Evaluating the simulation gap for training off-road self-driving.

Lars Zandbergen

Layout: typeset by the author using LaTeX.
Cover illustration: Aion Robotics

Evaluating the simulation gap for training off-road self-driving.

Lars Zandbergen
12093009

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

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

Supervisors
dr. E. Bruni, O. Ligthart
Secondary Supervisors:
D. Cerny, P. De Marez Oyens

Institute for Logic, Language and Computation
Faculty of Science
University of Amsterdam
Science Park 907
1098 XH Amsterdam

June 25th, 2021

Acknowledgement

First of all I want to thank my supervisors Elia and Oscar and my secondary supervisors David and Pieter for their support. They pushed me to constantly improve the heading of this research, encourage critical thinking and they are just overall good people. Secondly, thanks to Celena for providing constant mental support during this period by studying together and keeping me going. Lastly, a big thanks to my mom and dad. They are always there to support me, and were curious about any progress during this period. Big thanks to you all!

Abstract

This thesis researches the object detection and avoidance performance of a driving model trained in simulation. This model will eventually be deployed in off-road scenarios which are harder to be trained due to a lack of available data, thus the training in simulation. To evaluate its performance both its vision and driving model were tested with various experiments in the real world and in simulation. These experiments were focused on finding scenarios where performance of either was lacking or unreliable. In the real world detection suffered under conditions with little available light, or when objects were occluded. The driving model also failed to avoid any objects in the real world, although tests in simulation showed mainly issues with estimating its own size.

Contents

Chapter 1

Introduction

In the field of self-driving, specifically using artificial intelligence, a lot of research has already been conducted. Tesla for example is even famous for their Autopilot software on their cars which they claim to be a safer driver than human drivers [1] [2]. These systems are however designed to work properly on highways and in cities, but do not support driving in off-road or rural scenarios. There are however a multiple of applications for these self-driving systems in off-road scenarios: these systems could be used for example to enable autonomous farming, rescuing humans in hard to reach areas after a natural disaster or bringing essential supplies to remote locations to name just a few. Which shows that it is of the utmost importance to develop models which can perform well in off-road scenarios.

In this field quite a lot of research has already been performed into training algorithms to be used for segmentation of different terrains and the mapping of multiple objects. But while there are a lot of available data sets with images from city and highway driving, which are useful for training models specialised in city driving, the collection of data sets usable for off-road driving is limited. One of the reasons for this is that most driving occurs on highways and in cities, so there are fewer vehicles which could be used for gathering data. And since off-road driving is also more difficult and often slower, reaching the same average speed, and thus the amount of miles covered in a similar time-span, is near impossible. This makes the training of algorithms and models a lot more challenging.

Due to the limited availability of real-world data that can be used for training, it is thus interesting to consider other methods for training the models and gathering data. Generating synthetic data, training models in simulation, adding random noise or variation to existing data sets or finding methods to increase training efficiency such that less data will be needed are just a few of the options. This research will be focused on training a model in simulation and then evaluating its real-world performance. This heading was chosen since using a simulated environment for training means that the only limiting factor in terms of data gathering is the amount of processing power.

[1]Relevant article: `https://futurism.com/the-byte/tesla-safety-autopilot-safer-human`
[2]Tesla safety claims: `https://www.tesla.com/VehicleSafetyReport`

This heading is also relevant for the company under which this thesis was conducted, Saivvy. Saivvy specialises in training models using computer vision and reinforcement learning to operate and control robots which will eventually be used in off-road scenarios. Before this research was conducted a simulated environment to train these models for object detection was already set up with the help of other students. This research aims to further that development by evaluating the real-world performance of these models by finding their shortcomings and differences in the real world, since training in simulation rarely transfers perfectly to the real world.

The goal of this thesis is to evaluate these shortcoming by deploying the trained models in the real world. These insights will then be used to improve the accuracy of the simulation and to determine methods to increase performance and reliability. Which leads to the main research question for this thesis: *Does an autonomous self-driving model trained in simulation perform to a similar degree in the real world, specifically in off-road scenarios?*

To evaluate this question properly, multiple parts of the model will need to be evaluated, such as the object detection model, driving model and the overall controlling system. Only when these separate parts have been evaluated, we can get an understanding of the entire model's performance. To evaluate these the following sub-questions will be answered:

- *Does the real world object detection perform similarly compared to the simulated environment?*
- *Is the driving model capable of navigating to the same accuracy in the real world?*
- *Can the control system deal sufficiently with the inconsistencies of the real world?*

The rest of this thesis will be dedicated to explaining the set-up of the rover, the technical difficulties that occurred during set-up, the experiments that were ran to evaluate performance, the metrics that were used to measure performance, and the results we gathered. This will enable these research questions to be answered, which will help evaluate in what areas the simulated environment needs to improve to increase real world performance.

Chapter 2

Related work

As mentioned a lot of research has already been conducted in the area of autonomous off-road driving. For this thesis, two areas of research are relevant. First off, research into algorithms for localisation and segmentation of objects for obstacle avoidance is important. Secondly, research into transferring findings from specific environments to different and unrelated areas is of relevance. Both are investigated since it is important to understand the underlying algorithms which need to be trained, and the processes needed to transfer findings to other environments. Some of the most relevant articles will be discussed in this chapter to get a better understanding of this field.

2.1 Object detection and localisation

Previous research by Choi et al. 2012 describes an algorithm to detect landmarks with cameras that can not be detected by LIDAR. These landmarks include speed bumps, crossings and lane markings. For detecting these obstacles the paper used a set of three different camera sensors. The image from these cameras was transformed to a top down view and processed using image filters to extract only relevant details, as pictured in figure 2.1. By utilising these cameras the researchers were able to create a robust system which performed well in avoiding obstacles along its path. This research can be used for this thesis to inspire possible algorithms for obstacle avoidance.

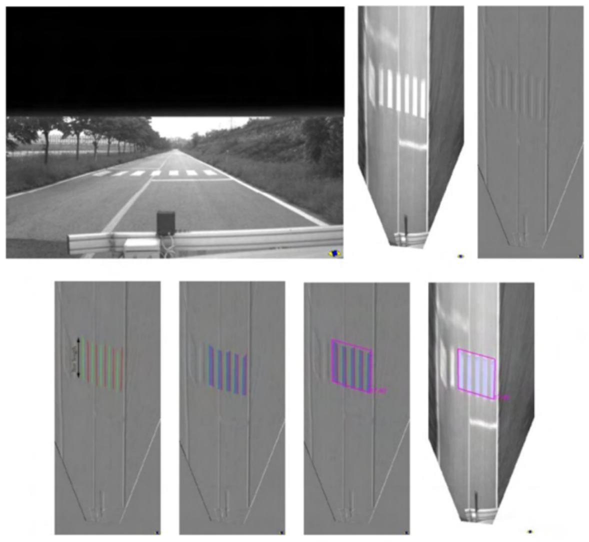


Figure 2.1: Landmark detection

In a paper by Blanke et al. 2012 an in-depth description of a controlling system deployed in a real orchard is provided, which is a similar environment to the one the robot eventually will have to navigate. This system is divided into multiple independent parts which communicate with each other. Dividing the system into multiple parts allows for self-regulation which improves the reliability of the system. There is also a controlling system which ensures higher-level control to enable path planning and obstacle avoidance. By choosing this design philosophy the robot can be controlled safely and efficiency while also allowing changes to individual parts. This design could prove useful for this thesis to allow for quick changes to certain systems of the rovers while still ensuring reliability.

Finally, in a paper by Bochkovskiy et al. 2020 another algorithm for object detection is proposed, called YOLOv4. This Convolutional Neural Network (CNN) is trained on the MS COCO data set which contains images of multiple classes of objects in a single picture. The model manages to outperform other object detection models on this same data set when taking both speed and accuracy into account, as pictured in figure 2.2. Although less accurate than some models, this model requires relatively little processing power to run, which allows for a higher frame rate, and thus a higher inference speed. These properties are crucial for autonomous driving since the rover is not that powerful, and the detection speed thus impacts the ability to make well timed decisions. This model was the inspiration for YOLOv5, which is deployed on this rover in this thesis.

