Autonomous Offroad Mobility using the F1TENTH-Platform

Scientific thesis for obtaining the academic degree
Bachelor of Science (B.Sc.)
at Department Mobility Systems Engineering
of TUM School of Engineering and Design
of Technical University of Munich

Supervised by	Prof. Dr.-Ing. Johannes Betz
	Felix Jahncke, M.Sc.
	Chair of Automotive Technology
Submitted by	Moritz Wagner
Submitted on	30.04.2024

Table of Contents

List of Abbreviations

2D	2-Dimensional
2WD	Two-wheel drive
3D	3-Dimensional
4WD	Four-wheel drive
ACFR	Australian Centre for Field Robotics
AI	Artificial Intelligence
AOV	Autonomous Offroad Vehicle
ASIC	Application specific integrated circuit
ATV	All-Terrain Vehicle
AV	Autonomous Vehicle
AVS	Autonomous Vehicle Systems
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DC	Direct Current
DGC	DARPA Grand Challenge
DGPS	Differential Global Positioning System
EDGAR	Excellent Driving GARching
ELROB	European Land-Robot Trial
EM	Electromagnetic
EMI	Electromagnetic Interference
FCN	Fully Convolutional Network
FPGA	Field Programmable Gate Array

FPS	Frames per Second
GB	Gigabyte
GLONASS	GLObalnaya NAvigatsionnaya Sputnikovaya Sistema
GPS	Global Positioning System
GPU	Graphics Processing Unit
HD	High Definition
HSI	Hue Saturation Intensity
IED	Improvised Explosive Device
IMS	Inertial Navigation System
IMU	Inertial Measurement Unit
JPL	Jet Propulsion Laboratory
LiDAR	Light Detection and Ranging
LiPo	Lithium Polymer
MCA	Modular Controller Architecture
MEMS	Micro-Electro-Mechanical Systems
MuCAR-3	Munich Cognitive Autonomous Robot car, 3rd generation
NASA	National Aeronautics and Space Administration
NMEA	National Marine Electronics Association
PC	Personal Computer
Radar	Radio Detection and Ranging
RASCAL	Robust Autonomous Sensor Calibrated All-terrain Land-vehicle
RAVON	Robust Autonomous Vehicle for Offroad Navigation
RC	Remote Control
RGB	Red Green Blue
RHex	Hexapedal robot
ROI	Region of Interest

ROS	Robot Operating System
RSC	Rockwell Scientific
RSPMP	Real-time Semantic Perception and Motion Planning
RTK-GPS	Real Time Kinematic Global Position System
SoC	System on a chip
TOF	Time-of-Flight
TTL	Transistor-Transistor Logic
TUM	Technical University of Munich
UniBw	University of the Bundeswehr, Munich
USB	Universal Serial Bus
VESC	Vedder Electronic Speed Controller
VW	Volkswagen
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WSL	Windows Subsystem for Linux

List of Symbols

Symbol	Unit	Description
a	-	Sine measurement for LiDAR scan angle
$angle_{\mathrm{inc}}$	-	Angle increment between LiDAR scan lines
$angle_{\max}$	-	Maximum LiDAR scan angle
$angle_{\min}$	-	Minimum LiDAR scan angle
apm	1/m	Angle change per meter traveled when turning
b	-	Cosine measurement for LiDAR scan angle
$boundary_{\mathrm{l}}$	m	Left boundary of the top-down map
$boundary_{\mathrm{n}}$	m	Near boundary of the top-down map
$center_x$	pixel	Center of the camera image in horizontal direction
$center_y$	pixel	Center of the camera image in vertical direction
$constant_x$	1/mm	Focal length scaling factor in horizontal direction
$constant_y$	1/mm	Focal length scaling factor in vertical direction
$\mathrm{depth}(u, v)$	m	Depth measurement at coordinates u, v
f	mm	Camera focal length
fac_{gps}	-	GPS heading influence factor
f_x	mm	Camera focal length in horizontal direction
f_y	mm	Camera focal length in vertical direction
h_{fused}	-	Fused heading measurement
$h_{\mathrm{gps\_mag\_avg}}$	-	Averaged GPS and magnetometer heading

h_{mag}	-	Magnetic heading measurement
$h_{\mathrm{mag\_current}}$	-	Current magnetometer heading reading
$h_{\mathrm{mag\_timed}}$	-	Magnetometer heading reading at GPS fix time
l	m	Length of a planning arm
mtd	m	Minimum turning diameter of the vehicle
m_x	T	Magnetic field strength in vehicle forwards direction
m_y	T	Magnetic field strength in vehicle leftward direction
μ_{s}	-	Temporal low-pass filtered mean saturation value
n	-	Total number of LiDAR scan lines
nos	-	Number of segments in planning path
res	m/pixel	Length-resolution scaling for image
sf	-	Steering factor modulating steering
s_{off}	-	Upper cut-off value for saturation normalization
$steer_{\mathrm{max}}$	-	Maximum possible steering angle
$steer$	-	Steering angle
$\mathrm{s_w}(x, y)$	-	Saturation value of a pixel at x, y in HSI
u	-	Pixel coordinate in horizontal direction
$\boldsymbol{u}$	-	Vector of all horizontal pixel coordinates
v	-	Pixel coordinate in vertical direction
$\boldsymbol{v}$		Vector of all horizontal pixel coordinates
v_{f}	m/s	Vehicle forwards velocity
v_{w}	pixel	Vehicle width in planning
x	m	Position coordinate in vehicle forward direction
y	m	Position coordinate in vehicle left direction

$\boldsymbol{y}$	m	Vector of position coordinates in left direction
z	m	Position coordinate in vehicle upwards direction
$\boldsymbol{z}$	m	Vector of position coordinates in upwards direction

1 Introduction

“It is anticipated that ordinary vehicles will one day be replaced with smart vehicles that are able to make decisions and perform driving tasks on their own” [1, p. 1].

The field autonomous vehicles (AV) profits from an ever increasing body of research focusing on on-road driving in urban or highway environments [2], where issues such as pedestrian safety [3] or traffic sign detection [4] are possible topics. However, less research exists on the topic of autonomous offroad vehicles (AOV), which nevertheless is an active and developing field [5] with tremendous potential in defense, agriculture, nature conservation or search and rescue [6]. Projects in this area include dirt-road and path following [7], the autonomous operation of heavy off-highway industrial equipment [8], mapping in cooperation with areal observers [9] or fully autonomous cartography on unknown terrain [10].

As an emerging field in the realm of modern automotive research, there are also approaches to teach and test autonomous vehicle systems with hands-on equipment made from commercially available hardware on a much smaller scale than full-size vehicles [11, 12]. Here, students and researchers can test and experiment freely on inexpensive hardware, and learn firsthand from scalable examples with real-world algorithms and hardware [13]. This is not only important to prepare prospective engineers and researchers for the topics emerging in the field, but can also further innovation necessary to improve transportation and logistics in the future [11].

These two subfields of autonomous vehicle systems research can lie on opposite ends of the spectrum, and this thesis will try to answer the question of how to combine them. Is it possible to achieve the same level of scalable, directly applicable research and development shown by miniature platforms when set to the task of autonomous offroad navigation?

Can a small-scale remote-controlled vehicle achieve realistic autonomous offroad navigation?

1.1 The Place of this Work in the Field of Autonomous Vehicle Systems

Researchers already work on the problem of autonomous offroad driving, developing many different vehicles for the job. However, these vehicles are without fail large and complex, because their creators build them on the basis of existing offroad vehicles or platforms [14, 15]. While this allows them to incorporate large suites of different sensors, powerful computers as well as complex actuation and control systems, the author of this thesis stipulates that this approach has the following drawbacks:

Firstly, operating large vehicles poses a certain danger in operation. Errors and accidents are potentially dangerous or fatal to bystanders, which necessitates strict, clear oversight and control over any given project.

Secondly, the development of such vehicles comes with an excessive cost. Full-scale research vehicles are prohibitively expensive, because they not only encompass an entire base vehicle or buggy, but must also provide sensors for multiple ranges, emergency systems and custom actuators for control. This limits them to senior teams at financially strong, distinguished universities, which confines the amount and range of research possible in the realm of offroad robotics.

Alternatively, studies and concepts for autonomous vehicles in a non-offroad setting, which take advantage of smaller platforms like remote control (RC) cars [13] are plentiful and have clear advantages in comparison to larger solutions. The procurement of their platforms, sensors, and hardware is much cheaper while the resulting vehicle is also less dangerous and complicated to operate, which leads to faster progress and more rapid cycles of development and testing.

The aim of this thesis is now to combine the aspects these two approaches (Figure 1.1), creating an inexpensive, small-scale, capable offroad vehicle testbed that utilizes full-scale algorithms and sensor technology for the purpose of research into autonomous offroad robotics.

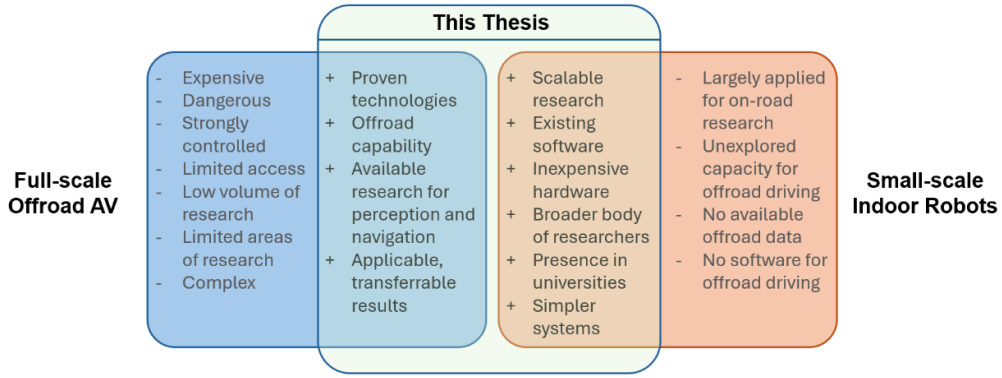


Figure 1.1: Intended characteristics of this thesis' proposed vehicle compared to existing solutions.

1.2 Structure of this work

This thesis will open by assessing the state of the art in the field of autonomous offroad vehicles in Chapter 2 'State of the Art', with examples on multiple scales, with differing tasks and objectives. This chapter will also include specific works relating to sensorics, algorithms used for perception purposes and navigational methods or techniques used by researchers in the field and in the laboratory.

Next, Chapter 3 'Method' will detail the chronological development of the vehicle platform it set out to create in the previous section, with first analyzing the state of the art to gain valuable insight into the specific hard- and software suitable to this task. Next, the chapter presents initial setups and tests, before moving on to the development of the perception software interfacing with the vehicles sensorics systems. It will also include a chapter showing the special integration of the GPS sensor, before moving on to the development and testing of the navigational system. Lastly, Chapter 3 will close with the final modifications made to the setup based on insight gained during the development and incremental testing.

After this, Chapter 4 'Results' will present the findings of these tests conducted during and after development. First, since the actual creation of the platform is an important part of the problem statement of this thesis, the chapter will give an overview of the hard- and software of the vehicle.

Next, it presents the results of hard- and software specific tests designed to give insight into the workings of specific components or their interactions, as well as system tests in the laboratory and in the field.

Then, the text will move onto Chapter 5 ‘Discussion’, which analyzes and interprets the results of the previous tests. This will include assessments of the systems computational and overarching performance in the field, as well an evaluation of the presented results’ validity and applicability. Finally, this section will deliver an in-depth comparison of the presented system with its perceived ‘parents’, the full-scale AOV and small-scale research vehicle.

Finally, Chapter 6 ‘Conclusion’ will draw the thesis to a close with a comprehensive listing of its contributions as well as possible further tasks for research on and with this platform. Figure 1.2 schematically presents the overarching structure of the work.

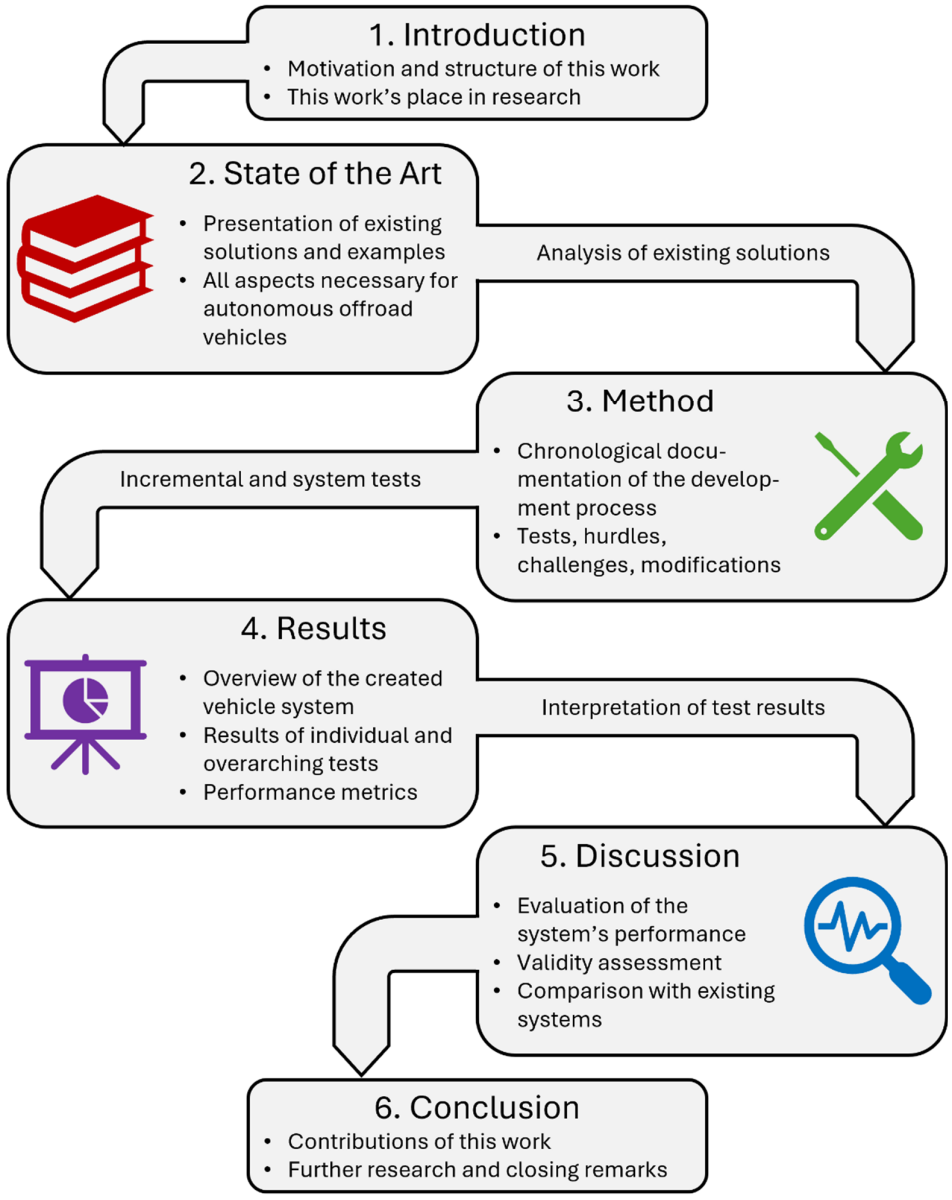


Figure 1.2: Schematic representation of the structure of this thesis.

2 State of the Art

The field of autonomous robotics is one of the most active research areas in modern technology. Within it, the specialized subject of autonomous offroad vehicles is one of the fastest developing subfields, with many different actors such as universities [10], private companies [16] and parts of the military industrial complex hedging interests [17]. It is the aim of this Chapter to provide an overview of the state of the art in this field, with a special focus on concepts applicable to wheeled, offroad robots using a standard car-like platform.

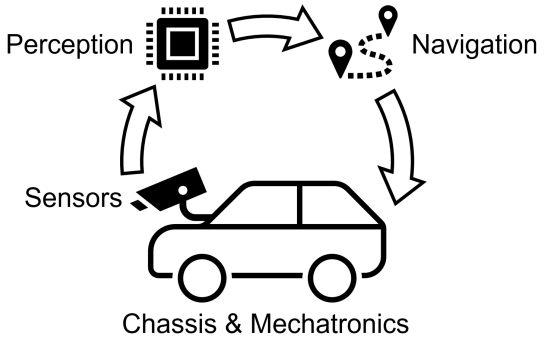


Figure 2.1: Overview of the autonomous offroad driving process, adapted and simplified from [4].

Analogous to Figure 2.1, this chapter will group relevant preliminary, advanced, and alternative works and their results into the subcategories of Mechatronics and Chassis, Sensorics, Perception as well as Navigation. These constitute the main building blocks for current, state of the art autonomous offroad robotics, and indeed autonomous vehicles as a whole [2, 4, 18].

While a concise framework for this thesis may suggest a focus strictly on technology and techniques applicable in off-road settings, this thesis deems the presentation of a limited number of relevant on-road solutions beneficial. As such, the scope of this chapter slightly widens to include a select number of road-based works from more ‘traditional’ autonomous vehicle concepts over the range of the presented subtopics.

2.1 Mechatronics and Chassis

To facilitate effective integration of autonomous driving technologies, a project must first adopt a suitable mechanical platform. Where standard on-road driving is a topic broad enough to have entire books – such as [19] – detailing it, AOV are a relatively scarce topic in scientific literature. So, it seems plausible to assess necessary modifications or novel approaches for a prospective platform supporting a project in offroad robotics, as opposed to the established systems of on-road vehicles.

2.1.1 Laying the Groundwork: DARPA and ELROB

One of the most influential events in the field of off-road autonomous driving and also for Autonomous Vehicles as a whole where the United States' Defense Advanced Research Projects Agency's (DARPA) Grand Challenges (DGC), in 2004 and 2005 [17]. These proved to be a great driver of innovation in the field and led to the creation of many forward-thinking Autonomous Offroad Vehicles such as Stanley by Stanford University [14], or the entries of the SciAutonics teams from Rockwell Scientific (RSC) and Auburn [15, 16] (Figure 2.2).

Figure 2.2: Different (and similar) approaches to Offroad Autonomous Driving. On the left, Stanley of Stanford University [14], and on the right AVIDOR-2004 of Team SciAutonics [16].

Stanley is the winning prototype developed by Stanford University for the 2005 installment of the DGC (Figure 2.2, left). The desert course provided in the DGC was demanding, because while it technically had a road, said road was little more than a simple track of slightly compacted dirt through the Nevada desert [17]. To combat these difficult driving conditions and avoid building an entirely new vehicle, the Stanford researchers opted to instead modify an existing vehicular platform: a Volkswagen (VW) Touareg R5. Their paper [14] states the reasons for this: "The Touareg has four-wheel drive (4WD), variable height air suspension, and automatic electronic locking differentials" [14, p. 664]. Additionally, they installed skid plates and a reinforced front bumper.

