sufficient for complex environments.
2. **Model-based sensing:** Model-based sensing involves using an internal model of the robot's environment to plan its path. This approach is more effective than reactive sensing in complex environments but requires more computational resources.

CHAPTER II

Mobile localization with obstacle avoidance robots require efficient control mechanisms, reliable actuators, and appropriate actuation mechanisms to navigate through their environment while avoiding obstacles. This report discusses the essential components and functionalities necessary for effective control and actuation in such robots.

Control Mechanism.

The control mechanism is the backbone of a mobile localization with obstacle avoidance robot(acting as the brain), enabling it to make informed decisions and execute actions based on sensor data and predefined strategies.

Reactive control is a control mechanism used in this mobile localization with obstacle avoidance robots. It is designed to enable real-time response to immediate sensor inputs, allowing the robot to react and adjust its behavior instantaneously based on the perceived environment. Reactive control focuses on the direct mapping between sensor inputs and corresponding actions without explicit long-
term planning.

Some key aspects regarding this control mechanism include;

Real-Time Response: Reactive control emphasizes immediate reaction to sensor inputs. As the robot continuously receives sensor data, it processes this information and generates control signals to execute actions in real-time. This enables the robot to quickly adapt its behavior to changes in the environment and respond promptly to obstacles or other dynamic elements.

Sensor-Driven Behavior: Reactive control heavily relies on sensor inputs to guide the robot's actions. Sensors such as proximity sensors, vision systems, or depth sensors provide information about the robot's surroundings, including the presence of obstacles, distance measurements, or other relevant environmental cues. Based on these inputs, the robot determines the appropriate action to take, such as adjusting its trajectory or speed to avoid collisions.

Simple Reactive Rules: Reactive control typically employs a set of simple reactive rules that map sensor inputs to specific actions. These rules are designed to handle specific scenarios or conditions. For example, if an obstacle is detected on the left side of the robot, a rule may dictate turning right to avoid the obstacle. These rules are often based on if-then conditions or lookup tables, providing a straightforward and intuitive way to define the robot's behavior.

Lidar (Light Detection and Ranging) scans provide detailedinformation about the surrounding environment in the form of 3D point clouds. The classification of data from Lidar scans can be useful for informing the control mechanisms of mobile localization with obstacle avoidance robots.

Lack of Long-Term Planning: Reactive control does not involve explicit long-term planning or consideration of the robot's global state. Instead, it focuses on immediate sensory information and local interactions with the environment. While this can limit the robot's ability to anticipate future obstacles or plan optimal paths, reactive control excels in situations that require quick reflexes and maneuverability.

Limitations of this control mechanism: Reactive control has certain limitations.

It may struggle in complex and dynamic environments where multiple conflicting inputs are present simultaneously or when dealing with unpredictable situations.

Reactive control tends to exhibit reactive behaviors without considering the long-term consequences of actions, which may lead to suboptimal or inefficient behavior in certain scenarios.

Complementary Approaches: Reactive control can be complemented with other control mechanisms to enhance the robot's capabilities. For instance, higher-level planning algorithms or decision-making processes can be integrated to provide more sophisticated behaviors, combining reactive control with elements of rule-based or model-based control.

Other control mechanisms to use may include;

Proportional-Integral-Derivative (PID) control offers feedback-based adjustment of control signals, and model predictive control (MPC) utilizes dynamic models to optimize control actions.

Actuators

Actuators are the physical components responsible for translating control commands into mechanical movements and actions. They enable the robot to move, steer, and interact with its environment. Common actuators used in mobile localization with obstacle avoidance robots include motors, stepper motors, and actuated grippers.

Motors:

Motors are the types of actuators used in this robot. They provide the necessary torque and rotational motion required for locomotion and maneuvering. Two types of motors often employed in mobile localization robots are:

A. DC Motors: DC (Direct Current) motors are widely used in robotics due to their simplicity and controllability. They convert electrical energy into rotational motion through the interaction of a magnetic field and current-carrying conductors. DC motors offer speed and direction control, making them suitable for driving wheels or tracks.