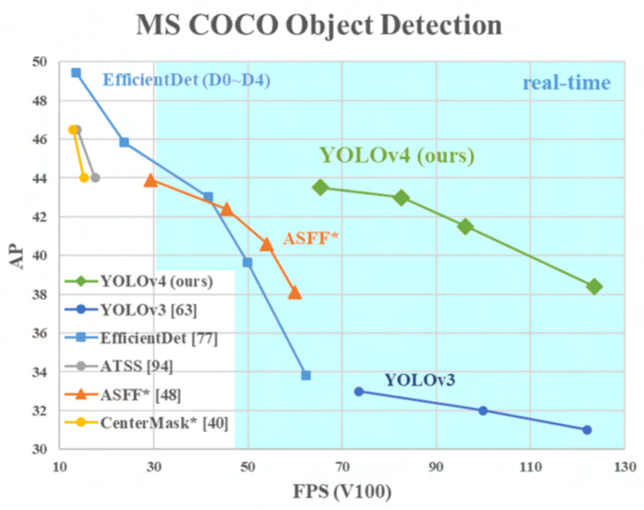


Figure 2.2: YOLOv4 performance

2.2 Transfer Learning

The paper by Sharma et al. 2019 is a relevant paper in the field of transfer learning, the second relevant field for this thesis. It describes a new way of training CNNs for semantic segmentation. Normally these neural networks are trained using a data set which is as closely related to the actual task as possible, ideally gathered from the real world. However, for this training data sets with irrelevant labels and synthetic data sets were used. The irrelevant labels were filtered out which allowed for a reduction in the size of the network, and the synthetic data was afterwards used to further train the network. The researchers also took inspiration from other well performing networks for the architecture of their own CNN. By combining all these techniques the researchers were able to achieve fast and accurate results in segmentation for off-road driving, as shown in figure 2.3. This research is relevant since it shows unique ways of achieving good performance while having access to sparse amounts of data.

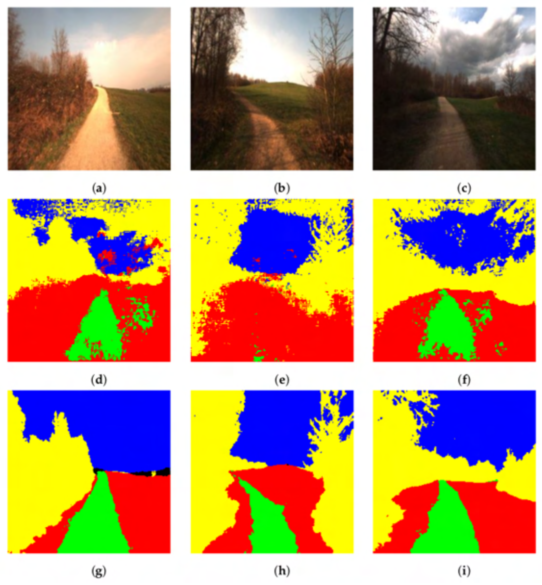


Figure 2.3: Evaluation of Learned model

Another paper by Osinski et al. 2020 describes the training of a Reinforcement Learning based approach in a simulated environment. For this simulation they used built-in environments for their engine and provided a goal which needed to be reached. Using this goal the driving policies could be trained completely in simulation. While for the training of semantic segmentation real and simulated images were used to enable diversified learning. Testing the trained model resulted in decent performance in reaching the goal, but it showed inaccurate or over-fitted results in some cases when the simulation was too simple. This was because small idiosyncrasies in the design of certain environments made for strange behaviour. They also observed worse performance in the real world compared to simulation, which was caused by an increased amount of noise. This paper is relevant since it shows that even though learning in simulation can be useful, there is still a need for real world training and testing.

Chapter 3

Technical Explanation

Since this thesis uses simulation programs, controlling systems, machine learning techniques and other technical terms, it is important that these are explained clearly. This section of the thesis will thus be dedicated to explaining the most relevant terms.

3.1 ROS

ROS is the main operating for the control system of the robot. This operating system allows for several components of the robot to operate independently. Each component or system is represented as a node, which functions as a small section of a network. Nodes can then subscribe or publish messages to certain topics, which allows these nodes to communicate with each other by sending and receiving messages. The set-up of this operating system is very similar to the system proposed Blanke et al. 2012, and thus allows for more independence, adaptability and the ability to properly detect errors in the program. An overview of the general control system of the rover is also shown in 8.3 to show the inspiration it took from this filosophy.

3.2 Rover

The off-road robot, or rover, used for this thesis is the R1 Autonomous Rover UGV Platform by Aion Robotics. This is a relatively small but capable all-wheel driven off-road skid-steer platform, which means no wheels can be angled for steering. Inside is an Nvidia control system tasked with the processing of sensor inputs and controlling movement. This robot can also be simulated within the Gazebo simulation environment so that a comparison can be made between simulated and real-world performance. Below in figure 3.1 the rover is pictured along with an external battery, the external computer and the RealSense camera.

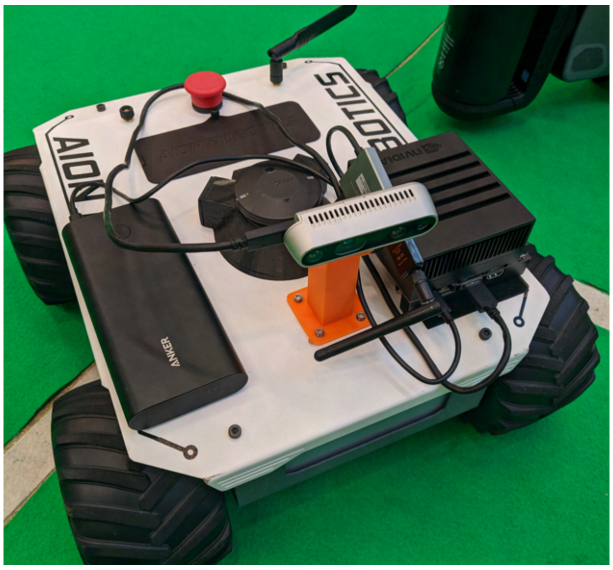


Figure 3.1: Picture of the rover

3.3 Gazebo

Gazebo is a simulation toolbox designed to allow easy and accurate simulation of robots within different environment. It uses physics simulations to accurately mimic movement of the robot. It also allows for the placing of multiple objects to test obstacle avoidance, while including multiple maps and environments built-in as well, which can be used to evaluate performance in different scenarios. One caveat with this simulator is that there is no direct way of simulating computer vision. The state machine uses topics which contain the position and size of each existing object. This means the rover is not restricted in its knowledge of the world, which makes an impact on the gap we are evaluating. Pictures of this environment will be shown in the experiments and results sections of this thesis.

3.4 RealSense

The camera system used on the rover is an Intel RealSense depth camera. This system uses multiple cameras to measure the depth of objects while also trying to understand their dimensions and shape. This system is well known for good performance and can be easily installed on the robot to extend its functionality. The system creates an output for the location and size of detected obstacles which is used as an input for the driving model. The system is mounted nearly at the front of the rover on a mount to ensure a higher viewing point for better overview, it can be seen in figure 3.1.

3.5 State machine

The state machine is is responsible for making decisions for the rover on where to go, what objects to avoid and how to reach its goal. To make these decisions, the model has a list of options (states) which it evaluates and chooses from, depending on the situation. To choose from these states it uses multiple different parameters. These determine at what distance it should start avoiding an obstacle, what speed it can use to do so, what distance to maintain while performing the manoeuvre, and when to make an emergency stop among other options.

While in motion, the model can perform a basic forward, backwards and idle movement, depending on the input of the user. Once it detects an object it should automatically start manoeuvring around it by entering an avoidance state when in a certain range. This distance is configurable as well as many other parameters such as the safety margin used when avoiding an obstacle. Once this move is completed it should then continue moving in the specified direction. If an object is however detected too late to start a manoeuvre, or no suitable path is found, the rover enters an emergency stopping state to protect it from running into an object. An overview of all states of this model is included in Appendix A. Overall the state machine allows for good configuration while remaining simple and efficient in its decision making.

3.6 YOLOv5

To detect obstacles in front of the rover the YOLOv5 [1] model is used, which was also trained on the same MS COCO [2] data set. This model is an improved version of the YOlOv4 model which was mentioned previously in Chapter 2. It is an even further improved version, mainly with regards to its performance and optimisation. By utilising the PyTorch framework it is possible to run the model on less powerful hardware. This is ideal for this application since including a massively powerful computer to process the visions would add weight, increase power consumption, and decrease range.

The implementation for this model is achieved by the creation of a vision node within ROS. This vision node receives the raw data of the RealSense camera and

[1]YOLOv5 model: `https://github.com/ultralytics/yolov5`

[2]Coco dataset: `https://cocodataset.org/`

scans it for any objects. Once an object is detected, it is marked with a bounding box, which describes the size of the detected object by a best fitting rectangle. This rectangle is then processed to give an output of the predicted size of the object and its location. Interestingly, the model can also be configured to include uncertain predictions or only include predictions with a high certainty by changing its sensitivity value. Normally this value is set to 50%, which was unchanged for these experiments to reduce the amount of incorrect detections, while also ensuring most objects are detected.

Chapter 4

Thesis Background

This thesis was performed, as mentioned before, under supervision of the Saivvy company. Since this project is quite intensive and demanding, it was performed by two students, Rhuben Vlugh and me. We were asked to work together to set-up a simulated and real-world test to evaluate the differences between these environments so the company can further improve its training for future models. This meant that we had to find a way to split tasks and create a personal heading for the project, while also having our research fit together. Our theses were however written completely independently, including the interpretation of results. We also ended up splitting our efforts mostly between the two different environments, where I focused my efforts on the real world, Rhuben focused on the simulation.