RSC's SciAutonics-Teams pursued an interesting alternative approach contrary to that of the Stanford researchers. Instead of opting for a standard on-road, production vehicle, they utilized modified All-Terrain-Vehicles (ATV) from two different manufacturers. The first team used a Prowler vehicle from ATV Corporation, a 4WD buggy with a roll cage and manual differentials, which they then modified to fit their sensory systems. Their second Team utilized a similar vehicle, made by TOMCAR, which is only equipped with two wheel drive (2WD) (Figure 2.2, right) [15, 16]. Achieving an overall second place in the tournament, this second team's vehicle failed after 10.8 km when it veered off course because of an GPS error. After leaving the planned route, the vehicle dug into a sandy patch next to the track and became stuck, ending the trial [7].

Another important benchmark and competition in offroad AV is the European Land Robot Trial (ELROB), which led to the creation of similar AOV systems, which Appendix A further presents.

2.1.2 Small-Scale Dedicated AOVs

Moving from existing platforms and vehicles which still fit human occupants to smaller, more specialized offroad robots enables researchers to individually improve and specialize the design of the chassis, mechatronics, and propulsion systems. Thus, they can implement yet entirely

untried and unexplored approaches to offroad mobility in robotics, such as tracked or legged robots, changes in the number of wheels and other modifications, examples of which the following paragraphs will present.

The first example, Endeavor Robotics' (formerly iRobot) PackBot, is an industry standard [20]. A joint team from the Southwest Research Institute, the University of Texas at Austin, and Griffin Technologies, Inc. developed a mobility test suite, and tested it on two small AOVs in [20]. PackBot, which contains a "combination of 'standard' tracks, front-mounted articulator arms, and a dual-motor drive system" [20, p. 5] is capable of flipping itself right-side-up if the need arises. It is a remarkably capable platform in comparison to the other tested robot, with a particular advantage in speed, slope climbing ability, resistance to stepped obstacles such as curbs, and energy efficiency [20]. However, researchers in [21] have noted that its low build and center of gravity together with poor odometry because of its tracked nature and low quality Inertial Measurement Unit (IMU) limit its usefulness in applications that require Localization and/or Mapping.

The other robot analyzed in [20] is RHex (Hexapedal Robot) (Figure 2.3, left). A joint team from Boston Dynamics, McGill University in Montreal and AutoVu Technologies developed the mobile platform, with [22, 23] outlining the robot's capabilities. In comparison to PackBot, RHex appears to have better mobility and speed only in certain types of terrain, for example Rock Beds [20]. For its locomotion, the RHex robot utilizes six compliant (e.g., bendable) legs, each of which rotates about its pivot with only one motor. The advantage of this system as compared to a robot utilizing a wheeled approach is greater, more precise control over ground reaction forces (e.g. traction) in terms of direction and magnitude. [23]. Several different versions of RHex exist, introducing shock proofing, water ingress resistance and amphibious capabilities to the platform [22].

Lastly, researchers at the University of Sydney's Australian Centre for Field Robotics (ACFR) have developed a wheeled mobile robot in their Swagbot (Figure 2.3, on the top left). It is an all-wheel independently driven and steered, 4WD-capable electric ground vehicle, which can move omnidirectionally, allowing it to excel in uneven terrain [24]. The wheel-strut assemblies connect to a central rocker mechanism, allowing the Swagbot to traverse low lying obstacles and stay level even in challenging terrain [25].

Figure 2.3: On the right, one example of the RHex platform [22] and on the left, ACFR's Swagbot, a wheeled robot for agricultural applications [25].

2.1.3 The Challenges of extreme off-road driving on fully unknown Terrain

Development of most of the examples up until now relied on prior knowledge of the route through the terrain or are designed to operate on preexisting paths formed by other vehicles or human

movement, e.g. Stanley and the other DGC vehicles on their desert track [17], or ACFR's Swagbot on relatively flat and traversable fields [25]. As such, researchers build these vehicles specifically to excel on a given type of surface or route, and thus the precomputed route itself – or the remote operator – can compensate for vehicle capabilities and shortcomings in the path planning and driving. Researchers working on mobility in fully unknown, extreme terrain however create AOV with different setups, shown with the following examples.

Figure 2.4: Robots for extreme off-road terrain: left 'Tracked', and right 'Ackermann', which [26] tests and evaluates in tunnels and mines.

Researchers in [26] present tests of multiple AOV in several extremely challenging terrains such as mines and tunnels. Of particular interest are 'Tracked' and 'Ackermann' (Figure 2.4), the former using combining tracked and wheeled capabilities, while the latter utilizes a traditional four-wheeled RC-car chassis. The researchers built the 'Tracked' Vehicle on the basis of a Faulhaber Telemax PRO [26], which is a four-wheeled, four-tracked platform, capable of up to 10 km/h, 45° ascend over both sloped surfaces and stairs and a 10h battery life [27]. This system was tested in the Berkley Exhibition Coal Mine, traveling a total of 0.76km at 0.57m/s [26].

The Ackermann platform starts with a Traxxas X-Maxx RC monster truck platform [26], sporting all-time full 4WD with oil filled differentials to limit slip between tires when one loses traction, large tires and a high suspension for clearing obstacles combined with a low center of gravity and a comparatively short wheelbase [28]. The researchers tested it in the DARPA 'SubT' Competition, where it drove 0.39km at the second highest speed of the tested vehicles: 0.83 m/s [26].

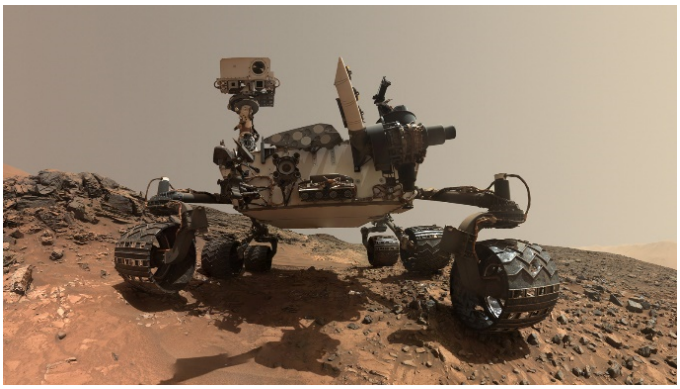


Figure 2.5: The Perseverance Mars rover [29], which NASA's JPL developed and built to explore the Martian surface in search of water

No compendium of extreme off-road rovers can be complete without mentioning the achievements of the various extraplanetary Mars Rovers built by NASA's Jet Propulsion Laboratory (JPL), the latest active example of which is the Perseverance Mars rover (Figure 2.5). Its creators envisioned it to combat the entirely unexplored and fully remote surface of Mars, so

they expressly designed its mobility system is to master this challenge without any human intervention. Perseverance is a six-wheeled rover equipped with a suspension of the rocker-bogie-type [30], six independent motors driving each wheel, and four additional motors above both front and rear wheels provide steering. Each of these motors also features a brake to prevent unwanted rotations. Featuring a maximum speed of 4.2 cm/s and the entirely passive suspension, which limits vehicle tilt and increases stability when traversing obstacles, it has traveled up slopes as steep as 25° and more than 21 km total as of August 2012 [30].

2.1.4 The F1TENTH Platform

Figure 2.6: On the left, a stock Traxxas Slash 4x4 model driving in its intended environment [31], on the right an example finished configuration of an F1TENTH vehicle based on the Slash chassis and drivetrain [32].

The F1TENTH Platform – shown in Figure 2.6 on the right – is an AV development platform which its creators designed to mimic real F1 racecars at ⅒ scale, in order to enable a scaled approach to research and development on the AV front [11]. At the same time it sets up an obvious path into competition, thus allowing for the comparison of developed algorithms and solutions [12]. This enables both researchers and students alike to develop algorithms and hardware for AVs rapidly, with fast access to real word testing, while negating risks, tribulations and cost connected to testing on an actual one-to-one scale vehicle [12].

The basis for the platform is forms a Traxxas Rally ⅒ scale RC car, specifically a variant of the Traxxas Slash 4WD Electric Short Course Truck [33], shown in Figure 2.6 on the left. In its base configuration, it features Servo-powered Ackermann Steering, a brushless direct current (DC) motor capable of propelling the car up to speeds of 100 km/h, and a 324mm wheelbase 4WD drivetrain, with Limited-Slip-like silicone-filled differentials to reduce traction loss. It also utilizes oil-filled shock absorbers with long travel and progressive springs, a modular chassis and a high torque-capable Aluminum Driveshaft [31].

To reconfigure this platform for the task of autonomous driving, several modifications are universally necessary to transform it into an AV platform. To facilitate easy, reliable and accurate motor control of both servo and brushless DC Motor, the platform uses an open source motor controller: the Vedder Electronic Speed Controller (VESC) [13, 33]. Additionally, researchers added a power distribution board to reliably power all onboard electronics off of the battery, as well as a mounting plate to mount components, additional sensors and computer hardware to [11].

Researchers of the University of Pennsylvania created the F1TENTH platform in 2016 [34] and first presented it in [13], where it is envisioned to be “an open-source evaluation framework with

virtual environments and a low-cost hardware counterpart which enables safe and rapid experimentation suitable for laboratory research settings" [13, p. 85]. As such, universities extensively use it to teach autonomous systems as shown in [11], where it is the basis for a full class. The F1TENTH platform there operated full-scale AV code and algorithms seamlessly, proved reliable, low-cost, and safe with access to a simulator to accelerate development.

Extending this concept, university research utilizing small-scale robots such as [33] and competition between different research institutes [12] is currently based on the F1TENTH platform, describing it as convenient and efficient, without the risks and costs associated with full-scale vehicles [12].

2.2 Sensorics

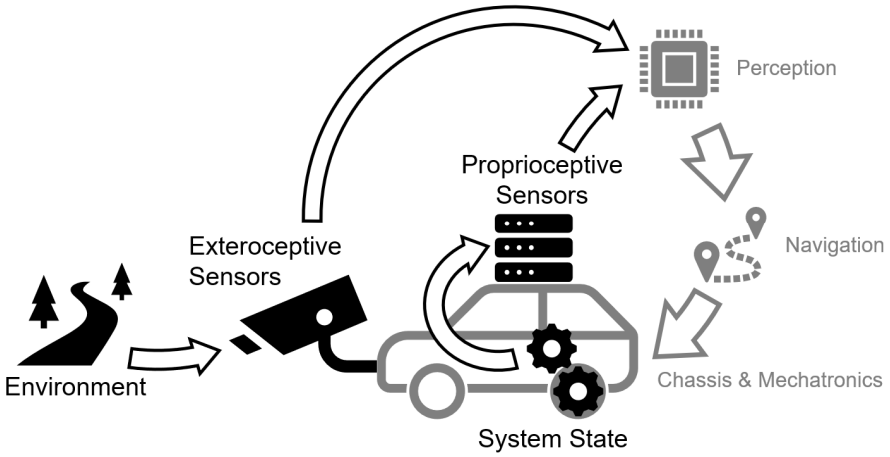


Figure 2.7: Overview of sensing systems on AV, on the basis of information from [2]

The second Pillar of AV are the sensorics systems which the vehicles employ to facilitate the detection and perception of its position and orientation, surroundings, other actors, as well as obstacle or signage detection (Figure 2.7). As there is a broad range of different sensor suites and combinations available, this section presents a selection of the most common, useful, or specialized to the task of offroad driving, along with several vehicle examples and studies using said sensors.

"Sensors have to be able to create both a perceptive and locational view of the environment so that the vehicle can make decisions in real-time" [35, p. 1]. Following this thinking, the following paragraphs group sensors and sensor systems into exteroceptive sensors, which are used for perceiving the environment, and proprioceptive sensors, which measure values within the system, e.g. speeds or positions [3, 35].

2.2.1 Exteroceptive Sensors

Exteroceptive sensors are devices capturing the external state – e.g. the environment – of the vehicle, for example distance measurements or light intensity from the system's surroundings [3]. Sensors which fall into this category are LiDAR sensors, Cameras, Radar sensors and ultrasonic

sensors [3]. The following sections will present LiDAR and Cameras, with Appendix B detailing radar and ultrasonic sensor capabilities and usage.

LiDAR Sensors

LiDAR – which stands for Light Detection and Ranging [3], sometimes called a laser range finder – uses a laser to determine the distance between the sensor and a target. Most LiDAR sensors use infrared light in the 900nm range, though longer wavelengths have been observed to perform better in adverse conditions such as fog and rain [2, 3]. The distance measurement works by sending out pulses of light, which then reflect off objects within the LiDAR's field of vision. Upon the return of the reflected light, a sensor detects the reflected light pulse, calculates the time lapse between the sending and the receiving of the pulse – the time-of-flight (TOF) [35] – and thus is able to calculate the distance between the sensor and the object. Most LiDAR then return this data in a point cloud, which represents individual scan-points with x-y-z coordinates, typically with the sensor in the origin of the coordinate system, and have a range of up to 250m [35, 36].

There are two main types of LiDAR sensors: mechanical LiDAR and solid-state LiDAR [3]. In a mechanical LiDAR, a spinning element – which spins around the vertical axis if produces a horizontal scan – enables both the laser and the detector to scan continuously, achieving up to 360° coverage around the sensor. Many of these sensors can scan on multiple lines at different heights or inclinations, adding a third dimension to their measurements. Different amounts of scan-lines and vertical Field-of-View (FOV) determine the spacing and vertical resolution of these devices, while the imaging capabilities of the detectors limit the horizontal resolution, which however is usually much higher than the vertical resolution [2, 3].

Solid State LiDAR come in three types. Flash-based means that they operate in part like a traditional photo camera, sending out a flash of light and recording an entire frame-array of reflections at the same time. Optical phase array-based devices use integrated photonics technology with micro-structured waveguides to direct the laser, and MEMS (Micro-Electro-Mechanical Systems) based systems use piezoelectric elements to vibrate a mirror to scan the laser beam. All of these require no mechanically moving parts and provide a smaller horizontal FOV, typically limited to 120° [3]. However, they provide an evenly spaced resolution in both directions, a more rugged and reliable design, as well as lower cost and size [2, 3, 37].

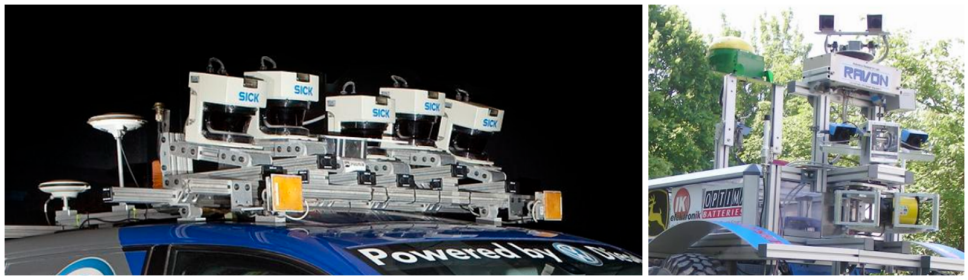


Figure 2.8: Different sensor systems on early AOVs, on the left Stanford's "Stanley" Robot's roof rack, with the SICK LiDAR sensors on top, a monocular camera below the middle sensor, and two orange-colored Radar sensors on either side of the rack [38]. On the right RAVON's sensor suite [39].

Many of the vehicles presented here use LiDAR: some of the earliest ones such as Stanley [14], the University of Kaiserslautern's Robust Autonomous Vehicle for Offroad Navigation (RAVON)

[40] (Shown in Figure 2.8) as well as SciAutonics' Robust Autonomous Sensor Calibrated All-terrain Land-vehicle (RASCAL) [15]. All of these use one or more LiDAR units manufactured by SICK in different configurations and setups. More current examples are the vehicles which [24] tests and the example configuration of the F1TENTH platform in [13]. The SICK units used are all of the rotating Line Scanner variety [41] and include both vertical and horizontal versions [15], as is the horizontal Hokuyo UST-10LX unit of the F1TENTH platform [13, 42]. Stanley and the F1TENTH platform use several planar two-dimensional (2D) scanners [14, 42] (which report only one line), while RAVON uses both 2D and three dimensional (3D) units [40], where the 3D LiDAR reports more than one line [41]. The University of the Bundeswehr's (UniBw) Munich Cognitive Autonomous Robot car, 3rd generation (MuCAR-3) also uses a 3D LiDAR [43].

Cameras

"Cameras are one of the most adopted technology for perceiving the surroundings [of autonomous Vehicles]" [3, p. 6]. By focusing reflected and scattered light from nearby surrounding surfaces in through a lens and detecting it on an image sensor, a camera can visualize and image an area of the environment. They provide a high definition, high refresh rate stream of information, which has a variety of applications. Downsides of Cameras include susceptibility to weather and environment conditions such as snow, light levels, sun glares or hazy weather, reducing effective quality of the images, as well as the large computational power required to analyze and process images. In addition, by increasing the FOV further and further into the range of fish-eye cameras, or with cheaper, sub-par optics, distortions appear in the camera feed, requiring computationally intensive intrinsic calibration to generate usable data. Common uses include perception of obstacles and other actors in the environment, their position and velocity, semantic segmentation (Section 2.3.1) and detection of barriers and signage. AV camera systems can employ either monocular or binocular camera systems, which could employ either color or black-and-white sensors [2–4, 35].

Conventional red-green-blue (RGB) monocular cameras provide 2D information, which is useful for classification and interpretation of terrain [2]. Due to the planar nature of the information received, their applications are limited compared to stereo cameras, which possess an advantage in areas such as position and velocity calculations [3]. One type of monocular cameras are fish-eye cameras, which [44] employs for the task of near-field sensing, where researchers use their particularly high FOV to create a "surround-view" system with only 4 cameras.

Stereo, or Binocular cameras mimic the depth perception mechanism found in animals, where the stereoscopic difference between the left and right images formed in each eye provide a sense of depth [3]. In these sensors, which separate the cameras by a specific distance, the disparity information is then used to calculate a depth map, thus inferring 3D data from the two planar camera sensors [3]. Some systems use 'stereo' camera systems employing more than two cameras, such as the Bumblebee XB3 FireWire, a triple-camera system which [45] employs, improving overall accuracy and range of the distance measurements.

Cameras are a ubiquitous technology [3] on AOV, with almost all of the examples in this thesis utilizing them. Stanley uses a singular, monocular color camera for long-range road perception [14] – shown in Figure 2.8 on the left in the center of the array – whereas MuCAR-3 uses a tri-camera system, but without a depth calculation [43]. The example F1tenth Platforms in [11] and [13], as well as RASCAL and RAVON use stereo camera systems [15], the latter of which uses two separate systems with different baselines for long and short range perception [40] (Figure 2.8 on the right). Additionally, Swagbot [25] and the vehicles in [26] use cameras in their

sensor arrays. The Mars rover ‘Perseverance’, currently on mission on the red planet [46], employs 23 cameras, which include stereo, monocular and fisheye cameras in both color and monochrome [47] (Figure 2.9). Some platforms notably omit cameras, such as the vehicle of the second SciAutonics team, AVIDOR 2004 [7], which instead uses a Global Positioning System (GPS) receiver, IMU, as well as ultrasound, LiDAR, and mm-wave radar. One configuration of the F1TENTH Platform [33] also opts to not utilize a camera.

