Gazebo Simulation Fidelity for the Turtlebot3 Burger

Louis van Zutphen

Layout: typeset by the author using LaTeX.
Cover Photographer: Mirka Schoute

Gazebo Simulation Fidelity for the Turtlebot3 Burger

Louis van Zutphen
11851910

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisors
dhr. prof. dr. K.B. Akesson MSc and dhr. ing. E.H. Steffens

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Jun 26th, 2020

Abstract

Simulation software allows programmers to test their code without the need for the physical robot. As a result, a group of people can share a single robot, while working efficiently on individual projects. One requisite for this to work is for the simulation software to accurately replicate the real world, so that proceeding from the simulation environment to the physical world poses as few problems as possible.

This thesis examines the fidelity of the Turtlebot3 Burger bot in the simulation software called Gazebo in order to determine how efficiently the robot can be shared. This is done by conducting experiments in both Gazebo with the virtual robot and the real world with the physical robot. The experiments are designed to both test the fidelity of the individual sensors and actuators as well as to examine how this sensor fidelity affects the fidelity of more complex applications that use them. The results of these experiments show that there are several substantial differences, such as the implementation of too much sensor noise in Gazebo. These differences should be taken into account when testing applications in Gazebo in order to minimise the time required to move from the simulation to the real world. Solutions are suggested for the infidelities. One solution for the implementation of too much sensor noise is to reconfigure the noise in Gazebo to more closely resemble that of the real robot.

Contents

Section 1: Introduction

The Embedded Software and Systems[1] course of the University of Amsterdam (UvA) includes a project wherein a LEGO Mindstorm EV3 robot is programmed to follow a line using a colour sensor. The LEGO robot has fairly unsophisticated sensors that require a lot of fine-tuning and testing to properly work. This process was time consuming for the students following the course and should not be their main focus. Every group of students had one robot, this made it difficult for the students to work efficiently as only one student could use the robot at a time. Additionally, there was a bonus assignment which required two LEGO robots. This made it even more complicated for the students to work efficiently, as an even larger group of people now had to share two robots. Because of these shortcomings of the LEGO robot, the UvA is considering to replace the LEGO robot with a new robot called the Turtlebot3 Burger bot[2]. This robot is more expensive than the previously mentioned LEGO robot. Therefore, there will be less robots available overall as the budget for the course is limited. As a results, it must be possible to share the Turtlebot3 Burger bot efficiently between groups of students.

The robot must be evaluated in order to determine how efficiently students can work with a single robot simultaneously. The Turtlebot3 Burger bot uses simulation software which allows programmers to develop code without needing the physical robot. One aspect that determines how well the robot can be shared is fidelity of this simulation software. The fidelity signifies how well the simulation software represents the real world and how well the behaviour of the robot in an application in the simulation will resemble that of the robot in the same application in the real world. Determining the fidelity of the robot simulation is the goal of the thesis. It is important for the course to know how much time and work is required for code, that works in the simulation software, to function with the physical robot. The fidelity of the simulation dictates how much effort will be required to adjust the code to the real world. Knowing the fidelity will help determine how many students can work with a single robot efficiently and, as a consequence, how many physical robots will be necessary for the course. The simulation fidelity is examined by creating and running a number of tests in the simulations and on the real robot. The results of these are then compared. The tests are kept simple at first in order to examine how accurately the individual sensors and actuators of the robot are emulated in the simulations. Once this accuracy is established, a more complex and higher-level application is assessed.

The Turtlebot3 Burger mainly uses the simulation software called Gazebo[3]. The second section of this thesis examines previous research that has been conducted on the fidelity of Gazebo as well as the fidelity of the models for different robots in Gazebo. The third section covers the background information for the Turtlebot3 Burger bot. This includes the default operating system of the Turtlebot3, the simulation software and the general description of the robot. In the fourth section, the test methods are explained and substantiated. The results of these tests are shown in the fifth section and discussed in the sixth section. The seventh section contains the conclusion of the simulation fidelity derived from the previously discussed results as well as the implications for

[1] https://datanose.nl/Course/Manual/71473/Embedded%20Software%20and%20Systems/2018
[2] http://www.robotis.us/turtlebot-3-burger-us/
[3] http://gazebosim.org/

how the robot can be shared. In addition, potential future work is described in this section as well.

Section 2: Previous Work

Several papers have been written about similar work. Most of these papers look into the fidelity of Gazebo in some way. As of the time of writing, no paper has been published on the fidelity of the Turtlebot3 Burger bot for Gazebo specifically. Papers have been written on Gazebo simulation fidelity for Quadrotor UAVs and custom robots. This section contains information on previous work that relates to this thesis as well as an explanation of how they differ.

Mobile Robot Performance in Robotics Challenges: Analyzing a Simulated Indoor Scenario and its Translation to Real-World[1]

This paper looks into the issues that must be taken into account when moving from simulation to the real world. It also looks into common mistakes that researchers make during this transition. A custom robot is implemented in Gazebo and tested on several aspects, such as execution time, which is also examined in this thesis, and CPU usage. One notable result of the paper is that the execution time in Gazebo turned out to be shorter than in the real world. This is the opposite of what was discovered for the execution time of the Turtlebot3 Burger in Gazebo, which turned out to be slightly longer. This most likely has to do with inaccuracies in the implementation of the robot model. The Turtlebot3 Burger bot used for this thesis is a popular open-source robot that has an official model implemented in Gazebo. Therefore, it can be expected that the Gazebo implementation has a higher fidelity than the model of a custom robot. The paper is similar to the thesis as it also evaluated Gazebo on fidelity. However, it looks mostly at different aspects and uses a custom robot as well.

A High Fidelity Simulator for a Quadrotor UAV using ROS and Gazebo[7]

This paper compares the differences in performance between Gazebo and the real world by running an obstacle course with a Quadrotor UAV. The paper concludes that one of the causes of the difference between these two environments is that the simulation assumes an ideal system. In reality, a substantial amount of noise is introduced. The performance is measured for high-level algorithms such as SLAM, FastSLAM and GraphSLAM. In contrast, mostly low-level applications will be used in this thesis. The paper also uses a flying robot whereas the robot used in this thesis only drives on the ground.

Comprehensive Simulation of Quadrotor UAVs using ROS and Gazebo[2]

This paper implements a dynamic model for Quadrotor UAVs in Gazebo and validates it with a trajectory containing transitions between different velocities. This trajectory is then compared between the real world and the simulated Gazebo world. The dynamic model introduced in the paper implements configurable noise. The paper states that the default method used by Gazebo for simulating a camera is of limited fidelity, as it does not exhibit effects such as motion blur. The robot in this thesis uses the default noise implemented by Gazebo and is not equipped with a camera. The lack of fidelity for the camera can therefore be neglected for this thesis.

Simulation Environment for Mobile Robots Testing Using ROS and Gazebo[6]

This paper claims that, after properly creating a robot model in Gazebo, the code developed for the simulation process can be directly implemented in the real robot without modifications. The paper also propose an effective method for creating precise 3D maps using a combination of ROS packages. The paper assumes that, based on previous research on implementation of robots in Gazebo, the fidelity of this simulation software is high enough to transition from simulation to the real world directly. The paper does not examine the fidelity of Gazebo directly, while this thesis mainly contains experiments showing this fidelity.

Section 3: Background Information

This section will explain the background information required for understanding the simulation fidelity of Gazebo. This section contains several subsections explaining: the operating system of the robot, the simulation software that is assessed, and a description of the relevant components of the robot.

The Robot Operating System

The Robot Operating System (ROS)[4] is the default system used for operating the Turtlebot3 Burger bot. ROS is a powerful open-source framework for creating robot applications. ROS provides a structured communication network in the form of nodes, topics, messages and services. A node is an independent process capable of peer-to-peer communication with other nodes. These nodes are connected to other nodes based on topics. A node can send information over a topic by publishing to it, while other nodes can receive the information that is published on a topic by subscribing to it. Any number of nodes can publish or subscribe to the same topic, and a single node may publish or subscribe to any number of topics. Nodes communicate in the form of messages, which are sent over topics. All messages that are sent over a single topic have a strictly typed data structure. This means that all messages sent over a certain topic must contain the exact same data structure. This strictly typed structure can contain a standard primitive type, such as an integer or a float, or it can be an array containing multiple different primitive types and constants. Such a ROS network can be visualised using a graph. A default ROS network can be seen in Figure 1. The nodes in the graph are also the nodes in the ROS network, the edges between these nodes are the topics and the information that is transferred over these edges are the messages. Apart from topics, nodes can also communicate in the form of services and clients. A node can provide a service that accepts messages in the form of a strictly defined request and response. A client node can call a service node with a request and will then wait for a response from the service node.