The next section will highlight the progress that was made on the hardware and software side of the rover, and other tasks that were performed. These achievements might normally not be mentioned, but they highlight the difficulties that were overcome to set-up the rover and actually achieve desired performance. While setting up the Rover I also took the responsibility of organising the weekly meeting between the supervisors and students, suggested changes to the internal documentation of the company and tested the impact of changes in the code. Working with hardware provides difficulties which would not show up when doing only software development, or when training and performing experiments in simulation. These difficulties originate from the fact that unexpected behaviour can both be caused by hardware or software errors which are often hard to diagnose, costing significantly more time.

4.1 Software

First of all it was important to understand the software side of this project. Since this project was performed with help from Saivvy, quite a lot of development and thinking had already been performed to make the controlling of the robot possible. As mentioned above, ROS is the control system that is used to enable controlling the rover using software while also reading data from its sensors. To get an actual understanding of how this system functions, multiple tutorials [1] were followed to grasp the different relevant terms and properties of this system.

Once a thorough understanding of ROS was achieved, Docker images needed to be unserstood. Docker is a way of running applications in a portable and self sufficient environment, which makes them function independently from the computer they are run on. This allows for changes in the code of the applications to be immediately implemented on multiple systems, while also ensuring that all dependencies are actually installed. To use Docker it is needed to actually understand the management of files and the function of images, which are basically the packages containing all relevant applications. Since these are separate from the main operating system, making changes to code is also a bit more involved.

After understanding the operation of both of these concepts, Gazebo was the next relevant piece of software to understand and familiarise with, even though Rhuben eventually was the main user of this software. As mentioned above, Gazebo is the simulation environment where experiments can be performed with the rover. To run these experiments, objects need to be placed in the environment and the robot needs to receive instructions in a consistent manner. Gazebo also simulates the sensors of the rover which means that this environment can be used to create ways of measuring and storing the path which the rover travelled, which is useful to evaluate its performance. The environment itself is pictured below in figure 4.1.

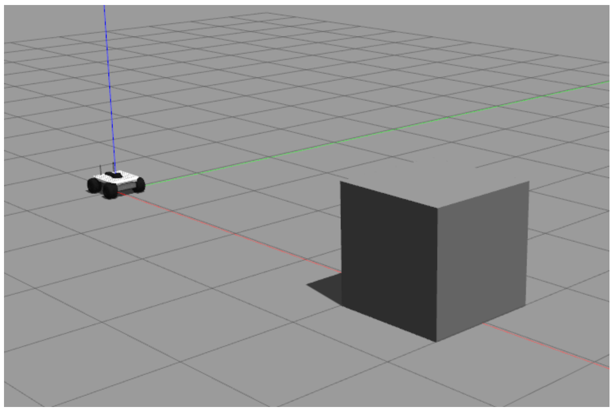


Figure 4.1: Overview of Gazebo Simulator

The last part of the software configuration that needed to be investigated was the detection of the current turning angle of the rover. To allow the state machine

[1]Ros tutorials: `https://wiki.ros.org/ROS/Tutorials`

to properly make decisions it needs information about its current rotation in terms of position, and rotation. For the rotation it needs information specifically about its current yaw, rotation over the y-axis, to determine its current heading. While Gazebo provides this information for every object, in the real world other sensors need to be used. The task was thus to figure out what sensors could provide accurate information such that navigation around obstacles was possible. After trying topics from the internal accelerometer, GPS compass and an estimated orientation based on previous movements, we were unable to get an estimation of the yaw. This means the rover is not able to navigate around objects or change its direction, which impacted the experiments directly.

4.2 Hardware

After covering and creating an understanding of the software side of the rover and this project, there were also quite a lot of hardware changes that needed to be made. The rover itself comes installed with an Nvidia module, called the TX2, which is responsible for actually running all of the software needed for control, while also processing data from the sensors. While it is directly integrated into the rover and links to other components, it is not powerful enough to perform all processing related to computer vision, and thus needed to be upgraded to a Xavier module. Since the TX2 component is built-in, it took quite a lot of thinking to figure out a way to connect the motors, controllers and camera to the new module. They all connected with strange connectors and often didn't directly communicate which meant figuring out if this was due to a software or hardware problem.

The first issue that was encountered was the communication between the Rover's built-in controller and the Nvidia module. The built-in controller, a Pixhawk PX4, is responsible for sending commands to the wheels and reading from their sensors, while also sending information about the GPS, Wheel sensors and other sensors back to the module. This module communicates to the Nvidia modules over what is called the `FCU_URL`, which specifies what port is used for communication and at what speed. The TX2 is directly connected to the PX4 through a serial port directly on its board. This port is not available on the Xavier so it needed to be converted to a USB connector, which meant figuring out the wiring used for this connector and switching to USB. This connection is shown below in figure 4.2 for context. After that it was possible to verify what `FCU_URL` could be used to enable communication, which could be found after some online research. This allowed the Xavier to gather updates about every sensor and send commands to the motors.

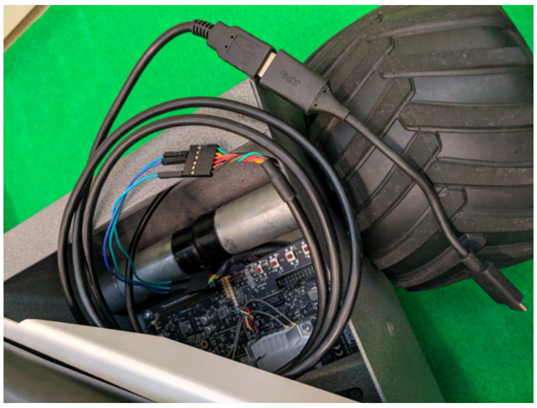


Figure 4.2: Connector conversion to USB

After enabling this communication the focus was shifted on figuring out how to fake a GPS location. The rover uses GPS to constantly check its location and account for any variations. The problem is that the rover needs a solid GPS lock before the it is actually allowed to do any sort of driving, which can not be disabled. In bad weather conditions the Rover can not go outside since multiple parts are not weather proof, so finding a way to use the Rover indoors was deemed worthwile. There is a built in ROS node that uses the internal sensors of the rover to estimate a position and print the position as if coming from the GPS. A few days were spent unsuccessfully trying to enable this node.

Since it turned out not to be possible to enable this node, focus shifted to enabling control in outdoor environments. Even though the ability to communicate with the built-in controller through the USB port on the Xavier now existed, it turned out that it was not yet possible to actually send driving commands, although all other functions of the PX4 were available and the responses were identical when compared to the TX2. The error code seemed to indicate a software error since communication with the motors was reported to be impossible. But after a week of trial and error with no results, the motors turned out to be powered only when a separate battery on the rover was enabled. A simple solution which was difficult to find since all other systems powered on perfectly using the power from the Xavier.

Chapter 5

Methodology

As mentioned before, this thesis is aimed at evaluating the gap between a model trained in simulation and its real world performance. To evaluate this gap, multiple experiments will need to be performed for evaluation of the model in both environments. Due to the rover not being able to navigate around objects, no direct comparisons could be performed between the rover and the simulation. The simulation is also not capable of simulating the actual output of the vision model as mentioned before which further restricts the direct comparisons that can be made. Since the simulator is however capable of navigating around obstacles, it can be used as an example of the performance that is actually desired.

To make the testing results from both environments relevant for further development, it is focused on pushing both environments to their respective limits. This means that scenarios were developed where we can evaluate current performance and also evaluate edge cases where performance is not expected. For the simulation this included small gaps and complex environments to evaluate performance. While in the real world performance the following challenges were evaluated: vision under different lighting, avoidance with multiple objects, detection of objects with multiple sizes and performance with partially occluded objects. The next section will highlight why these scenarios were chosen and how performance was measured.

5.1 Simulation Performance

First off, we want to challenge the simulated environment in multiple ways. Since this functions as a demonstration of the desired behaviour we will create multiple scenarios to evaluate its performance. These scenarios will be varied in terms of configuration and the amount of objects to avoid. In these scenarios the size of objects is changed, the robot is tasked with avoiding multiple objects, and it is tasked with evaluating whether it can fit in between a small pathway. Using these scenarios we want to measure and evaluate the path the rover takes to avoid an object, and the amount of safety margin the rover takes.

The consistency of the state machine in making choices about what direction to take and when to start avoiding the obstacle will also be evaluated. This consistency needs to be evaluated since it will show whether the model itself has some randomisation in its decision making, or if any random behaviour is caused by external factors when evaluating the real world. The actual set-up of these scenarios will be explained with the experiments.

5.2 Computer vision

The next part of the rover we want to evaluate is the computer vision model, YOLOv5, which is used to detect objects. The output of this model mainly influences the decision making of the state machine and thus needs to show consistency in terms of detection in multiple scenarios. For this thesis the model will be challenged with different lighting conditions, multiple objects in one scene, different sizes of objects and the occlusion of objects by others. The next section will highlight the importance of each of these and the metrics used for evaluation.