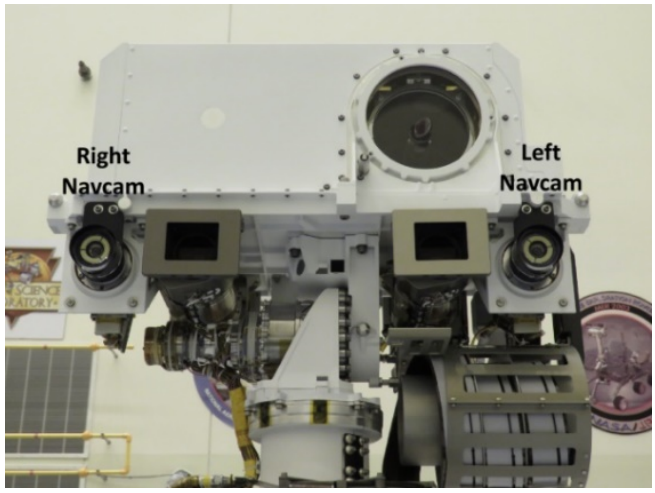


Figure 2.9: On the left, the cameras mounted on the Remote Sensing Mast of the Perseverance Rover, including the outer stereo Navcams, with another pair of stereo cameras in between them, and the SuperCam topping the Mast [47].

2.2.2 Proprioceptive Sensors

“Proprioceptive sensors, or internal state sensors, capture the dynamical state and measure[…] the internal values of a dynamic system, e.g. force, angular rate, wheel load, battery voltage, et cetera” [3, p. 5]. This section will present these, including encoders, IMU, magnetometers and GPS [3, 35] because they provide robust positional information import for offroad navigation [5].

Encoders

Providing information on the status of a control system, e.g., an electric motor, encoders provide odometer data via dead reckoning – which measures wheel rotations to account for distance traveled – or provide information for the state of steering servos to gain information about a vehicles rate of turn [35]. This is a cost-effective way to gain real-time information about the position of a vehicle relative to a starting point [35], although measurement errors due to traction loss and subsequent wheel slip can accumulate over the course of a journey, leading to large error increasing over time [48]. Researchers in [49] however show that it is possible to reduce this error with the use of neural networks.

The F1TENTH Platforms’ [11, 13, 33] VESC [50] uses encoders for odometry, as do the Werrimbi robot [48], RAVON [40] and all of [26]’s vehicles. The MuCAR-3’s camera system [43] explicitly mentions the use of an encoder to read an internal motor’s position, but other systems using motors for steering or other applications likely also utilize them, especially on other full-size vehicles requiring complex control mechanisms.

Inertial Measurement Units and Magnetometers

Inertial Measurement Units or IMU are electronic devices measuring a body's force, acceleration, angular rate of rotation and change thereof [35]. They accomplish this with accelerometers, gyroscopes and – if present – magnetometers arranged in an orthogonal configuration in order to obtain relevant data in all three dimensions [35]. Excluding the magnetometer, some systems combine the IMU's data and wheel odometry to integrate it into an Inertial Navigation System (INS). With an IMU, the vehicle can gain information about its current orientation, position and linear as well as rotational speed relative to a starting pose [35]. Since they operate on a similar principle for providing odometry as compared to encoders, IMU suffer from similar issues, such as drifting of the measurements due to accumulated errors [35].

Magnetometers are sensors which calculate a heading direction with reference to the magnetic north pole [51]. When calculating heading from the magnetic field, the sensor system must account for the tilt of the vehicle, for example with the IMU's accelerometers to determine the direction of gravitational pull. In the absence of such a correction, or when the corrections are inaccurate, a tilt out of the horizontal plane would reduce the measurement accuracy [51]. It is also worth noting that local variations of the earth's magnetic field and ferromagnetic materials in the proximity of the sensor can significantly influence a magnetometer, necessitating calibration [51].

As one of the standard sensors for AOV [5], virtually all examples shown here use IMU systems, especially the larger ones, while fewer ones employ magnetometers. RAVON [40] and RASCAL [16] use IMU and magnetometers in conjunction, while MuCAR-3 only uses an IMU coupled to a GPS [43], which can be effective at combating the drift issues by providing an absolute reference [35]. F1TENTH platform's [11, 13, 33] VESC unit provides IMU data only [50], similar to the examples shown in [26].

GPS Receivers

"GPS is a satellite-based radio-navigation system that provides geolocation and time information to a GPS receiver anywhere on Earth as long as there is an unobstructed line of sight to four or more GPS satellites" [35, p. 2]. It does this by trilateration, a method with calculates the location of a point relative to others using the distances between them, where these distances are obtained by TOF of the signals [52]. The accuracies of these receivers lie between one to three meters [35, 52], and other global navigation satellite systems such as GLONASS and Galileo [53] can further increase this accuracy, with the drawback that such receivers are generally more expensive [35]. Another way to increase GPS accuracy is with the use of a differential GPS (DGPS) system whereby a base station relays corrections to the GPS data to a mobile unit, enabling accuracies in the centimeters [54]; however these systems usually come at a high cost [55]. The other main drawback of GPS systems is the quick degradation of their accuracy when the line of sight to the satellites is obstructed, for example by trees, tall buildings or overhangs in the surrounding terrain [35].

All of the larger vehicles in this work such as RAVON [40] and RASCAL [15] employ GPS Systems. Stanley uses a GPS compass system in addition to a normal GPS [14], which allows it to receive high accuracy heading information without the use of an magnetometer; similar to MuCAR-3 [43]. The AVIDOR-2004 vehicle of the second SciAutonics team relied on an early DGPS System, however it encountered GPS difficulties, causing it to drift of course, and – without other significant guiding systems such as cameras – get stuck, ending its race [7].

2.2.3 Sensor configurations overview

Considering this works' many different sensor and AV examples, it is beneficial to gain an overview of the different configurations and sensor combinations. The intended tasks performed, costs, ease of use and of course the target environment strongly influence sensor selection and system design [2].

Table 2.1: Exteroceptive sensors usage percentage in this thesis' examples, grouped by type, in comparison to EDGAR's sensor configuration, for which checkmarks and crosses denote sensor presence and absence respectively. Expanded table in Appendix C.

	LiDAR		Camera			Ultrasonic
Vehicle Types	2D	3D	mono	stereo	Radar	Sensors
Full-Size [14–16, 40, 43]	80%	20%	40%	60%	60%	20%
F1TENTH [11, 13, 33]	66%	/	/	66%	/	/
EDGAR [54]	✗	✓	✓	✓	✓	✓

To now compare the presented configurations to a newer AV system, this section will comparatively list the examples in addition to the sensor system used on EDGAR (Appendix C), an Autonomous Driving Research Platform developed at the Technical University of Munich (TUM) [54].

As a current example AV research, its sensor suite is a valuable sample of current sensor technology, even though it is not purpose-build for offroad applications like the other, older examples shown before. Table 2.1 shows a comparison of exteroceptive sensor configurations on this thesis' examples, with proprioceptive sensors compared in Table 2.2. For the purposes of these tables, 'Full-size' vehicles encompass AVIDOR-2004 [16], Stanley [14], RASCAL [15], RAVON [40] and MuCAR-3 [43].

Table 2.2: Proprioceptive sensors usage percentage in this thesis' examples, grouped by type, compared with EDGAR's setup. Values in parentheses are assumptions with not explicitly stated sensors, which this thesis assumes to be present. Full table in Appendix C.

		IMS		GPS	
Vehicle Names	Encoders	IMU	Magnetometer	position	heading
Full-Size [14–16, 40, 43]	60% (100%)	80% (100%)	40%	100%	40%
F1TENTH [11, 13, 33]	100%	100%	/	/	/
EDGAR [54]	(✓)	✓	✗	✓	✓

2.3 Perception

The ability to infer information and gain knowledge from the provided sensor data is the next core element of Autonomous Vehicle systems [18, 56, 57] and this section will split it into environmental perception and localization [56]. Designing a perception system is a uniquely

challenging task, with many different schemes available for different combinations and setups of specific sensors, and fusion algorithms combining data and information from different sensors to make the system more robust and reliable [18]. Figure 2.10 shows the general structure of perception systems on AV. Additionally, it is worth analyzing the hard- and software platforms which run and coordinate different algorithms and process their respective data [57].

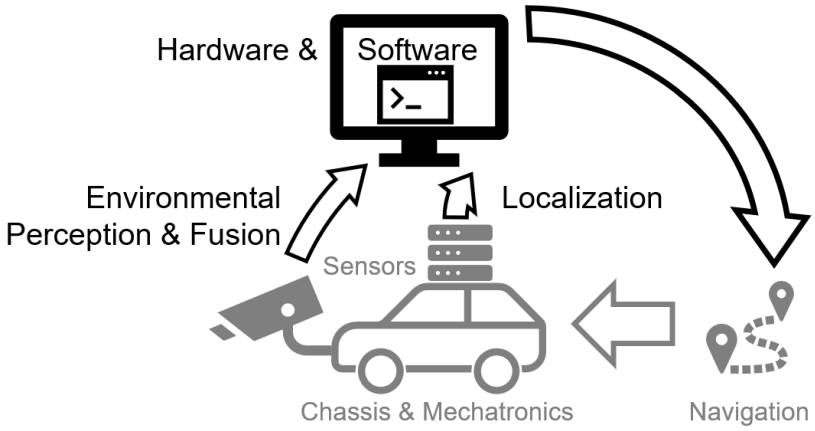


Figure 2.10: Symbolic representation of perception systems on AV, based on [18]

The following sections will present these areas of research along with examples and a particular focus on areas specifically relevant to offroad AV, as these face unique challenges compared to on-road solutions [10, 58]. Equipment and algorithms not only have to be more rugged and deal with adverse conditions, but also have to work with non-uniform geometries and non-rigid objects like trees and grass [5].

2.3.1 Environmental Perception

Firstly, relying on exteroceptive sensors [18], “Environmental perception refers to developing a contextual understanding of the environment, such as where obstacles are located, […] and categorizing data by their semantic meaning” [56, p. 3]. This is one of the biggest challenges especially for offroad AV owing to the non-uniform, complex nature of the environment, compounded with the fact that geometric information alone is insufficient for assessing the traversability of the terrain, for example with high grass [58, 59]. In offroad AVs, the main task of environmental perception is semantic classification of the ground surface – at the highest level into ‘drivable’ and ‘non-drivable’ – as well as the detection of obstacles [58, 60, 61].

To facilitate this, most AOV mainly use LiDAR, cameras, or a combination/fusion of these sensors for the task of environmental perception [56] (Section 2.2.3, Table 2.1). Cameras specifically require computationally expensive, dedicated algorithms to process their dense data stream [3], which the next section will present, and following that, this thesis will introduce recent fusion algorithms. Appendix D presents LiDAR perception strategies as well as older fusion approaches not directly relevant to this thesis.

2D Camera Perception

Cameras on road vehicles can be used to detect and road signs and traffic lights, as well as the borders of drivable or allowed road areas or obstacles [3, 4] (Section 2.2.1). Offroad vehicles

mainly use them for path or obstacle detection and classification [40, 43, 56, 58], which will be this section's focus.

Two separate ways to analyze camera data exist. Firstly, an algorithm can apply direct mathematical calculation (Appendix D). Secondly, Artificial Intelligence (AI) in the forms of neural networks and/or machine learning approaches can analyze camera feeds, which this section will present, while Appendix D presents the preliminary knowledge for this.

This section's first example of AI-based semantic segmentation in offroad AV comes from researchers at Carnegie Mellon University and Yamaha Motor Corporation USA in [62], who use custom-built Fully Convolutional Networks (FCN) to segment camera images. Both of their architectures are modifications of exiting network architectures, the first of which, '`cnns-fcn`' utilizes an input of 227×227 pixels and produces an output of 109×109 pixels, while the second, called '`dark-fcn`', bases itself on the Darknet architecture with an input and output of 300×300 pixels. The researchers report that their architectures yield results comparable to state-of-the-art networks, except for the detection of obstacles, where performance is significantly worse. In real world, closed loop tests, their vehicle successfully traversed various trails on which a previous AV had failed, with the camera perception system only struggling occasionally with roadside grass and on one occasion with the detection of a bush as an obstacle [62].

Elander's master's thesis at the Royal Institute of Technology in Stockholm [63] similarly uses FCN to facilitate the semantic segmentation of off-road scenery using a transfer learning approach. Here, a UNet architecture combines with a ResNet model as a backbone. Building on top of a FCN, the UNet architecture provides up- and down-sampling to capture context and precise location, combined with convolutions and skip layers to retain further detail, while the ResNet model introduces functionality to combat degrading performance with increasing number of layers. Starting with a model pretrained on Google's open image dataset, Elander then trains the models on a forestry dataset of the University of Freiburg. This provides comparable results to other work on the same training data; however, it also shows that semantic segmentation provides more accurate results with fewer classes. The work states that the usage of deeper networks provides a less-then-linear increase in performance when measured against compute time – a comparison between the different networks is shown in Figure 2.11 – as well as showing that reducing color resolution to 8-bit improved performance with a negligible effect on segmentation accuracy [63].

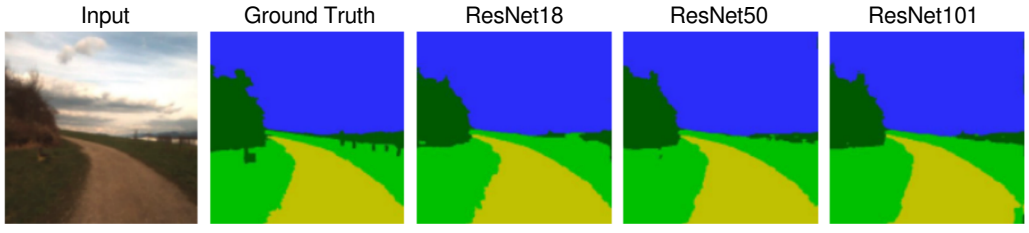


Figure 2.11: Comparison between results obtained in [63] from different FCN's, with differing amounts of layers. From left to right: Input image; provided annotation e.g. the desired output; the segmentation results of a network with 18, 50, and 101 layers [63].

Researchers from Xiamen University present another approach to semantic segmentation in conjunction with the University of Oxford in [59], developing a modification of the LedNet segmentation architecture called MiniLedNet. LedNet is an efficient solution for the task of semantic segmentation which builds on the ResNet model, and as such is FCN based. As this model lacks

performance for real-time applications, the researchers modified LedNet's encoder structure to reduce computation overhead and complexity, yielding a speed improvement of ca. 31% with only a maximum of about 3% loss on accuracy, which the researchers state has negligible effect on vehicle performance. Compared with the '`dark-fcn`' model from the previous paragraph – albeit with at a lower input and output resolution – MiniLedNet yields a ca. 30% increase in accuracy, although the work does not assess processing speed differences. While some segmentation errors occur in the researchers' real-word testing, their vehicle compensates for these inconsistencies by way of information obtained from a LiDAR [59].

Lastly, researchers of the Indian Institute of Science Education and Research Bhopal and the Texas A&M University propose a novel, two-stage approach in [61] called 'OFFSEG'. This uses a semantic segmentation transfer learning approach to extract a region of interest from the image, after which a color segmentation algorithm identifies subclasses in the region of interest which are then classified again; an example of this system can be seen in Figure 2.12. Firstly, the classes in the offroad vehicle datasets 'RUGD' and 'RELLIS-3D' pool into four superclasses: traversable, non-traversable, obstacle, and sky according to semantic contributions to the environment the researchers identified. They do this to eliminate issues with class-balance, which decrease model accuracy. They then perform training for two semantic segmentation architectures on these simplified datasets, namely BiSeNetV2 and HRNETV2+OCR, which both are FCN models capable of semantic segmentation [64, 65]. These architectures showed satisfactory performance, but since the researchers did not compare them to other network architectures, no relative statements are possible. Researchers however did state that BiSeNet-V2 is overall faster, while HRNetV2+OCR proved to be slightly more accurate on one tested dataset. Figure 2.12 shows a result of the segmentation process in in the second image from the right. The traversable class now contains several additional subclasses like dirt, mud, or gravel, which can still play a key role in pathfinding, and as such, the researchers decided to provide them. For this task, the traversable area found in the segmentation masks the input image. Then, on this subset of the original image only containing the traversable area, a k-means algorithm runs to find color clusters, which then run through a MobileNetV2 FCN to identify the specific subclass of terrain (Figure 2.12, rightmost image). The researchers obtained the MobileNetV2 model pretrained on an 'ImageNet' dataset, and subsequently modified it through transfer learning on the 'RUGD' and 'RELLIS-3D' offroad datasets, with results showing about 97% accuracy in the detection of subclasses [61].

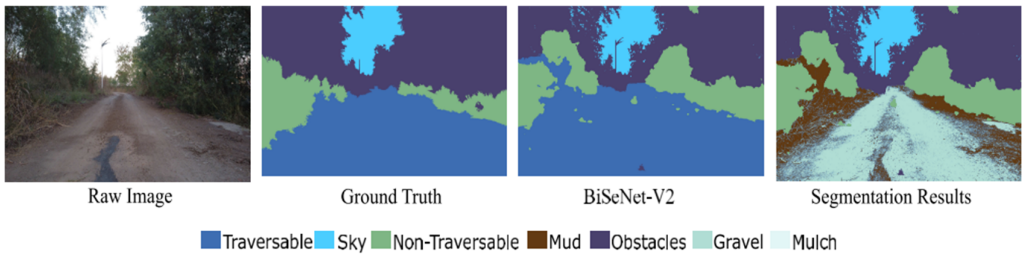


Figure 2.12: Example of the OFFSEG algorithm. From left to right: Input image, desired output from first segmentation, segmentation result, final result after clustering and classification.

Recent Exteroceptive Sensor Fusion

Sensor fusion refers to the combination of data received from multiple sensor sources such that the resulting information provides a more coherent image of the environment [2], with the goal

of creating more reliable, accurate and precise data. For Environmental perception, sensor fusion algorithms mainly use camera and LiDAR/Camera information, in order to augment RGB camera feeds with reliable 3D point cloud information [1, 2, 56].

Researchers of Xiamen University and the University of Oxford have developed a different approach of LiDAR-Camera fusion, called Real-time Semantic Perception and Motion Planning (RSPMP). It fuses the result of semantic segmentation of a camera feed with 3D point cloud information of a LiDAR, which it then translates into a 2D top-down map. Next, it associates the segmentation class data of each pixel with a 3D coordinate by transforming camera pixel coordinates into LiDAR coordinates, and then interpolating the LiDAR range data to calculate the 3D coordinate of the pixel in space. By doing this the algorithm generates a point cloud containing the semantic segmentation results as well as geometric LiDAR information, which it then translates into a voxel grid according to the current data, previous and predicted future occupancies in the grid. RSPMP then calculates the 2D representation by merging the voxel grid down along the z-axis, where it ignores voxels over the height of the vehicle, and otherwise the highest voxel determines the grid value at a specific position. Figure 2.13 shows an example of the complete pipeline. In practice, this system proved capable – however it was not compared to other solutions – where the vehicle managed to traverse terrain to reach a goal point, and additionally could compensate for segmentation errors with the geometric information [59].

Figure 2.13: RSPMP perception pipeline, left, example RGB camera input, middle, semantic segmentation results for that image, right, 2D-projected result map containing semantic segmentation results.

