Tartu University
Faculty of Science and Technology
Institute of Technology

Uroš Petrović

Small-scale cars for autonomous driving research

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervisor:
Dr. Naveed Muhammad

Tartu 2023

Resümee/Abstract

Väikesemahulised autod autonoomse sõidu uurimiseks

Isejuhtivad autod (autonoomsed autod, roboautod) on sõidukid, mis saavad sõita ilma inimese sekkumiseta. Viimastel aastakümnetel on see teadusvaldkond saanud palju tähelepanu. Kasvab vajadus liikluse ohutuse ja tõhususe parandamise järele. Isejuhtivad autod lubavad olla ohutud. Peamine põhjendus on see, et inimlike vigade kõrvaldamise ja algoritmide kasutamisega saaks enamikus olukordades ettetulevate vigade ja õnnetuste arvu drastiliselt vähendada. Autonoomne sõit on praegu põhjalikult uuritud valdkond, kuid olulised väljakutsed on endiselt lahtised, mistõttu on järgmistel aastakümnetel vaja teha märkimisväärseid uuringuid. Kuna elusuuruses katseplatvormide maksumus autonoomse sõidu uuringute jaoks on väga kõrge, tekib vajadus kasutada katseplatvormidena väiksemaid sõidukeid. See lõputöö uurib sellise meetodi elujõulisust, hinnates väikesemahulise isejuhtiva auto platvormi Donkey Car jõudlust ja sobivust autonoomse sõidu uuringute erinevate aspektide jaoks. Uuriti platvormi tajumist, lokaliseerimist ja kaardistamist, planeerimist ja täielikke võimalusi ning võrreldi neid avalikult kättesaadavate päriselus isejuhtivate autode katsetega. Saavutatud tulemused näitasid, et väikesemahulised autod on võimelised teostama autonoomse sõidu uuringuid vastuvõetaval tasemel võrreldes pärisautodega..
CERCS: S274 Teaduse uurimismetodoloogia, T125 Automatiseerimine, robootika, control engineering
Märksõnad: väikesemahulise autod, autonoomsed autod, SLAM, hindamine, isejuhtivad

Small-scale cars for autonomous driving research

Self-driving cars (autonomous cars, robo cars) are vehicles that can drive without human input. In the recent decades, this field of science has received a lot of attention. There is a growing need to improve the safety and effectiveness of traffic. Self-driving cars have the promise of being safe. The main reasoning is that by removing human error, and utilising algorithms, the number of mistakes and accidents that would arise in most situations could be drastically reduced. Autonomous driving is a heavily investigated area currently, but significant challenges remain open, thus requiring significant research during the decades to come. As the cost of life-size test platforms for autonomous-driving research is very high, the need to use smaller-scale vehicles to be employed as test platforms arises. This thesis investigates the viability of such a method, by evaluating the performance and suitability of a small-scale self-driving car platform, the Donkey Car, for different aspects of autonomous driving research. Perception, localization and mapping, planning and end-to-end capabilities of the platform were investigated and compared with publicly available real-life self-driving car experiments. The results achieved showed that small-scale cars are capable of performing autonomous driving research at an acceptable level in comparison with real-life cars.
CERCS: S274 Research methodology in science, T125 Automation, robotics, control engineering
Keywords: Small-scale cars, autonomous vehicles, SLAM, evaluation, self-driving

List of figures

List of tables

Abbreviations

ROS - Robot Operating System
SLAM - Simultaneous Localization And Mapping
LIDAR - LIght Detection And Ranging
RC - Remote Car
IMU - Inertial Measurement Unit
CNN - Convolutional Neural Network
UT - University of Tartu
CAD - Computer Aided Design
RP - Rotating Platform
SBC - Single Board Computer
DIY - Do It Yourself
AV - Autonomous Vehicle
ADL - Autonomous Driving Lab

Contents

1 Introduction

Autonomous vehicles have become prevalent in the last decade, gaining commercial popularity. The knowledge surrounding this technology is still in its early stages, with many companies (such as Tesla, Waymo, etc.) [1] racing to become the best in this field. The main issues autonomous driving currently faces are issues of safety and economics [2].
There is a growing need to increase the testing as most vehicles rely on data to be able to successfully learn to drive. One potential method of providing easy and cost-effective research is by utilising small-scale self-driving platforms. This would hasten the research by easy deployment of small vehicles and cut costs with budgeting. This thesis tackles the suitability of such a solution by demonstrating that small-scale vehicles can perform autonomous driving research on par with regular size self-driving vehicles.

1.1 Contributions

The work done in this thesis aims to answer the question of suitability of small-scale cars for autonomous driving research by:

- Researching previous work on both small- and large-scale autonomous driving platforms
- Proposing and developing experiments necessary to give an answer to the question of suitability of small-scale cars for autonomous driving research by performing these experiments, and validating the results by comparing with real-life vehicle where applicable

There are also indirect technical contributions resulting from thesis work:

- Creation of Donkey Car image capable of running ROS and donkeycar Python library
- Fixing of Donkey Car IMU unit to enable its use in ROS
- Several Donkey Car parts:
    - Object detection
    - Path planning
    - Trajectory creator
    - Path planning controller capable of sending images and receiving trajectories
- Several python scripts:
    - Socket server capable of receiving images and sending generated trajectories
    - Google Cartographer and Hector SLAM configuration files necessary to enable mapping on the Donkey Car

All of the data and files have been made available online, on Github [3].

1.2 Structure of the manuscript

The remainder of the manuscript is structured as follows. Chapter 2 will tackle literature review on autonomous vehicles, the challenges that current autonomous driving research poses, and give a brief introduction to small-scale platforms. Chapter 3 will focus on the methodology of the thesis, introducing the car used and the experiments performed. Chapter 4 will give the

results achieved in the experiments and discussion regarding those results. Finally, chapter 5 will give the conclusions that arise from the work and any future work that would be necessary.

2 Literature review

This chapter covers the introduction to autonomous vehicles, common autonomous driving architectures, challenges in autonomous vehicle research and overview on small-scale cars and its research.

2.1 Autonomous vehicles

An autonomous vehicle (Figure 1) [4] is a vehicle that can perform all its functions without the need of any human intervention, through its ability to sense and perceive its surroundings. To do that it employs different types of sensors such as cameras, lidars, GPS, odometry, inertial measurement units (IMU) etc.
Autonomy in autonomous vehicles is divided into six levels [5], each representing different ratio between human input necessary and autonomy of the car:

- Level 0 - no automation: The vehicle is equipped with no autonomation, requiring the human driver to be fully in control of the vehicle.
- Level 1 - hands on: The vehicle can do one or more features automatically, such as cruise control. The human driver needs to perform all the other tasks. The Mercedes-Benz was a pioneer in this with its radar-managed cruise control [6].
- Level 2 - hands off: Vehicle can perform two or more features autonomously, such as steering and acceleration. The human driver needs to monitor all tasks and perform the rest. This level of autonomy is what most commercial autonomous vehicles are equipped with, such as the Mercedes S-Class [6].
- Level 3 - eyes off: The vehicle has environmental detection capabilities and uses it to drive. The human can take his eyes off the road but still needs to be able to intervene should the need arise. This level of autonomous driving is starting to be introduces more and more in specific regions of the world (such as Mercedes-Benz Drive Pilot [7]).
- Level 4 - mind off: The vehicle can perform all driving tasks under certain conditions. The human driver still has the option of intervention. This level of autonomy is very much in the early stages of research, with only a small set of companies (Waymo is introducing driverless taxis in the USA [8]).
- Level 5 - steering wheel optional: The vehicle can perform all driving tasks under all conditions. No human attention or intervention is needed. This level of autonomy is currently not commercially available, however, there are some predictions as to when this will be available. Experts state that until the year 2035, level 5 vehicles will not be available commercially [9].

Figure 1: A Tesla autonomous vehicle driving on the road. Source: Micheal Simari, Car and Driver [10]

To perform driving, most autonomous vehicles use two distinct methods:

- Modular pipeline approach
- End-to-end driving

2.2 Modular pipeline approach

The modular approach (Figure 2) splits the autonomous driving pipeline into several separate tasks such as perception, localization, planning, decision making, control...

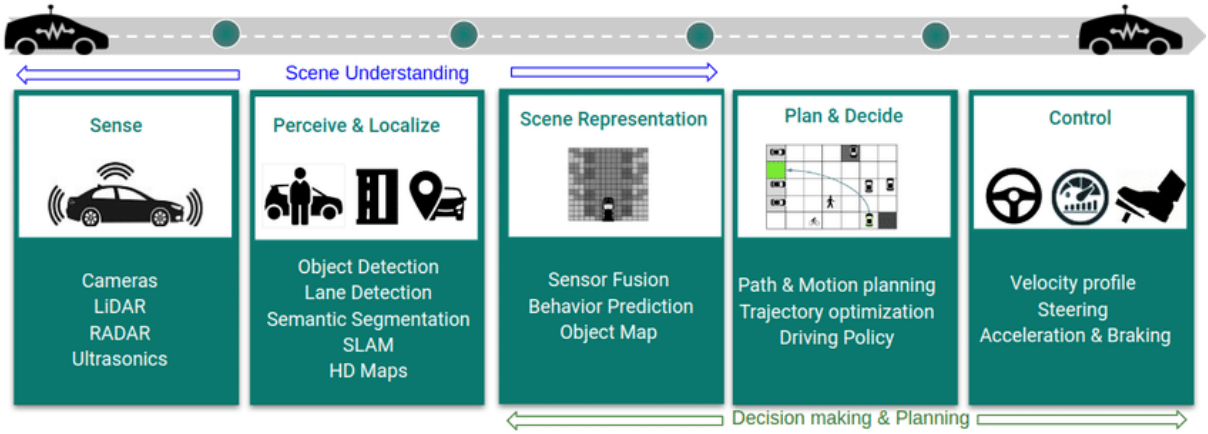

Figure 2: An example of a modular autonomous driving pipeline [11]

The advantage of such a method is that it allows for division of labour, which allows for development of autonomous tasks onto specific teams which can be independent of each other. Such systems are then able to be operational if the intermediary outputs are working. While the "divide and conquer" paradigm has its benefits, it also has the drawback of being complex and requiring significant manual design. The end-to-end approach will be explained in section 2.6 End-to-end driving.