B. Stepper Motors:

Stepper motors are another type of actuator used in mobile localization robots, especially for applications that require precise control over rotational or linear movements. Stepper motors divide a full rotation into multiple steps, allowing for accurate position control without the need for external feedback. They offer high holding torque, precise positioning, and the ability to hold a position without additional power. Stepper motors are commonly used in robotic arms, camera gimbals, or any application requiring precise motion control.

Actuation Mechanism

Actuation mechanisms determine the type of locomotion and manipulation a robot can achieve. They facilitate the physical movement and operation of actuators to enable the desired actions. Differential drive systems is the main actuation system used in implementation of this robot.

Differential Drive:

The differential drive mechanism is used in the implementations of our mobile robot for omnidirectional movement. It consists of two independently driven wheels or sets of wheels on each side of the robot. By controlling the speed and direction of the wheels, the robot can achieve forward and backward motion as well as rotation by driving the wheels in opposite directions. Differential drive mechanisms provide simplicity, maneuverability, and the ability to turn on the spot. However, they can result in skidding or slipping when navigating challenging terrain or obstacles.

CHAPTER III

This chapter discusses the simulation process for our robot. The main aim of this project is to code or develop a mobile localization with obstacle avoidance robot. A brief description of this robot project is that we’re meant to simulate and show a good understanding of the basic ROS(Robot Operating System) concepts. These include; Ros-Nodes, Ros-Topics, Subscribing, Publishing, and simulating robots in Gazebo.

To begin with, **ROS** is a free and open source robotics software framework that is used in both commercial and research applications. The ROS framework provides a number of robot-programming capabilities which include Message passing interface between processes, High-level programming language support and tools, Availability of third party libraries, Operating system-like features among others.

Ros-Nodes refer to the processes that use ROS APIs(Application Programmer Interface) to perform computations.

Ros-Topics refer to the named buses in which ROS nodes can send a message. A node can publish or subscribe any number of topics.

Some of the technologies used to develop and simulate this robot include;

- **Ubuntu distro** : 16.04 LTS - this is the host operating system.

- **ROS distro**: Kinetic – this is the ROS distro used to obtain the robot programming concepts and practices.

- **Gazebo version** : 7+ - this provides the simulation environment for our robot.

- **Python:** rospy – this is a high-level programming language used in the implementation and rospy is a third-party library utilized.

Fig. 1 rqt_graph

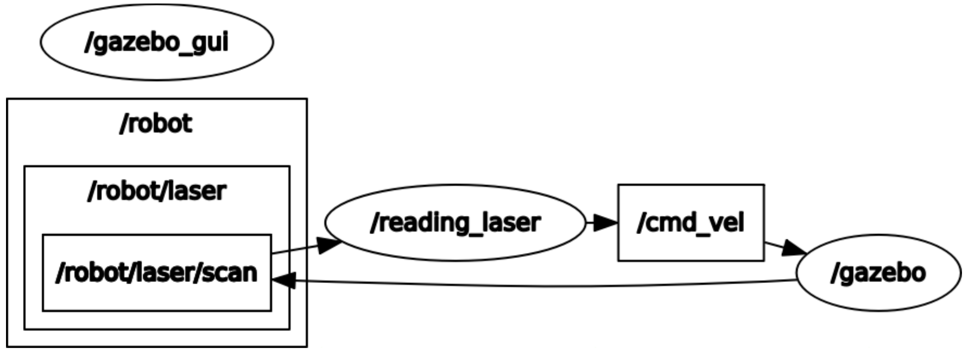


Fig.1 is a snapshot obtained from **“rqt_graph”** command in ROS and shows the different Nodes for example **/reading_laser** and Topics like **/cmd_vel** in this robot.