The network system employed by ROS provides a number of benefits. It allows for easier single-node debugging due to the independence between the nodes. Because nodes can run independently of other nodes, a single node can be debugged while the rest of the network is left running. This is beneficial for the students following the course as it allows them to test their applications without the need to restart the entire network. Logged sensor data can be used in a controlled environment and logged message data can be played back similarly. The re-usage of data allows for multiple tests to compare different system implementations without the need for more measurements. Networks

[4] ros.org

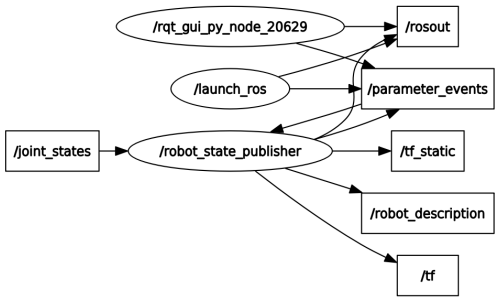


Figure 1: ROS network when only running the robot bring up and the visualisation tool (rqt).

and network configurations can be stored in packages allowing for easier deployment and sharing of work. As previously mentioned, the networks can be visualised using graphs. These graphs make the process of debugging an entire network straightforward.

The latest long-term support version, called ROS2 Dashing Diademata, is used in this thesis. ROS2 was opted for as it will be relevant for a longer time in the future when the course is given at the university. The main differences between ROS 1 and 2 is that the newer version does not use a master node to manage the other nodes and it utilises the Data Distribution Service (DDS)[5]. ROS2 is also available on platforms other than Ubuntu.

Simulation Software

ROS2 has two default simulation environments: Fake Node and Gazebo. Fake Node is used for testing robot models and movement. However, it does not support using sensor data and is therefore not suitable for this thesis. The other simulation software, Gazebo, is used instead. Gazebo is a free open-source simulation software that claims to have a physics engine with a high fidelity. It has competed in multiple competitions, such as the DARPA robotics challenge [3]. Gazebo is considered an overall capable simulation software that has a wide variety of features while maintaining a simple interface [4].

Robot Description

The robot that is used for the course is called the Turtlebot3 Burger bot. The relevant sensors for this robot will be described in this section. Refer to the official documentation of the robot for a description of the other sensors[6]. The robot is equipped with a 360 LiDAR laser distance sensor LDS-01, two differential wheels and Inertial Measurement Units (IMU). The LiDAR is capable of accurately measuring distances between a distance of roughly 120 and 3500 mm. This sensor measures the distance to the object, as well as the intensity of the surface. The LiDAR spins roughly 5 times per second, which allows it to measure distances at the full 360-degrees around

[5]https://www.dds-foundation.org/
[6]https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/

the robot roughly every 2.1 seconds. The robot can achieve a maximum translational speed of 0.22 metres per second and a maximum rotational speed of 2.84 radians per second (162.72 degrees per second). The IMU of the robot consist of a gyroscope, a magnetometer and an accelerometer. The magnetometer and the accelerometer are too noisy in both Gazebo and the real world, with values fluctuating inconsistently. These sensors are therefore not tested in this thesis.

The gyroscope gives the orientation of the robot in the form of a quaternion. This notation is converted to Euler in the gyroscope experiments, as this notation is straightforward to use. The quaternion notation is more relevant for a robot that moves in three dimensions, such as a Quadrotor UAV robot. This is not as applicable to the Turtlebot3, as it can only move on a horizontal plane. The quaternion notation would be more useful than the Euler notation, in case the robot drove on a slope.

Section 4: Method

The fidelity of Gazebo is first tested on a primitive level. This is done with simple experiments where individual or pairs of sensors or actuators are tested. The LiDAR, the actuators, and the gyroscope are tested. Getting a better understanding of the fidelity at a lower level helps when inspecting higher-level applications that are constructed using these primitive parts. One such higher-level experiment is done in this thesis on a wall following application. This section explains what is tested and how the experiments are carried out. All the experiments are done in the Robolab at the UvA. This is a controlled environment, which allows the limiting of many factors. Some of these factors are: a consistent light level throughout all the experiments, the robot is initialised in the same environment, and the robot uses the same ground and wall surfaces for most tests.

The LiDAR

The package that was used for the LiDAR experiments is called "LiDAR_Test" and the code for this package can be found in "LiDAR_Test.py" on the GitHub repository[7]. This package logs the LiDAR measurements to a file in the logs folder with the name format: *laser_log_[time].txt* where *[time]* is the time of day when the data was logged. The code first queries the user for the number of measurements that should be logged. The measured LiDAR distances for 0, 90, 180 and 270-degrees and the measured intensity at 0-degrees are then written to the log file roughly every 0.1 seconds until the desired amount of measurements is reached.

The LiDAR sensor will be tested using three different experiments: the detection range, the noise at different distances and the intensity on different surfaces. These experiments test the different aspects of the LiDAR sensor that will be important in the classroom environment.

The detection range of the LiDAR given in the description is specified as being roughly between 120 millimetres and 3.5 metres. This range is an estimation and can differ between robots. The detection range of the LiDAR will be tested by moving the robot towards a cardboard object from a

[7] https://github.com/LvZut/Turtlebot3_Gazebo_Fidelity

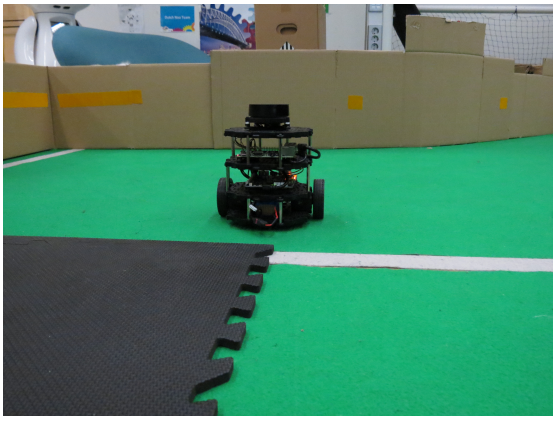


Figure 2: Setup for measuring the noise at different distances in the real world

distance of 4.5 metres to 5 centimetres while logging the LiDAR measurements at a 0-degree angle. The highest and lowest measured values contained within the logged data show the detection range of the robot. The robot will drive at a speed of 0.01 metres per second towards the cardboard object. The logged data will be visualised in the results as a graph showing the measured distance over time as the robot was moved closer to the object.

The noise of the LiDAR will be tested at different distances. During this experiment, the accuracy of the LiDAR is also tested as the measured distances will be compared to the real distance. The distances used will be within the detection range of the LiDAR and more noise tests will be done for shorter distances, as these are more relevant for a classroom environment. Measurement data will be acquired by placing an object with a cardboard surface in front of the robot at the specified distance and logging the LiDAR distance measurements of the robot at 0-degrees. Once the data has been collected, frequency histograms will be created with the distance measurements which show the noise distribution of the LiDAR at set distances. The robot is placed at different measured distances facing a flat cardboard surface. This is shown in Figure 2. A similar experiment will take place in Gazebo. The robot will be placed in front of a cube at varying distances. The set up for the experiment in Gazebo can be seen in Figure 3.