2.3 Perception in autonomous driving

Perception is the ability of an autonomous vehicle to utilize many different sensors such as cameras, LiDARs, ultrasonic distance sensors and more, to perceive the world.
For humans to perceive the world around us, they utilize different senses such as sight, smell, vision... The same goes for autonomous vehicles. Utilizing many different sensors an autonomous vehicle can more accurately perceive the world around it to perform the necessary tasks.

2.3.1 Camera

A camera (Figure 3) is a specialized image sensor that detects visible light spectrum reflected from objects. In terms of autonomous driving, it is mostly used in object detection such as traffic lights, people, vehicles, roads and other section necessary for driving. The advantage of such sensors is that they can achieve high quality necessary for accurate image processing while being on the lower end of cost. The main disadvantage is that they have trouble working in adverse weather conditions such as rain or snow [12].

Figure 3: An Allied Vision Mako G-319C Camera. Source: University of Tartu Autonomous Driving Lab

2.3.2 Lidar

Light Detection And Ranging (LiDAR) is one of the most common sensors for autonomous vehicles [13]. Used for localization and mapping, this device uses light in the form of a pulsed laser to measure distances. The Lidar device (Figure 4) emits pulsed light waves from a laser into the environment. These laser waves are then returned from objects they collide with and return to the sensor, where the time it took for the wave to return is calculated. The device is also rotated at very high speeds to create a set of points that can be processed into a 2D or 3D map depending on the type of sensor. This set of points is called a point-cloud map.

Figure 4: A Velodyne Puck 32MR. Lidar device. Source: Velodyne

2.4 Localization in autonomous driving

Localization (and mapping) is the ability of the car to orient itself in the real world, and its ability to create a map of the environment using various sensors that it has access to.
For the vehicle to localize itself in the environment, data acquired from the car sensors is used. The most common method autonomous vehicles use for localization and mapping is SLAM (Simultaneous Localization And Mapping) algorithms. This is the method of construction of a map and orientation of a vehicle simultaneously. There are many different algorithms that are used to do this, and some of them are:

- Particle filter [14]
- Kalman filter [15]
- GraphSLAM [16]

The particle filter (also known as the Monte-Carlo localization) is a localization algorithm that utilizes possible states (i.e., Particles) to estimate the likely position of the vehicle in a map (Figure 5). The algorithm uniformly distributes a set of particles on the map with the same certainty. In the next measurement all particles are updated with the likelihood of the vehicle location based on the observation of the environment. The main advantage of such algorithms is that they are relatively easy to implement and scale very well. However, they are computationally expensive.

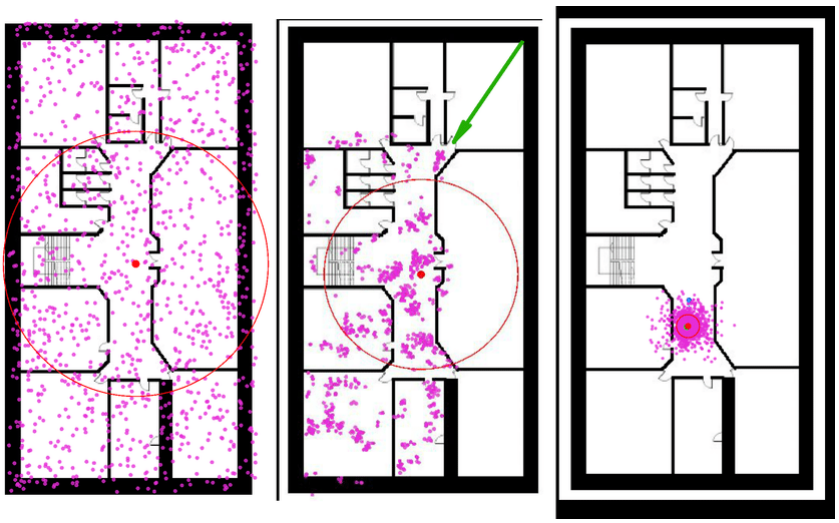

Figure 5: Particle filter visualization [17]

Graph based SLAM, as the name suggests, works on the principle of graphs, whose nodes represent the environment (location of the vehicle or other objects) and the edge between nodes encodes a sensor measurement that constrains the connected poses (Figure 6). As the vehicle moves in the environment the algorithm detects when it revisits a location and uses this information to improve map consistency. It then performs a global optimisation of the graph to find the most likely position of all objects and update the map. This optimized graph is then used to determine the vehicle's current pose in real time.

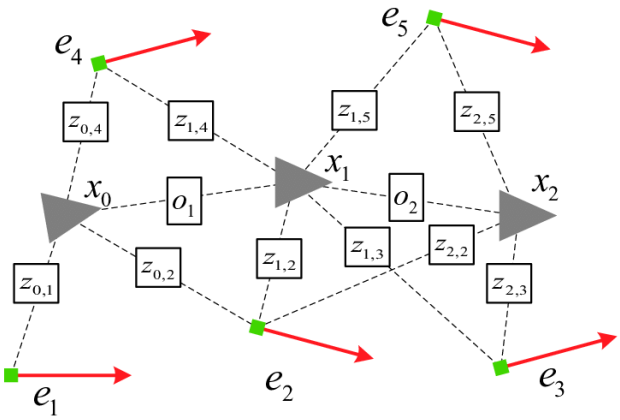

Figure 6: GraphSLAM visualization [18]

2.5 Planning and control in autonomous driving

Planning [19] is the ability of an autonomous vehicle to utilize data acquired from sensors to plan a trajectory to reach a specific destination. Planning methods are generally divided into three functions: global route planning, local behaviour planning and local trajectory planning. Global route planning (Figure 7) is responsible for finding the best road level path in a network, represented as a directed graph containing edges and nodes. Such planners search in the graph to find the sequence that links the start and end goal with minimal cost. Common global route planning algorithms are Djikstra and A* algorithms.

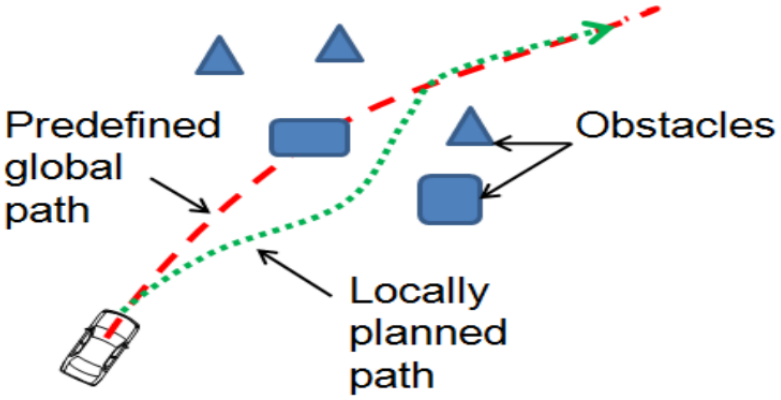

Figure 7: A global and local route planner visualization [20]

Local behaviour and trajectory planning algorithms are responsible to create a safe trajectory based on the global route already created. Common local planning algorithms include the pure pursuit and potential field method.

Pure pursuit (Figure 8) works on the principle of calculating a goal that is a specific distance away from the vehicle called the lookahead distance, giving an angular velocity necessary for it to reach the position. The goal point is then updated, and the process repeated until the final point is reached. While this algorithm is easy to implement and to work with. It has difficulty following the exact path needed to reach and will sometime not stabilize and reach it.

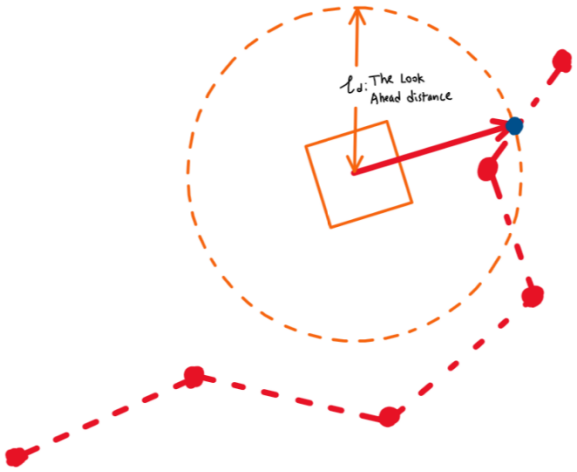

Figure 8: An example of pure pursuit [21]

The potential field method (Figure 9) works on the principle of potential fields. Every object found in the environment, and the goal position is assigned to what is called potential force, which is attracting the vehicle towards the goal position, and repelling it from any obstacle, thus preventing collisions. Similarly, to pure pursuit, it has the advantage of easy implementation, however the algorithm sometimes reaches a "local minimum", where the attractive and repulsive forces are the same, thus stopping the vehicle and preventing it from reaching the goal.