2.3.2 Localization

The second type of perception is localization [18], which relies “only on on-board vehicle sensors to find the global position of a vehicle in a specified coordinate system” [66, p. 832]. While mostly relying on proprioceptive sensors like GPS and IMU [18], localization may use exteroceptive sensors such as cameras or LiDAR to augment its algorithms [66]. Localization contains two subtasks: global localization for high level navigation tasks to reach a target point, and local localization, providing a position relative to obstacles and environmental limitations. Localization can work via prior knowledge of a map, where the system then compares map elements with perception data in order to locate itself, however, the required prior map knowledge [10, 18] complicates the localization system. As such this work will neither present nor use this technique. Instead, the following sections present, global localization tasks, while Appendix D explains local localization techniques.

Global localization and localization sensor fusion

GPS is the most commonly used localization system for all types of autonomous vehicles – not only in offroad applications – as it can offer an inexpensive and easily implementable source of

global positioning data [66]. Most of the full-size off-road AVs presented here utilize GPS systems [7, 10, 14, 15, 40]. However, because it suffers from poor reliability and accuracy, some researchers augment it with other systems to increase its viability [1, 66].

Although DGPS and other methods can improve the reliability of GPS readings – which its creators initially designed to only be accurate to 15 m – such systems usually require base stations that correct signals or provide an additional baseline, which may not always be feasible [1]. The most common solution to this is to fuse the GPS system with an IMU. This is prone to drift-errors, and as such is not suitable for global localization tasks, but in conjunction with an GPS can provide drastically improved levels of precision and accuracy than either system. The fusion of the information streams happens mostly over a Kalman filter, which itself is prone to assumptions and limitations, but researchers regard it as a solid solution to the task of global localization. More recently, researchers have shown that it is advantageous to use neural networks and machine learning approaches to fuse IMU and GPS data [1].

Early off-road AVs heavily rely on GPS to achieve accurate localization. However, especially for the purposes of SLAM and situations of GPS outage, vehicles require accurate pose information, which – in these early systems – IMU systems provide. Stanley uses an IMU system to both gain information about the vehicles orientation to generate accurate maps and facilitate environmental perception, but also uses it to gain reliable position information during GPS outages of up to 2 minutes. Experiments showed that the vehicle state estimation only veered 1.7m off the actual position after 1.3 km of driving [14]. Other full-size AOV examples such as MuCAR-3 [43] and RAVON [40] utilize similar systems. They often combine GPS-loss and subsequent IMU operation with a reduction in speed, in order to slow down the degradation of the position information as well as to give the system more time to react to changing conditions [14, 15].

Advancing from purely IMU/GPS coupling, studies have combined a GPS system with a LiDAR, camera, or radar sensor instead of an IMU to offset the low precision of the GPS. Such a system works in much the same way as local localization methods (Appendix D), however their usefulness is questionable when considering the prevalence of IMU systems [1, 66, 67].

2.3.3 Perception Hardware and Software Basis

Algorithm research and hardware development are crucial for the advancement of autonomous vehicle technology, because every kind of data processing needs a platform to run on [57]. To gain an overview of the components which enable the sensorics, mechatronics, perception and navigation, the following sections will discuss the hard- and software used in autonomous vehicles, with special focus on offroad applications.

Computation hardware and variants

“For autonomous driving, a powerful computing system is required to interpret a large amount of sensing data and perform complex perception functions in real time” [57, p. 151]. Many different approaches to this problem exist, encompassing multicore central processing unit (CPU) computers, graphics processing unit (GPU) equipped systems, and distributed solutions.

Early examples such as Stanley [14] primarily use consumer computer platforms, where the former combines six Pentium M personal computers (PCs) with a network switch. RASCAL [15], RAVON [40] and MuCAR-3 [68] utilize a similar ‘computer cluster’ approach with multiple PCs working in tandem. Researchers considered in RASCAL and implemented in MuCAR-3 what is

known as ‘Real-Time’ compute solutions, which run low-level controllers and access I/O in order to provide more consistent and reliable timing and control [15, 68]. The image in Figure 2.14 on the left shows an example of such a combined computer systems in AV.

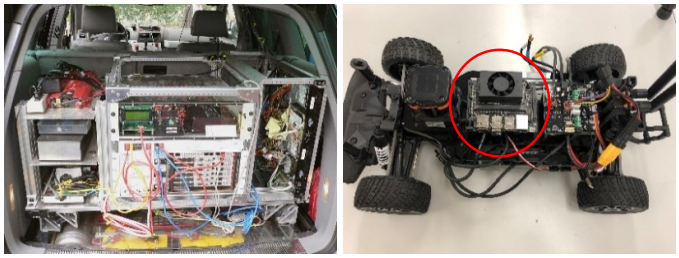


Figure 2.14: Different compute solutions in autonomous vehicles, left, Stanleys trunk-mounted compute network of multiple computers and batteries [69], Right, a partially completed F1TENTH vehicle with the NVIDIA Jetson SoC circled in red [34].

With the advancement of technology came the usage GPUs [57]. For their demonstration of their Semantic Mapping system, researchers in [62] utilized two commercial, off-the-shelf laptops, linked by network switches, which use laptop-grade GPU hardware in order to facilitate real-time capabilities with the computationally expensive neural network architecture. This is echoed in state-of-the-art systems such as EDGAR [54], which also provides GPU capabilities to the platform’s algorithms.

Further advancements make possible the utilization of CPU and GPU in [63], which uses an NVIDIA Jetson Xavier, a system-on-a-chip (SoC) providing specialized tensor cores, high speed memory for both GPU and CPU as well as hardware accelerated en-/decoding and deep learning [70]. These systems enable Neural Network, AI, and Machine Learning workloads in energy efficient ways on small-scale, autonomous systems. Common in many recent AV projects and testbeds, testing of semantic segmentation and mapping algorithms in [59] and [71] uses similar Jetson Units, and Jetson SoC are standard equipment on two of the F1TENTH platforms [13], (Figure 2.14 on the right). One of the F1TENTH platforms analyzed instead opts for a Raspberry Pi microcomputer [33], which enables even lower power consumption and cost. However [72] shows that it is largely incapable of running significant neural network-based analysis in a real time setting.

Lastly, work has shown viability of Field Programmable Gate Arrays (FPGA) and Application Specific Integrated Circuits (ASIC) to both increase performance and decrease power consumption for neural network computations [57]. While FPGA are reconfigurable, they are slower, less efficient, and larger compared to dedicated ASIC solutions, and they both suffer from exceedingly high implementation costs, especially for ASIC.

Software frameworks, platforms, and architectures

All the sensors, algorithms and hardware need a software fabric to create a complete vehicle. Researchers developed different solutions over time, and this section will present relevant examples. It is worth noting that many early prototypes do not explicitly mention specific software frameworks and thus this section cannot present their data in a beneficial way.

Stanford’s Stanley vehicle uses a Linux operating system as a base layer for its software, as it provides excellent networking and time sharing capabilities [14], which is an approach that RASCAL [15] and RAVON [40] also take. Additionally, RAVON uses the Modular Controller

Architecture (MCA) C++ robot control framework, a system of the University of Kaiserslautern researchers, who used it on other Robotic systems and research projects, and, as such, provides many user libraries and additional features [40]. MuCAR-3 utilizes a Debian Linux operating system modified with a special kernel for lower latency operations, and a KogniMobil real-time database accomplishes interconnections between software and data sharing [43, 68].

Advancing further, researchers in [45] implement the Robot Operation System. “The Robot Operating System (ROS) is a set of software libraries and tools that help […] build robot applications” [73]. Because it introduces infrastructure, algorithms and developer tools, many offroad AV implementations such as [3, 8, 11, 13, 26, 54, 59, 62, 74] use it to fully integrate sensors, perception, navigation and control into a functioning system [3, 73]. It also provides opportunities for developers to insert their own, custom software containers called packages, has message-based interconnects between individual software packages as well as logging and diagnostics, an online store of available software ready to download and embed into a solution, and implementations for many commercially available sensors.

Notably, many – presumably all – of the smaller F1TENTH applications utilize the ROS meta-operating system, such as [11, 13, 33]. Notable benefits of the system include easy interaction of the driving and perception application layer with the low-level sensor and actuation interfaces, as well as many reliable, pre-built packages with easy integration, adaption or change depending on individual project needs. ROS also minimizes development time loss due to coding of infrastructure and logistics and provides overall simplification of the software design process as well as assistance with wireless development.

2.4 Pathfinding and Navigation

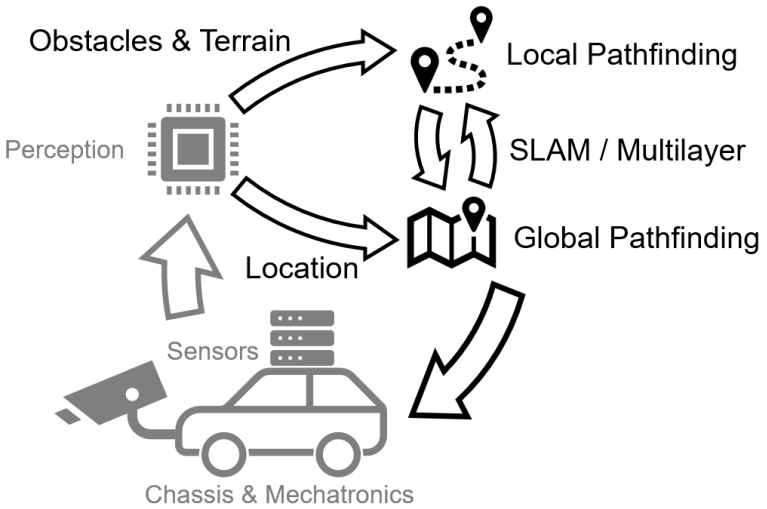


Figure 2.15: Schematic representation of AV navigation systems, based on [75]

“In order for a robot to navigate successfully, it must be capable of finding a path from its current position to its goal position” [76, p. 12]. To accomplish this, a robot most be able to avoid obstacles – both stationary or moving – and may need to optimize its path to reduce travel time and power usage, using a navigation system (Figure 2.15). Following this, mobile robot pathfinding splits into two subcategories, global (or offline) pathfinding and local (or online) pathfinding. In

the former the robot has complete prior knowledge about its environment including surface types, terrain shapes and obstacle locations, whereas in the latter the robot has to make decisions dynamically based on its perception of the environment [76], with Table 2.3 presenting a comparison. The following sections will present, local pathfinding – along with certain combined techniques – as it burdens a vehicle with fewer constraints of prior knowledge, training data and computational complexity. As such it is more suited to autonomous off-road navigation [71], while global pathfinding is explained in Appendix E.

Table 2.3: Comparison of Global and Local Pathfinding [77]

Global Pathfinding	Local Pathfinding
Offline	Online/Real-time
Often needs a pre-processing stage	Does not require any pre-processing
Relying on maps as input	Based on sensors such as LiDAR

2.4.1 Local Pathfinding

Local path planning assumes that the robot has little or no data about its current environment and situation, which stands in contrast to global pathfinding (Appendix E). Instead, the vehicle has to adapt to the current situation to plan its path based on its perception of the environment [76]. In comparison to global pathfinding – see Table 2.3 – it happens in real time, does not require pre-processing and is based on exteroceptive sensors [77]. Local pathfinding can work via either trajectory-based methods [5] which the following section will present, or via genetic algorithms and artificial intelligence [76] (Appendix E).

Trajectory-based navigation systems

“Trajectory based navigation systems calculate a set of potential robot trajectories and then select the most suitable one for execution” [5, p. 111]. The three algorithms laying the groundwork for trajectory-based methods are the Bug 1, Bug 2, and Vector Field Histogram algorithms [76], and this thesis will quickly present them in the following.

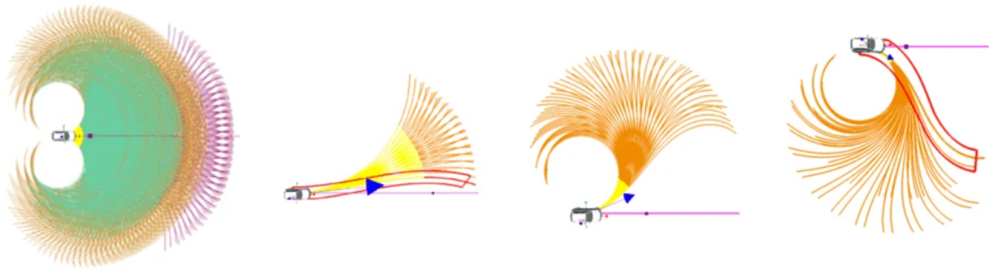
The Bug 1 algorithm, upon detection of an obstacle on the path to the global goal will follow the contour of said obstacle while constantly computing the further path to the goal point. Once it finds the point on the contour with the shortest distance to the goal point, the robot again proceeds on the contour until it reaches that point, and then continues onward to the goal point from there. As a modification of Bug 1, the Bug 2 algorithm aims to improve efficiency by constantly monitoring the heading angle of the vehicle and comparing it to the initial heading for the unobstructed path to the goal. Once – after starting the obstacle avoidance mode – the current heading of the vehicle equals the precomputed one it will then switch back to normal path-to-goal driving, having circumvented the obstacle enough to provide a clear path.

Lastly, the Vector Field Histogram method implements a 2D histogram grid, which accumulates the obstacle data obtained by the onboard sensors. Then, the algorithm takes a fixed size and distance subset of the 2D grid around the robot and filters it into a 1D polar histogram, which represents obstacle density in a certain heading direction around the robot. Next, the algorithm uses this data to select the sector containing the lowest concentration of obstacles, and the robot then subsequently heads into that direction. Many other algorithms use these or similar concepts

as a basis to facilitate obstacle avoidance in the field [76]. The following paragraphs will present such derivative and contrasting algorithms.

First, as an early example, Stanley [14], the University of Stanford's entry into the DGC, uses an entirely separate algorithm. Here, the navigational algorithm maps obstacles using their distance to the proposed path of the vehicle – which it precomputes using the checkpoints given by the challenge – enabling the vehicle to execute obstacle avoidance by changing its lateral offset to said precomputed trajectory in the event of obstacle detection [15].

Researchers present another approach to local pathfinding in [78], which is based on a map containing obstacle boundaries and the boundaries of slopes too steep for the vehicle. First, if a straight path from the current position to the goal position is drivable, the algorithm chooses that path, otherwise, it computes a set of straight line fixed-length trajectories starting at the vehicles position on the map. If such a line hits no obstacle, the algorithm considers it viable, and from all viable paths, it chooses the one which ends closest to the goal. This provides a viable solution for local navigation, however, if a global planner or generated waypoints include dead ends farther away than the planner's horizon, the vehicle might get stuck [78].



(a) (b) (c) (d)

Figure 2.16: Tentacle configurations of MuCAR-3. (a) shows the total set of tentacles, colored according to the current vehicle speed from green to orange to magenta. For the other example subsets of tentacles (b) to (d), orange tentacles are feasible, yellow indicates predicted stopping distance on a tentacle, a blue arrow denotes the current steering angle, where purple shows an arbitrary target trajectory, and the selected path has a red border in (b) and (d).

Expanding on a similar methodology, work done on the MuCAR-3 vehicle [43, 68, 79] also implements the filtering of multiple possible trajectories. Called 'tentacles', the approach mimics the usage of antennae in nature, which represent driving options of the vehicle, e.g. speed and steering inputs, which the algorithm then assigns a value to for the drivability analysis of the individual tentacles, based upon which it selects one trajectory for the vehicle to drive. The algorithm constructs tentacles with respect to a maximum steering angle considering maximum lateral acceleration and physical steering limitations. Using this maximum, the system constructs a set of discretized tenacles between the maxima in increments of $0{,}001\ ^{rad}/_{s}$ of change in the steering angle. This generates tentacles which start in all directions between the maximum steering angles and then execute a specific amount of steering change over time, resulting in the total space of possible tentacles shown in Figure 2.16 (a). For any given instant, the navigation algorithm then selects a subset of feasible tentacles based on the current vehicle speed, steering angle; drivability, clearness, flatness, distance and heading to target, and an additional visual cue (Appendix D) of the tentacles (Figure 2.16 (b) to (d)). If no tentacle is feasible, the vehicle will either drive backwards (where the algorithm implements a similar tentacle approach for backwards driving) or stop and expand its search to include other starting steering angles which the

vehicle can then adopt before continuing driving operations. This approach proved viable, as it allowed the vehicle to complete the 2009 European Land-Robot Trial (ELROB) in Finland as the only competitor driving the entire distance.

Many offroad AV implement similar approaches. In [75], 625 possible trajectories are evaluated for a potential driving time of 2 seconds, evaluating robot performance via a simulation, which integrate the robot acceleration. The algorithm also evaluates tentacles using a cost function that includes distance to obstacles, distance to the goal and to the optimal path as well as remaining velocity after a maneuver to increase overall speed.

Researchers present a GPS-independent tentacle navigation method in [80]. Here, the navigation algorithm evaluates tentacles with respect to obstacles blocking the path, obstacles near the path and obstacles near the path which have another obstacle opposite them on the same path. In tests comparing their approach with a variant of the potential field method, the researchers show that tentacles provide a useful look-ahead functionality combined with more stability enabling higher overall speeds, as well as the tentacles based-navigation reaching the goal in more scenarios than the potential field method.

In [62], researchers provide a navigation approach which samples 30 trajectories from -15°/s to +15°/s at intervals of 1°/s and a constant velocity of 9 km/h . To evaluate tentacles, the navigational algorithm employs a reward function, which summarizes the movement cost based on semantic segmentation perception (see chapter 2.1.3) over 20 m. To account for vehicle width, the algorithm additionally evaluates multiple paths with an offset to the original to cover the vehicle footprint. Researchers observed that this algorithm worked, however it veered from side to side on larger trails on wider trails.

Most recently, researchers in [59] also implement a tentacle-based approach, where their algorithm samples a given set of trajectories with different characteristics according to the current speed of the vehicle, and in an additional step optimizes them according to the distance to the goal and the estimated movement cost of the traversed terrain. The researchers proved that this approach is viable in tests, where their system allowed the AOV to reach its goal consistently.

2.4.2 Distributed, Multi-Layer and SLAM Navigation Approaches

This section presents solutions for offroad AV which implement multi-layer planners for navigation purposes [40, 81] or employ a SLAM or other mapping approach to then use higher level, global pathfinding in conjunction with local pathfinding solutions [9, 18].

Researchers in [81] present the first example for a robot utilizing both global and local planning. Here, they employ a two-stage system with a local and a global planner. The former uses a local map, which remains fixed relative to the vehicle position, delivering environment data relevant to navigation to the local navigation algorithm. This then determines candidate waypoints for the robot to follow to avoid obstacles and high movement costs to move toward the global goal. The global planner utilizes a global map, into which it transfers data from the local map after each processing step (see SLAM in Appendix D) to keep knowledge about already traversed areas in the robot's memory. It then extends the local pathfinding module with an A* algorithm to proceed towards the goal [81].

The RAVON vehicle of the University of Kaiserslautern employs another example of a multi-layer navigation system [40]. First, the short-range navigation provides local pathfinding, avoiding collisions and reaching a goal point up to 3m away, which the mid-range navigation provides. This

attempts to detect ‘passages’ through obstacles, which allow the system to avoid dead-ends and reduces the amount of backtracking and slow-speed driving needed with just the low range system. Lastly, long range or global navigation starts at 10m ahead, using a D* algorithm and SLAM type map knowledge generated from sensor data [40].