The last aspect of the LiDAR that will be tested is the intensity and the effect of different surfaces measured. The intensity measurement is partly based on the reflectivity of the surface struck by the laser. Intensity can be used to recognise certain surfaces. This makes it possible to place landmarks that the robot can more easily recognise. By default, the intensity values are set to 1 in Gazebo and do not change depending on the distance, even when a higher intensity is assigned to an object. This contrasts with how the intensity behaves for the real robot, as this value decreases when the robot is further away from the object. This substantial difference is further noticed with the wall following application. Intensity values first have to be measured with the robot in the real world before implementing them in Gazebo. The noise test uses a cardboard surface for the measurements. Apart from the cardboard surface, reflective tape and aluminium foil surfaces will be tested at a set distance of 0.5 metres to inspect the effect of these three different surfaces on the intensity measurements of the LiDAR. In addition to the intensity distributions, the noise of the

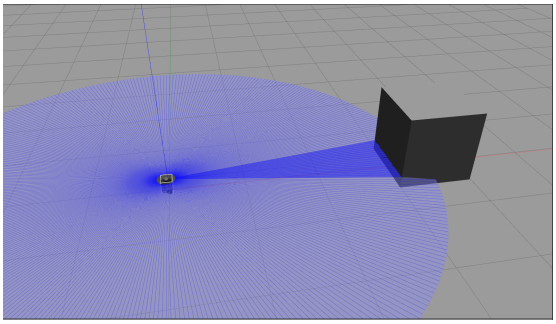


Figure 3: Setup for measuring the noise at different distances in Gazebo

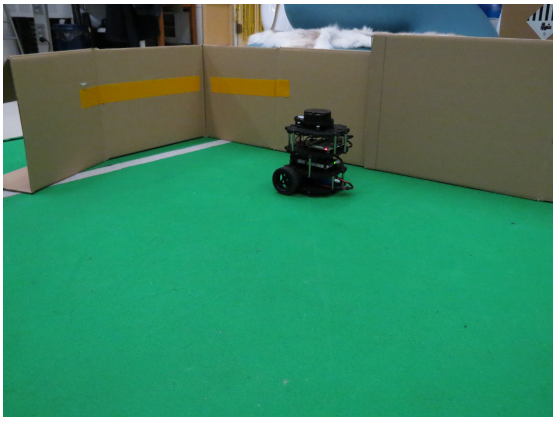


Figure 4: Setup for measuring the intensity on reflective tape at a distance of 50 centimetres

distance measured on these different surfaces will be compared. The set up for this experiment for reflective tape can be seen in Figure 4.

The Actuators and Odometry

The Package that was used for the actuators and odometry experiment is called "Odom_Test". The code for this package can be found in "Odom_Test.py" on the GitHub repository[8]. This package drives the robot in a straight line after receiving a distance to drive and a maximum driving speed. This package uses the odometry of the robot to measure the driven distance and additionally measures the distance in front of the robot at the start and at the end of the drive and logs the absolute difference between the two. The LiDAR measurement serves as verification of the odometry, the reliability of which is determined in the previously described experiment. The test accelerates the robot at the start and decelerates at the end in order to minimise the slip caused by sudden significant changes in translational velocity. The odometry of the robot was logged using the package called Odom_Log.py.

[8]https://github.com/LvZut/Turtlebot3_Gazebo_Fidelity

The actuators and odometry will first be tested by driving the robot in a straight line at a speed of 0.18 metres per second. The odometry of the robot will be compared to the LiDAR measurements. This distance comparison will then be compared between the real world and the simulated world. Additionally, the execution time between Gazebo and the real world will be compared. The distances driven are between 0.3m and 1.5m. The distances were chosen based on their relevance for applications in a classroom environment. There are more tests for shorter distances, as these are more important in a classroom.

The effect of different maximum driving speeds will be examined in a similar manner. The maximum speed will be varied from 0.1 metres per second up to the maximum driving speed of the robot at 0.22 metres per second. All the experiments with different driving speeds will be done over a distance of 1 metre. The odometry will be logged in order to visualise the trajectory of the robot in both the real world and Gazebo. These trajectories will then be visualised in the results.

The Gyroscope

The package used for testing the gyroscope is called "Gyro_Test". The code for this package can be found in "Gyro_Test.py" on the GitHub repository[9]. This package turns the robot a given angle between -180 and 180-degrees and stops within a given threshold. The gyroscope of the robot is then used to check if the robot is within 0.05π of the desired rotation. This is executed on a 0.005 seconds timer, which is the interval at which it checks if the robot is within the threshold of the desired rotation.

The last primitive test will consist of turning the robot at a speed of 2 radians per second to a given rotation using the gyroscope. This will be verified with the use of markings on the ground and measuring the amount of degrees turned. A circle is used in the real world in order to ascertain the rotation angle in the real world. The situation in the real world can be seen in Figure 5. The process of verification in Gazebo is more precise as there already is an orientation attribute present for the robot model in Gazebo..

[9]https://github.com/LvZut/Turtlebot3_Gazebo_Fidelity

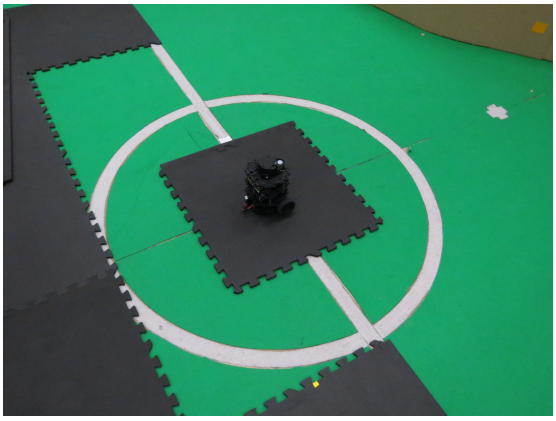


Figure 5: Setup for measuring the rotation of the robot in the real world.

Wall Following Application

The high-level application is a wall following application. The goal is for the robot to stop at the end of the wall with the use of reflective tape. The robot has to follow the wall at a distance of roughly 30 centimetres. In addition to following the wall and stopping at the end, the robot additionally has to avoid an obstacle and count two pieces of reflective tape (called gems in the application) along the wall. For a more detailed description of this application, refer to the thesis by Mirka Schoute [5].

This application will be tested on four different aspects:

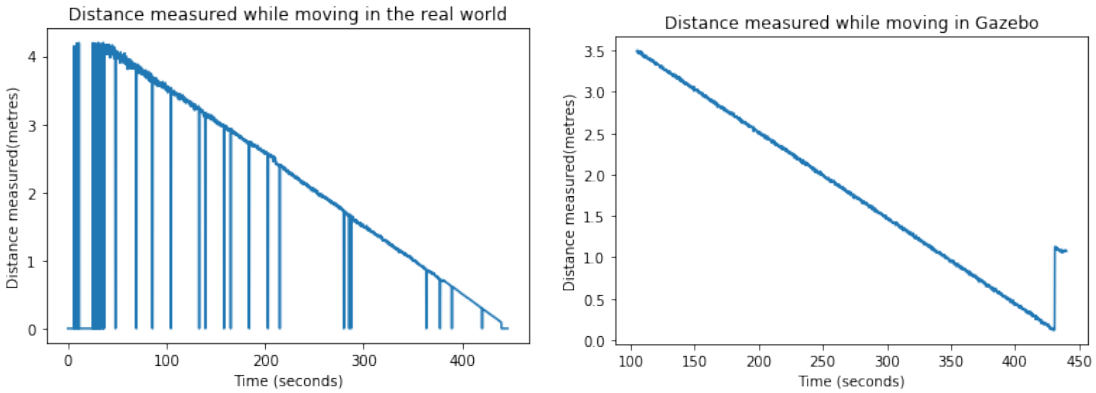
1. The execution time of the application from start to finish
2. The number of gems counted by the robot
3. If the robot managed to avoid the obstacle
4. If the robot stopped at the end of the wall

Section 5: Results

This section shows the results acquired from the experiments described in the previous section. First. the results of the three individual sensors are shown. In the last part of this section, the results of the wall following application are shown.

LiDAR Detection Range

Figure 6a shows the detection range of the LiDAR on the real robot. The figure shows the detected LiDAR distance at the front of the robot, while it was moving towards a wall at a speed of 0.1 metres per second. A steadily decreasing line can be seen with regularly occurring measurements of 0 metres. 9.81% (438 out of 4463) of the values were 0. The highest and lowest measured values were 4.200 metres and 0.091 metres, respectively. Figure 6b shows the same experiment in Gazebo.



(a) LiDAR detection range in real world

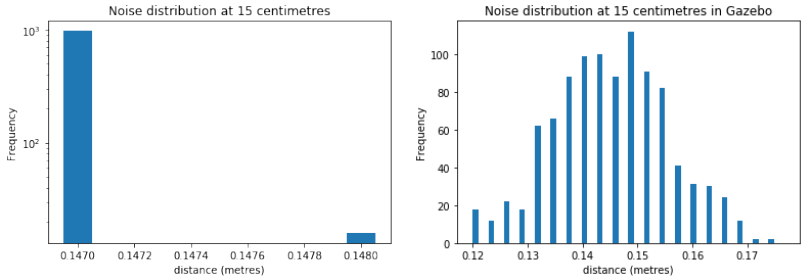
(b) LiDAR detection range in Gazebo

Figure 6: LiDAR detection range of both the real world and Gazebo. The data was acquired moving the robot from outside the LiDAR detection range towards an object at a speed of 0.1 metres per second. The horizontal axis shows the time, while the vertical axis shows the distance measured by the LiDAR.