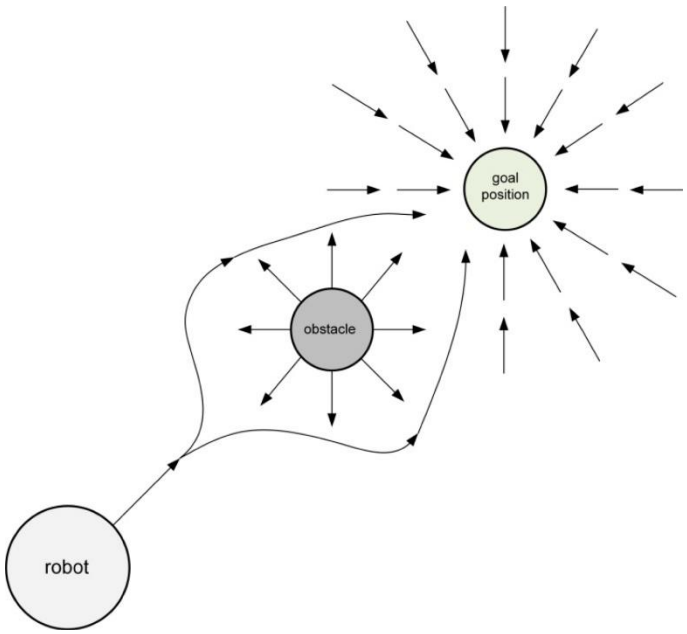

Figure 9: An example of the potential field method [22]

2.6 End-to-end driving

An end-to-end architecture (Figure 10) aims to combine all modules into one, i.e., to feed all inputs into a single algorithm that then directly outputs the driving policy.

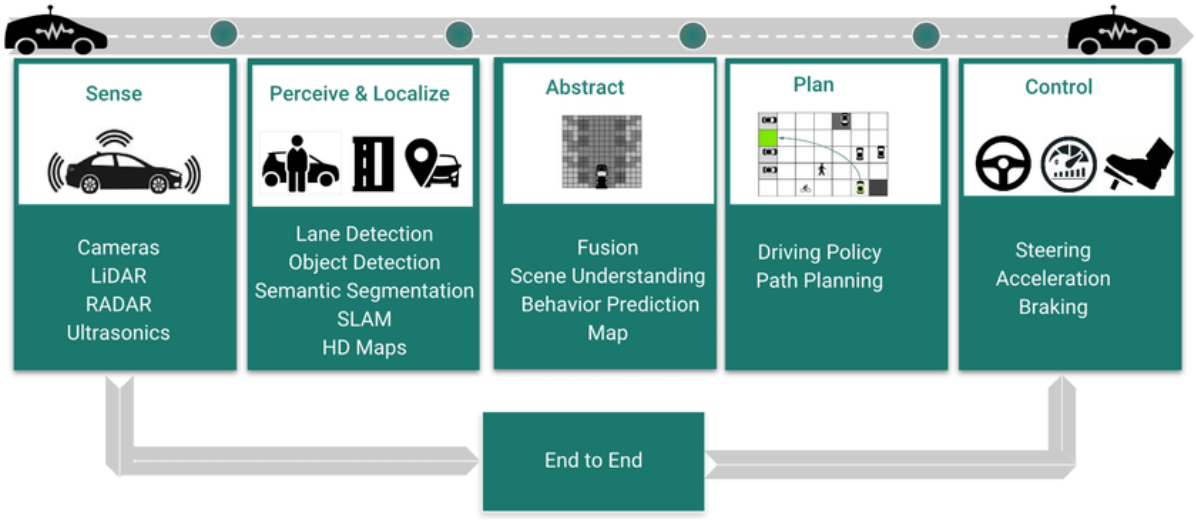

Figure 10: Visual comparison of end-to-end vs modular approach [11]

This architecture was first introduced in 2016 by Nvidia [23]. It was a 9-layer Convolutional Neural Network that used a camera image as input and output a steering angle and acceleration. While this method has the advantage of simplicity of implementation and integration, it is also difficult to interpret due to the complexity of deep learning models. In the following subsections two dominant paradigms in end-to-end will be explained – imitation and reinforcement learning.

2.6.1 Imitation learning

Imitation learning [24] is the more dominant paradigm used in end-to-end driving. It is a supervised learning approach in which a model is trained to mimic a specific behaviour. In the case of autonomous driving, this is the driving of a human driver, and the mimicked behaviour is driving commands such as steering, acceleration and braking. This method is reliant on the quality of the driver; hence it can be very accurate if the driver is driving perfectly, however, this also has the problem of distribution shift, i.e., the model might not know how to behave in unseen situations.

2.6.2 Reinforcement learning

Reinforcement learning [24] is an approach where the system aims to receive the maximum reward by acting in an environment. It can perform better than imitation learning in diverse driving situations, however, it is less data efficient and more challenging to use in the real world. In contrast to Imitation learning, the agent responsible for driving is not told how to reach the destination, rather it is given instructions on possible commands such as steer left/right, accelerate, break, and using these instructions it receives "feedback" in the form of a reward function. This gives positive value if the vehicle reaches its destination or driving correctly, and negative value if it's driving incorrectly by crashing or not following the road rules.

2.7 Challenges in autonomous driving

There are many challenges current autonomous systems face [2], some of them being:

- Safety: Current technology must be tested in detail before being put into commercial use. The World Health Organization states that vehicle accidents are one of the leading causes of human death [25]. As the goal of autonomous vehicles is to reduce this, they need to be able to make little to no mistakes during driving.
- Technical barriers: modern technology for autonomous driving has yet to be developed enough to be able to perform in all kinds of weather, and situations. Therefore, it is necessary that more research and development is made to create reliable sensors and computers to be able to accommodate different scenarios that autonomous vehicles must learn to react to properly.

Tackling these issues is very important. To develop these methods, however, most researchers do not have access to a large-scale vehicle, or no financial capacity to perform it. Thus, the growing need to scale down the vehicles for testing has arisen.

2.8 Small-scale car platforms

Figure 11: A small scale car. Source: TU Berlin

Research shows [26][27] that small-scale vehicles have the potential to create a low-cost, easy to integrate solution that will further the area of autonomous vehicle research. These platforms are designed to mimic commercial AVs (Figure 11):

- Equipped with a sensor suite necessary for AVs (lidar, camera...)
- On-Board Computer (OBC) to perform necessary calculations and processes.
- Chassis to have an appearance close to a real AV and to house all components.

After each platform is designed, experimentation is done to assess the viability of such a platform for the task they were designed to do (which is usually one or more areas within the broader field of autonomous-driving research).

Object detection proved to be robust and scalable on a small-scale AV. Research [27] shows that the results achieved are acceptable enough, with some small issues regarding the confidence of the detected objects. The self-driving aspects of platforms gave results acceptable for academic learning.

Self-driving aspect testing on the Go-CHART [28] platform confirmed the use of small-scale AVs to test human-driver and driver-driver situations in a safe and controlled environment. The platform was able to perform lane following and object detection at an acceptable rate. There were issues with the inference rate of some configurations due to the hardware limitation but ultimately this platform was stated to be able to execute autonomous driving capabilities at a level acceptable enough for research.

Small scale platforms exhibit state-of-the-art localization and mapping research capabilities [26]. Through testing Lidar based SLAM methods, results achieved were sufficient for further improvements in this area of autonomous vehicle research.

Small-scale platforms also show acceptable results in specific autonomous driving concepts like parking [27]. Testing was done on different areas necessary to perform this specific task

(perception, self-driving, localization…) and the results achieved were acceptable to further research in autonomous parking.
Verti Wheelers have shown promising results [29] in driving in rough terrain and allowed for development of different control algorithms like open-loop, rule based and end-to-end driving.

2.10 Other small-scale and robotics platforms

It is important to note that the topic of the thesis is to investigate small-scale car platforms, and not just any robotics platform. There are many robotics platforms that can perform autonomous driving tasks such as path planning, perception etc. that are not cars but are small scale. For example, there are many small-scale autonomous robotic vacuum cleaners, small-scale drones, other robotics platforms (such as Robotont) that can perform most (if not all) autonomous driving tasks but are fundamentally different both in their kinematics/dynamics, the environment they are used in and their visual aspects. As the topic of the thesis is to explore small-scale cars, they will not be covered but are worth mentioning.
In that regard, for a platform to be considered as a small-scale car it should have 4 wheels, rectangular-prism shape and have the capabilities to steer, break and accelerate.

2.11 Summary

Research done on many different small-scale platforms have shown that using them is quite cost-effective and easy to deploy. These platforms were tested on many different aspects of autonomous driving research (self-driving, localization and mapping, parking, object detection…) and all achieved quite acceptable results individually. This leads to the question of more systematically investigating the capabilities of small-scale car in terms of autonomous driving in general, by demonstrating that they are suitable for any kind of autonomous driving research, not just specific tasks they are designed to perform.

3 Methodology

This chapter describes the methodology used to investigate the suitability of small-scale vehicles for autonomous driving research. The small-scale platform used in experimentation is introduced – Donkey Car, as well as a real-life vehicle which will be used as the common ground for comparison – the Lexus Car [37]. Next, several experiments are introduced to investigate different autonomous driving modules and end-to-end driving.

3.1 Donkey Car

Donkey Car (Figure 12) [30] is an open-source DIY self-driving platform for small-scale cars. This platform allows for the custom creation of a self-driving vehicle that can be controlled remotely (using a smartphone or a computer) or drive on its own. The focus of this platform is fast experimentation and easy contribution.

Figure 12: A Donkey Car. Source: Donkey Car

The Donkey Car library supports different SBCs like the Raspberry Pi or Jetson Nano. Its code is written in the Python programming language.