Lastly, the thesis in [9] uses aerial unmanned vehicles to first create a detailed, pre-classified map of the terrain to be traversed, which it then uses to facilitate global navigation with the high-quality mapping made available before.

3 Method

The previous sections of this thesis present the existing technology in the diverse and dynamic field of autonomous navigation with a special focus on offroad technology. They show that a variety of hardware, software and algorithms exist and that researchers with many diverse backgrounds and subsequent focal points in their works tested them for the purpose of autonomous offroad navigation. With Section 1.1 identifying a niche in current AOV development, which – in short – applies to small scale offroad AV, this chapter will now focus on the development and implementation of said vehicle.

It achieves this by first analyzing the state of the art (Chapter 2) to gain relevant knowledge. Said knowledge will provide the foundation for the path this project takes toward the goal of this thesis, a necessary inclusion before then presenting the step-by-step process developing a functional miniature offroad AV. Figure 3.1 shows a schematic overview of this process.

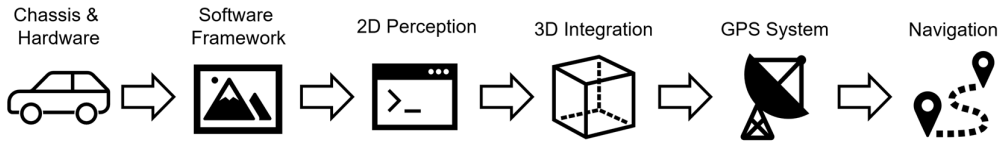


Figure 3.1: Overview for the miniature AV development process in this thesis.

3.1 Initial Analysis of existing Solutions

This section performs detailed analysis on the work previously presented, identifying key elements and concepts used for the vehicle that this thesis aims to develop. Its contents include the selection of a chassis, sensor suite and other hardware, as well as selecting specific concepts for software as well perception and navigation algorithms, presented in the following.

3.1.1 Decisions for the Chassis and Platform

All the early, fully functional, and capable offroad AVs that competed in DARPA and ELROB are based on full-size chassis, which the researchers have modified to suit their needs (Section 2.1.1). While the setups of these challenges suggest an approach such as this, and the resulting vehicles are capable, development of a vehicle on such a scale is fully outside of the scope of a bachelor's thesis. Reasoning for this is analogous to the drawbacks already presented in Section 1.1, but to summarize, such vehicles are expensive, dangerous, and complex. So, while this thesis will not move forward in the direction of full-size vehicles, it is nevertheless important and useful to analyze the findings of researchers that worked in this area alongside smaller scale solutions.

Stanford University's Stanley vehicle as well as the MuCAR-3 team, who competed in the DGC and ELROB respectively, both utilize a VW Touareg as the basis for their vehicles. The researchers state that among the reasons for this choice are 4WD, variable height air suspension and locking differentials. That two independent, well performing teams choose this vehicle leads to the assumption that these elements are vital for off-road performance of an AV. Furthermore, the AVIDOOR-2004 vehicle by SciAutonics notably does not possess 4WD, and got stuck in offroad environment, further cementing the relevance of 4WD. Both SciAutonics Teams utilize preexisting ATV with locking differentials, which are thus important.

Alternatively, some researchers developing smaller scale vehicles chose to utilize non-wheeled systems. This thesis' examples for such systems either use tracks, or in one case compliant, rotating arm-like structures for their locomotion (Section 2.1.2 and 2.1.3). While this provides advantages over wheeled vehicles in offroad terrain, these vehicles present clear drawbacks. Researchers note that tracked vehicles possess low quality IMU readings, which make autonomous navigation tasks difficult (Section 2.3.2). The presented platforms come as experimental systems at high price points, further making their use in this project undesirable.

The observation that wheeled approaches are viable in offroad scenarios compounds this and the University of Sydney's swagbot (Section 2.1.2) as well as the Mars rovers and the 'Ackermann' vehicle (Section 2.1.3) prove this. However, both swagbot and the Mars rovers utilize complicated suspension and wheel setups to further improve offroad capabilities. While swagbot's omnidirectionality may be useful in agriculture and the Mars rovers 6-wheel setup performs exceptionally well in the sands of Mars and at slow speeds, both capabilities are less relevant for the task of this thesis, and as such, a setup like the 'Ackerman' vehicle is most practical. As it is based on a Traxxas scale model RC-Truck, it sports the following features noticed in full-scale systems: 4WD and differentials which limit slip in the event of traction loss, large offroad tires and a high suspension. Additionally, it has a notably low center of gravity, making it difficult to roll on steep terrain.

This translates to the F1TENTH platform (Section 2.1.4), which is based on a Traxxas RC-truck. While smaller than the 'Ackermann' vehicle, it is a proven system for AV operations that includes all the advantages and requirements from the full-scale platforms because its creators based it on an offroad-ready RC-car. Systems and methods to control the mechatronics for AV purposes have proven themselves and this thesis directly reuses them for this project. With existing knowledge and infrastructure already present in universities around the world, it further fits perfectly for the intended contributions of this thesis (Section 1.1). As such, it is the ideal choice as a basis for this thesis, combining offroad capabilities with a proven AV-platform that is already at home at many research institutions.

3.1.2 Sensor Selection

This thesis discusses sensor systems on AOV in detail (Section 2.2) and specifically recaps them with summarizing tables (Section 2.2.3). For exteroceptive Sensors, 80% of full-size platforms as well as 66% of the F1tenth platforms use 2D LiDAR, with only minor usage of 3D LiDAR. Considering that 3D systems introduce more cost and increase computational load, and with many older full-size AOV showing that autonomous offroad navigation is possible with the use of 2D LiDAR only, a 2D sensor is sufficient for the purpose of this work. The overall necessity for a LiDAR sensor stems from its lower computational load as well as the proven methods for obstacle and surface detection (Section 2.3.1 and Appendix D). 2D LiDAR are also easy to integrate and use, which is why this project may benefit from them especially.

Analyzing Camera usage reveals that sensor's frequent usage in AV, with 60% and 66% of examples respectively using stereo cameras on full-size and F1TENTH platforms, where EDGAR utilizes multiple camera systems (Appendix C). While EDGAR and other full-size examples utilize monocular cameras, they are overall less prevalent. Section 2.3.1 explains in detail what information algorithms gain from a camera image using modern technology, making it a valuable addition to an offroad AV despite the additional computational load. This is especially true in the absence of 3D LiDAR capabilities for the purposes of perception as Appendix D shows. Additionally, even though 2D LiDAR are more prevalent as discussed in the previous paragraph, many full-size AOV use multiple sets of them to provide a 3D point-cloud. This is impractical on a small RC car and as it would introduce computational hurdles and calibration issues. Thus, a point-cloud generated from a stereo camera seems a valuable substitute, although it likely lacks enough precision for pure point cloud perception (Appendix D and Section 2.2.1). The point-cloud may additionally be useful for sensor fusion (Section 2.3.1), introducing additional capabilities. With all of that in mind, this project will use a stereo camera.

For additional exteroceptive sensors, some full-size examples use Radar and Ultrasonic sensors. However, such sensors and systems provide contingency or emergency use (Appendix B) which are difficult to integrate on a small RC car, leading to the decision to not integrate them on this thesis' vehicle.

Lastly, for proprioceptive sensors, the chosen F1TENTH platform (Section 3.1.1) already includes a motor controller, the VESC (Section 2.2.2), which provides encoder support and an IMU, which are present on all platforms (Section 2.2.3). This is appreciated, since these are capable sensors for the task of localization and can augment other sensing systems (Sections 2.2.2 and 2.3.2), which is vitally important for navigation (Section 2.4). When it comes to other proprioceptive sensors, a large split emerges between full-size AOV and small-scale AV. Here, we see all large AV utilize GPS for their offroad navigation requirements, with 80% of them additionally employing either a magnetometer or heading information, while none of the F1TENTH based vehicles include such capabilities. As global localization is critical for offroad navigation, this project will include a GPS sensor on the vehicle, but without a GPS heading system as those are more complicated, and thus expensive (Section 2.2.2), with this thesis' vehicle instead opting for a magnetometer.

3.1.3 Goals for Perception and Navigation

All of the prior considerations lead to the purpose of the vehicle, which is offroad navigation and its perceptional requirements. Section 2.3.1 shows extensively that AI-based semantic segmentation is highly capable and of significant use for the task of autonomous offroad navigation. While Appendix E lists relevant alternative techniques, they rely on sensor technology which is unavailable for this project, while the presented perception techniques simultaneously supply a greater amount of information. As such, this project chooses the two-stage OFFSEG system as the primary perception method, as it provides a large amount of offroad-relevant ground classes as well as pretrained models ready to use. It is also open source, and as such poses less hurdles for adaption in this project while still showing adequate results. Additionally, navigation requires a 2D top-down map for the techniques of Section 2.4, and so the developed system will run a 2D-image and point cloud fusion algorithm (Section 2.3.1).

Next, for navigational purposes, while many advanced localization techniques and corresponding hardware exist (Sections 2.2.2 and 2.3.2), the combination of IMU, encoder odometry and GPS has proven itself as sufficient when combined with Section 2.3.2's fusion techniques, as

many full-size examples such as Stanley demonstrate. All advanced systems are too difficult to calibrate, too large to set up or too expensive, making their use impractical.

For navigation, this project will focus on local pathfinding, as Section 2.4 describes the drawbacks of utilizing global navigation without a prior, established map. For this task, the vehicle will utilize a trajectory-based tentacle navigation system (Section 2.4.1), since it is a proven, effective, and simple solution to short range navigation. This is especially true considering the limitations of our sensing suite (Section 3.1.2), which would make mid-range navigation difficult without more accurate point-cloud data supplied by unavailable 3D LiDAR sensors.

As to methods such as SLAM and subsequent multi-layer navigation (Section 2.4.2): SLAM is too complicated and difficult to implement with the aforementioned limited sensors and without a more accurate positioning system (Section 2.3.2). As such a system is outside of the scope of this work, and without longer range sensing or local mapping, this project's navigational system has no use for longer range planners. This eliminates the need for complex multilevel navigation, which this project will thus not use.

3.1.4 Considerations for Compute Hardware and Software

This thesis further decides on a compute hardware platform that is a proven, widely used system. While early systems employ complex computer systems, the current state of the art are NVIDIA Jetson SoC (Section 2.3.3). They are capable systems, used in both semantic segmentation development and in actual, running AV. Researchers use them on two of the F1TENTH platform examples shown before, further validating their capabilities on small-scale platforms. While one of the F1TENTH platforms uses a Raspberry Pi, it has less compute power. With the computationally taxing AI-based semantic segmentation system needed for perception (Section 2.3.1), this thesis' vehicle will use the higher performance hardware proven in this field.

Lastly, deciding on a software framework is similarly straightforward. Section 2.3.3 states that, while early examples use custom solutions or early real-time systems, almost all later implementations of AOV utilize the Robot Operating System or ROS combined with a Linux operating system. As a comprehensive and capable suite of infrastructure, algorithms, and software systems for autonomous driving, the F1TENTH platform uses ROS by default on all presented examples, further proving its capability and applicability.

3.1.5 Summary of initial Analysis

To finish this section, Table 3.1 displays a combined list of the hard- and software picks made for the vehicle developed in this thesis, based on the state of the art in offroad driving.

Table 3.1: Intended Setup of this thesis' platform based on prior work and analysis thereof.

Chassis/Electronics	Sensorics	Perception/Software	Navigation
Traxxas Slash 4x4 Short course desert truck	2D LiDAR	AI-based semantic segmentation	Local Pathfinding
NVIDIA Jetson	Stereo Camera	2D-image/point-cloud fusion	Short range 2D top-down map
	GPS receiver	Linux operating system	Trajectory based tenacle methods
	VESC	ROS	
	Magnetometer		

3.2 Setup and first System Tests

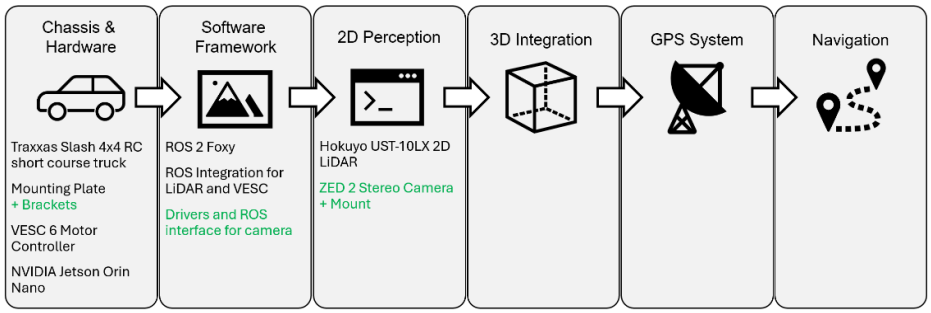


Figure 3.2: State of the thesis at the start of the physical project, grouped by subsystems in order of their development in this thesis. Black text signifies components already present, green denotes elements developed in this section.

After the previous section's analysis, this section will now begin documenting the construction of the robot, with Figure 3.2 showing changes currently planned for this initial step. The Technical University of Munich's Autonomous Vehicle Systems (AVS) department, which oversees this thesis, supplied a partly preconfigured F1TENTH platform for this project (Figure 3.3).

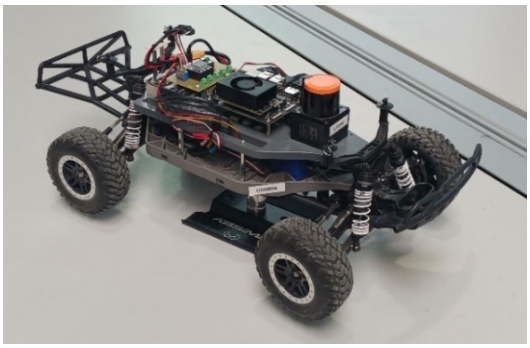


Figure 3.3: Initial configuration of the F1TENTH platform supplied by TUM's AVS department.

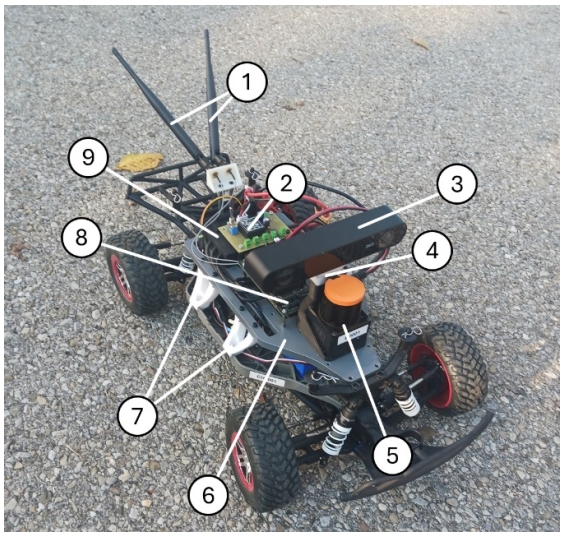
This system builds on a Traxxas Slash 4x4 short course desert RC truck, similar to other F1TENTH platforms (Section 2.1.4), which Section 3.1.1 declares a suitable basis. Like these platforms, the vehicle receives a mounting plate, which contains an NVIDIA Jetson Nano, a Hokuyo UST-10LX 2D LiDAR, the VESC 6 motor controller, dual WLAN antennas and a 12 V power supply board to regulate battery voltage for the Jetson and the LiDAR. The vehicle powers off of a Traxxas 5000 mAh 11.1 V 3-cell Lithium Polymer Battery. The Jetson has the Ubuntu 20.04 LTS distribution of Linux preinstalled, together with ROS 2 Foxy, which already integrates the VESC and LiDAR via the installed `f1tenth_stack` ROS package [82]. This package also sets up robot control via a Logitech F710 gamepad and facilitates LiDAR data access via ROS. This setup is similar to other F1TENTH platforms in sensors (Section 2.2.3) and overall software setup as documented on the official F1TENTH website [34].

3.2.1 Chassis modifications and Sensors

After receiving the initial F1TENTH depicted in Figure 3.3, it does not contain provisions for mounting the ZED 2 stereo camera. This project utilizes this camera due to its availability at the

TUM's AVS department, and with 110 horizontal and 70° vertical FOV stereo vision, 3D point cloud data via an integrated neural depth engine, plus IMU and magnetometer data, it more than satisfies this project's requirements (Section 3.1.2). A 3D-printed mounting stalk utilizes the existing LiDAR mounting provisions present on the mounting plate to attach the camera. This stalk allows both the camera and LiDAR to function relatively unobstructed and without interference from each other or other components, analyzed in detail in Sections 4.2.1 and 4.2.2.

A closer investigation of the provided F1TENTH platform shows that the mounting plate attaches to the chassis only with hot-glued standoffs. This raises safety concerns for the hardware when subjected to offroad driving, so specially designed, additional 3D-printed mounting brackets increase structural integrity. These interface with the handle mounting holes on the chassis as well as the mounting holes on the mounting plate used by the hot-glued standoffs. To allow access to the battery compartment, the vehicle uses three brackets instead of four to hold the mounting plate. A lateral offset between the mounting holes in the plate and the chassis necessitates the integration of this offset in the brackets, thus the 3D printer prints the model of the third bracket mirrored. Figure 3.4 shows these additional elements mounted on the vehicle.



1. Dual Wi-Fi antennas
2. 12 Power distribution board
3. ZED 2 stereo camera
4. Camera stalk
5. Hokuyo LiDAR
6. Mounting plate
7. Mounting plate brackets
8. Jetson Nano (behind ZED 2)
9. VESC 6

Figure 3.4: Initial modifications to the provided F1TENTH platform, with annotations. The NVIDIA Jetson Nano is easier to see in Figure 3.3, where it mounts at the same place.

3.2.2 Sensor Capabilities and Software Architecture

As Section 3.1.4 previously explains, ROS 2 forms the basis for the Software Architecture, with Section 2.3.3 expanding on its capabilities. The initial platform includes a preconfigured version of ROS 2 Foxy – which this project adopts – providing working integrations for the LiDAR and VESC via the F1TENTH software package [82]. The Hokuyo UST-10LX LiDAR has a 270° horizontal field of view and provides one scan line with an angular resolution of 0.25°, providing 1081 total scan points at 40Hz refresh rate. The VESC 6 controls both the brushless main motor driving all four wheels as well as the steering servo. In addition, it features an additional IMU providing odometry at 50 Hz.

However, the ZED 2 Camera still needs software integration, provided by the `zed-ros-wrapper` package available on GitHub [83]. This package makes the full capability of the camera available directly to ROS, including camera feeds, depth information, magnetometer data and camera

odometry readings. While the camera can provide its video streams at up to Full High Definition (HD) which is 1920 by 1080 pixels while running at 30 Frames per Second (FPS), this project and its hardware cannot handle such data density. Thus, the setup initially runs at HD720 (1280 by 720 pixels) and 15 FPS with 'neural' depth mode, while running the IMU at its max of 400 Hz. Depth data ranges out to 20m, with resolution and refresh rate matching the main camera feed.