5.2.1 Lighting conditions

When using an autonomous driving system, it needs to be capable of functioning in light and dark conditions. This is mostly relevant for night time operation since vision is very limited in those cases, but these systems could also be deployed in dark forests or weather conditions which limit vision. To properly evaluate the impact of lighting conditions on the detection of objects, the vision will be tested from a stationary position, for a period of 1 minute. During this minute an object will be placed in front of the sensor and the amount of frames where the object was properly detected will be counted for evaluation. The number of detected frames is divided by the total amount of detected frames to produce a detection rate, which evaluates the performance of this model. This test will be performed in lighting conditions ranging from being outside during daytime, to a dimly lit room, and nighttime conditions.

5.2.2 Occlusion

When using the previously described model to detect obstacles, it is also important to evaluate the ability to detect objects which are partly covered by others. This can both help in creating an understanding of an environment to ensure proper navigation, as well as in reacting in a timely manner, since an object is already detected before the avoidance of another obstacle is for example performed. To measure the detecting accuracy of the model when part of an obstacle is occluded, all observations over a period of 1 minute were once again stored. It was then counted how often the partially occluded object was detected during this period to create a detection rate. To evaluate multiple scenarios, the objects were occluded for 10, 30 and 50 percent.

5.2.3 Multiple Objects

Another important factor for evaluating the performance of the model is the ability to separately detect multiple objects. If the model had a limit on the amount of objects it could detect, it could possibly run into obstacles or create dangerous situations. Thus we want to evaluate how the model deals with multiple objects. To evaluate this, a varying amount of obstacles were placed in front of the sensors of the robot. During a period of 1 minute, the detection rate was once again measured to evaluate performance in this scenario.

5.2.4 Inference rate

The last part of the computer vision model on its own that was evaluated was the average publishing rate for detected objects. This rate is important since a low amount of updates means that the model can also evaluate its choices at a lower rate, since it depends on new information being available. If the amount of updates per second (Hertz or Hz) is low, this means that the model will take longer to react to changes and it might lead to worse understanding of the scene. The main impact on this rate is the amount of detected objects, which was thus the variable for this method. ROS has a built-in command which measures the publishing rate on a certain topic, in this case the object detection topic, that was used for this evaluation. Over a period of 1 minute this produced an average update rate, and a standard deviation of this update rate. The same test was also performed in the simulation to give a comparison on the relative performance gap.

5.3 Real world

To evaluate the performance of the state machine when placed in a real world situation, multiple different scenarios were created. The main focus was on evaluating the impact that the distance of an object, the size of an object, and the amount of objects made on the ability of the rover to stop at a safe distance from the objects. This metric was chosen since it was, as mentioned above, not possible to actually navigate around objects in the real world. To further enhance the depth of the testing we also chose to measure the distance at which the objects were actually detected. Due to a lack of time because of difficulties with setting up the rover, each of the following scenarios were tested 3 times. This ensured that some average was actually measured, while also allowing a variety of scenarios to be tested. The following section will explain what variables were tested in these scenarios.

5.3.1 Distance

The first variable that was tested is the impact that the initial distance of the object to the rover makes. The thinking is that first of all objects should be detected from a further distance. But when an object is further away, the rover is also afforded more time to think and predict whether it will collide with said object.

5.3.2 Size

The second variable that was tested is the impact of an object's size for detection and avoidance. Larger object might be easier to detected, are more likely to be positioned on a collision course for the rover, and can also possibly be detected from further away. Once again the distance till detection was evaluated, while also noting any peculiar behaviour during the test.

5.3.3 Amount of objects

The last variable that was tested was the impact that placing 4 different objects in front of the rover instead of 1 would make. The ideas was that, when placed closely together, at least one of the obstacles would be guaranteed to be in a direct path of the rover. These obstacles were placed closely together to ensure that there would not be a gap between them that the robot might consider to be large enough to fit through. Once again performance in detection and avoidance was evaluated in the same way as the other variables.

Chapter 6

Experiments

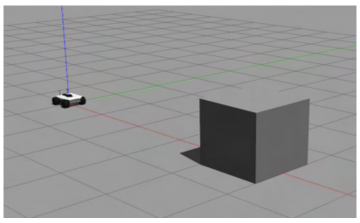
To measure the performance of the rover in each of the scenarios which were mentioned above, multiple experiments had to be performed in a consistent and repeatable manner. In the simulation this was quite easy to achieve since a scene can be reloaded such that all obstacles are placed at the exact same starting location, which ensures they can be repeated. This made experimenting quick and easy, and not easily influenced by other conditions.

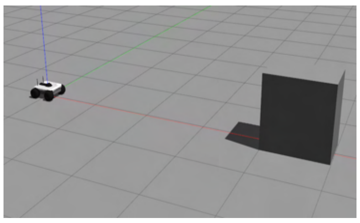
When testing in the real world this repeatability can not be achieved so easily. One of the main factors which impacted this was the sensitivity of the GPS sensor on the rover. This sensor is highly sensitive and needs direct line of sight with multiple satellites to ensure accurate navigation and control. Unfortunately some locations worked at one moment, and were too obstructed the next day. To account for this variation, the experiments were always ran on the same type of pavement, at the Science Park. The camera was also always angled down at 15 degrees such that mostly the objects directly in front of the rover occupied its vision.

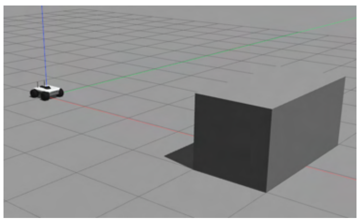
Another challenge was finding suitable objects for the rover to detect and possibly collide with. In the end mostly plastic bottles, filled partly with water were used. These were stable enough to stay standing, are a trained class in the YOLOv5 model, and wouldn't damage the rover when a collision did occur. A 500ml bottle was used as the baseline object for most testing. While in the simulation unit boxes, square or rectangular boxes with a standard size of one unit, since they were the only enabled object that would actually be avoided by the state machine. Using this object reduces the complexity of testing since the dimensions can easily be changed.

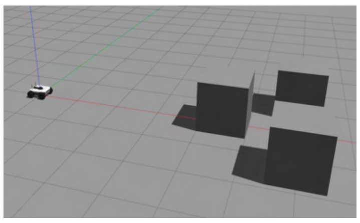
6.1 Simulation

For testing in the simulation, 5 scenarios were evaluated. In figure 6.1 below these scenarios are shown. Scenario a) evaluates the performance of the rover in the most simple case. A cube unit box with size 1 was placed in front of the rover. This scenario tests its ability to detect this object and efficiently navigate around it. Scenarios b) and c) vary this test slightly by increasing and decreasing the width of the object ahead of the rover. Scenario d) places three objects in front of the rover such that it has to evade two objects in a row, and Scenario e) evaluates the ability of the rover to take its own size into account for fitting between a gap. As mentioned before, each scenarios was ran three time to evaluate consistency, and the rover always starts at a distance of 4 units from the obstacles.


(a) Unit box


(b) Slim box


(c) Wide box


(d) Multiple boxes

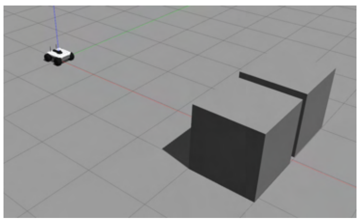

(e) Small gap

Figure 6.1: Simulation testing scenarios

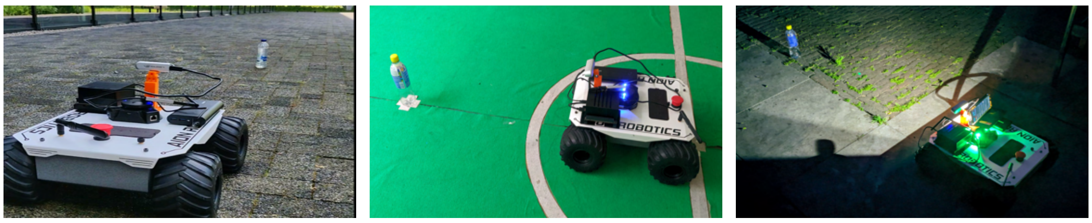
6.2 Computer vision

6.2.1 Lighting conditions

To evaluate the performance of the rover in different lighting conditions, multiple scenarios were considered. In each of these scenarios the amount of available light was tested with the use of a cellphone. Using the sensor on a phone, the level of luminance can be measured. This detection is for a very specific area. Thus the phone was placed directly in front of the rover and pointed at the object to be detected. Then an average value for the light availability, measured in lux, was measured over a period of 15 seconds. For these tests the rover was placed at a distance of 0.5, 0.75 and 1 meters. Since these are the kind of distance where detection is crucial to still ensure a possibility to a avoid the obstacle at low speed.