As this is a custom-made car there are a lot of different configurations hardware-wise that can be made. For this thesis the following configuration is used:

- Raspberry Pi 4B [31]
- RoboHAT MM1 robotics controller board [32]
- Raspberry Pi Camera [33]
- 1/16th 4WD Electric Power R/C Off-Road Truck [34]
- RPLidar A1 [35]
- MPU9250 Inertial Measurement Unit [36]

3.2.1 Donkey Car software architecture

The Donkey Car code is organised into parts (Figure 13) that take various inputs and return outputs. A part is a Python class that wraps a functional component of a vehicle (such as sensors, actuators, pilots, etc.) Each part is constructed and then added into a vehicle loop.

```
class DonkeyCarPart:
    def run(self, input_variable):
        output_variable = proecess(input_variable)
        print(f"{output_variable}")
        return output_variable

# create the vehicle and it's internal memory
V = dk.Vehicle()

# initialize a variable in the vehicle's memory
V.mem['input_variable'] = 10

# add the part to the vehicle.
V.add(DonkeyCarPart, inputs=['input_variable'], outputs=['output_variable'])

# start the vehicle loop
V.start()
```

Figure 13: An example Donkey Car script that executes a part

Donkey Car comes with a premade template called manage.py which contains the main code to run the car. This code is run in a vehicle loop that will be executed at a rate specified by the DRIVE_LOOP_HZ value. Most of these values are stored in myconfig.py and config.py for ease of editing. There are also a couple of concepts that are important in the Donkey Car template:

- Memory: this is a hash map of all various vehicle values that are shared by all parts, such as inputs, outputs and conditions
- Inputs: these are memory values passed to the run() method of any part. Every time the run() method is called, the vehicle loop will take the input from the memory and pass it to the run() method of the part for execution
- Outputs: these are memory values that will be returned by the run() method. These methods are written into the vehicle memory after each loop is executed

3.2 Lexus RX450h

The Lexus RX450h car (Figure 14) is utilised by the Autonomous Driving Lab (ADL) in the University of Tartu. It has all necessary sensors that an autonomous vehicle should have for research such as:

- Velodyne VLP-32C and Ouster OS1-128 lidars capable of detecting obstacles
- NovAtel PwrPak7D GNSS device – for determining the position of the vehicle using satellite positioning
- Allied Vision Mako G-319C cameras – for tracking traffic lights
- AStuff Spectra computer – for housing the software necessary to perform all driving decisions
- Aptiv ESR 2.5 radar – for determining the distance and speed of vehicles around the car

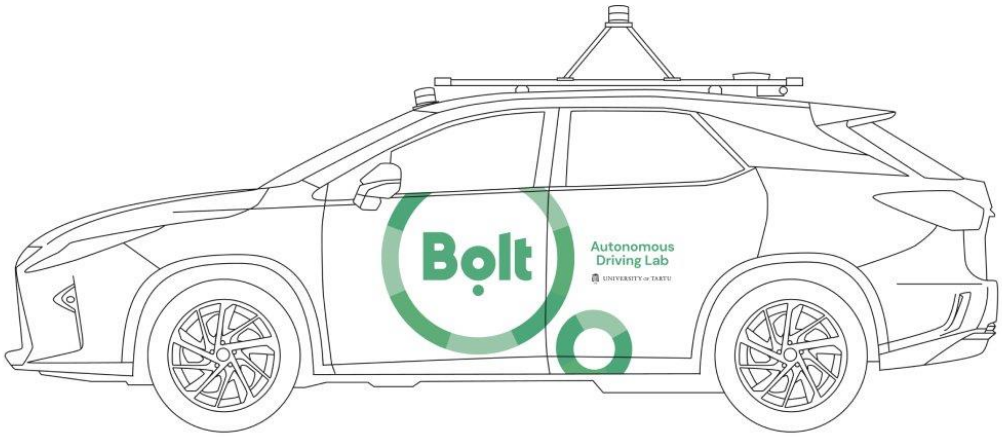

Figure 14: The Lexus Car [37]

This vehicles data will be used as the ground truth for a real-life car during the end-to-end driving experiments.

3.4 Experiments

To investigate the suitability of small-scale platforms for autonomous driving research the proposed method is to show that these platforms can achieve all the important tasks an autonomous vehicle should be able to perform, and that is:

- Perception
- Localization
- Planning and control
- End-to-end driving

For each of the following tasks, experiments are devised to both validate and explore the viability of the platform capabilities.

3.4.1 Perception experiment

It is stated that the most critical tasks perception needs to be able to do is road segmentation and object detection [38]. To confirm perception capabilities of small-scale cars, object detection will be evaluated. Using a camera, as it is one of the most commonly used perception sensor in autonomous driving, several object detection algorithms will be tested, and compared with the performance of the algorithms on real cars. The algorithms evaluated will be:

- YOLOv4-Tiny
- MobileNetSSD
- EfficientNetLite

Each of these algorithms are lightweight object detection algorithms, and they are run on the vehicle in order to measure the inference time and compare it with data from a real vehicle.
Utilizing Donkey Car's threaded parts, an object detector is created. All models are loaded using pre-trained weights in order to be as objective as possible.

For the experiment to be successful, the Donkey Car should be able to successfully load the model and detect an object using the camera without impeding the driving loop of the car.

3.4.2 Localization and mapping experiment

In order to validate localization capabilities, we are using a method similar to this paper [39]. In most localization experiments, the vehicle is using several algorithms to map and localize itself in the environment, and those results are then compared to a ground truth. This is the case for the small-scale cars as well. Like the perception experiment – two parts are tested: indoor and outdoor to showcase similar points that are being challenged.
For this experiment two SLAM packages are ran: Hector SLAM [40] which is a particle-filter based localization algorithm and Google cartographer which combines grid-based SLAM with scan matching [41].
For the indoor part of the experiment, the Donkey Car is driven around a section of the University of Tartu Delta building (Figure 15). Using the CAD drawing of the building as the ground truth a small section of a floor is mapped.
The first section (Figure 16) is on the 2nd floor near room 2018. This section is to demonstrate a small area with a "crossroads" to simulate a more condensed road.
The second section (Figure 17) is a part of the 3rd floor hallway to simulate long roads and harsh 90° turning.

Figure 15: Experiment setup of the localization experiment.

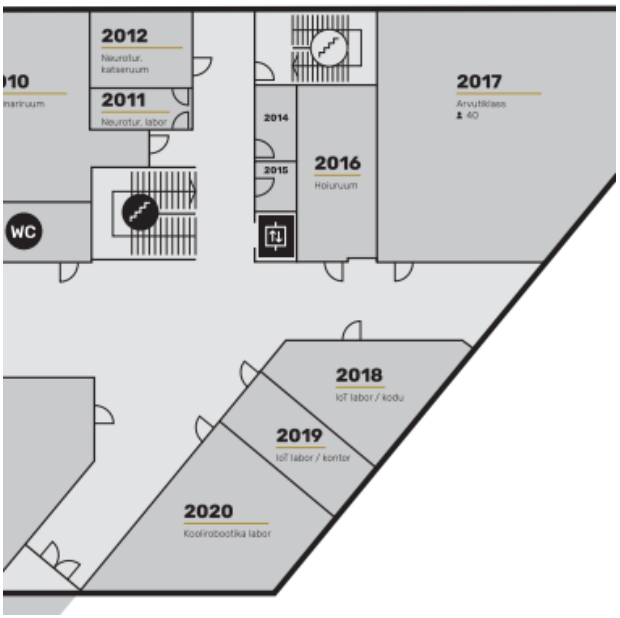

Figure 16: Section one of the localization experiment. Figure 15 corresponds to the space between rooms 2018 and 2016

Figure 17: The CAD drawing of the second location that will be mapped indoors

The second part of the experiment: The outdoor section is Müürivahe street in Tallinn to demonstrate the difference of using a small-scale vehicle on a large-scale environment. The ground truth in this case is the Google Maps representation of the area (Figure 18).

Figure 18: Google map representation of the outdoor experiment

To evaluate the localization and mapping aspect of the Donkey Car platform, ROS Noetic is used. As the hardware of the car is not powerful enough to handle heavy computing, communication between the car and a more powerful computer is necessary, hence the use of ROS [42]. Its publish and subscribe methods are excellent in evaluating both the sensors and the localization methods.
To compare with a real-life vehicle, we are following a similar system as in this paper [43]. Instead of showing the trajectory, the CAD drawing/Google map section is overlayed with the created map.
For a successful experiment, the Donkey Car should be able to map an accurate section of the environment without causing the main driving loop to slow down.

3.4.3 Planning and control experiment

To demonstrate planning and control capabilities, the Donkey Car is used in order implement a path planning algorithm that follows a similar methodology as the pure pursuit algorithm.
The pure pursuit algorithm is modified to facilitate small-scale capabilities. In that regard, the pipeline created is as follows:

- the object detection algorithm detects an object (person, car)
- the detected object is fed into a algorithm that creates a trajectory using Bezier curves
- the final created trajectory is used by the pure pursuit algorithm to follow the target

The detection algorithm outputs the detected object as the coordinates of the bottom half of the object.
To generate what the trajectory the vehicle should create to reach the object a Bezier curve is constructed (Figure 19) between the points of the location of the car (which is the bottom-middle of the image) and the detected object. The 3-point Bezier curve is used as it most closely relates to what the proposed trajectory of such a path planning algorithm should follow. It is calculated as:

$$P(t) = P_0 t^2 + P_1 2t(1-t) + P_2(1-t)^2,$$

Where t is the number of points the curve should have ranging from 0 to N, P_0 is the start point of the curve, P_1 is the end point and P_2 is the control point of the curve.
This curve is then finally fed to the pure pursuit-like algorithm which follows a different methodology in how the next point the vehicle should reach is: the lookahead distance which will be defined as LOOKAHEAD_DISTANCE is the n^{th} point in the Bezier curve trajectory created in the trajectory creator algorithm.