3.2.3 Lessons from first Drives

Initial test results are promising; however, several problems manifest. Firstly, data capture utilizes the ROS bagging feature, which enables data capture and replay by interfacing with the message-based data infrastructure to save time-stamped messages to a file. Due to the high bandwidth of both the video and depth data (quantified in Section 4.2.2), the slow transfer speeds of the internal Jetson micro-SD card compromise data bagging. To combat this, for future tests a Samsung 970 EVO 1 TB internal NVMe solid state drive provides the necessary disk speeds for high bandwidth data gathering on the Jetson SoC.

Next, the system responds slowly with occasional, up to second-long lags, which result in loss of control for that timeframe. With data bagging still impacting the system, a further reduction of camera capture framerate and resolution solves the lag issue. The resolution is now set at the Video Graphics Array or 'VGA'-option (672 by 376 pixels) with a 10 FPS refresh rate. However, even after this further reduction, a test via the internal ROS message infrastructure shows that the camera package only provides around 7 FPS continuously. Section 4.2.2 extensively studies this effect, with quantified results for multiple configurations.

Modifications to the original Traxxas plastic outer shell for the vehicle such as holes for the Wireless Fidelity (Wi-Fi) antenna, camera, and LiDAR allow its continued use on the vehicle, combating ingress of dirt, mud, or water into the sensible and unprotected electronics on the mounting plate (Figure 3.5). With these modifications, the vehicle is ready for a first data gathering run in an offroad environment, with Figure 3.5 showing a picture of said run. The system records data for 30 minutes while driving without issues, bagging 150 GB of data.

Lastly, after witnessing the vehicle struggle in offroad environments due to its low ride height caused by suspension sag under the high weight of the additional sensor and compute hardware, the vehicle needs a suspension system upgrade. This however only arrives towards the end of the project due to shipping issues and thus Section 3.6.1 will present it at the end of the chapter.

Figure 3.5: Modifications of the provided F1TENTH platform vehicle. Left, modifications with plastic outer shell. Right, data gathering run in off-road environment.

3.3 Perception Software Development

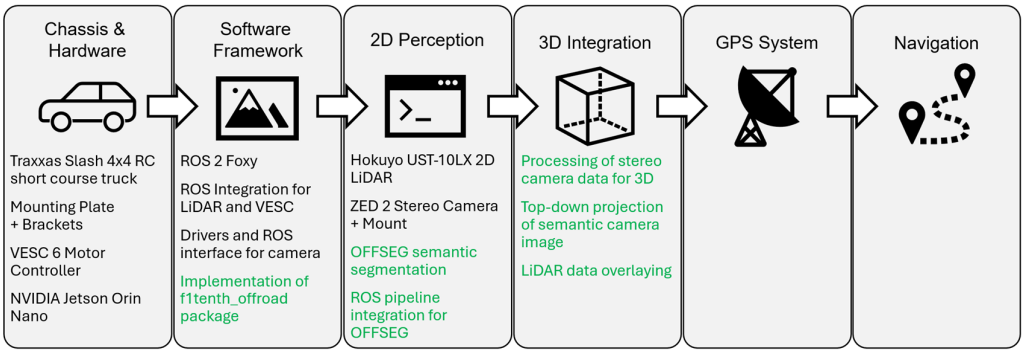


Figure 3.6: Progress of the project at the start of software development, black items signify current implementations, green items are this section's additions.

This chapter covers the perception elements of this thesis' software development. The ROS package into which all of the software elements – including navigation, presented in 3.5 – will be integrated is called `f1tenth_offroad`, available on GitHub [84]. The perception system splits into two different subsystems – shown as the middle two steps in Figure 3.6 – which are the 2D perception and the 3D Integration, and this chapter splits into the same compartments. More specifically, the 2D perception comprises the semantic segmentation algorithm and its integration into the ROS pipeline, while the 3D Integration handles depth data processing, top-down projection, and LiDAR fusion. Figure 3.7 shows an overview of the perception system, with external input, data transfers and internal processing steps.

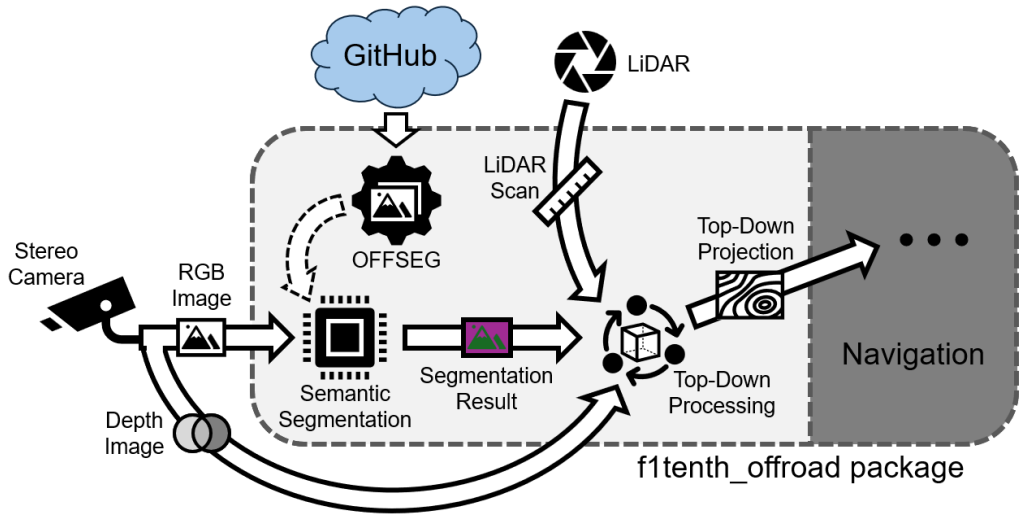


Figure 3.7: Schematic Overview of the `f1tenth_offroad` package's perception system also containing a placeholder for the navigational components.

The software development is largely conducted on a Windows platform, utilizing an NVIDIA GTX 2060 and the Windows Subsystem for Linux (WSL) with a Linux distribution corresponding to the one used on the vehicle (Section 2.3). The reason for this is the slow software development on the Jetson, where compilation and running code takes considerable amounts of time. It is complicated to use Visual Studio Code to remotely connect over `ssh`, a process for which a Wi-

Fi connection, hotspot and internet adapter bridging are necessary. Additionally, the Jetson crashes without warning when the battery charge gets too low, a phenomenon which will later cause problems in field tests (Section 3.6.2).

Software development with WSL also is not without problems. While it is much faster and easier to develop software on a desktop computer – which allows the software development without the presence of the physical vehicle – several problems exist with GPU integration. Since the software stack requires the presence of an NVIDIA GPU, a custom kernel and integration for WSL are needed [85]. WSL Software development requires installation of the ROS system in precisely the same manner as on the physical platform, and matching the packages and infrastructure during development poses a challenge. Once the setup is complete, most software development happens on the desktop platform, while using the GitHub repository to share code between the systems, which is highly effective.

3.3.1 2D Perception: OFFSEG Semantic Segmentation Code and ROS Integration

This project uses the OFFSEG Semantic Segmentation project and neural network model (Section 3.1.3). The official implementation, available on GitHub [86], provides said model, pre-trained neural network weights, as well as utilities and code for performing the segmentation and facilitating training of the models. OFFSEG is a multistage system, which utilizes two different models for coarse and fine segmentation (Section 2.3.1). Additionally, OFFSEG provides two different options for the coarse segmentation step, of which this project uses the faster BiSeNetV2 architecture.

To integrate this system into the perception pipeline testing ensues. Initial trials prove troublesome, as the provided implementation does not function right out of the box, with missing initial requirements for the code to execute fully. After adapting the `requirements.txt` file to use newer or different packages available for the selected Linux distribution and hardware (Section 3.1.4), the next problem is the second stage segmentation. This requires a pre-step of K-means clustering, which is non-functional in the original GitHub code. Downloading the source code for a GPU-based K-means algorithm from GitHub [87] proves effective, although modifications were necessary to alleviate bugs and make the code run on the vehicle platform. After the implementation of the custom K-means package, further work fixes other, smaller bugs present in the provided OFFSEG pipeline, enabling the OFFSEG algorithm to successfully work as demonstrated in its paper.

Next, the Algorithm requires integration into the ROS pipeline, where it must operate on the message-based image feed from the camera driver and push finished, segmented images back into the message infrastructure. This is simple on a conceptual level, since the provided OFFSEG code iterates through a given folder of images, successively running the algorithm on each found file. If this ‘inner’ algorithm cloud instead operates on an image feed from a camera – which also provides subsequent images – integration would be effectively complete. The code achieves this modification by setting up the ROS package called ‘`f1tenth_offroad`’, which provides a ‘node’ – a piece of code running within and interacting with the framework – facilitating segmentation. The specific node running this OFFSEG segmentation is ‘`perception.py`’ and consists of the initial OFFSEG pipeline code from the `pipeline.py` file and the modifications together with required libraries from the ‘libs’ subdirectory. By subscribing to the camera feed via the provided ROS node infrastructure, the algorithm receives images, extracts them, runs them

through OFFSEG without any changes to the inner algorithms, and publishes received results – instead of saving them to a file – back into a ROS message available to the system. This requires modifying the package configuration to include the required libraries and configuration files in the package compilation, after which the algorithm achieves a complete integration of the OFFSEG pipeline into the ROS infrastructure.

This structure allows OFFSEG code to run on recorded data from the actual vehicle. In addition to the segmented image, an 'overlayed' image, showing the segmentation results overlayed over the camera input image is available. Figure 3.8 shows a side-by-side display of a camera image and its first- and second-stage segmentations by OFFSEG. Note that all camera pictures – RGB or depth – shown in this paper stem from the left camera of the stereo setup.

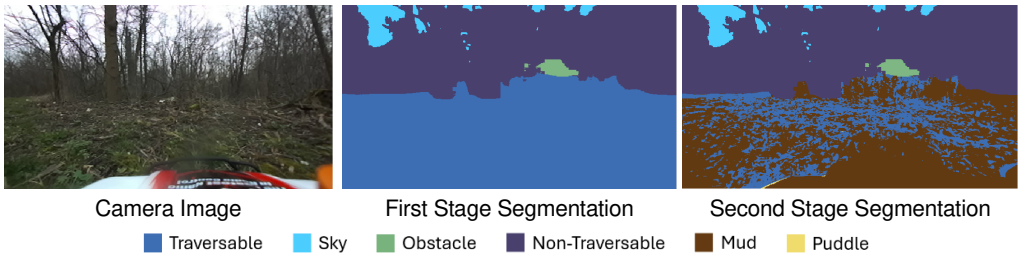


Figure 3.8: Results of the OFFSEG semantic segmentation pipeline.

However, the following issue appears during the initial testing of the 2D Perception system: a large part of the computational cost lies in second stage algorithm. Section 4.3.1 further discusses the specifics and ramifications of this, but the implementation of the second stage is both computationally unfeasible as well as unnecessary. Thus, the final integration of the OFFSEG algorithm for this project only performs the four-class segmentation into the categories sky, traversable, non-traversable and obstacle as shown in Figure 3.8 in the middle. Additionally, the system uses a mask to block out occluded parts of the camera image (Section 4.2.2).

3.3.2 3D Integration: Depth Data Processing and Top-Down Image Generation

Next, another node called `transform.py` subscribes to the ROS message stream containing the segmentation results. This node contains the sensor fusion code, which merges the segmentation results, the LiDAR scan, and the 3D stereo camera data into a 2D top-down map of the terrain in front of the vehicle. This task necessitates the following steps:

1. Creating a point cloud from the stereo camera depth data.
2. Merging the point cloud with the segmentation results, creating a 3D segmentation.
3. Performing a birds-eye-view/top-down projection of the segmentation point cloud.
4. Overlay the laser scan results over the 2D top-down segmentation image.

Depth Data Processing and RGB Data Merging

The ZED 2 depth data required for step one is available in the form of an ROS image message, where each pixel contains a distance measurement in the corresponding view direction as a floating-point number instead of RGB data. The system initially implements the creation of the required point cloud from this data with code taken from the official `image_pipeline` ROS

package, specifically the `point_cloud_xyz.cpp` file at [88]. Here, the code iterates over the RGB and depth data pixel-by-pixel, while utilizing additional data from a `camera_info` message published by the ZED 2 camera driver, which contains necessary information for 3D operations. This `camera_info` message comprises of the coordinates of the optical image center in pixel coordinates, $center_x$ and $center_y$, as well as focal length f dependent scaling factors:

$$\begin{aligned} constant_x &= \frac{1}{f_x} \\ constant_y &= \frac{1}{f_y} \end{aligned} \tag{3.1}$$

When the algorithm multiplies a pixel's coordinate and distance measurement by the respective factor, it gets the 3D point's lateral offset from the z-axis pointing into the image frame, from which it can calculate the x or y coordinate in the camera reference frame. It does this as follows: let u, v be the coordinates a pixel in the horizontal and vertical direction respectively and let $\mathrm{depth}(u, v)$ be the distance measurement at this pixel coordinate, taken from the depth image. Camera coordinates here do not align with vehicle coordinates, since for camera the camera, u and v denote directions in the image plane, with $\mathrm{depth}(u, v)$ denoting 3D depth 'forward' into the image plane. The vehicle coordinate frame aligns the x-axis into the driving direction, e.g., forward (corresponding to the direction of the camera's depth measurement) and the z-axis to point upwards. Thus, the algorithm calculates the coordinates of a point x, y, z in the vehicle coordinate system from the depth data and scaling factors from (3.1) as follows:

$$\begin{aligned} x &= \mathrm{depth}(u, v) \\ y &= -\big((u - center_x)\ \mathrm{depth}(u, v)\ constant_x\big) \\ z &= -\Big(\big(v - center_y\big)\ \mathrm{depth}(u, v)\ constant_y\Big) \end{aligned} \tag{3.2}$$

After obtaining these coordinates, the algorithm assigns the corresponding segmentation image RGB values to this point, generating a RGB point-cloud. Figure 3.9 shows an example of a resulting point cloud together with its input segmentation and depth image.

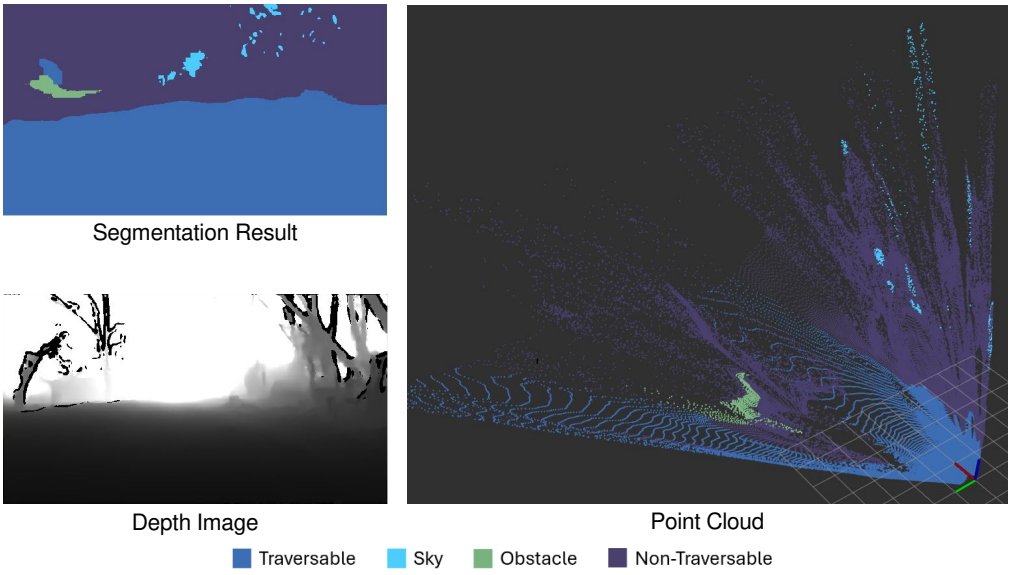


Figure 3.9: Visualization of steps one and two of the 3D integration pipeline, with a OFFSEG semantic segmentation result, corresponding depth image and the resulting point cloud. For the depth image, brighter points are farther away, brightest being 5 m.

However, performance of this initial system is inadequate – further expanded upon in Section 4.3.2 – leading to an alternative implementation. The existing code runs in python, where the `for`-loops utilized for iterating over the pixel space are the main culprit for the lacking performance as they are notoriously inefficient in python code. `numpy` usage solves this problem. Specifically, utilizing `numpy`'s `fromfunction` and `apply_along_axis` methods, it is possible to create vectors of offset factors in the u and v image directions, which can then apply along the axes of the depth data array. Specifically, a subsequent method adapts the functions from (3.3) to provide the following vectors with $\boldsymbol{u}$ and $\boldsymbol{v}$ as vectors of all image coordinates:

$$\begin{aligned} \boldsymbol{y} &= -\big((\boldsymbol{u} - center_x)\ constant_x\big) \\ \boldsymbol{z} &= -\Big(\big(\boldsymbol{v} - center_y\big)\ constant_y\Big) \end{aligned} \tag{3.3}$$

The algorithm then uses $\boldsymbol{y}$ and $\boldsymbol{z}$ (3.3) as lambdas to fill vectors spanning the dimensions of the image in the u and v directions, respectively. By multiplying these vectors with the depth data either with each row for y or each column for x – using `numpy`'s `apply_along_axis` method and another lambda – the algorithm calculates the y and z coordinates, where x is already supplied efficiently from the depth data in (3.2). This method improves performance (Section 4.3.2).

Top-Down Image Projection and LiDAR Fusion

With the segmentation point cloud now available, the next step is transforming it into a top-down 2D map. Code from [89] serves as a baseline, transforming a ROS 2 point cloud into a top-down image by splitting the point cloud in the z direction into slices of points, and then subsequently iterating over them. Within these slices, it discards points which lie outside of predefined limits for the top-down image, while translating the coordinates of all remaining points into pixel coordinates. The limits for the 2D top-down image initially limited the map to $\pm$7.5 m sideward and 15 m forward but are subsequently reduced to $\pm$2.5 m sideward and 5 m forward due to the navigation algorithm not requiring such a far map.

The algorithm achieves this by way of introducing a resolution value res, which defines the length in meters per pixel of the output image. Here, res is 0.025 m/p, resulting in a final top-down image resolution of 200 by 200 pixels for the 5 m by 5 m input. Next the algorithm translates the meter-based coordinates of the point cloud in the planar x and y direction into pixel coordinates: Here, u, v are again pixel coordinates corresponding to $-y$ and $-x$ respectively, correctly orienting the map vehicle's forward x-axis pointing upwards in the map. Then, the following formula computes the pixel coordinates, where $boundary_{\mathrm{l}}$ denotes the top-down images' leftward extent in meters and $boundary_{\mathrm{n}}$ denoting the same for the minimum forward distance:

$$\begin{aligned} u &= -\frac{y}{res} - \frac{boundary_{\mathrm{l}}}{res} \\ v &= -\frac{x}{res} + \frac{boundary_{\mathrm{n}}}{res} \end{aligned} \tag{3.4}$$

With these the pixel coordinates computed, the algorithm fills RGB color information into the output image array at the specified location, overwriting anything that is already present, and by iterating through the point cloud slices from the bottom to the top, it creates a top-down image. Section 3.6.3 later refines this algorithm using GPU acceleration via `torch`, because of performance concerns (Section 4.3.2).