To establish a baseline, the rover was first tested outside on a day with consistent but light cloud coverage. The light level was measured to be around 400 lux which should be more than enough. The test was performed on pavement with a neutral background to ensure little interference, pictured in picture 6.2a. To add to this experiment the rover was tested next in the RoboLab of the University with the lights switched off to create a lighting value of around 100 lux for the objects. In terms of positioning, the rover was facing a white wall with little pattern to ensure little impact. This situation is pictured in picture 6.2b.

To increase the difficulty even more the rover was also tested in an outside environment in the evening at dusk. In these scenarios the lux was reported to be around 0, which was a clear shortcoming of the sensor since dusk should still be around 1-10 lux. Thus as a final experiment a phone was placed on top of the rover behind the camera with its flashlight enabled, pictured in image 6.2c. A phone was chosen since it highlight a wide area, where a normal flashlight focuses the light to a smaller area. Since this produced inconsistent amounts of light, the light level needed to be measured at each distance. From 0.5 meters, the light level was around 40 lumens, 30 lumens were measured from 0.75 m, and at 1 meter the level was around 25 lumen.



(a) Daytime (b) Dim room (c) Dusk with flashlight

Figure 6.2: Three different lighting conditions

6.2.2 Occlusion

To measure the performance when objects were occluded, as mentioned before, one object was placed partly behind the other. These experiments were also ran in the RoboLab with full lighting since no GPS was needed and performance was consistent in this environment. To ensure consistent location and distance between all experiments, the robot was placed on the middle of the football pitch, with the objects on a white patch 1 meter in front of the rover. Once again plastic bottles were used since these could be detected consistently and were easily available. The percentage of occlusion was calculated by measuring the width of the object and dividing this by the width of the covered area. Pictured below in figure 6.3 is a cropped image of the rover's view for each of these experiments.

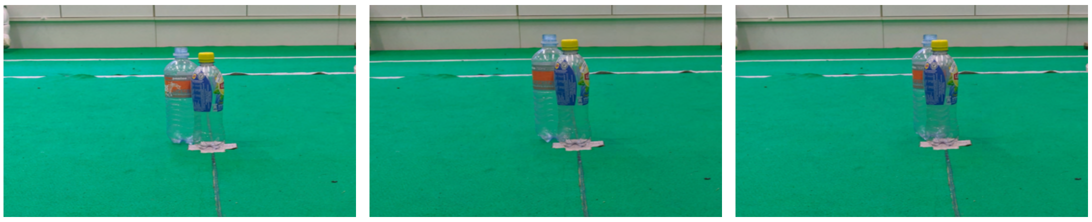

(a) 10% (b) 30% (c) 50%

Figure 6.3: Three different levels of occlusion

6.2.3 Multiple Objects

To evaluate the performance in differentiating multiple objects, the rover was once again placed in the RoboLab with the lights on. The objects were all placed on the same location as described above. After the baseline of one object and its detection rate was established, two, three and five bottles in total were placed in front of the rover. The picture below, in figure 6.4 was taken by a phone directly above the real camera of the rover, so it is not completely representative but illustrates the view of the rover quite well.

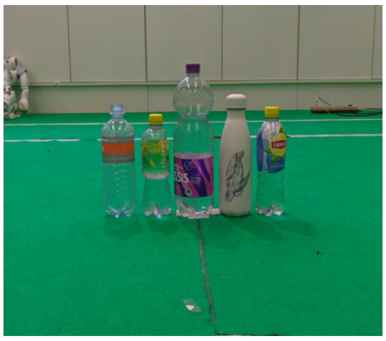


Figure 6.4: 5 objects in front of rover

6.2.4 Inference rate

While evaluating the ability of the rover to deal with multiple objects, the same environment and set-up was also used to evaluate the inference rate, as described before. From the same location a baseline was established with no objects in front, then one, two and three bottles were added to evaluate the impact these made on performance. In the simulation one unit box was placed in front of the rover, and the inference rate was measured in the same way as the real world.

6.3 Real world

To evaluate the performance of the rover in the real world, a baseline was first established. This meant using the same plastic bottle as used in previous experiments. This bottle was placed at a distance of 2 meters from the rover on a cloudy day. This distance was chosen since most objects could be decently detected from this distance, while there was still enough distance to make a decision and evaluate performance of the rover. The rover was situated on the same pavement at Science Park and was tasked to drive towards the bottle. This driving speed for each experiment was capped at 0.3 of the max speed of the rover, which equalled around 2 kph. This speed was chosen to allow the rover enough time to detect objects, while also not being unrealistically slow.
While experimenting, the distance where detection of the object occurred in real life was estimated by looking at the detection location and marking the spot to measure the distance to the bottle. Since this is an inaccurate way of measuring, these serve more as an indication rather than an accurate measurement, there was unfortunately no time to implement a stopping command once an object was detected to accurately measure this distance in the real world. 4 experiments were ran since in one of the the vision failed to recognise the bottle, this run was eliminated to create a more consistent baseline. The testing set-up is also pictured below in figure 6.5.

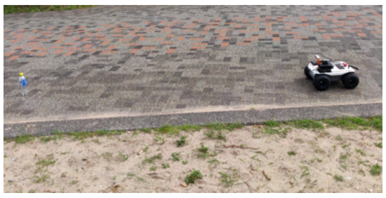


Figure 6.5: Baseline experiment

6.3.1 Distance

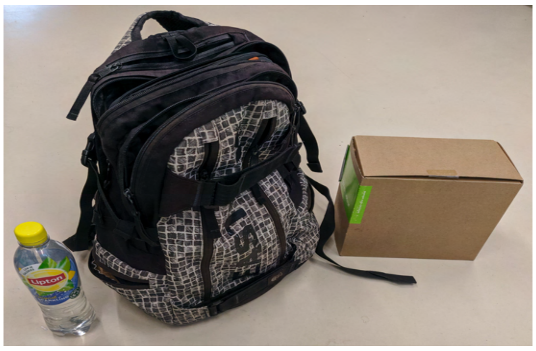
After establishing this baseline, the rover was placed at the same location but at a distance of 5 meters from the bottle. The rest of the experiment was unchanged but the bottle needed to be placed slightly to the right of the rover since its path was not directly straight. This did not seem to be caused by any incline since the behaviour was consistent throughout other experiments. The placing of the object means that it might not be considered to be in the collision path from the start since it is not directly in front.

6.3.2 Size

To evaluate the performance of the rover when objects of different sizes are placed in front, two objects were used. The first object was a plastic bottle of 1.5 litres, used to increase the footprint and the chances of the object being detected as: "In the way." Once again this object was placed at two meters distance from the rover although at a slightly different location. Once again on the pavement but the GPS was not cooperating so the tests were performed about 50 metres away from the previous experiments. After the bottle was placed, a backpack, which was around 50 by 30 cm in front profile, was also placed in front of the rover. The class of this object was not really detected by the vision model but its position and size seemed accurate, so it was used to experiment with substantially larger objects. Once again placed at two meters distance, and this bag compared to the normal bottle are shown in figure 6.6a below.

6.3.3 Amount of objects

To finally evaluate the performance of the rover when driving towards multiple objects, one extra scenario was considered. In this scenarios 4 different bottles were placed in front of the rover at 2 meters distance. The rover was then driven towards these obstacles, once again outside on a cloudy day. When the rover came within touching distance of these objects, they basically were directly in front of the rover. Below, in figure 6.6b, is a picture of these obstacles from the viewpoint of the rover.


(a) Different sized objects

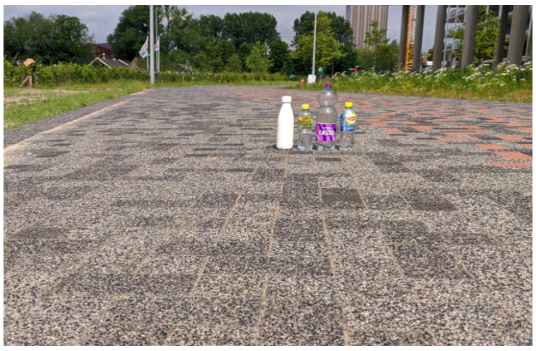

(b) Multiple objects outside

Figure 6.6

6.4 Miscellaneous

While running these aforementioned experiments, and while setting up the rover some other miscellaneous experiments were also ran. These experiments were not in-depth enough to be scientifically significant since they were often only ran once, and not measured to the same accuracy as other experiments. They are however worth highlighting since they give some extra insights into the functioning of the vision model and the performance of the state machine in the real world.

The first experiment that was performed when the wheels could actually be turned was to manually instruct the rover to turn around its y axis on pavement and in the grass while specifying different turning rates. These experiments were ran to evaluate if different surfaces impacted the ability of the rover to complete a turn, while also finding the minimum amount of power needed to perform a turn.

The second experiment that was performed was the evaluation of the model to deal with other objects than it was trained for. A cardboard box, pictured in figure 6.6b, was used for this experiment. It was originally meant to represent an object of medium size for evaluating the impact of size, but was eliminated for reasons specified in the Results.