The figure shows a steadily decreasing line without 0 measurements throughout. The highest and lowest measured values were 3.5 metres and 0.12 metres, respectively. The measurements below the LiDAR detection range, the last 92 values, show noisy data with a mean of 1.087 metres. Additionally, it should be noted that, in the real world, the LiDAR measures 0 when the distance is above the detection range, while the LiDAR in Gazebo will measure inf.

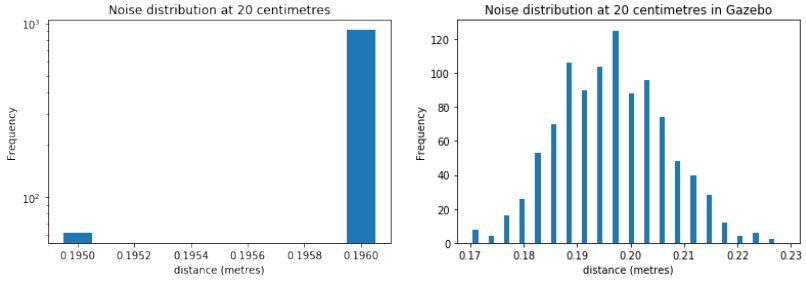
LiDAR Noise Distributions

Figures 7a - 9c show the comparison of the noise distributions between the real world and Gazebo at 15, 20, 30, 40, 50, 75, 100, 150, 200 and 300 centimetres, respectively. For the distances below 75 centimetres, the LiDAR measurements in the real world show little noise except for the occasional zero measurement. At and above 75 centimetres, noise becomes visible and the amount of zero measurements decreases. The mean and deviation for the figures can be seen in Table 1. The mean and deviation shown in this table were calculated without using the 0 measurements of the data. In this table, it can be seen that the deviation in the real world overall increased as the distance increases. The standard deviation is roughly 0.09% of the mean at 15 centimetres and increases up to roughly 0.4% of the mean at 300 centimetres. The noise distributions of Gazebo shows a consistent noise distribution across all distances and no zero measurements. The deviation for Gazebo in table 1 agrees with this as the deviations are roughly the same for all distances. The noise present in Gazebo is the most similar to the noise at 300 centimetres in the real world.



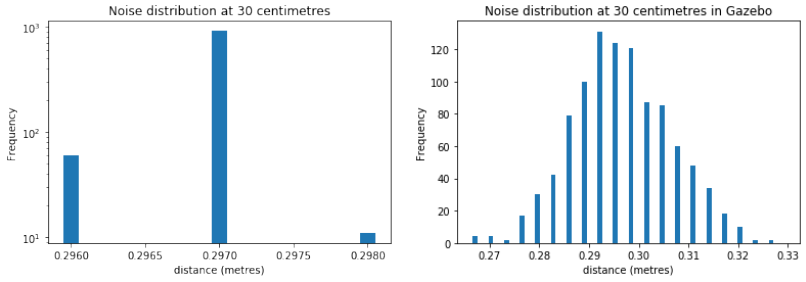
(a)

(b)



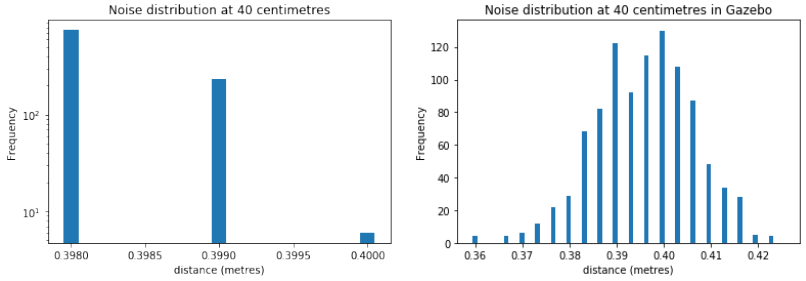
(c)

(d)



(e)

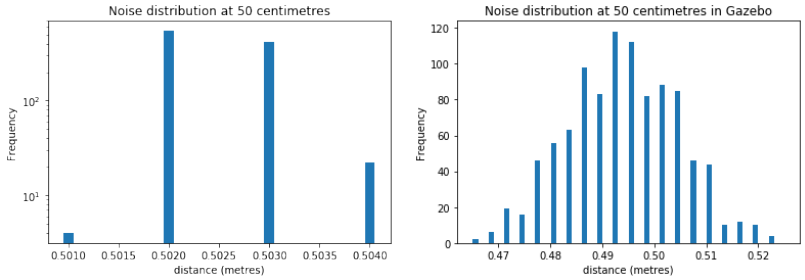
(f)



(g)

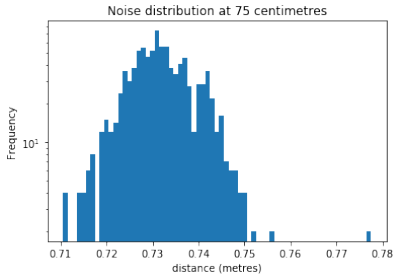
(h)

Figure 7: Distributions of the LiDAR noise at a distance of 15, 20, 30, and 40 centimetres. The left column shows the real world distributions and the right columns the Gazebo distributions. Measured on a flat cardboard surface in the real world. The horizontal axis shows the distance measured by the LiDAR while the vertical axis shows how often the corresponding value was measured. The zero measurements are filtered out for the real world distributions.

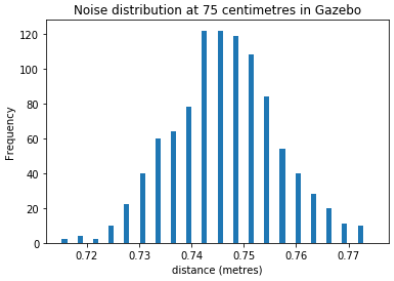


(a)

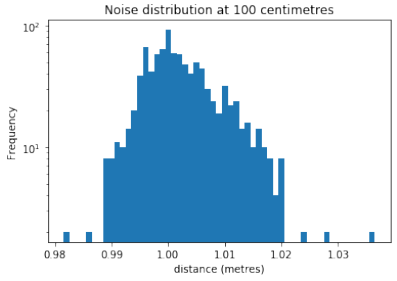
(b)



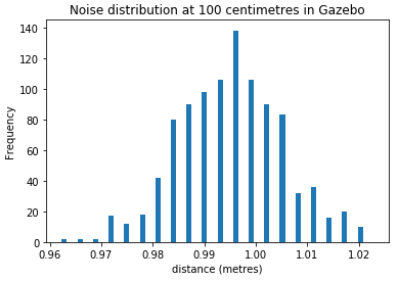
(c)



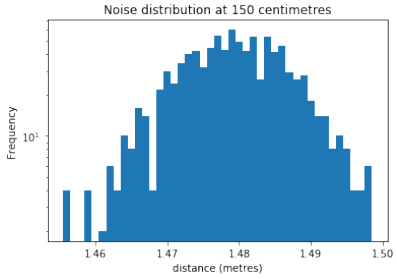
(d)



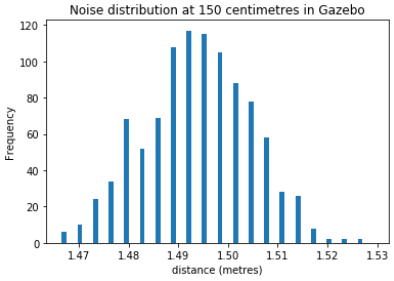
(e)



(f)



(g)



(h)

Figure 8: Distributions of the LiDAR noise at a distance of 50, 75, 100, and 150 centimetres. The left column shows the real world distributions and the right columns the Gazebo distributions. Measured on a flat cardboard surface in the real world. The horizontal axis shows the distance measured by the LiDAR while the vertical axis shows how often the corresponding value was measured. The zero measurements are filtered out for the real world distributions.