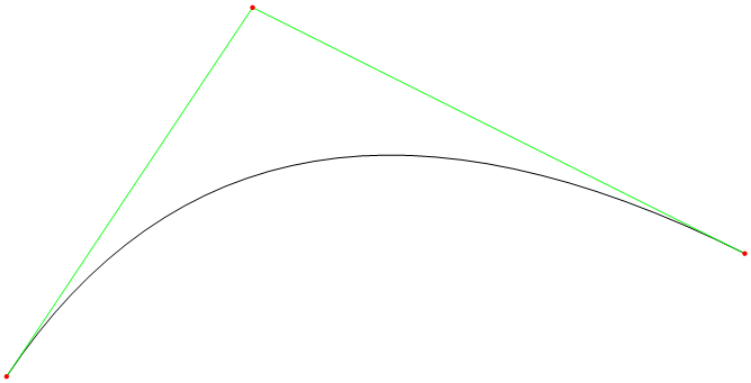

Figure 19: A 3-point Bezier curve

Considering the compute heavy requirements of object detection models there are two cases of evaluation; one where the whole system is done on the Donkey Car, and one where only the pure pursuit portion is run on the Donkey Car, i.e., the object detection and trajectory creation are performed on a different device to only test the planning and control portion of the pipeline. For the experiment to be successful, the algorithm should be able to infer steering values from the pipelines.

3.4.4 End-to-end experiment

As end-to-end architecture combines several common autonomous driving modules, it is an excellent way of validating the abilities of an autonomous car.
To compare performances of two different scaled platforms, data from the real-life car is used to create a similar track for a smaller car by scaling down the size. The dataset used for the real-life car is from WRC Rally Estonia 2020 and 2021 mentioned in this thesis [44].
To compare the results achieved with the models, on-policy evaluation is used. Two parameters are used to compare the cars:
- intervention (the number of times the user had to stop the model and adjust the position of the car manually)
- whiteness (the smoothness of the sequence of the predicted action of the model)
Whiteness is calculated as:

$$W = \frac{1}{D}\sum_{i=1}^{D}\left(\frac{\delta P_i}{\delta t}\right)^2,$$

Where D is the size of the dataset, δP_i is the change in predicted command between each timestep, and δt is duration of each timestep. This value is 0.02 for both cars.
The end-to-end models used for the small-scale cars are the standard recommended methods from the developers of the platforms to showcase what small-scale car can do in terms of end-to-end driving.

To be as objective as possible as the thesis tackles suitability not end-to-end driving development, the recommended guidelines are used from the developers of the Donkey Car platform, so the default procedures that the developers state should be followed.
The Donkey Car recommends driving 10 perfect laps, collecting between 5 and 20k images depending on the track length. That data is then cleaned up using the DonkeyUI app, removing any collisions, crashes, reverses, or other undesirable actions. The collected data is then used to train the Keras Linear model, a Convolutional Neural Network consisting of 5 convolutional layers followed by 2 dense layers.

The track used to perform this experiment on is a toy city created in the UT Delta Centre for autonomous driving competitions and research. It features a small-scale version of a racing track (Figure 20).
The dataset and model results from a real-life car feature the Lexus Car driving 10km in the Estonia rally competition, in both directions [44].

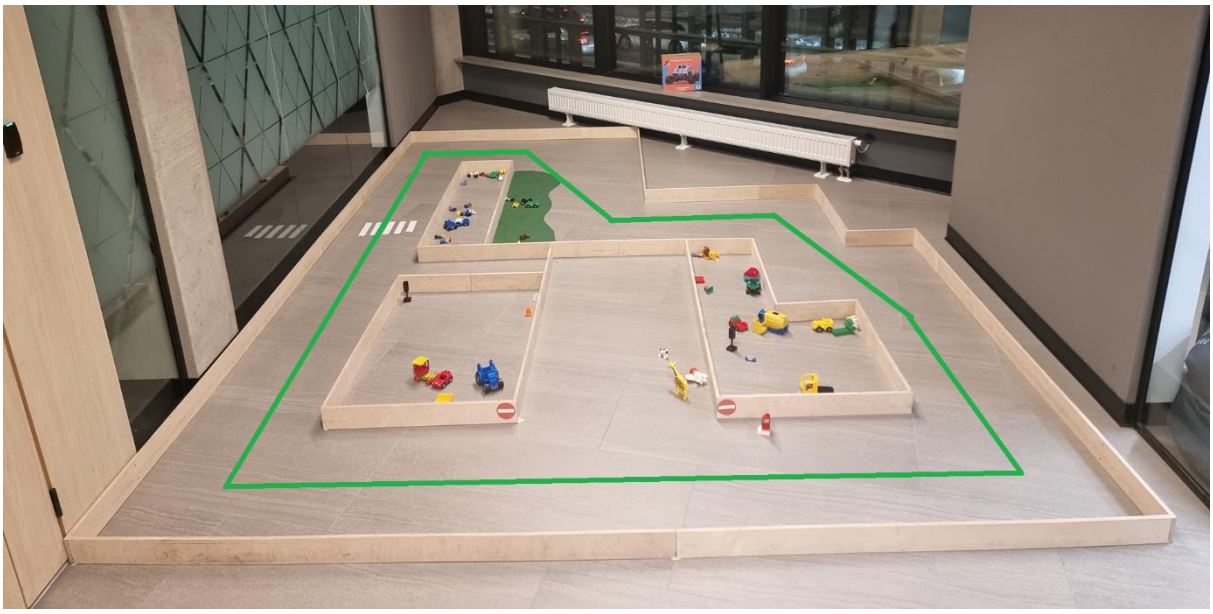

Figure 20: Picture of the track used in the end-to-end experiment. The green line is the path the car is trained on

For the end-to-end experiment to be successful, the vehicle should be able to learn how to drive on the track.

4 Results and discussion

This chapter showcases the results of the experiments and provides description and conclusions on the experiments themselves.

4.1 Perception experiment

The object detection models were used to create a custom Donkey Car part that would detect a specific object. This part is then run during the main driving script in order to test out the object detection capabilities.
As it was stated in section 3.4.1 the Donkey Car part was created using OpenCV's dnn module. The vehicle was unable to run the model and drive in loop, causing the car to drive irregularly. In order to mitigate these issues changes are needed to create a viable object detection system. Utilizing Donkey Car's threaded system, the part responsible for object detection was then changed to allow it to be run in a thread. This meant that it would try to run as fast as the python interpreter was able to run it, and not limit it to the vehicles drive loop. This allows for the object detection algorithm to detect the model as fast as python's interpreter would allow and allow the vehicle to drive independently of its performance.

Vehicle	Model name	Average frames per second [fps]
DonkeyCar	YoloV4-tiny-416	0.43
	EfficientNetLite	7.22
	MobileNetSSD	1.26
Real life vehicle	YoloV5	62.5
	EfficientNet	10.20
	SSD300	46

Table 1: Object detection model results comparison

The threaded part was successful at running object detection models on the vehicle. The performances of object detection models on small-scale vehicles are heavily influenced by the computing hardware of the platform. In comparison to a real-life vehicle the inference time is considerably slower than on a real vehicle (Table 1). In terms of autonomous driving, the object detection inference time makes it impossible to perform self-driving tasks using object detection. It was necessary to make the models threaded otherwise they would seriously impede the driving loop of the vehicle. To try to mitigate this the EfficientNet model was run utilizing the lite variant, which is made for lower end devices, and using TensorflowLite library the .tflite model was loaded which is said to be faster than regular models. This did improve the fps of the model up to 7.22 but this was still not enough. While such slow inferences would affect the safety of a real-life vehicle, this wouldn't affect a small-scale car, as it isn't capable of creating heavy damage or accidents on the scale that a real vehicle could. The inference of models can potentially be increased by introducing more powerful CPUs. This would potentially increase the cost of the vehicle itself, so this can be a potential area of research.
With all these issues in mind, a conclusion is reached that small-scale vehicles are suitable for autonomous driving research in the area of perception, albeit not at a level real-life vehicles are able to achieve.

4.2 Localization experiment

As the Donkey Car platform does not natively support ROS, a pipeline was created in order to enable the use of ROS and driving of the car. ROS has a plethora of libraries capable of performing autonomous driving research and sensor communication, hence its use in localization experiments.
During the mapping process, the calculations were done on the Donkey Car in order to confirm the assumption that the vehicle will not be able to handle the compute. This assumption was correct, hence the change to send data to another device capable of handling the heavier compute requirements of SLAM. Firstly, Hector SLAM is used to map the environment as it is a lightweight SLAM algorithm sensor-wise, meaning that it only uses the lidar to map, and it calculates and provides odometry using lidar data.
The beginning of the mapping process proved to be difficult, with the vehicle producing very inaccurate readings immediately after moving. This was both the problem of the speed of the vehicle and the lack of sensors. Slowing down considerably caused the mapping process to run much more smoothly. There were still issues with the accuracy, as can be seen in the results of the mapping process (Figure 21). This was due to the vehicle odometry not being provided.