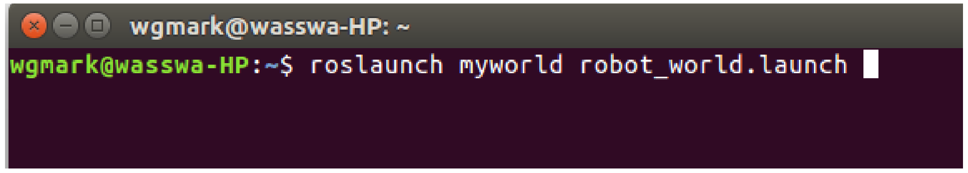
Data processing

This involves analyzing the input dataset which include the position of the robot, position of the obstacles, the distance between the robot and the obstacles, and some of this information is obtained by help of the **LaserScan**.

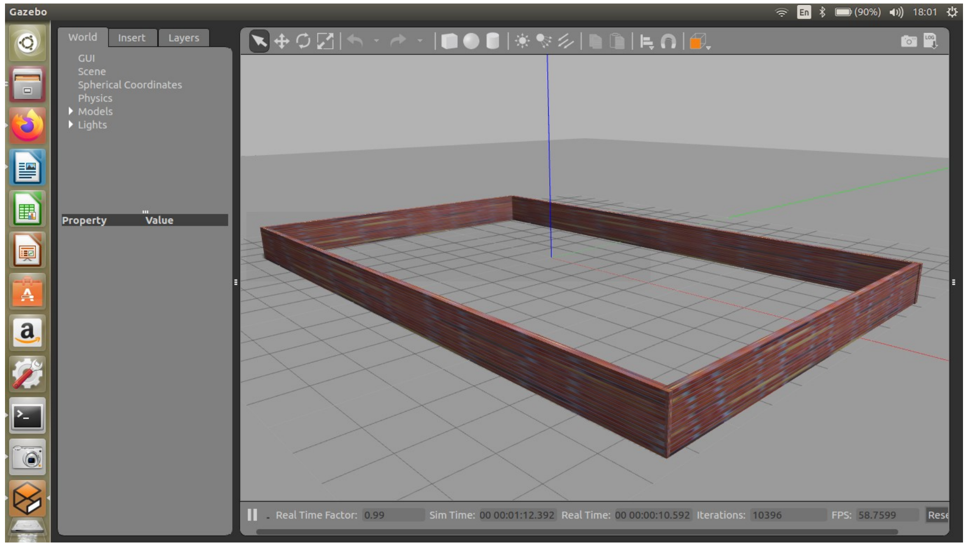
What's a "clearest path"?

The clearest path is the direction in which the distance to the nearest object is larger than the threshold defined in the code. This threshold is introduced so that the robot would know which objects to consider as obstacles and which to ignore. The threshold in this project is defined to be 1.5 meter (it depends on the environment the robot will operate in, and the size of the robot, along with some other factors)

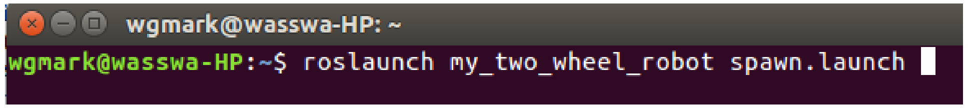
To begin our simulation, we start by launching our simulation environment or world by running the following command in terminal, “ **roslaunch myworld robot_world.launch** ”, this will launch a gazebo environment where our robot will navigate and avoidance new and existing obstacles. **myworld** is the package which holds the robot world.