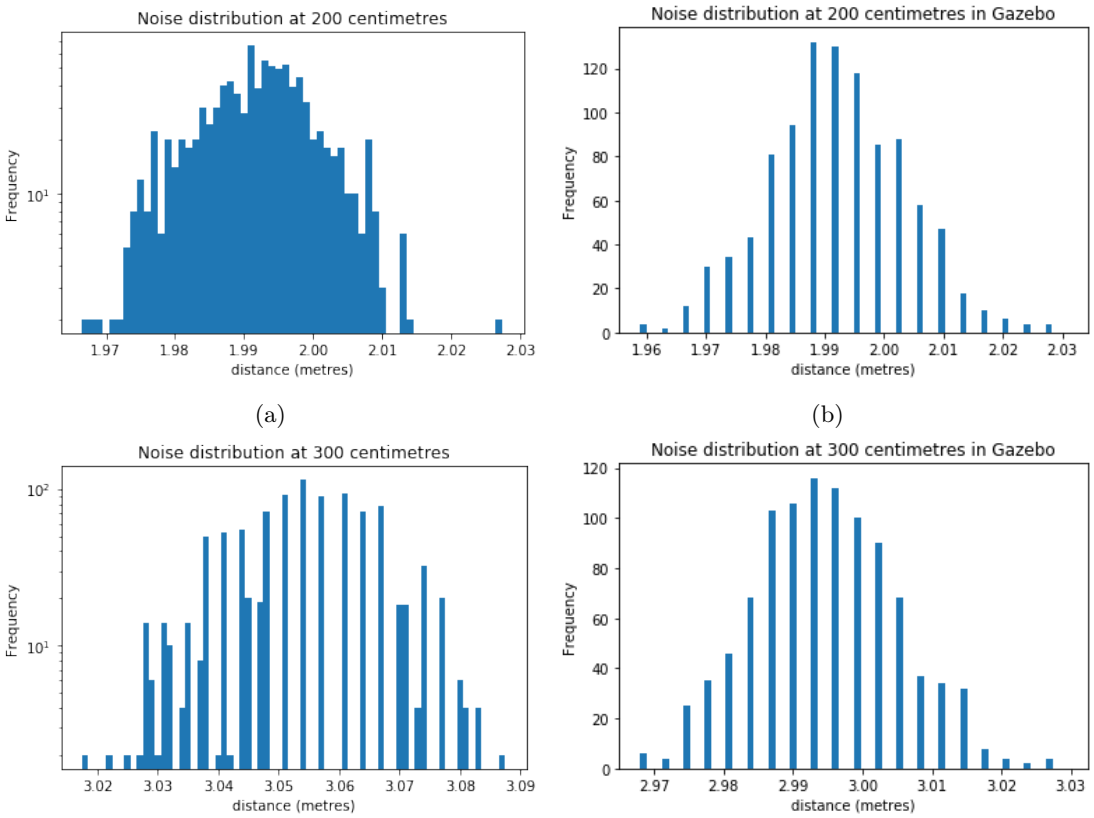


Figure 9: Distributions of the LiDAR noise at a distance of 200 and 300 centimetres. The left column shows the real world distributions and the right columns the Gazebo distributions. Measured on a flat cardboard surface in the real world. The horizontal axis shows the distance measured by the LiDAR while the vertical axis shows how often the corresponding value was measured. The zero measurements are filtered out for the real world distributions.

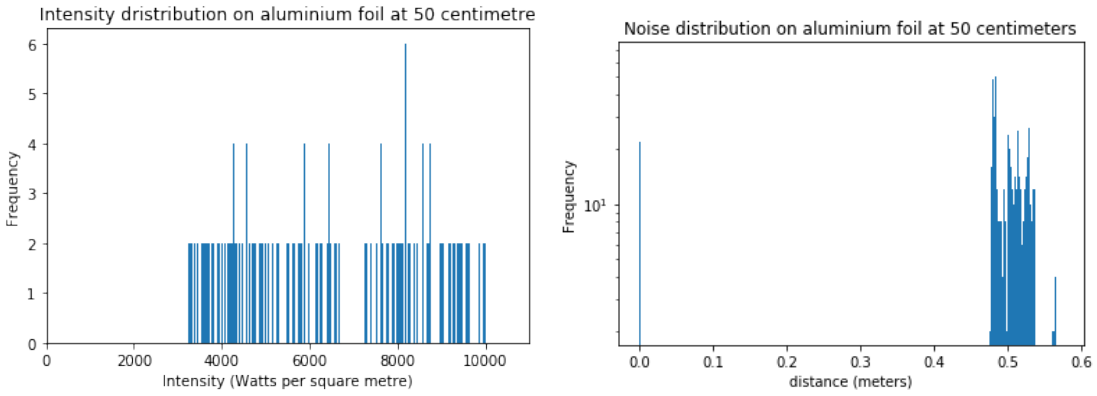
Table 1: Mean and σ for Figures 7 - 9. Distance measurements of 0 centimetres were filtered out before the calculations. The second and third column show the mean and σ for the real world. The fourth column shows the amount of zero measurements in the real world. The last two columns show the mean and σ for Gazebo. There were no zero measurements in Gazebo

Mean and σ of noise distributions

Distance (cm)	Real world			Gazebo	
	Mean (m)	σ (m)	Zeros	Mean (m)	σ (m)
15	0.147	0.000126	11	0.145	0.0105
20	0.195	0.000243	23	0.197	0.0100
30	0.296	0.000263	15	0.297	0.0099
40	0.398	0.000445	12	0.397	0.0105
50	0.502	0.000548	15	0.494	0.0106
75	0.732	0.00759	31	0.747	0.0100
100	1.002	0.00691	24	0.996	0.0101
150	1.479	0.00767	24	1.494	0.0106
200	1.991	0.00864	20	1.993	0.0115
300	3.053	0.0122	0	2.995	0.0103

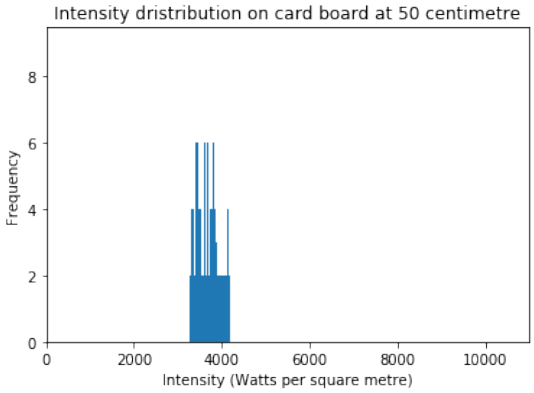
LiDAR Surface Influence

The LiDAR intensity and noise distributions at a distance of 50 centimetres for aluminium, cardboard and reflective tape can be seen in Figure 10. Figure 10a shows the intensity distribution for aluminium foil. From this figure, it can be seen that the intensity fluctuates between 3000 and 10000 W/m^2. The noise distribution is visible in Figure 10b. This figure shows that an aluminium foil surface at 50 centimetres distance results in more noise and more zero measurements than a cardboard surface. Figure 10c shows the intensity distribution for cardboard. For this surface it can be seen that the intensity fluctuates between 3200 and 4200 W/m^2. The noise distribution is visible in Figure 10d. This figure is equivalent to Figure 8a. Figure 10e shows the intensity distribution for reflective tape. From this figure it can be seen that the intensity fluctuates between roughly 9500 and 11750 W/m^2. The noise distribution is visible in Figure 10f. This figure shows that there is more noise on reflective tape at a distance of 50 centimetres than on cardboard but less noise than aluminium foil. In addition, there are more zero measurements on reflective tape at this distance than on cardboard. It can be speculated that zero measurements are caused by the surface not being entirely flat which causes the LiDAR laser to not reflect straight back. In addition, this explains why more zero measurements occur on reflective surfaces, as the laser is less likely to be reflected back at the LiDAR.

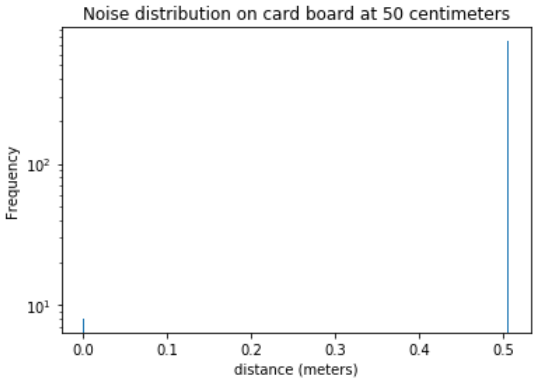


(a)

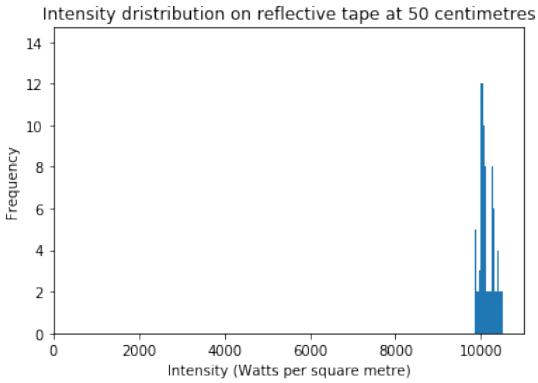
(b)



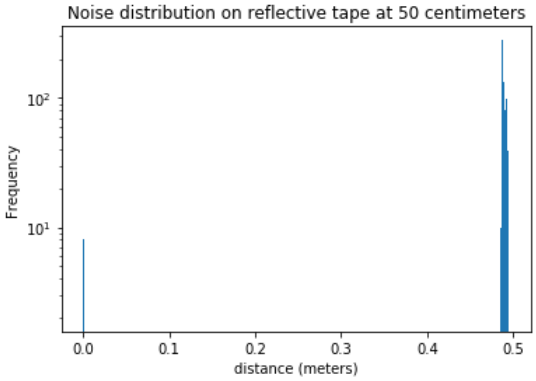
(c)



(d)



(e)



(f)

Figure 10: The intensity distributions and corresponding noise distributions of the LiDAR measurements on aluminium foil, cardboard and reflective tape at a set distance of 0.5 metres. The horizontal axis of Figure 10e shows the measured intensity and the vertical axis shows how often each intensity was measured. The horizontal axis of Figure 10f shows the measured distance and the vertical axis shows how often each distance was measured.