6.5 Expectations

6.5.1 Simulation

In terms of expectations for the experiments, the most consistent results are expected to be shown in the simulated environment. In this environment the rover should for example be able to avoid any obstacle perfectly fine when that obstacle is placed directly in front, since it should not be impacted by possible detection flaws. When dealing with multiple objects however we expect it to detect the object in front, start avoiding and after this avoidance return to a regular driving mode, while it has neared the next obstacle too close. Thus it is expected that the rover will perform an emergency stop.

6.5.2 Computer vision

When evaluating the computer vision model, the different lighting conditions are expected to make the biggest impact on performance. In the dark, there is a lot less light to be used for the sensors, and a lot more noise to make an impact on the detection model. The inference rate of the rover is also expected to suffer when multiple objects are placed in front, since more calculations will need to be performed. When looking at multiple objects at the same time however, detection should still remain accurate since the model is trained to detect multiple objects in a single image. On the contrary, when detecting occluded objects the performance is expected to suffer, especially when a large percentage of the object is covered. When an object is covered it shows a lower amount of significant features and the shape differs, which should make detection more difficult.

6.5.3 Real world

For the experiments in the real world, the predictions are a bit harder to make since there are many variables. We do however expect the best performance to be shown when multiple objects are placed in front. Since the vision model should be able to handle this, the multiple objects have a greater chance of being in the way of the rover such that it needs to make a stop. Bigger objects should also show good performance, but they might not completely cover the field of view when at a close distance, which might result in failed detection. Detection from a distance should not make much of an impact, since detection should still be fine from a closer distance if the rover fails for some reason.

Chapter 7

Results

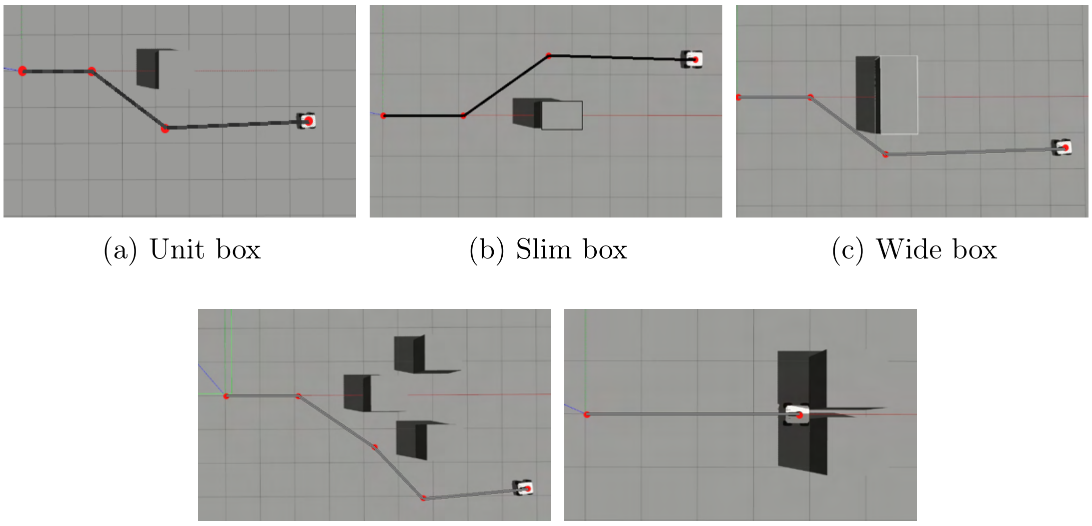
For this results section, the performance metrics of each of the experiments are shown below in their respective graphs and tables. These provide numeric insights in the results of the performance, but there will also be descriptions of the discoveries about behaviour during some of the experiments. This is done since numbers don't always show this behaviour, or because some behaviour was hard to measure in this way. All results for the simulation side were provided by Rhuben, while I provided him with my data from the real world.

7.1 Simulation

When testing in the simulation, as expected the results are very consistent. During 3 runs the performance of the rover was even identical enough that only one line trace had to be drawn in the top down view, shown below in figure 7.1 to represent the path the rover had taken. No metrics were measured by Rhuben to see at what distance, but the top down views provided below of each experiment do show a grid on the floor. Each cell in this grid has a size of 1 unit by 1 unit, so they can be used to roughly estimate the distance where an avoiding manoeuvre was started, and the safety margin during this manoeuvre. Both were estimated manually and shown in table 7.1, with their respective mean and standard deviation. Experiment e was excluded for both of these measures since the rover was unable to avoid the obstacles, thus scoring 0.

Label	Distance (unit)	Safety margin (unit)
a.	1.94	0.7
b.	2.10	0.87
c.	2.50	0.35
d.	1.91	0.58
e.	0	0
Mean	**2.11**	**0.63**
Standard deviation	**0.24**	**0.19**

Table 7.1: Manually estimated results from simulation



(a) Unit box (b) Slim box (c) Wide box

(d) Multiple boxes (e) Small gap

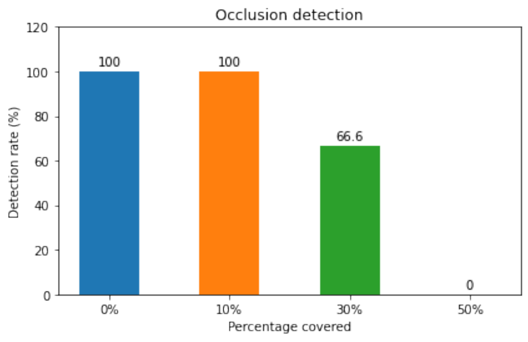
Figure 7.1: Top down view of simulation experiments with visualised path

7.2 Computer Vision

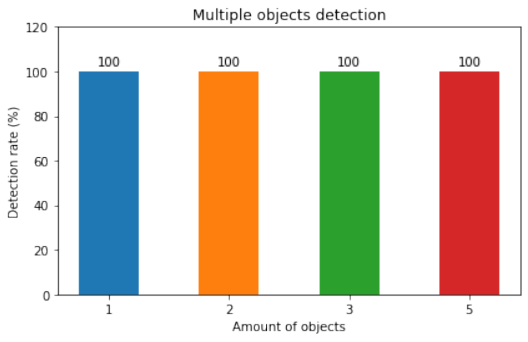
The results for testing the computer vision are summarised below in the following graphs. Each experiments is explained by their individual graph. First off, table 7.2 shows a decrease in inference rate for every added object. This inference rate of itself is however also quite low considering the rate is at 1000 Hz in the simulated environment, no matter the amount of objects. The graph after this table shows the results from the occlusion testing. In figure 7.2a a drop in performance is shown once 30% of the object is occluded. While the results from detecting multiple objects are shown in figure 7.2b, where no performance drop was measured. The results from the the detection in different lighting conditions are shown in figure 7.3, where the best performance overall occurs at a distance of 75 cm, with a distance of 1 m showing second best results.

Amount of objects	Inference rate (Hz)	Standard deviation (σ)
0	0.88	0.20
1	0.76	0.22
2	0.68	0.15
3	0.60	0.21
1 in Gazebo	1000	0.01

Table 7.2: Inference rate in real world and simulation



(a) Occlusion detection



(b) Multiple objects detection

Figure 7.2

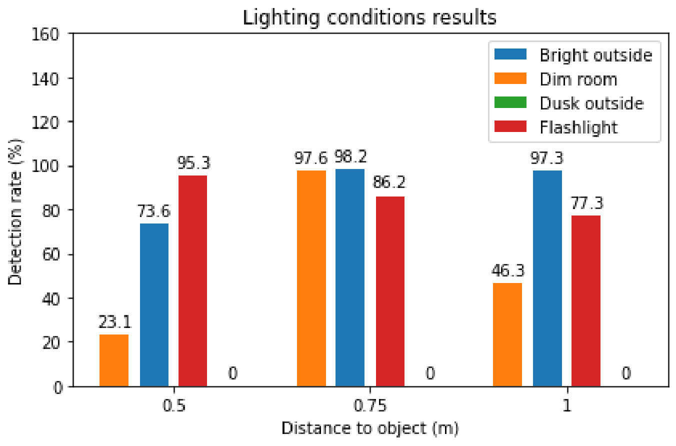


Figure 7.3: Performance in varying lighting conditions

7.3 Real world

Performance during the real world testing, the average results of the 3 runs in each experiment are shown in table 7.3. The measured distances until detection do not represent actual meters, but are distances for the internal coordinate system of the rover, which are identical to the coordinates used in Gazebo. A differentiation was also made between the distances for the first detection, and the correct classification. The distance for first detection evaluates at what distance the rover realises something is ahead of him, even though not knowing what that is. While the correct class distance measures when the correct class of the object was actually detected.

The results of these experiments seem to be pretty consistent in terms of detection distance for most objects, while for some reason the multiple objects were identified correctly almost at the start of the experiment, whereas for other experiments correct classification only occurred from a distance of around 1.5m. The bag was also not classified correctly at all. The standard deviation for this correct identification was however more consistent overall than the standard deviation for first detection. Which shows there are pretty consistent points where the rover actually identifies an object. To explain these sections a bit further a description is provided for the testing in each scenario, where any noteworthy discoveries of behaviour is also discussed.