Lastly, the `transform.py` node overlays LiDAR Data into the map in the same way, starting with the creation of a planar point cloud from the LiDAR data. A laser scan message, which the ROS drivers for the LiDAR provide, contains a list of range measurements, together with minimum and maximum angle $angle_{\min}, angle_{\max}$. The node creates an angular map by iterating over the angles of the laser scan points with respect to the forward direction, recording the tuples (a, b) of sine and cosine measurements for these angles. It does this via the incremental change of angle between two scanlines $angle_{\text{inc}}$ and the scanline index i ranging from 0 to the total number of scanlines n:

$$(a, b)_i = (\sin(angle_{\min} + i\ angle_{\text{inc}}), \cos(angle_{\min} + i\ angle_{\text{inc}})) \tag{3.5}$$

The node then multiplies this map elementwise with the corresponding LiDAR distance measurements to obtain x, y tuples denoting the measured point's position around the lidar in meters. It then applies (3.4) again with the x, y coordinates received from the LiDAR data to generate points in image coordinates, which it then connects with lines on the navigational map. Figure 3.10 depicts an example of the final output of this system together with an overlayed segmented input, and an evaluation of the systems performance is available in Section 4.3.2.

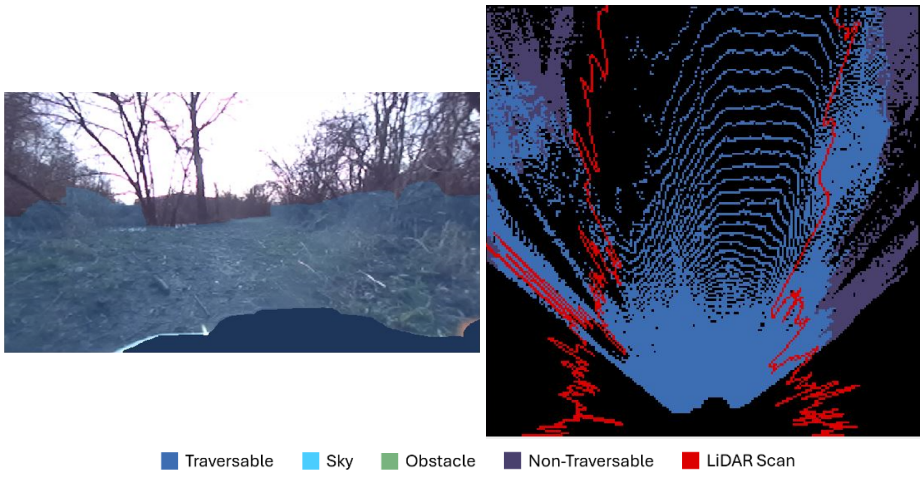


Figure 3.10: Example of the resulting top-down map from the 3D integration pipeline, together with the corresponding segmentation result overlayed over the camera input.

3.4 GPS Integration on a Small Vehicle

After the implementation of the perception system, the next step is the integration of the GPS system needed to move on to navigation.

This vehicle uses a Whadda WPI430 GPS module, sporting a u-blox NEO-7M GPS chip. It can make use of the global navigational satellite systems GPS, GLONASS, QZSS and Galileo (Section 2.2.2) and connects directly via a 9600 baud micro-Universal Serial Bus (USB) serial connection. While the module operates with its internal ceramic antenna in this project, it may alternatively operate with an external antenna via an SMA connector. Using the serial connection, the module will directly send National Marine Electronics Association (NMEA) messages containing status information and positional data to the computer. This is advantageous, because other modules require extra, complicated USB transistor-transistor logic (TTL) adapters.

This section will cover the mounting solution for the GPS sensor and its integration into the ROS infrastructure. In addition, it will present influences on GPS signal acquisition time and position data accuracy, together with improvements to the GPS system. These include EMI shielding for the plastic chassis components, standoffs, and other elements. Figure 3.11 presents these elements together with the initial GPS integration.

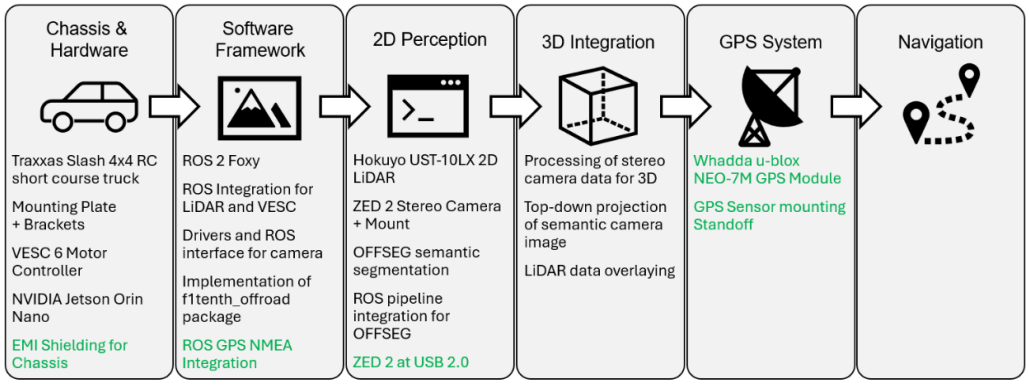


Figure 3.11: Thesis progress at the start of GPS development. Black text denotes items which the system currently implements, this section implements green text items.

3.4.1 Initial GPS Testing

Initially, the GPS sensor mounts simply to a standoff on the main mounting plate, and after connecting the receiver to the Jetson SoC, the next task is ROS integration. To facilitate this, the system integrates the `nmea_navsat_driver` ROS package into the software framework, which provides a serial NMEA driver node. Said node translates serial NMEA messages from the GPS module into ROS messages, publishing them into the system. These messages include velocity data, a 'fix', e.g., positional data as well as a time reference from the satellite network, all published to ROS at 1 Hz. In theory, these measurements should achieve the 1.5 m accuracy expected of GPS (Section 2.2.2), enough for the purposes of this work (Section 3.1.3).

In initial tests the ROS node however did not appear to send data to the ROS system. Testing inside, outside and with varying levels of building cover does not resolve the issue; no messages present in ROS. Next, analysis of raw serial data shows status information, which is not present in the ROS messages, providing grounds for the discovery that the GPS module has trouble acquiring satellite signals. Section 4.2.3 presents exact, quantified information on this together with the effects of improvements and modifications. Under the assumption that the module is in working order and because of the close proximity of the hardware on the mounting plate, the initial conclusion points to disruptive effects of Electromagnetic Interference (EMI) originating from other hardware elements. Further tests reveal that the GPS module works well when connected to another system – in this case a laptop – which further points towards an EMI issue with other hardware on the platform.

3.4.2 Analysis of GPS Electromagnetic Interference Effects

Thus, testing assesses the effects of other hardware, with Section 4.2.3 presenting results. The process of EMI analysis starts with selectively unplugging certain components from the System to measure their effect on the GPS module. First, tests assess the effect of the ethernet based

LiDAR by removing it, since quick research reveals occurrences of Ethernet-GPS interference. This proves effective at first, with the module acquiring a signal, however when activating the rest of the vehicle systems to record data, the signal accuracy and satellite acquisition is insufficient.

At this stage thinking turns towards electrical shielding, as it could possibly moderate the EMI effects on the GPS system while also mounting the sensor on a standoff on the roof of the vehicle – similar to sensors on full-size AV (Section 2.2) – to increase its distance to possible EMI sources. The application of aluminum-foil shielding to the inside of the vehicles hull and the custom designed 3D-printed standoff – both shown in Figure 3.12 – however do not provide conclusive results. Successive tests show that unplugging the camera dramatically improves signal accuracy and the number of satellites acquired, while observing the same effect achieved by unloading the camera driver during ROS initialization. Subsequent research again shows prior occurrences of this interaction, specifically with the ZED cameras used in the project. Thus, the tests conclude that the camera was the culprit for the GPS EMI effects.

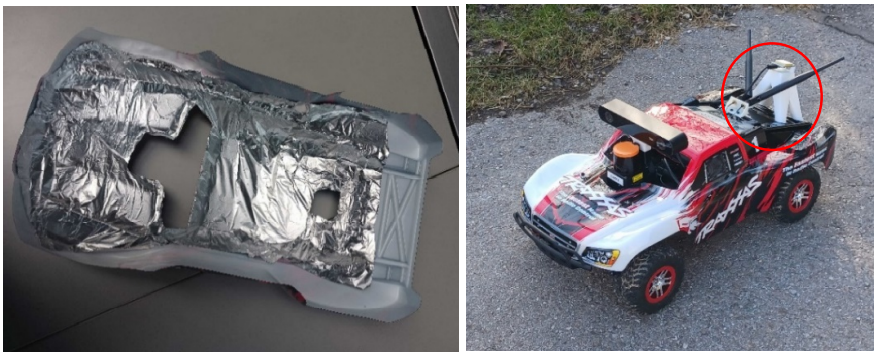


Figure 3.12: On the left: aluminum shielding applied to the inside of the vehicle's hull before installation of the grounding wire. On the right, the final mounting position of the GPS receiver at the rear of the vehicle on its standoff (red circle).

3.4.3 Modifications to Hardware and Software

Subsequently, modifications move the GPS receiver to the rear of the vehicle to increase its distance to the camera and reduce EMI influence. Other researchers, users and the manufacturer recommend stepping down the camera's USB interface bandwidth from the more capable 3.0 standard to 2.0 to further reduce EMI, so a USB 2.0 hub interfaces between the camera and the Jetson. This improves the number of GPS satellite acquisitions and subsequently the accuracy of the measurements into ranges acceptable for autonomous navigation purposes (Section 4.2.3). At the same time, it is limiting the camera to video capture at a maximum of VGA resolution and 12 FPS, which is inconsequential here since the camera runs at a lower resolution and framerate anyway (Section 3.2.3). As a safety measure, a further reduction of camera frame-frate to 7.5 FPS takes place. Figure 3.12 shows the final setup of the vehicle on the right, where it retains the aluminum shielding on the inside of its hull.

3.5 Navigation System

Looking at the big picture, the goal of all the modifications, sensorics and software developments which the sections up until present is true autonomous offroad navigation. With the chassis,

hardware integration, software framework, perception, and positioning components now preliminarily complete, this section will focus on the last element of autonomous offroad robots: navigation. The navigation system here will solely comprise of local pathfinding, utilizing the tentacle-based approach (Sections 2.4.1 and 3.1.3), with global positional information only having directing influence on the planner.

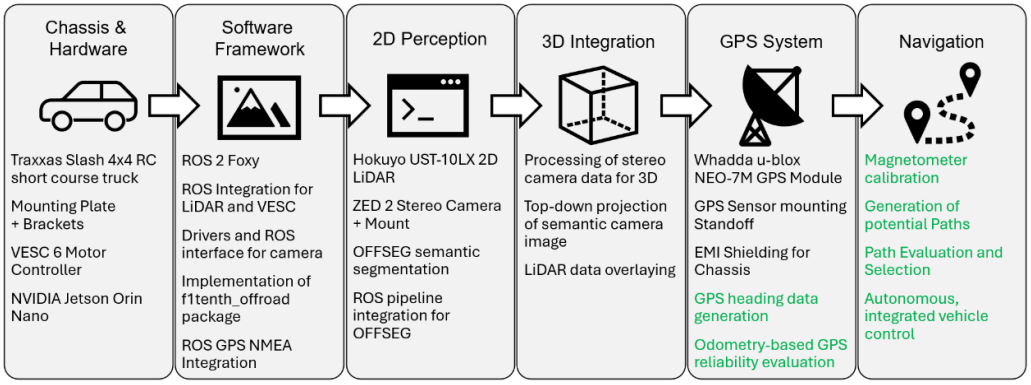


Figure 3.13: Project state and modifications for the navigation system. Black items signify the current state of the project, green items are modifications in this section.

This is nevertheless not a simple undertaking, with the navigation system utilizing not only data from the perception stages of the software package, but also data from the magnetometer, the GPS and VESC motor controller, leading to modifications of multiple systems (Figure 3.13). The following sections present the specific setup of the tentacle-based navigation approach, before explaining the challenge of acquiring reliable heading data and going over the path selection metrics and vehicle control. Figure 3.14 shows an overview of the navigation system, highlighting the multiple data sources from which the algorithm acquires data, with the previous perception stage supplying the top-down map.

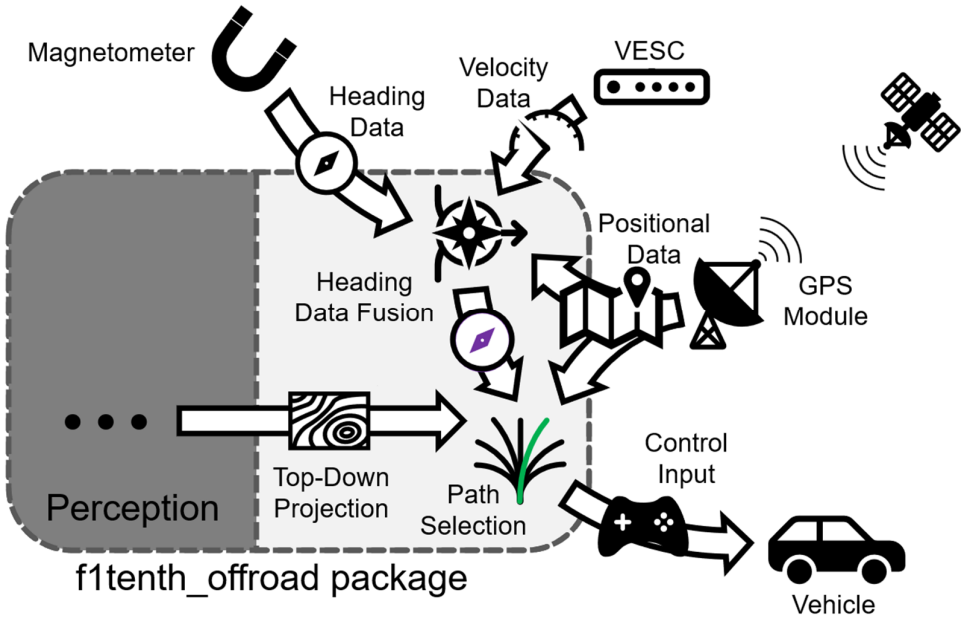


Figure 3.14: Schematic overview of the navigation side of the `f1tenth_offroad` ROS package.

3.5.1 Concept and Prospective Paths

As stated before, the pathfinding system in this project bases itself on the tentacle navigation approach (Section 2.4.1), and to use this method the algorithm must determine prospective paths (the 'tentacles'). Some researchers utilize a linear scaling approach for this, where they evenly space paths in terms of rate of turn between two extrema, however, for this project said approach is unsuitable. Due to the low vehicle speed – the maximum speed of the vehicle is only 1 m/s (Section 3.5.3) – the initial map range of 15 m is excessive (Section 3.3.2), justifying a reduction of the map range to 5 m. This results in a top-down map with a size of 5 m by 5 m, which the selection of prospective paths now needs to cover.

The specific path selection is the result of a development process, which implements many different numbers, angles, and rates of turn for the different paths. In the end, testing shows that 13 paths cover the image space adequately when additionally considering that the vehicle width v_w is 10 pixels during evaluation (Section 3.5.3). First, the path generation algorithm assigns lengths to the prospective paths, since the 110° FOV and limited side range of the map prohibit the computation of all paths from over the full 5 m range. From outermost to innermost, the navigation software sets the lengths of the paths in meters according to Equation (3.6) to not exceed the usable map space.

$$[1.45, 2.2, 3.2, 4.2, 4.8, 4.8, 4.8] \tag{3.6}$$

Then, it computes the points of a path, utilizing the maximum steering angle $steer_{max}$ and the respective number of segments in each arm nos_i, calculated with the values from (3.6) where the length of the arm with index i is l_i:

$$nos_i = l_i \cdot \frac{1}{res\ v_w} \tag{3.7}$$

Next, the algorithm needs the angle per meter apm, which is the maximum rate of change of angular direction in radian over 1 m, which it calculates using the minimum turning diameter mtd of 1.784 m. This is taken from the official F1TENTH documentation [34]. apm calculation then proceeds as follows:

$$apm = \frac{1}{0.5\ mtd\ res\ v_w} \tag{3.8}$$

Lastly, to set the direction and curve of each path, the algorithm requires the factor of maximum turn or scale factor sf, which determines the percentage of the maximum apm the vehicle will pursue for a specific path. For an arm of index i, where this index runs from 0 at the outermost, most curved path to 5 for least angled path – excluding the straight path – this factor is:

$$sf = \frac{1}{0.7i^{0.25i+1} + 1} \tag{3.9}$$

This equation yields the maximum turning angle $sf = 1$ for the outermost paths, and then scales back the rate of turn for the 'inner' paths, achieving an ideal covering of the navigational map. The last, straight path requires no curve, which the algorithm draws without any angling. Finally, the algorithm can now generate the paths, which it represents by their points. It generates these points for nos_i segments – the value of which is taken from (3.7) – of $0.5v_w$ pixels each, which

curve inward at the desired turn rate $steer$ at each step, the calculation of which utilizes (3.8) and (3.9):

$$steer = apm \cdot sf \tag{3.10}$$

For each run, the algorithm creates two mirrored paths curving both left and right, before adding the last, middle straight path to the prospective path list. This list is constant – with paths defined by the collection of their waypoints in the image space of the navigational map – and as such the algorithm reuses it in every computation step. Figure 3.15 shows the resulting paths forming a palm-like figure on said navigation map.

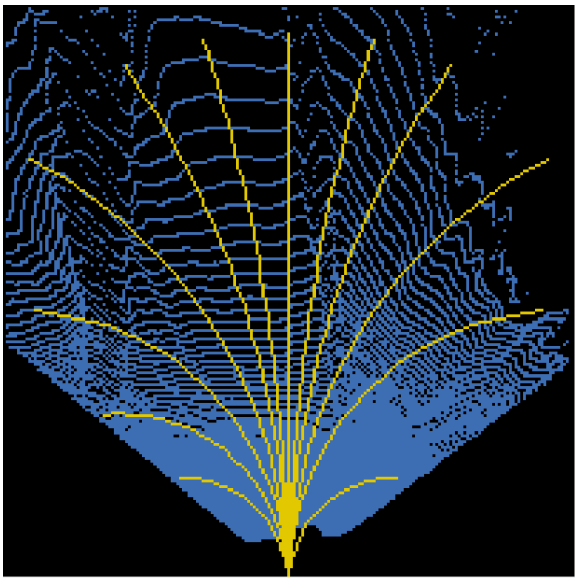


Figure 3.15: Representation of the prospective paths evaluated by the navigation system in yellow overlayed over a top-down navigational map showing traversable terrain in blue.

3.5.2 Magnetometer Calibration and Heading Data Fusion

With the creation of prospective paths finished, the next step is their evaluation, which however is impossible without heading data. If the direction of vehicle travel is unknown, the algorithm can make no decision over path suitability if it should drive towards a goal point.