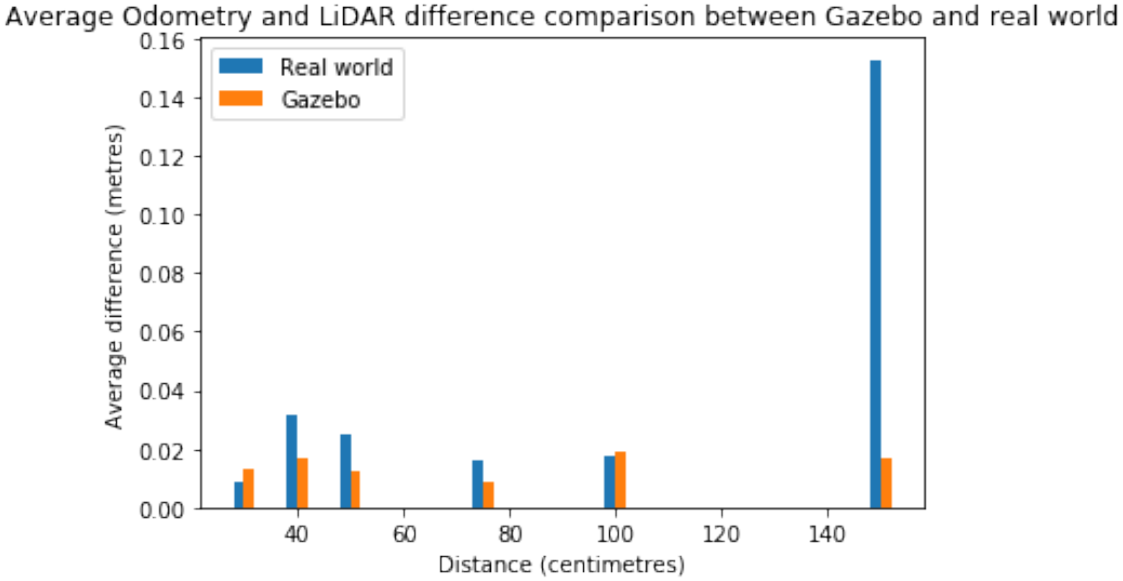


Figure 11: Difference of the measured odometry distance and measured LiDAR distance when driving different distances at a speed of 0.18 metres per second. These values are compared between Gazebo and the real world. The horizontal axis shows the distance driven while the vertical axis shows the average difference between the odometry and the LiDAR.

Odometry

Figure 11 shows the comparison between the real world and Gazebo by examining the difference between the odometry and the LiDAR. This figure shows that the odometry and LiDAR disagreed roughly the same amount when driving a distance of 100 centimetres or less. For a distance of 150 centimetres, there was a big difference between odometry and LiDAR in the real world, but not in Gazebo. This inconsistency was caused by the drift of the robot. The odometry measures the distance from the starting position, while the LiDAR measures the distance remaining to the wall in front of the robot.
Figure 12 shows the time it took for the robot to drive a given distance. The execution times for the real world were overall slightly lower than in Gazebo except for the two shortest distances, 30 and 40 centimetres.

Figures 13 and 14 show the trajectories of the robot when driving one metre at different driving speeds. The trajectories in Gazebo all follow a straight line where the robot did not deviate significantly. In contrast, the robot did deviate substantially when driving at higher speeds. At speeds above 0.15 metres per second the robot started to deviate noticeably in the real world. Additionally, this deviation does not follow the same pattern every trial and the trajectory deviation varies more when driving faster. Finally, there seems to be a substantial increase in deviation between 0.2 m/s and 0.22 m/s.

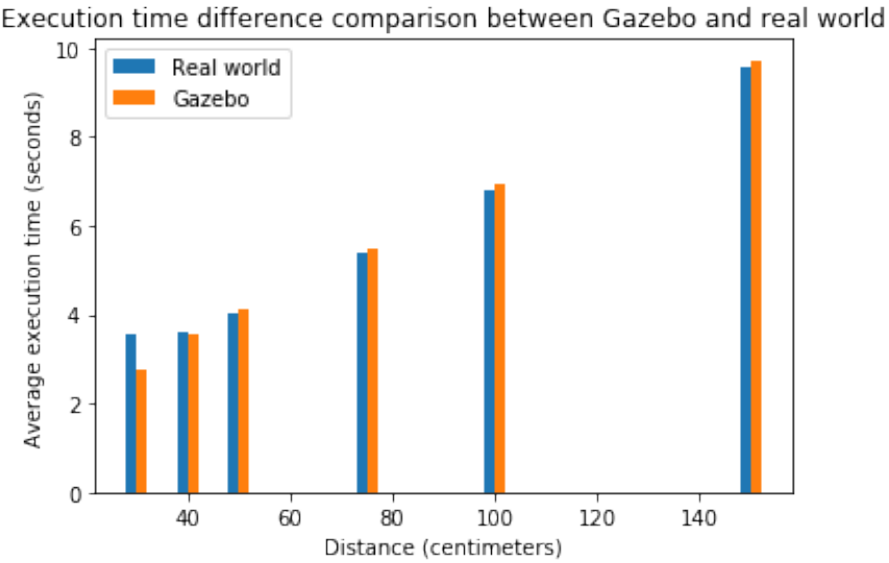


Figure 12: Comparison of the execution times between Gazebo and the real world when driving different distances at a speed of 0.18 metres per second. The horizontal axis shows the driven distance while the vertical axis shows how many seconds the robot drove.

Gyroscope

The results for the Gyroscope experiment are shown in Table 2. This table shows the average turning error in degrees for both the real world and Gazebo. The robot turned too far in all cases. In the real world, the error was overall larger when rotating counter-clockwise. In Gazebo, the error was overall larger when rotating clockwise. The error was larger on average for the physical robot. This suggests some sort of slight imperfection in the wheels of the physical robot.

Wall Following Application

Tables 3 and 4 show the results for the wall following application in the real world and in Gazebo, respectively. The application worked perfectly in the real world, as the robot always counted the exact amount of gems, stopped at the end, and avoided the obstacle. In Gazebo, the robot counted 2 gems 50% of the time and 3 gems 50% of the time. In addition, the robot managed to stop at the end of the wall half of the time. The average execution time of the wall following application was 326 seconds and 389 seconds in the real world and in Gazebo, respectively. The robot always managed to avoid the obstacle.