Experiment	First detection (unit)	Std. dev. (σ)	Correct class (unit)	Std. dev.(σ)
Baseline	3.52	0.42	1.64	0.29
Big bottle	3.31	0.25	1.67	0.11
Bag	3.3	0.24	-	-
Long distance	5.04	0.48	1.9	0.03
Multiple objects	2.7	0.14	2.7	0.14

Table 7.3: Real world testing results

7.3.1 Baseline experiments

When performing the baseline experiments correct classification of the bottle occurred quite consistently at a distance of around 1 meters, with one run even resulting in detection from a distance of 1.3 meters. The rover did however not make any attempts to slow down for these objects, and collided during all three experiments, while the bottles were basically straight ahead. One earlier run which showed identical behaviour has been uploaded to youtube [1] so this is more clear.

7.3.2 Size experiments

During the experimentation with objects of different sizes the behaviour was pretty similar when the larger bottle was placed in front. Even though it was immediately detected, although not classified, in 2 of the 3 runs, the robot failed to evade the object. It did in fact not even slow down on any of these runs. A video of one of these collisions is also available on youtube [2]. When the bag was placed in front of the rover it was also immediately detected, but it was not correctly identified and was also hit in all experiments.

[1] Baseline collision: `https://youtu.be/L41ZwVZY7pY`

[2] Big bottle collision: `https://youtu.be/wgnBrZIwKyE`

7.3.3 Long distance

For the experimentation with a longer detection distance, more interesting behaviour was shown. The object was detected from a further distance than the other experiments, since it was coming from a further distance. Correct classification however occurred a little earlier. While closing in the rover was also slowed down during one of the runs since it detected an extra unrecognisable object, which resulted in a slower approach. Unfortunately a hit was still registered in each run.

7.3.4 Multiple objects

While experimenting with multiple objects in front of the rover, detection was consistent and occurred from a further distance than other experiments. While it took longer for the bottles to be detected, the classification was correct once they were. This did unfortunately not result in any avoidance or slowing down of the rover.

7.4 Miscellaneous

During the other experiments that were not considered for actual evaluation, one of the findings was that the cardboard box, as pictured in 6.6a, could not be detected whatsoever at multiple distances. This hints to the limitations of the COCO data set. Furthermore, when manually instructed to start turning, the turning speed of the rover was highly variable. This behaviour was especially noticeable on a grass surface as can be seen in this youtube video [3]. There was no easy way to measure the turning speed however, so no further research was performed.

[3]Inconsistent turning speed: `https://youtu.be/9otK7wCtpoM`

Chapter 8

Evaluation

8.1 Discussion of results

8.1.1 Object avoidance

When evaluating the performance of the rover, especially in the real world, performance was a bit disappointing. The expectation was that some scenarios, especially with bigger objects would result in an avoiding or a stopping response from the rover. Given that the detection of objects was actually pretty consistent with an accurate prediction of their location, these expectations were thought to be confirmed. However, this did not happen, which makes it seem like there is some sort of configuration error in the state machine which causes errors in avoidance or slowing down.

To explain further, the state machine actually uses a different module which calculates the projected path of the rover and sees if any obstacles are in its path. Since in simulation this same module also sent the rover on a path between two obstacles where it was not actually able to fit, this might indicate an error in this module. Where it only considers one line, instead of its actual width. This could also explain the difference between real and sim, since in the simulation it is sure of the positions of each object which allows for good decision making. In the real world less information is available due to the lower inference rate, and this information is also less consistent due to small variations in the vision model. This might mean the rover is not sure enough where the detected object is and when it will collide, resulting in no avoiding action.

In other scenarios where the rover should have more of a chance to detect the fact that an obstacle was in its way, especially with larger or multiple objects, performance was not any better. The inability to correctly classify the bag or to detect the box at all shows that the vision model is also not sensitive enough to untrained classes of objects, which could cause more problems in the real world. When starting at a distance performance was also not that much better, which shows the limited range of the RealSense system. The most probable explanation is a lack of detail on the sensor to discern features of an object, but there could also be other reasons for this lack in performance.

The simulation on its own did however show good performance in all other scenarios, simple objects were easily avoided. And contrary to expectations, the model also performed well in accounting for the objects behind the first object and avoiding them as well. This calculation did however only seem to happen once the first avoidance manoeuvre was completed, since there are two clearly different paths, instead of one more optimal path to avoid both. The distance where the avoiding manoeuvre was started also seemed to vary, as well as the safety margin away from any object. This variation could however also come from the measuring methods used to calculate this distance since counting pixels from an image is not always accurate. Further testing would be needed.

8.1.2 Computer Vision

The evaluation of the computer vision lines up pretty well with the expectations that were described. The performance and accuracy of the model suffered mostly when the availability of light was lowered. In the lab and outside with good lighting, performance was consistent. But the dim room was already enough to dramatically decrease performance, and performance at dusk was lacking to the point that a flashlight needed to be used. Interestingly this flashlight helped significantly at close distances, which can possibly be explained by the fact that this configuration basically only lit the bottle, and little other background objects. The camera could thus theoretically adjust so that only the lit object was visible. At further distances this performance was lacking however. So either a bright light would need to be installed, or possibly an infrared camera if night performance is to be improved.

The dropping inference rate when multiple objects were detected was also pretty much as expected. It shows a limitation in either the processing power of the processor, or a possible need for further optimisation of the code. A higher inference rate could help in improving object detection and possibly in avoiding obstacles.

In terms of the actual detection of multiple objects the model did show good performance. As mentioned, it was trained to deal with multiple objects and thus this observation was not surprising. This does however mean that as long as the other components and the state machine are also capable of dealing with multiple objects, the rover should not be quickly confused in more complex environments.

Lastly, the performance when evaluating the ability of the rover when dealing with occluded objects met the expectations. The significant drop-off in performance was however not quite the desired result. It would definitely be a benefit for performance and reliability if the rover was able to actually detect partly occluded objects. When a navigation manoeuvre is performed it is important to get an accurate understanding of the size of the object. Otherwise errors could be made since an object is not detected. Possibly this error is however not that severe since during this manoeuvre more of the occluded object also becomes visible, which creates higher chances of detection.

8.1.3 Sub-questions

Using the discoveries of this thesis, it is now possible to evaluate each of the sub-questions that were posed for this research. First off: *"Does the real world object detection perform similarly compared to the simulated environment?"* While the object detection is also relatively consistent when evaluated in ideal conditions, in most scenarios the real world model doesn't match the simulated environment closely. The most obvious difference is the significant gap in terms of inference speed, where the simulated environment provide way more updates. But the vision model also struggles with accurate detection and is only reliable from a relatively close distance. This shows clearly that the real world object detection does not perform similarly to the simulated environment.

The second question: *"Is the driving model capable of navigating to the same accuracy in the real world?"* was unfortunately not evaluated to an extent where answering this question is possible. No comparison to the simulation in regards to the accuracy of its location using its GPS sensor, or the consistency of a driven path could be made. The short experiment with manually instructing the rover to turn on uneven grass did however demonstrate that some uncertainties will need to be accounted for.

The last question: *"Can the control system deal sufficiently with the inconsistencies of the real world?"* was however evaluated and can be answered with a clear no. The control system did not seem to understand when an object was actually in its way such that it would decide to take avoiding action. In one of the experiments it did slow down, but this seemed to be caused by an incorrect understanding of its situation since the actual object was still hit.

8.2 Conclusion

The goal of this project was to evaluate if a model trained in simulation was capable of avoiding objects in the real world with similar performance. Due to issues with setting up the rover the avoidance could not be tested, but it was still possible to evaluate the ability to detect objects and stop before they are hit. At the same time the simulated environment was tested to find situations where it showed incorrect behaviour, and to understand what behaviour was generally desired.

In the real world two different types of experiments were performed, aimed at evaluating the performance of the vision model, and the driving model of the rover. While the vision model showed decent performance in certain situations, its detection was certainly not flawless when challenged further. The driving model also showed an inability to avoid all objects, seemingly indicating some software error. Testing in simulation showed that the model only seems to take objects straight ahead into account, which could be a possible explanation for insufficient performance in the real world.

After evaluating these scenarios the main question for this thesis can be answered: *"Does an autonomous self-driving model trained in simulation perform to a similar degree in the real world, specifically in off-road scenarios"*. The answer to this question is an obvious no, the vision model is limited in its speed, detection accuracy and can not sufficiently deal with occluded models. The driving model also does not seem to stop or slow down for detected objects.

8.3 Future Research

The first heading for any further research into this subject, specifically using this rover could be done into the reason that the rover did not attempt to avoid any obstacles. As mentioned before, it seems to be some configuration fault within the driving model, but it could also have to deal with a lack of information about the current yaw of the rover. Making the collision detection is also a relevant heading for this research since it is also still not perfect in simulation.