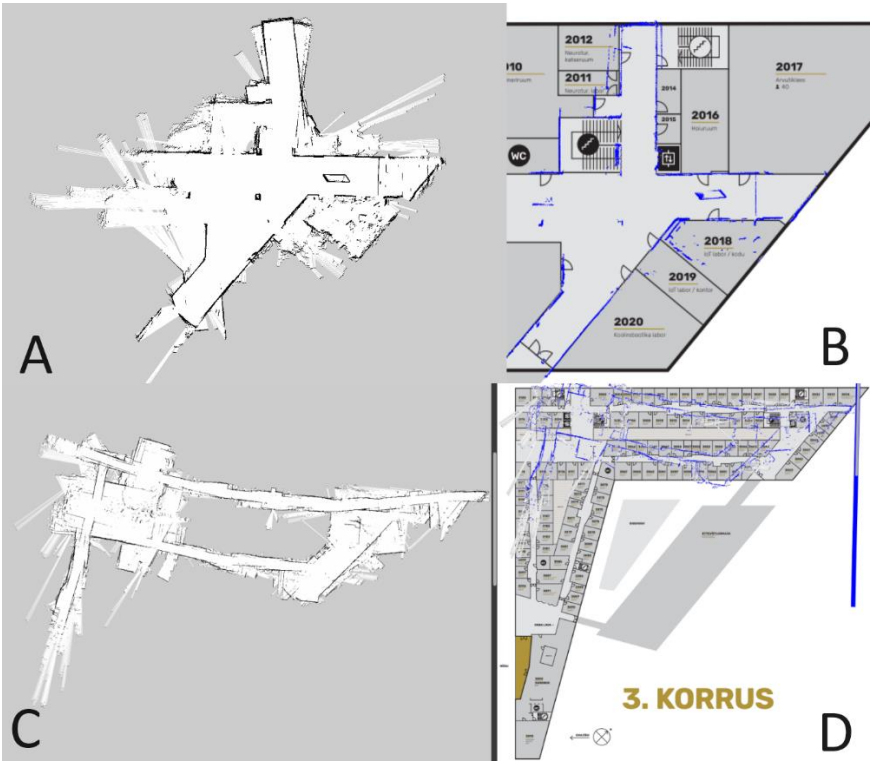

Figure 21: Hector SLAM results of the localization experiment. (A) Section 1 results, (B) Section 1 results overlayed (C) Section 2 results (D) Section 2 results overlayed

To improve upon the capabilities of mapping, more sensors are required, hence the use of Google Cartographer. This SLAM algorithm is able to utilize data from both a lidar sensor and an IMU unit to more accurately provide odometry. While the DonkeyCar is equipped with an MPU9250 sensor capable of providing IMU data, there were issues with the hardware setup of the sensor on the Donkey Car I^2C. Thus, the need to develop a solution where the car is able to read and produce sensor data is needed. Using the ROS package capable of sending IMU data

[45], and a python library which enables communication with the IMU [46], a ROS node was created capable of publishing IMU data. This enables the Cartographer to determine the orientation of the utilizing sensor data from the MPU's magnetometer, providing better odometry.

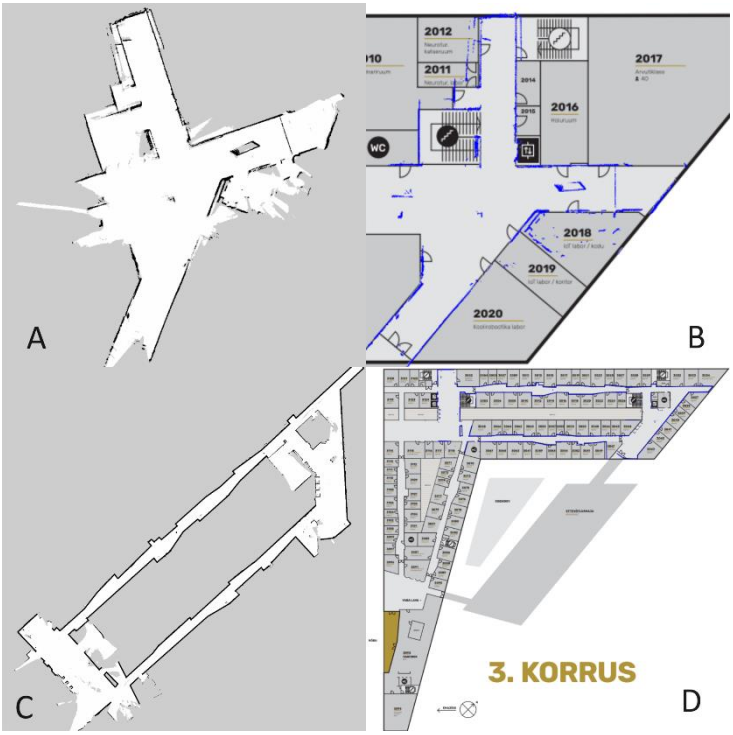

Figure 22: Cartographer results of localization experiments. (A) Section 1 results (B) Section 1 results overlayed (C) Section 2 results (D) Section 2 results overlayed

Utilizing a SLAM package combining multiple sensor data proved to be considerably more accurate than single sensor data (Figure 22).

Figure 23: Outdoor mapping results

Mapping in an outdoor environment was unsuccessful (Figure 23). The scale of the road in regard to the car itself, and the noisiness of the environment due to people and other vehicles constantly changing prevented any kind of usable mapping. This gives a conclusion that to perform any kind of accurate mapping with a small-scale car and the sensor configuration in this experiment should be done in a much more controlled environment. Comparing with the results achieved in the indoor experiments we can confirm these assumptions as that environment was scaled down less as the width of the "road" was more for humans which are closer to the scale of the small-scale car. There was also less traffic in the building allowing for less noise.

Overall, the vehicle was able facilitate localization and mapping successfully. There were several issues of note during the experiment executions:

- To have a more accurate map, the vehicle needed to be heavily slowed down, which increases the mapping time quite considerably. This goes in line with SLAM on real-life vehicles, so in that regard they share the main issue.
- The vehicle has difficulties processing any kind of turning. This is fixed by employing better odometry settings. This issue is fixed by either having the SLAM algorithms create odometry from laser scan data, or by attaching sensors that can provide this. Utilizing Donkey Car's IMU unit this issue was solved. Both issues have the potential to be a research topic into improving the performance of SLAM algorithms in situations where there is limited sensor availability such as difficult weather conditions or technical issues causing sensors to fail.
- Having environment that is noisy results in inaccurate map creation. This can be fixed by performing experiments in a more controlled environment.

With these issues in mind, and the results achieved during the experiments, a conclusion is reached that small-scale vehicles are suitable for use of autonomous driving research in terms of localization and mapping, sharing similar performances and problems real-life vehicles have.

4.3 Planning and control experiment

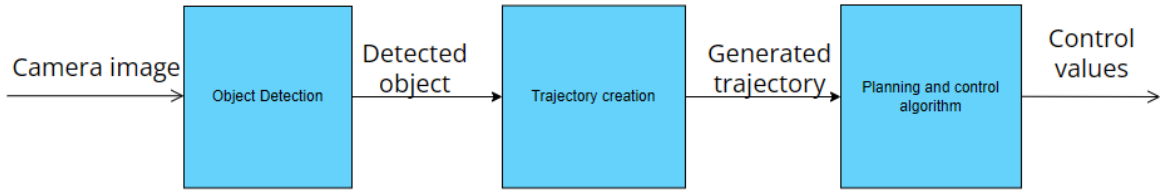

Figure 24: Block diagram of the planning pipeline

To create a planning pipeline necessary for this experiment, the part used in the perception experiment in section 4.1 was repurposed to be used as the object detection portion. This model then outputs the location of the object to the trajectory builder which then creates a Bezier curve to simulate a trajectory of the vehicle (Figure 24). The most intuitive way of determining the Bezier curve points was to use the start position of the curve as the location of the car itself. This is set as the bottom-half point of the image. The control point is the centre of the camera, i.e. the centre of the image. This way the vehicle is then only able to turn left and right to follow a detected obstacle. The created trajectory is then fed into the planning and control section to determine the steering angle that is needed to be inferred on the Donkey Car in order to align itself with the object.

Running the planner fully on the Donkey Car was successful. The planner was able to create the trajectory and follow several different obstacles. As it was already mentioned in section 4.1, the object detection models cause the inference time to be greatly reduced (up to 4 seconds of inference time, slowing down the main vehicle loop), making the planner unusable in a driving situation.

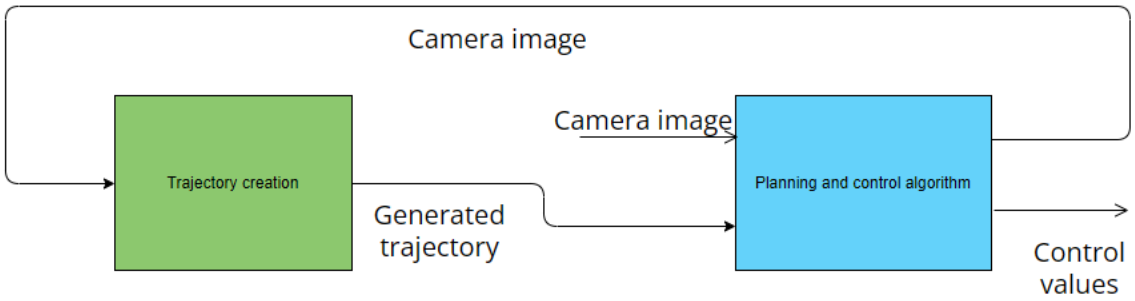

Figure 25: The faster Planning pipeline block diagram. The blue shaded block is the Donkey Car, which takes the input of the camera and sends it to another device, shaded green.

As planners are generally though to receive trajectory and goal position as an input, a faster pipeline scheme is then developed (Figure 25). Utilizing Python's socket library, communication is established between the Donkey Car and the trajectory creator. The client, Donkey Car, sends camera images to the host which using object detection and Bezier curve generation creates the trajectory and sends it back to the Donkey Car. The Donkey Car is then able to conserve resources to perform only the planning portion of the algorithm.