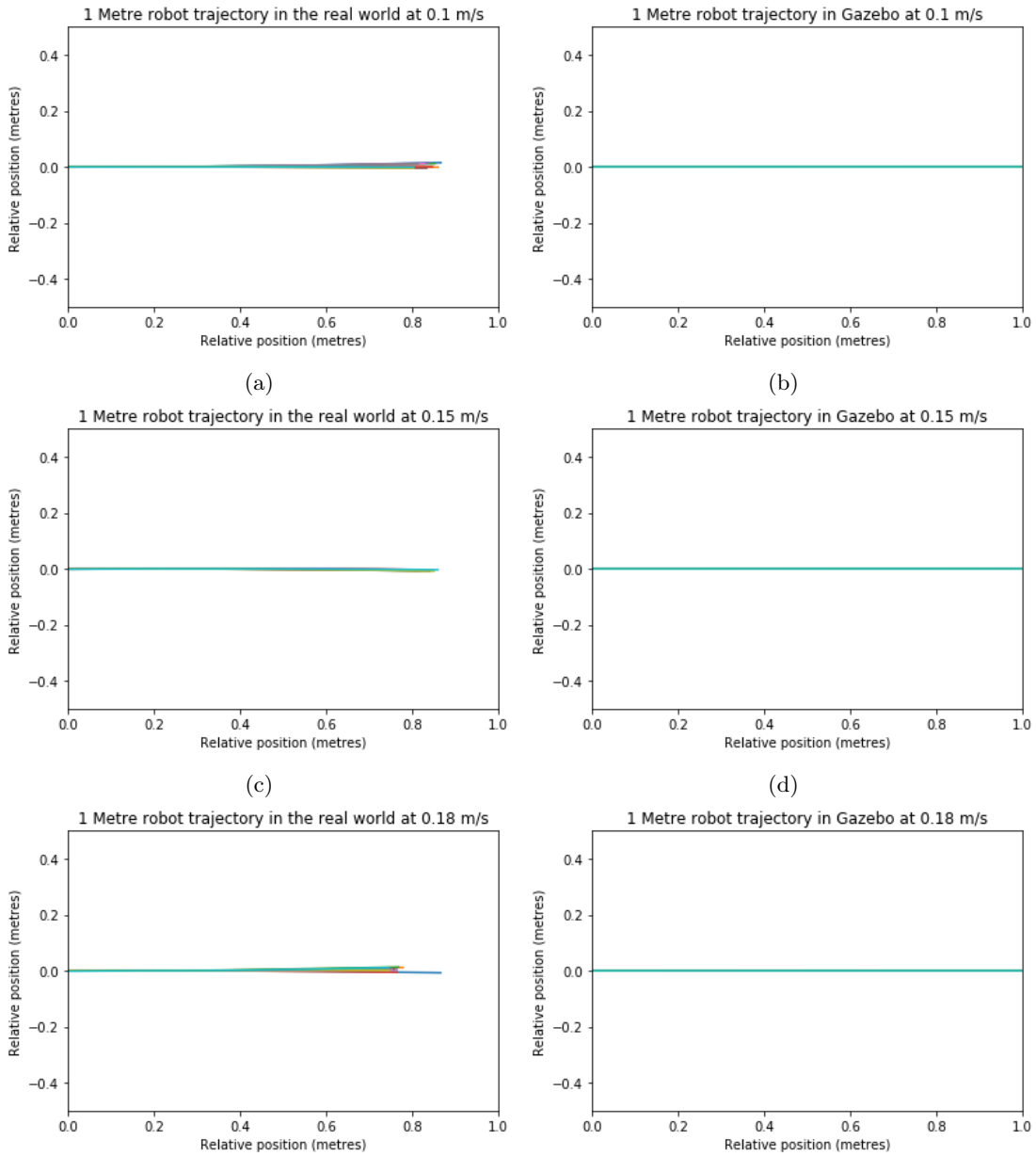
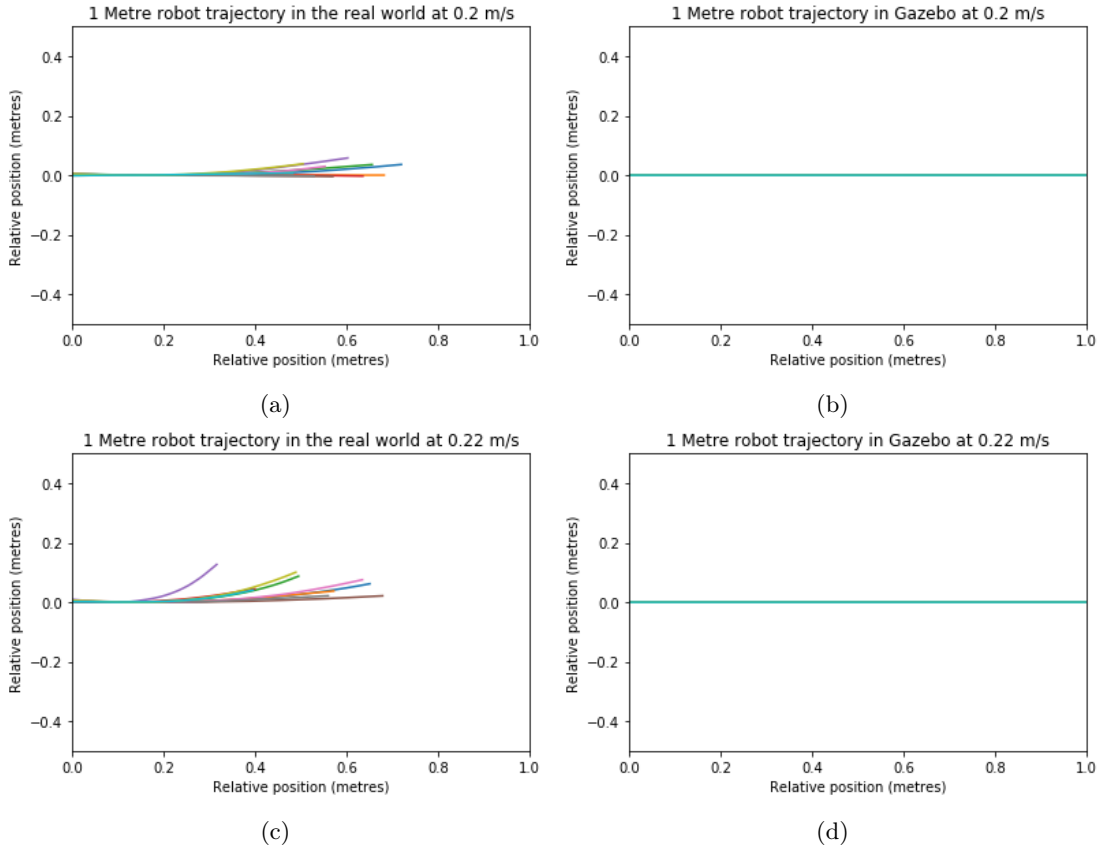


Figure 13: Visualisation of five trajectories of the robot for each driving speed. The figures show trajectories that cover a distance of one metre at a speed of 0.1, 0.15 and 0.18 metres per second. All trials are visualised relative to their own starting position. The x and y axes show their respective relative coordinates of the robot.



(a) (b)

(c) (d)

Figure 14: Visualisation of five trajectories of the robot for each driving speed. The figures show trajectories that cover a distance of one metre at a speed of 0.2 and 0.22 metres per second, with 0.22 metres per second being the maximum driving speed of the robot. All trials are visualised relative to their own starting position. The x and y axes show their respective relative coordinates of the robot.

Table 2: Average error in degrees for the different turning angles of the gyroscope test in the real world and in Gazebo. Positive errors signifies that the robot turned too far. Each angle was tested 5 times for both situations.

Turning angle error

Angle	Error real world (degrees)	Error Gazebo (degrees)
-180	10.35	1.2
-90	5.4	5.4
90	6.3	5.8
180	4.05	8

Table 3: Results for the wall following application in the real world. The first column shows the execution time in seconds. The second column shows how many gems the robot counted. The third and fourth column show if the robot stopped at the end of the wall and if the robot avoided the obstacle respectively.

Results for the wall following application in the real world

Time (seconds)	Gems	Stopped	Obstacle
312.03	2	True	True
340.86	2	True	True
324.03	2	True	True
322.72	2	True	True
323.75	2	True	True
340.01	2	True	True
329.93	2	True	True
317.68	2	True	True

Table 4: Results for the wall following application in Gazebo. The first column shows the execution time in seconds. The second column shows how many gems the robot counted. The third and fourth column show if the robot stopped at the end of the wall and if the robot avoided the obstacle respectively.

Results for the wall following application in Gazebo

Time (seconds)	Gems	Stopped	Obstacle
360.69	2	True	True
392.94	3	False	True
450.49	3	True	True
375.45	2	True	True
371.27	2	False	True
362.95	2	True	True
432.07	3	False	True
368.50	3	False	True

Section 6: Discussion

The results of the previous section are discussed in this section. The discussion includes the differences in fidelity as well as some solutions for the problems that arise. Each subsection within this section discusses the results of one of the sensors and the final section discusses the wall following application.

LiDAR

The results of the LiDAR detection range already show several substantial differences between the LiDAR sensor in Gazebo and the real world. The first of which is that the LiDAR in Gazebo never measures zero values. This means that LiDAR measurements in the real world are not as reliable as in Gazebo. Another substantial difference for the detection range of the LiDAR are the actual measurable minimum and maximum value. These differ slightly between these two situations: The maximum range is 4.2 metres in the real world and only 3.5 metres in Gazebo. The minimum range is 0.091 metres in the real world and 0.12 metres in Gazebo. It is a possibility that the detection range varies between robots. As a consequence, the specifications of the robot have to be on the lower end of this variance in order to ensure that all manufactured robots at the very least meet the specifications. This advocates the testing of additional robots if they are bought for the course in order to more precisely know the individual specifications. The next substantial difference is how measurements outside the detection range are handled. These values are measured as 0 in the real world, but as infinite in Gazebo. Lastly, distances below the detection range are recorded as 0 in the real world, but are recorded as noisy data in Gazebo. This difference with values outside of the detection range is fairly small and the solutions to solve it are mostly straightforward. However, problems can occur when the programmer of the application is not aware of this difference between the real world and Gazebo.