A second improvement on this research would be to dive further into the vision model. Using objects of different classes provides an extra challenge for the model since their shapes might be even more irregular or their position harder to estimate. It would also be very interesting to see how the rover would react to moving objects, since it would then also have to start predicting the path of these objects. Further challenges could be created by changing lighting conditions more, or possibly evaluating performance in different weather conditions.

One more interesting heading would be to do more research into the performance of the driving model. It is now only capable of driving in a straight direction, and was not really evaluated in terms of the accuracy of its chosen path. There are multiple parameters which could be changed to evaluate their impact on performance. Using better path estimations in the simulation should also provide more insights into the consistency of the model's decision making, and once the rover can also avoid objects it would be interesting to evaluate the similarity of the driven paths between the real world and the simulation.

As a final challenge, once the model is capable of performing consistently in more random scenarios and it shows consistent avoidance of objects, the rover could actually be evaluated in more rough scenarios. The impact of inclines, dust, loose surfaces and less visible paths should make for an interesting evaluation.

Bibliography

[Bla+12] Mogens Blanke, Morten Rufus Blas, Søren Hansen, Jens Christian Andersen, and Fabio Caponetti. "Autonomous Robot Supervision using Fault Diagnosis and Semantic Mapping in an Orchard". In: *Fault Diagnosis in Robotic and Industrial Systems* (2012).

[BWL20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *CoRR* abs/2004.10934 (2020). arXiv: `2004.10934`. URL: `https://arxiv.org/abs/2004.10934`.

[Cho+12] Jaewoong Choi et al. "Environment-Detection-and-Mapping Algorithm for Autonomous Driving in Rural or Off-Road Environment". In: *IEEE Transactions on Intelligent Transportation Systems Vol. 13, No. 2* (2012), pp. 974–982.

[Osi+20] Blazej Osinski et al. "Simulation-Based Reinforcement Learning for Real-World Autonomous Driving". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2020). DOI: `10.1109/icra40945.2020.9196730`.

[Sha+19] Suvash Sharma, John E. Ball, Bo Tang, Daniel W. Carruth, Matthew Doude, and Muhammad Aminul Islam. "Semantic Segmentation with Transfer Learning for Off-Road Autonomous Driving". In: *Sensors* 19.11 (2019), p. 2577. DOI: `10.3390/s19112577`.

Appendix

Appendix A

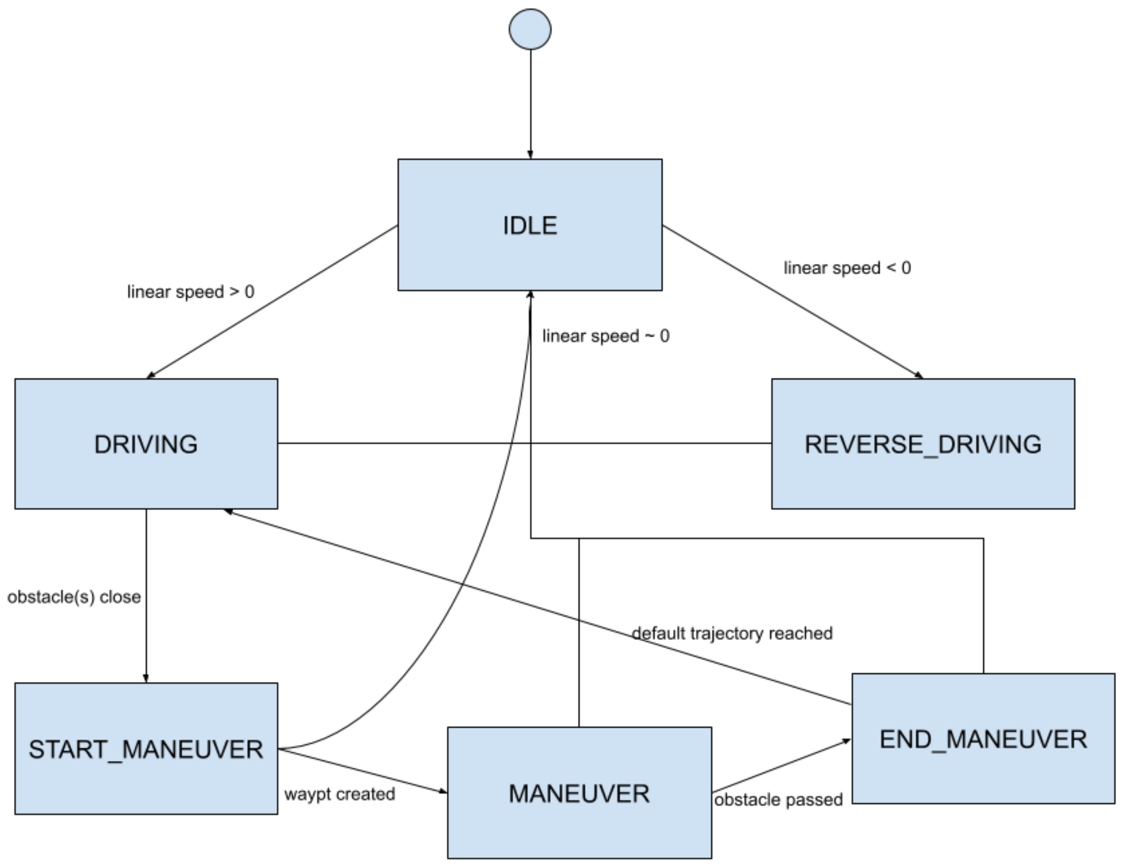


Figure 8.1: State machine configuration

Appendix B

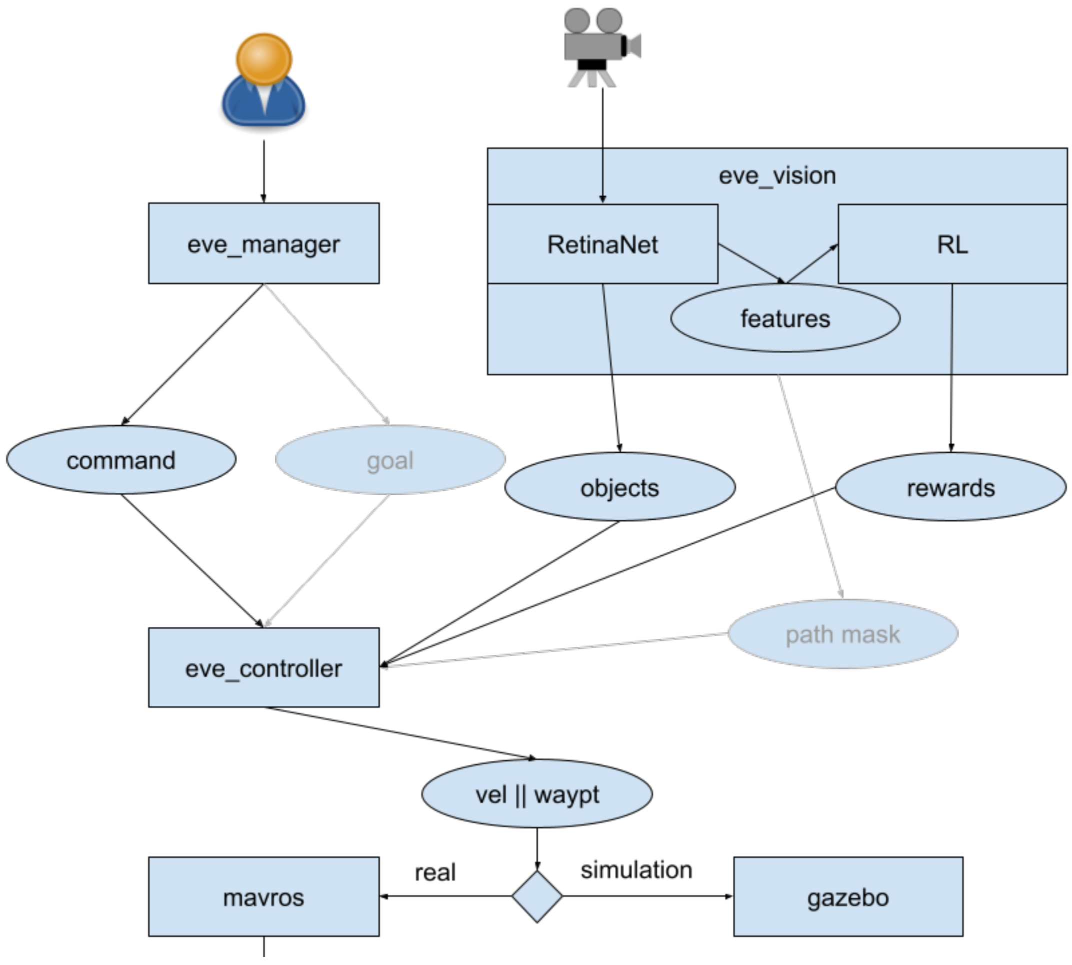


Figure 8.2: Controlling software information flow