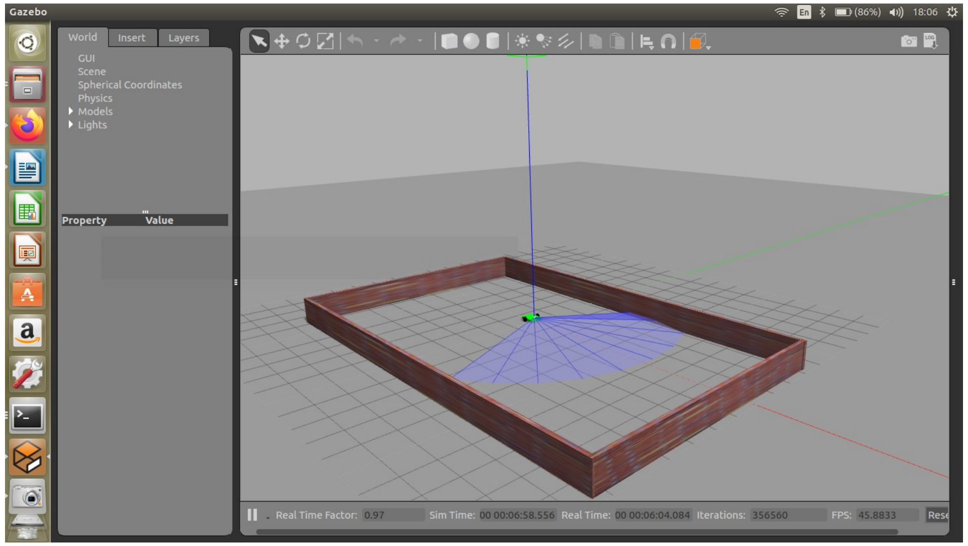
```
wgmark@wasswa-HP: ~
wgmark@wasswa-HP:~$ roslaunch myworld robot_world.launch
```



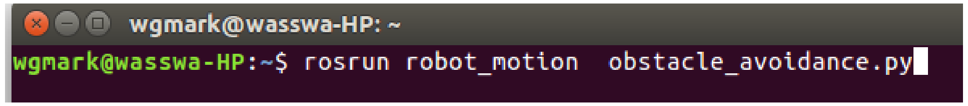
After this, we spawn our robot model into the gazebo environment by running the following command in another terminal, “**roslaunch my_two_wheel_robot spawn.launch** ”, this command will service a request created by the first command, and a robot will be put into the **Gazebo** world at the initial position os (0,0,0).



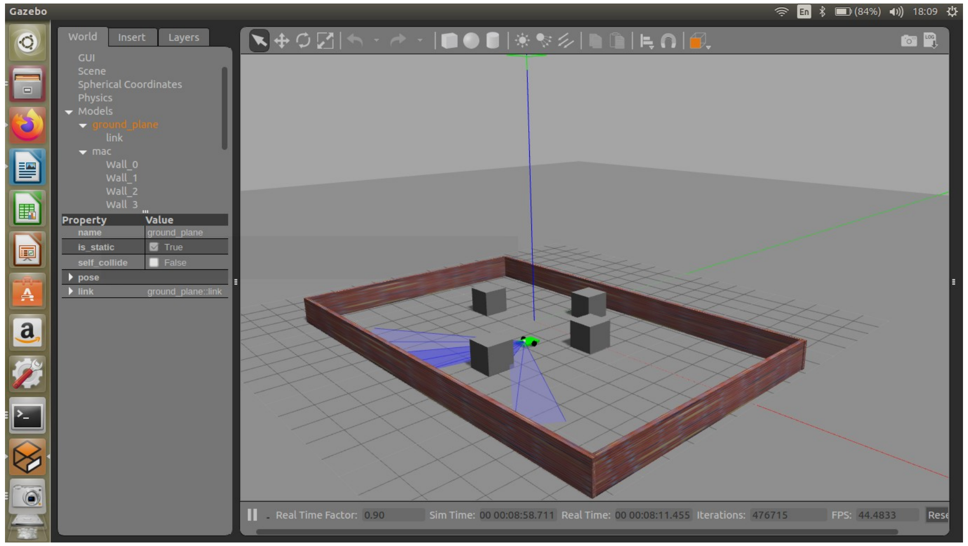
```
wgmark@wasswa-HP: ~
wgmark@wasswa-HP:~$ roslaunch my_two_wheel_robot spawn.launch
```