One solution for dealing with values outside of the detection range is to look at the values recorded before and after. If the LiDAR measures a distance that is near one of the edges of the detection range and the next measured value is 0, it is safe to assume that the robot measured a distance outside the detection range.

When looking at the noise distributions, it is clear that Gazebo always follows the same distribution, regardless of distance. However, the real robot shows an increasing amount of noise when the distance to the object increases. Furthermore, the noise at lower distances is much smaller than the constant noise in Gazebo. In the case of a classroom, this is mostly beneficial as lower distances are more relevant. An application that works in Gazebo with the excessive amount of noise at shorter distances will definitely work in the real world with less noise.

The results for the intensity experiments shows that the intensity of an object can reliably be used to distinguish cardboard surfaces and reflective tape, when these are the only two surfaces present. The intensity values for cardboard surfaces fall roughly between 3200 and 4200 W / m^2, while the intensity values for reflective tape fall roughly between 9500 and 11750 W / m^2. This makes it possible to classify a measured intensity value as one of these two surfaces at a distance of 50 centimetres if it falls within one of these two ranges. The intensity of an aluminium foil surface fluctuates excessively which makes classifying it unreliable.

Actuators and Odometry

The first experiment shows that there is only a very small difference between the odometry in the real world and in Gazebo. The substantial difference at 1.5 metres was most likely caused by the skewed trajectory of the robot, which is only present in the real world. The skewing can be caused by a number of factors. Some of these factors are: the driving speed of the robot, the material of the floor the robot is driving on, the slope of the floor, or even a marginal difference between the manufacturing of the two differential wheels. Only the effect of driving speed is further examined in this thesis. Other causes for the skewing of the trajectory can potentially be further looked at in future work.
The execution times show that a driving application executes slightly longer in Gazebo when driving longer distances, but slightly shorter when driving very short distances. This suggests that the actual speed that the robot achieves is higher while the acceleration is lower in Gazebo.
The trajectories of the robot at different speeds visualise the difference in skewing between the real world and Gazebo. The trajectory of the real robot is skewed substantially, while the trajectory in Gazebo follows a straight line. There is a slight difference at low driving speeds albeit significantly smaller than at higher speeds. Therefore, it is important for the programmer of an application to drive at low speeds when trajectory fidelity is important. The application used for this test did not take into account the possible skewing of the robot. One solution to counteract the trajectory skewing in the real world would be to continuously monitor the odometry of the robot and to correct any deviations in the heading of the robot as soon as they occur.

Gyroscope

The turning angle error is noticeable in both situations: the robot consistently turns too far both in the real world and in Gazebo. This indicates that the gyroscope is somewhat unreliable. Due to the fact that the error occurs in both cases, the gyroscope should not cause unexpected results when moving from simulation to the real world. The odometry could potentially be used in order to help determine the current rotation of the robot. This could result in a more accurate rotation.

Wall Following Application

The results for the wall following application show that the robot performed a considerable amount better in the real world than in Gazebo. The robot ran the application perfectly in the real world every time, while only successfully achieving this perfectly twice in Gazebo. One cause for the difference in executing time could be explained by looking at the execution times when driving in a straight line. When the robot drives a short distance, the execution time in the real world is considerably lower than in Gazebo. In the wall following application, the robot constantly drives short distances. A possible cause for difference in gems counted is that the intensity is implemented poorly in Gazebo. The measured intensity values do not change depending on the distance of the object. This makes it likely for the robot to detect a single gem multiple times as it will recognise a gem from a longer distance. This is not a problem in the real world, as the intensity values decrease with distance. The robot will only recognise a gem when it is close to the robot.
At first, the programmer of this application was not fully aware of the differences between the real world and Gazebo. This meant that the application had to be adjusted accordingly before it worked with the actual robot.
The values above the detection range were treated as being below the detection range instead as

these values are set to 0 in the real world. As a consequence, the robot recognised empty space as a wall that was too close to the robot to detect.
Furthermore, the detection of reflective tape using intensity had to be fine-tuned. This process was time consuming and resulted in needing the physical robot for a longer time than was originally expected.

Section 7: Conclusion

Overall, there are numerous differences between the real world and Gazebo that affect the fidelity. Most of these differences are trivial and can be dealt with individually as long as the programmer is aware of them. This process can nonetheless be time consuming and makes it harder to share the robot efficiently. However, once this knowledge is acquired, methods can be implemented that allow for the overall fidelity to be retained across applications, which can, in turn, greatly reduce the amount of time required with the physical robot. Once the trivial infidelities of Gazebo are ironed out, the programmers can focus on the actual task of designing and creating the application. This makes the size and complexity of the application the main bottleneck when sharing the robot. This is more in line with the intentions of the Embedded Software and Systems course as students are occupied with finding solutions for the assignments instead of tuning or testing sensors.
Sharing the robot can be done efficiently due to the structure of the Robot Operating System. The independent nodes of the ROS networks allow the efficient sharing of the robot. Students can connect simultaneously and run their packages sequentially without the need to restart or reinitialise the robot. Furthermore, the deployment of code is facilitated due to the robot being connected to the Internet and using Ubuntu. Additionally, the reliable fidelity of Gazebo indicates that students will require only a small amount of time with the robot as most work can be done in simulations. All in all, the biggest limiting factor when sharing the robot will be the assignment itself. If the testing of the assignment only takes a small amount of time, a large group of students will be able to work with a single robot. If the testing takes a longer time, it is recommended to invest in more robots and reduce the size of the groups so more students can test their implementations simultaneously.

Future Work

Due to the restricted amount of time for this project, choices had to be made for the direction of the research. There are several potential fields that could be explored in future work.
Multiple robots could be compared in order to examine the difference in their LiDAR, actuators and other characteristics. Only a single physical robot was available for this research. Therefore, it was not possible to inspect the differences between individual robots.
A proper implementation for laser intensity in Gazebo is potential work for the future. The current implementation is lacking and is one of the main reasons why the high-level application performed poorly in the simulation. A proper implementation would involve the implementation of variable noise depending on the surface and an intensity value that depends on the distance to the object. This would also make it more relevant to research the intensity on more surfaces, as well as the exact correlation between intensity and distance.
One aspect of the actuators that was not touched upon in this thesis was the impact of driving backwards. Due to the layout of the wheels and the weight distribution of the robot, it can be hypothesised that there is some kind of difference. The trajectory of the robot at different speeds

can be expanded upon. The trajectory would be more clearly visible if the robot drove a longer distance than one metre. An implementation of drift in Gazebo could also be added.
The turning of the robot with the use of the gyroscope was only done on a single rotation speed and with a small amount of trials. The effect of different speeds could be examined. In addition, the centre of the robot moved while rotating, the exact behaviour of the robot could be tested.
Lastly, the actual amount of students that can use a single robot could not be concluded theoretically. An accurate amount could be determined with the participation of multiple students.

References

[1] Francisco Rodrıeguez Lera et al. "Mobile robot performance in robotics challenges: Analyzing a simulated indoor scenario and its translation to real-world". In: *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*. IEEE. 2014, pp. 149–154.

[2] Johannes Meyer et al. "Comprehensive simulation of quadrotor uavs using ros and gazebo". In: *International conference on simulation, modeling, and programming for autonomous robots*. Springer. 2012, pp. 400–411.

[3] Umit Ozguner et al. "Simulation and testing environments for the darpa urban challenge". In: *2008 IEEE International Conference on Vehicular Electronics and Safety*. IEEE. 2008, pp. 222–226.

[4] Lenka Pitonakova et al. "Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators". In: *Annual Conference Towards Autonomous Robotic Systems*. Springer. 2018, pp. 357–368.

[5] Mirka Schoute. "Application programming for an embedded system in education on TurtleBot3 using statecharts". B.S. Thesis. Netherlands: University of Amsterdam, May 2020.

[6] Kenta Takaya et al. "Simulation environment for mobile robots testing using ROS and Gazebo". In: *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE. 2016, pp. 96–101.

[7] Mengmi Zhang et al. "A high fidelity simulator for a quadrotor UAV using ROS and Gazebo". In: *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2015, pp. 002846–002851.