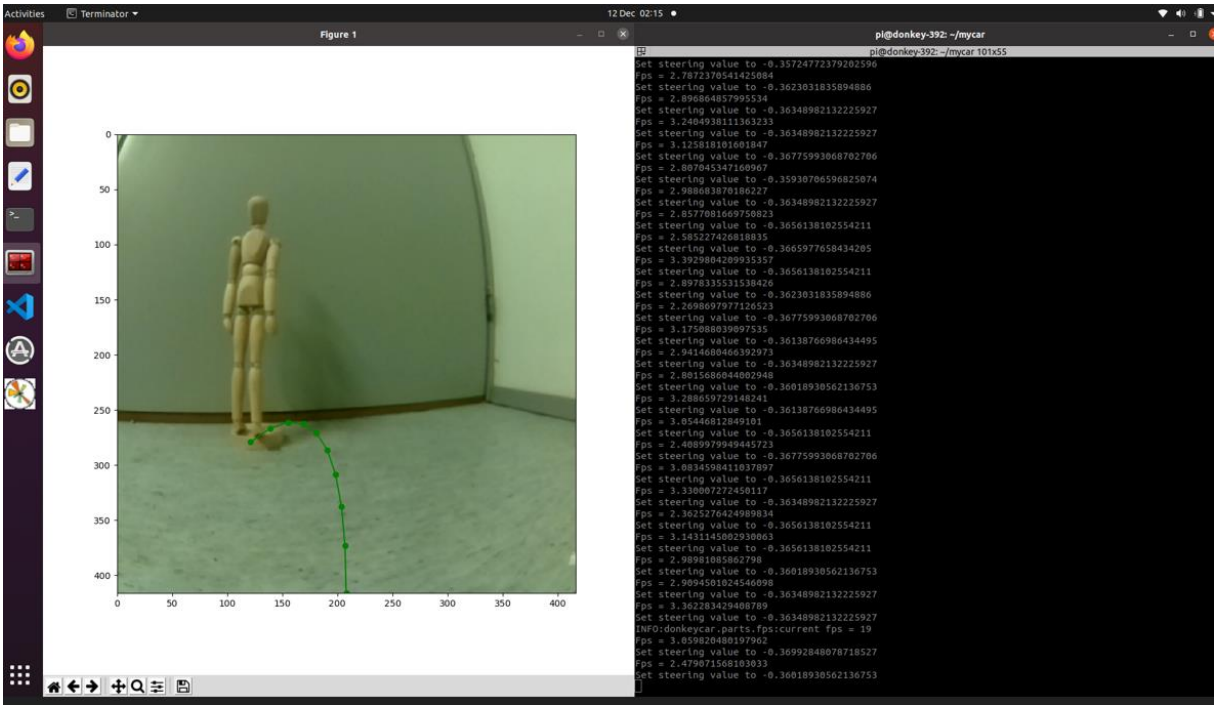

Figure 26: The faster pure pursuit pipeline in motion. The green line denotes the trajectory (bottom middle of the "person")

Running the planner partially on the Donkey Car was successful (Figure 26). The planner was able to infer steering values much faster than fully performing the pipeline on the vehicle (average inference rate was about 0.35s). While this is considerably faster than fully running the pipeline on the vehicle it is still not at an acceptable rate of inference. This could potentially be sped up even more by utilizing better data exchange sensors. Regardless, the algorithm was drastically improved, and it gives further cause into development of planning algorithms on the vehicle.

Based on the positive experiment results, a conclusion is reached regarding small-scale vehicles for autonomous driving research in the planning aspect of autonomous driving: small scale-cars are suitable for autonomous driving research in the area of planning and control, allowing the development and improvement of planning and control algorithms.

4.4 End-to-end experiment

Two variations of parameters were utilized in the creation of the models – the driving direction (counter-clockwise/left or clockwise/right) and time of day (day/night). These were used in order to create a variety of situations the vehicle might be introduced to (Table 2).

Description	Number of records
Left side driving during the day	20611
Right side driving during the day	21572
Left side driving during the night	22978
Right side driving during the night	22690

Table 2: Data collected for the end-to-end driving experiment

In order to drive the vehicle, there were two choices: the web controller provided by DonkeyCar, or a joystick. Both were attempted, and ultimately the joystick was chosen as it was the most intuitive to use, and matched more closely to how a steering wheel would be used to drive.

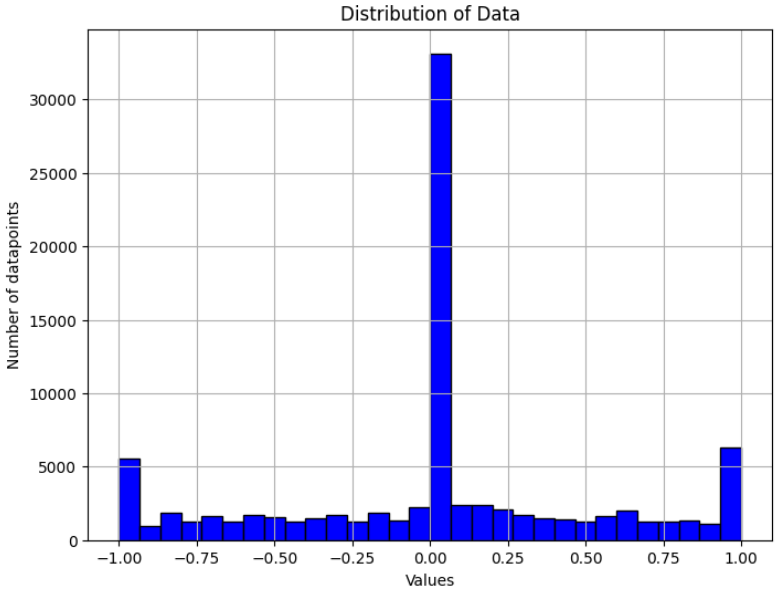

Figure 27: Distribution of data collected for end-to-end experiments

The data heavily was focused to the steering value of 0.0 (Figure 27), as the track itself has less turns then straight lines. As it was stated in the paper [44], data balancing was not necessary, hence it won't be performed.
All models were successfully trained and ran as can be seen in Table 3.

Model	Interventions	Whiteness [°/s]
DonkeyCar v1	18	38.64
DonkeyCar v2	20	36.76
Lexus Car (steering angle v1)	18	33.05
Lexus Car (steering angle v2)	12	31.94

Table 3: End-to-end experiment results. DonkeyCar v1 model represents the model trained only of right-side day data. DonkeyCar v2 model represents the model trained with all data collected. Both models were inferred using the .tflite model file generated

As we can see from the table above, the Donkey Car was able to match the performance of a real-life vehicle quite successfully, having similar number of interventions and even the same smoothness.
The models that were used in experiments are the standard models provided by the developers, which would imply that better performing models could be developed and possibly improve these results.
During both data collection and model testing, the vehicle was slowing down as time went on, which affected the performance of the models. This meant that the vehicle was able to successfully execute end-to-end driving around 50% of the battery life (corresponding to 10 minutes), as the slowdown of the car speed caused it to stop moving. This also meant that it was not possible to create a model that is able to reliably accelerate.
This is further confirmed by the issue of the locations in which the interventions were most frequent. Those were the bottom left, and top right corner of the track seen in Figure 20. These sections allowed less room for errors during driving and needed a specific speed of the vehicle set while turning, otherwise they would crash or get stuck. Having a better-quality battery, or a system which takes into account battery life to modify the speed would potentially solve this problem. While this impacts the development of any algorithms that would need long-term testing, the platform is able to perform on par with real life vehicles on small-scale tracks.

Based on the experiment results, small-scale cars are suitable for end-to-end autonomous driving research, closely matching real-life car models in their performance.

5 Conclusions and future work

This chapter explains the conclusions reached from the thesis work and any future work that should be performed.

5.1 Conclusions

This thesis tackled the suitability of small-scale vehicles for autonomous driving research. Many different aspects, which are important in terms of a small-scale platform's suitability for autonomous-driving research (such as perception, localization, end-to-end), were covered in the study, and evaluated through multiple experiments and investigations.
Regarding perception, the small-scale car is able to utilize sensors to perceive the environment, however the hardware prevents the object detection to perform at an acceptable rate. Regarding localization and mapping, the small-scale car is able to perform on par to a real-life vehicle. Regarding planning and control, the small-scale car is able to facilitate planning and control research, albeit on a smaller level than a real car due to hardware. Regarding end-to-end, the vehicle is able to match a real-life car with smoothness.
Overall, the experiment results shown in this thesis confirmed that small-scale vehicles can be used for autonomous driving research instead of a large real-life vehicle, producing acceptable results in areas evaluated on.

5.2 Future work

The work done in this thesis has revealed several avenues that could be researched further. During testing, there were several issues with the hardware of the vehicle platform (primarily compute and battery issues) not matching the performance of a real-life vehicle. It is necessary to find a hardware configuration that would create an acceptable platform for autonomous driving research without sacrificing the cost and the size of the vehicle.
There is also the question of demonstrating all the autonomous driving capabilities in the same environment, thus an experimental environment could be designed for both a large-scale vehicle and small-scale vehicle in order to create a more grounded comparison and demonstration of the capabilities of small-scale vehicle. This city could then be used as a testing ground for research.

Acknowledgements

I would like to dedicate this thesis to my sister Marija Sponza, as writing this thesis would not be possible without her invaluable support and belief in me. I would also like to thank my brother-in-law Mateja Sponza, my mother Nada Petrović and father Miroslav Petrović for their continued support. I would also like to thank my supervisor Naveed Muhammad for the support and help during the writing and thesis work.

Uroš Petrović

References

[1] Chandavarkar, N. D., & Nethravathi, P. S. (2023). SWOT Analysis on AI-based Self-driving Car Companies. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(3), 89-102. DOI: https://doi.org/10.5281/zenodo.820571

[2] B Hettige, RGS Deemantha. "Autonomous Car: Current Issues, Challenges and Solution: A Review". Presented at the 15th International Research Conference. Draft accessible online: https://www.researchgate.net/publication/366986201_Autonomous_Car_Current_Issues_Challenges_and_Solution_A_Review

[3] Uroš Petrović. (2023). GitHub - uroshpet/master_thesis. GitHub. https://github.com/uroshpet/master_thesis/

[4] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," in IEEE Access, vol. 8, pp. 58443-58469, 2020, doi: 10.1109/ACCESS.2020.2983149. Draft accessible online: https://ieeexplore.ieee.org/abstract/document/9046805

[5] SAE levels of driving AutomationTM refined for clarity and international audience. (n.d.). https://www.sae.org/blog/sae-j3016-update

[6] Pollard, T. (2023, November 1). What are autonomous car levels? Levels 1 to 5 of driverless vehicle tech explained. CAR Magazine. https://www.carmagazine.co.uk/car-news/tech/autonomous-car-levels-different-driverless-technology-levels-explained/

[7] DRIVE PILOT Automated driving. (n.d.). Mercedes-Benz USA. https://www.mbusa.com/en/owners/manuals/drive-pilot

[8] Dow, J. (2023, October 10). We tested Waymo's driverless taxi in LA in the perfect chaos of Venice Beach. Electrek. https://electrek.co/2023/10/08/waymos-driverless-taxi-tested-perfect-chaos-venice-beach-weekend/

[9] Robson, K. (2023b, March 9). Fully self-driving cars unlikely before 2035, experts predict. Verdict. https://www.verdict.co.uk/fully-self-driving-cars-unlikely-before-2035-experts-predict/

[10] Connor Hoffman. "How Capable Is Tesla's Autopilot Driver-Assist System? We Put It to the Test" 2021. https://www.caranddriver.com/news/a35839385/tesla-autopilot-full-self-driving-autonomous-capabilities-tested-explained/ 18.05.2023.