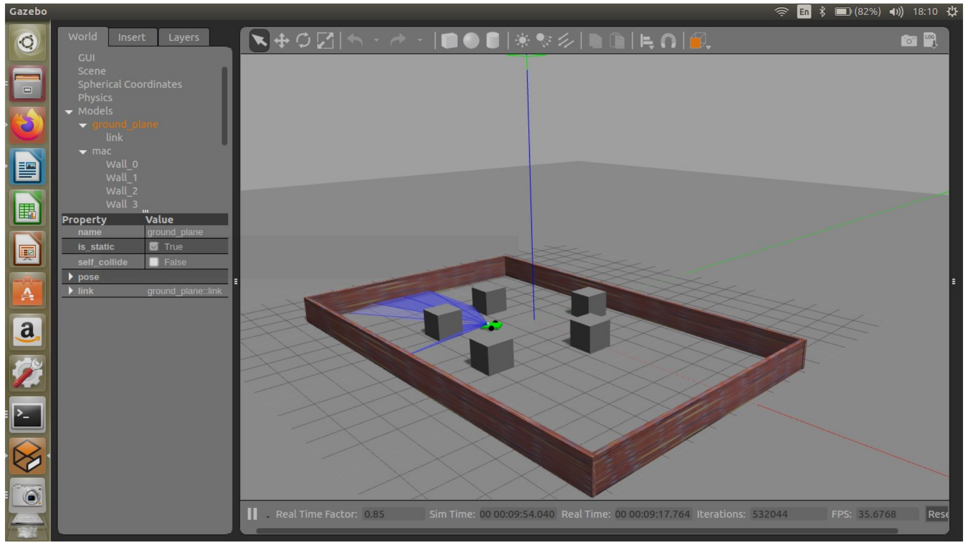


Lastly we have to run the Obstacle Avoidance algorithm which is an executable python file by executing the following command, "**rosrun robot_motion obstacle_avoidance.py** ".

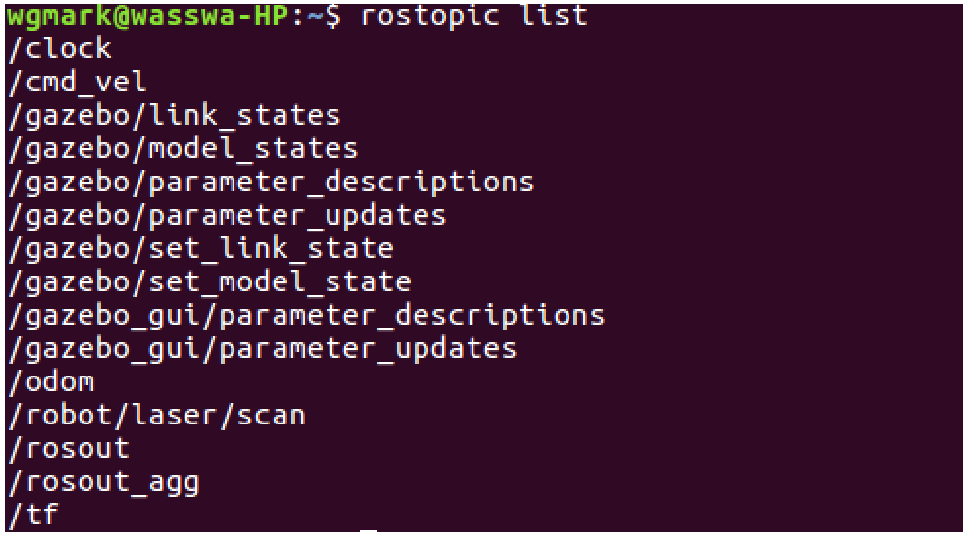


```
wgmark@wasswa-HP: ~
wgmark@wasswa-HP:~$ rosrun robot_motion  obstacle_avoidance.py
```





This is a list of **Ros-topics** running in the system during the simulation process.



```
wgmark@wasswa-HP:~$ rostopic list
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/gazebo_gui/parameter_descriptions
/gazebo_gui/parameter_updates
/odom
/robot/laser/scan
/rosout
/rosout_agg
/tf
```