[11] Ranga, Adithya & Giruzzi, Filippo & Bhanushali, Jagdish & Wirbel, Emilie & Pérez, Patrick & Vu, Tuan-Hung & Perrotton, Xavier. (2020). VRUNet: Multi-Task Learning Model for Intent Prediction of Vulnerable Road Users.

[12] Wang, C., Wang, X., Hu, H. et al. On the Application of Cameras Used in Autonomous Vehicles. Arch Computat Methods Eng 29, 4319–4339 (2022). https://doi.org/10.1007/s11831-022-09741-8

[13] Syed, S. U. H. (2022). Lidar sensor in autonomous vehicles. ResearchGate. https://www.researchgate.net/publication/359263639_Lidar_Sensor_in_Autonomous_Vehicles

[14] Yatim, Norhidayah & Buniyamin, Norlida. (2015). Particle filter in simultaneous localization and mapping (Slam) using differential drive mobile robot. Jurnal Teknologi. 77. 10.11113/jt.v77.6557. Draft accessible online: https://www.researchgate.net/publication/286763834_Particle_filter_in_simultaneous_localization_and_mapping_Slam_using_differential_drive_mobile_robot

[15] Inam Ullah, Xin Su, Xuewu Zhang, Dongmin Choi, "Simultaneous Localization and Mapping Based on Kalman Filter and Extended Kalman Filter", Wireless Communications and Mobile Computing, vol. 2020, Article ID 2138643, 12 pages, 2020. https://doi.org/10.1155/2020/2138643

[16] Thrun, Sebastian & Montemerlo, Michael. (2006). The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. I. J. Robotic Res.. 25. 403-429. 10.1177/0278364906065387.
PDF:http://robots.stanford.edu/papers/thrun.graphslam.pdf

[17] V. Landa, B. Ben-Moshe, S. Hacohen and N. Shvalb, "GoIn - An Accurate 3D InDoor Navigation Framework for Mobile Devices," 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Nantes, France, 2018, pp. 1-8, doi: 10.1109/IPIN.2018.8533810.

[18] H. Gao, X. Zhang, J. Wen, J. Yuan and Y. Fang, "Autonomous Indoor Exploration Via Polygon Map Construction and Graph-Based SLAM Using Directional Endpoint Features," in IEEE Transactions on Automation Science and Engineering, vol. 16, no. 4, pp. 1531-1542, Oct. 2019, doi: 10.1109/TASE.2018.2883587.

[19] Teng, S., Hu, X., Deng, P., Li, B., Li, Y., Ai, Y., Yang, D., Li, L., Xuanyuan, Z., Zhu, F., & Chen, L. (2023). Motion Planning for Autonomous Driving: the state of the art and future Perspectives. IEEE Transactions on Intelligent Vehicles, 8(6), 3692–3711. https://doi.org/10.1109/tiv.2023.3274536

[20] Do, Huy & Mita, Seiichi & Tehrani, Hossein & Han, Long. (2013). Dynamic and Safe Path Planning Based on Support Vector Machine among Multi Moving Obstacles for Autonomous Vehicles. IEICE Transactions on Information and Systems. E96.D. 314-328. 10.1587/transinf.E96.D.314.

[21] Sarah Xiang. Basic Pure Pursuit - BLRS Wiki https://wiki.purduesigbots.com/software/control-algorithms/basic-pure-pursuit

[22] Lazarowska A. A Discrete Artificial Potential Field for Ship Trajectory Planning. Journal of Navigation. 2020;73(1):233-251. doi:10.1017/S0373463319000468

[23] Bojarski, M. (2016, April 25). End to end learning for Self-Driving cars. arXiv.org. https://arxiv.org/abs/1604.07316

[24] Tampuu, Ardi & Matiisen, Tambet & Semikin, Maksym & Fishman, Dmytro & Muhammad, Naveed. (2020). A Survey of End-to-End Driving: Architectures and Training Methods. IEEE Transactions on Neural Networks and Learning Systems. PP. 1-21. 10.1109/TNNLS.2020.3043505. PDF: https://arxiv.org/pdf/2003.06404.pdf

[25] World Health Organization. “Road traffic injuries” https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries 18.05.2023

[26] Vincke, Bastien, Sergio Rodriguez Florez, and Pascal Aubert. 2021. "An Open-Source Scale Model Platform for Teaching Autonomous Vehicle Technologies" Sensors 21, no. 11: 3850. https://doi.org/10.3390/s21113850

[27] T. Yu, W. Lu, Y. Luo, C. Niu and W. Wu, "Design and Implementation of a Small-scale Autonomous Vehicle for Autonomous Parking," 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 2021, pp. 398-402, doi: 10.1109/CACRE52464.2021.9501311. Draft accessible online: https://ieeexplore.ieee.org/document/9501311

[28] Kannapiran, Shenbagaraj & Berman, Spring. (2020). Go-CHART: A miniature remotely accessible self-driving car robot. 10.1109/IROS45743.2020.9341770. Draft accessible online: https://www.researchgate.net/publication/345339459_Go-CHART_A_miniature_remotely_accessible_self-driving_car_robot

[29] Datar, Aniket & Pan, Chenhui & Nazeri, Mohammad & Xiao, Xuesu. (2023). Toward Wheeled Mobility on Vertically Challenging Terrain: Platforms, Datasets, and Algorithms. https://cs.gmu.edu/~xiao/Research/Verti-Wheelers/

[30] Donkey Car official website. https://www.Donkey Car.com/ 18.05.2023

[31] Raspberry Pi 4B documentation. PDF: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf 18.05.2023

[32] RoboHAT MM1 documentation. PDF: https://robohatmm1-docs.readthedocs.io/en/latest/ 18.05.2023

[33] Raspi Camera documentation: PDF: https://datasheets.raspberrypi.com/camera/camera-module-3-product-brief.pdf

[34] 1/16 wheel off road truck product description. https://www.amazon.com/YONGSHUO-High-Speed-RC-Cars/dp/B09LQ8T9NN?th=1 18.05.2023.

[35] RPLidar A1 documentation. PDF: https://bucket-download.slamtec.com/e680b4e2d99c4349c019553820904f28c7e6ec32/LM108_SLAMTEC_rplidarkit_usermaunal_A1M8_v1.0_en.pdf 18.05.2023.

[36] MPU9250 product specification datasheet. PDF: https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf Last access: 17.12.2023

[37] University of Tartu Autonomous Driving Lab Vehicle. (2023). https://adl.cs.ut.ee/lab/vehicle Last access: 21.12.2023

[38] Pendleton, S.D.; Andersen, H.; Du, X.; Shen, X.; Meghjani, M.; Eng, Y.H.; Rus, D.; Ang, M.H. Perception, Planning, Control, and Coordination for Autonomous Vehicles. Machines 2017, 5, 6. https://doi.org/10.3390/machines5010006

[39] Zheng, S.; Wang, J.; Rizos, C.; Ding, W.; El-Mowafy, A. Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis. Remote Sens. 2023, 15, 1156. https://doi.org/10.3390/rs15041156)

[40] Saat, S., Rashid, W. A., Tumari, M., & Saealal. (2020). HECTORSLAM 2D MAPPING FOR SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM). Journal of Physics, 1529(4), 042032. https://doi.org/10.1088/1742-6596/1529/4/042032

[41] Zhi, Cui. (2019). Research on Cartographer Algorithm based on Low-Cost Lidar. International Journal of Engineering Research and. V8. 10.17577/IJERTV8IS100060.

[42] Robot Operating System official website. https://www.ros.org/ 18.05.2023.

[43] Dwijotomo, A.; Abdul Rahman, M.A.; Mohammed Ariff, M.H.; Zamzuri, H.; Wan Azree, W.M.H. Cartographer SLAM Method for Optimization with an Adaptive Multi-

Distance Scan Scheduler. Appl. Sci. 2020, 10, 347. https://doi.org/10.3390/app10010347

[44] Aidla, R. (2022). Comparing Output Modalities in End-to-End Driving. [Master Thesis]. University of Tartu. Retrieved from: https://dspace.ut.ee/server/api/core/bitstreams/7770ca99-a16f-4534-888b-e62bdf518da0/content

[45] Bjajoh. (n.d.). ROS driver with imu + magnetometer fusion for a Raspberry Pi. GitHub. https://github.com/bjajoh/ros-mpu9250-ahrs

[46] Niru. (n.d.). GitHub - niru-5/imusensor: Python library for communication between raspberry pi and MPU9250 imu. GitHub. https://github.com/niru-5/imusensor

Non-exclusive licence

I, Uroš Petrović

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

“Small-scale cars for autonomous driving research”

supervised by Naveed Muhammad

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons’ intellectual property rights or rights arising from the personal data protection legislation.

Uroš Petrović
23.12.2023