Initially, the navigation algorithm attempts this solely via the integrated magnetometer of the ZED 2 camera, with the camera driver publishing magnetometer messages containing the strength of the magnetic field in all three dimensions. The software translates this into an angular heading h_{mag} – where pointing straight north is heading 0 – by utilizing the magnetic field strength in the x and y directions m_x, m_y and the $\arctan 2$ function with Equation (3.11).

$$h_{mag} = \arctan 2(m_y, m_x) \tag{3.11}$$

However, measurements prove to be inaccurate and unreliable. Said measurement responds non-linearly to smooth, linear turning, further underlining the magnetometer heading's inadequacy. Assumed cause for this behavior is the magnetic influence of other components such as the main propulsion motor's magnets, magnetic casings of sensors or fields generated by electric currents in wires.

Following these results, calibration of the magnetometer is paramount. The calibrations procedure starts by rotating the entire assembly in constant, known steps while the full system is running, spinning the wheels, and recording the readings from the magnetometer. Additionally, tests determine a fixed offset relative to true magnetic north and the magnetic north recorded by the sensor. Then, creation of a magnetic map follows, using the measurements taken in 10° intervals to create a look-up table containing the actual magnetic heading for each magnetometer reading in one-degree increments; the `mag_map.py` file contains code for this. Additionally, the software buffers the 40 Hz magnetometer readings over 10 readings – equating to 0.25s – and subsequently averages them using the `scipy` libraries' `circmean` function to reduce jitter. This, however, still proves insufficient. Further error sources such as the vehicle not being level when ascending or descending slopes – causing drifting of the magnetic heading measurement – or when coming close to large metal objects – like parked cars or solid metal fences – prove insurmountable even by repeated calibration attempts.

Thus, as an additional point of measurement, the software introduces the GPS position data into the heading calculations. By calculating the heading between two subsequent GPS fixes using the `pyproj` libraries' `geodesic` system, it creates another heading measurement via the continuous stream of GPS positioning data. As long as the GPS module can provide fixes, this system – in theory – can provide accurate heading measurements. At first, the software simply averages the two measurements, but that proves difficult since the GPS heading only updates every second. This means that heading changes within that second would only show as half their actual magnetometer measurement value to the navigation system, introducing erratic and inaccurate behavior, especially once the GPS heading updates. To combat this, the software only computes the average heading $h_{gps\_mag\_avg}$ once at the time of the GPS ping, while recording the current magnetometer heading h_{mag_timed}. Using these values, the software calculates the fused heading h_{fused} at each magnetometer heading update $h_{mag\_current}$ using formula (3.12), where all heading readings are in degrees and % is the modulo operation.

$$h_{fused} = h_{gps\_mag\_avg} + (h_{mag_{current}} - h_{mag\_timed}) \,\%\, 360 \tag{3.12}$$

This results in a dynamic average, where subsequent changes in magnetometer reading after the initial GPS heading average display at a one-to-one ratio in the fused heading – without 50% dilution as before – until the next GPS fix, which starts the next averaging process.

While this system is effective, it is troublesome when the vehicle is at standstill. In this situation, the GPS heading reading jumps unpredictably around the vehicle with the natural error of the GPS data, throwing off the fused heading. The heading fusion system combats this by introducing wheel odometry readings from the motor controller, specifically measurements of current forward speed v_{f} averaged over one second. It uses this to calculate a GPS trust factor fac_{gps}, calculated according to the following equation, the value of which it clamps between 0.0 and 1.0:

$$fac_{gps} = (v_{\mathrm{f}} - 0.2) * 5 \tag{3.13}$$

This results in a linearly scaling factor between the speeds of 0.2 m/s and 0.4 m/s, where it is at 0 for 0.2 m/s and at 1 for 0.4 m/s. The software then uses the value from Equation (3.12) to modulate the average between GPS heading and magnetometer heading, where a factor of 1.0 would equate to an equal average, and a value of 0.0 would negate the influence of the GPS heading. Said system allows the vehicle to switch to magnetic heading data at standstill and fuse it with GPS data to determine the current heading more accurately while moving, with quantified

results available in Section 4.4.2. Figure 3.16 shows an example of the algorithm in action, with odometry readings at standstill and while moving.

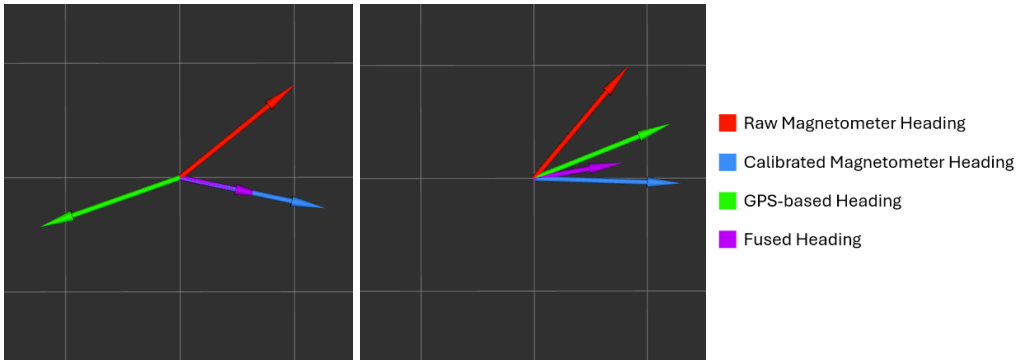


Figure 3.16: Heading odometry provided by the fusion system at standstill and while the vehicle is moving. Arrows indicate the heading reading of a specific sensor/system, with heading 0 defined in the upwards direction. At standstill, the GPS heading has no influence, and an average between GPS and magnetometer heading appears while moving.

3.5.3 Path Evaluation, Selection, and Robot Control

Now, the paths the algorithm generates (Section 3.5.1) require analysis to evaluate their viability. For this, the navigation system analyzes the established path points with the navigational map provided by the perception system. Since the map is an image, the algorithm establishes a color dictionary to recognize the four colors of the segmentation image. Additionally, it assumes that empty pixels are of the traversable class, while the red pixels of the LiDAR belong to the obstacle class.

Then the algorithm performs evaluation along the points of the paths. It does this by sectioning off square areas of the combined top-down segmentation and LiDAR map around the points of the path with size $v_w + 1$. It then analyzes the points in the following way: when a selected area contains any pixels not of the traversable class (e.g., sky, LiDAR, obstacle or non-traversable), it is impassable and the path evaluation ends, assigning the number of segments traversed until the obstruction as the path's evaluation result. It also regards the sky class as a not traversable class, because sky pixels should always be outside of the 5 m range of the map, such that when they fall inside the map, they are either reflections on glass or water, or mischaracterizations of real obstacles. When the algorithm detects no untraversable terrain on a path, it assigns that path the evaluation result nan, since the distance to an obstruction is larger than the visible path length.

Finally, the software compares the angle into which the path points – which is by extension an offset of the vehicle heading, hence its importance here (Section 3.5.2) – with the heading towards the global goal of the robot, the location of which is hard-coded into the navigation mechanism. If the algorithm finds one or more traversable paths with evaluation result nan, it chooses the path that points closest towards the direction of the goal, even if the directional difference is large. In this case, the algorithm discards all obstructed paths. If no unobstructed path is available, the navigation algorithm choses the path with the longest possible travel distance until an obstruction, and if the path shares this distance with more than one path, the algorithm chooses the one with the least amount of steering input, preferring straighter paths.

This algorithm proves to be effective but flawed in field testing, and Section 4.4 presents quantified results. Figure 3.17 shows an example of the algorithm at work.

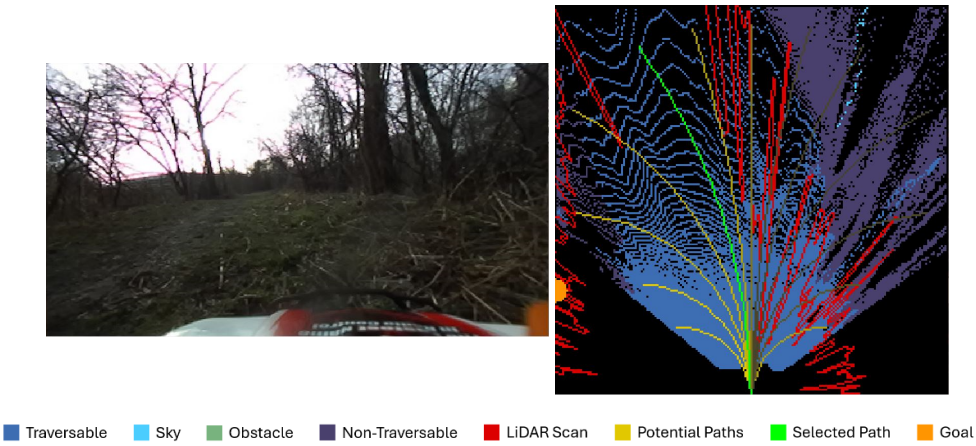


Figure 3.17: Result of the navigational algorithm (right) with the corresponding camera image (left). The color of a path indicates its evaluation result, with better results indicated by more intense yellow. The orange dot signifies the direction of the goal point.

The factor of maximum turn sf of this selected path steers the vehicle. First, if the distance to the target is larger than 5 m, the navigation algorithm assigns a desired speed of 1m/s, otherwise it is set to zero, halting the vehicle. Then, it multiplies the sf value with the maximum turn rate for the steering servo, 0.34, and passes that on as the steering control value. Next, it packs this information into a 'drive' message and publishes it into the ROS system, where the motor controller driver picks it up to forward the inputs to the motors. Section 3.6.4 further improves this control scheme.

3.6 Late-stage Modifications and final Developments

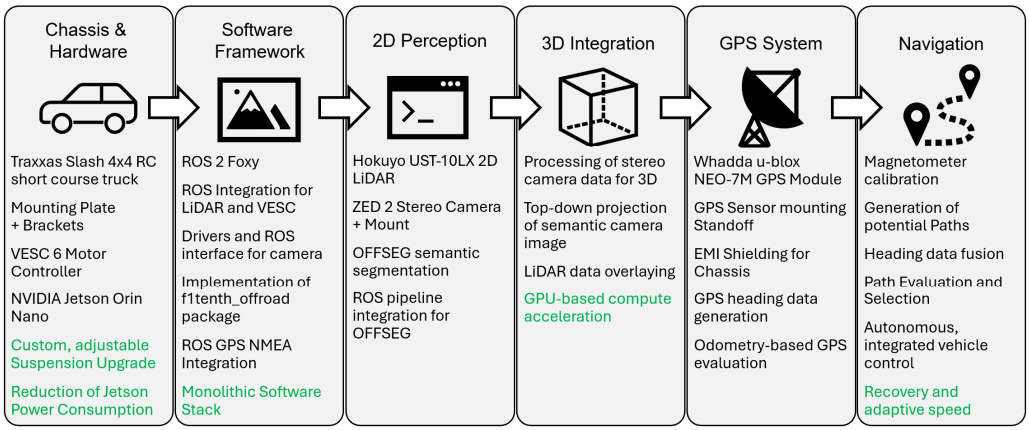


Figure 3.18: Project state at the end of if the initial development cycle. Final fixes, modifications, and improvements to earlier systems in the later stages of testing and development are green, with the existing state of the project in black.

During the development of the project and its incremental tests, problems appeared with some of the previous systems or components. Since these issues (Figure 3.18) are not part of the

initial development of the individual elements, they the previous sections detailing the initial development process do not cover them. This chapter now explains the final modifications and improvements to the current state of the project. These modifications mostly happened during development and testing of the navigational system, since they were only detectable once the entire system was running and ready for tests.

3.6.1 Suspension Upgrade

Firstly, an issue already detected during Section 3.2.3's initial tests: the additional weight of the sensing and compute equipment causes the vehicle to ride significantly lower than expected from the Traxxas Slash 4x4 chassis, reducing its offroad performance. To combat this, aftermarket, adjustable springs from Monster Hopups supplant the original springs and dampers, allowing adjustment to a more adequate ride height (Figure 3.19). This proves effective, while also allowing for the releveling of the car due to fix the uneven weight distribution.

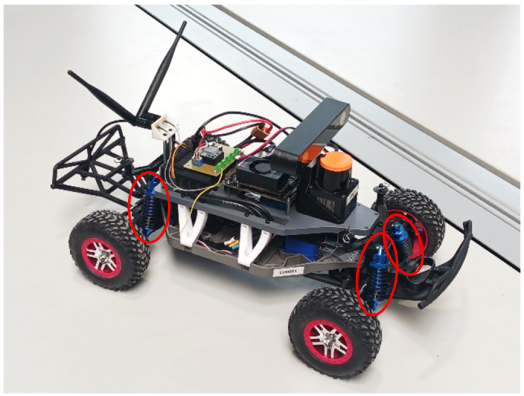


Figure 3.19: Traxxas Slash 4x4 chassis with the custom suspension upgrade, new shock-and-spring combo circled in red.

3.6.2 Power Delivery and Battery problems

During the active development of the software stack, the developing software proved troublesome on the live Jetson platform (Section 3.3). Among other problems, battery power loss is one of the issues listed. During tests with the newly completed navigation system these battery issues reappear when testing the autonomous navigation system in the field, where the vehicle would consistently shut off after 10 to 15 minutes of driving.

Endurance tests utilize a bench setup to evaluate different combinations of shut off sensors and software components until the system fails and noting the remaining battery charge afterwards. Section 4.2.4 presents quantified results, however the tests show that overheating is not the critical issue, even though cooling elongates the runtime time slightly. Reducing the internal power setting of the Jetson from 20W to 10W however allows tests to continue for up to 1 hour and 20 minutes while not impacting system performance, which is sufficient for outdoor operation.

3.6.3 Performance Improvements

Initial navigational testing showed that performance was a major concern, since the system could barely produce one image a second, which is insufficient for navigation and control. Thus,

performance improvements of the overall code are necessary, with the following subsections documenting the improvement process.

Monolithic Software Approach

First, tests analyze the ROS infrastructure. The tests provide information on significant delays and overhead introduced by the ROS message system, and thus the integration of a monolithic stack minimizes ROS overhead. This monolithic stack is a singular node, combining the functionality of all three existing nodes – 2D perception, 3D integration and navigation – into one. A separate node no longer publishes segmentation results as messages – since testing shows that large messages negatively affect the ROS system (Section 4.2.2) – because the monolithic node simply saves them as local variables before directly progressing to the next step. Similarly, the algorithm only locally saves the navigational map to avoid overhead.

The monolithic approach introduces several complications. First, different sensors supply data at different rates, and since the Camera is the slowest of the used sensors, it is the trigger for the combined process. All other sensors asynchronously deposit their readings into internal buffers, which the node then accesses once the computation starts. To further reduce performance issues, the system limits the pipeline to one instance running concurrently and drops all other camera images supplied during the computation.

While this makes debugging and development more difficult, it shows promising results and is thus the method of software execution used during in-the-field operation.

GPU-based Compute Acceleration

Analysis showed that the top-down image processing method (Section 3.3.2) took up a considerable amount of processing time every cycle, up to 700ms, with Section 4.3.2 presenting further quantified results. This section describes the process of moving the top-down transformation into `cuda` – which can execute the iterative approach from before faster while utilizing the `torch` libraries' `index_put_` function – since this fixes the problem.

Instead of filtering and assigning slices according to a vertical resolution, the new algorithm transforms the entire space of x and y coordinates at once into pixel coordinates and puts them into the top-down map. Since every pixel in the RGB input image has a corresponding pixel in the depth image, it has a corresponding 3D location, yielding matching array lengths with perfect index correlation. Given the transformed, image-space coordinates extracted from the x, y coordinates and the RGB segmentation data, `torch`'s `index_put_` completes the entire operation of putting the RGB data into the output top-down image array at the corresponding x, y position in one single, GPU optimized step.

This operation has one problem, however. If two datapoints fall into the same pixel – e.g., when vegetation grows vertically upwards, occupying the same x, y footprint in the point cloud – the behavior is undefined. In this case the top-most point should naturally be the one which takes precedence over all points below it. However, testing reveals that the opposite is the case when utilizing the data in its given format. There, the camera driver lays out the image's pixels – and thus the 3D points – starting at the top left corner, before then stepping through the data row-by-row, arriving at the bottom right corner last. Under the assumption that `index_put_` iterates through the array in the same way, the algorithm would give the lower points precedence, resulting in behavior opposite to the required variant.

The improved algorithm solves this by reverting the direction of the data array prior to computation. Thus, it lays the data into internal memory from the bottom right to the top left, which results in the correct behavior when encountering preoccupied cells in the map. Since this method is not reliant on the orientation of the input image – since the 3D data entirely governs the output orientation – this internal data layout has no further effect on the output image except the intended one. Section 4.3.2 presents the final performance gains from this strategy.

3.6.4 Navigational System Modifications

During final system integration testing and autonomous driving evaluation, the vehicle would occasionally continue driving straight into obstacles when it came to close to them. This is a direct result of the navigation algorithm (Section 3.5), where, when obstacles or terrain blocks all prospective paths, the vehicle will simply attempt to continue onwards on straight path.

An improved navigational algorithm solves this by – should the no path continue unobstructed for $2\ v_{\mathrm{w}}$ or 25 cm – commanding the vehicle to drive backwards in a straight line at 0.5 m/s until the next processing step. Thus, if the vehicle is facing an obstacle or terrain, the navigational algorithm will now back away until it finds an unobstructed path continuing for more than 25 cm, which will provide the vehicle with enough space to maneuver. Section 4.4.3 presents the precise effects of this minute change but suffice it to say here that this provided the vehicle with a simple and effective way to pilot itself out of previously unrecoverable situations, even if it requires multiple attempts.

Another problem the vehicle encounters in early tests is the following: when the vehicle attempts to take the straight path to the obstacle, as it the navigation algorithm compels it to do, situations can occur that cause it to take paths diverging from the intended route. This can result in the vehicle getting stuck in dead end terrain geometries or necessitate the vehicle operator to interrupt a test when the vehicle would otherwise drive into water. While a navigational algorithm would ideally automatically circumvent such situations, this is not within the scope of hard-, software and time available in this thesis. The algorithm thus implements an alternative solution: a waypoint system. This system supplants the initial singular goal point with a set of waypoints. The vehicle drives towards the first point on the list, and, upon reaching within 5 m of it, the navigational algorithm swaps to the next point, prompting the vehicle to continue from point to point until it reaches the final waypoint, e.g., the new goal. This allows for the customization of routes, which, while slightly less capable than full autonomous start-to-goal navigation, compensates for many of the systems issues and shortcomings, and as such the vehicle uses it going forward.

Lastly, the system limits itself to a maximum of 1.5 executions of the `f1tenth_offroad` pipeline to not overload the vehicle hardware and cause crashes, which appear during tests without such a cap on processing pipeline execution frequency. This 666 ms delay between control inputs however proved to be a major problem when turning tightly, as the vehicle's orientation, FOV and general situation can change significantly during this time, especially when navigating close to obstacles. Thus, the last change to the control system is the implementation of adaptive speed control. When executing the tightest turn with a diameter $mtd = 1.784$ m, the vehicle slows from 1 m/s to 0.5 m/s. Additionally, the vehicle drives the second tightest turn with a diameter of $2\ mtd$ at 0.75 m/s. Further testing and validation prove this to be highly effective (Section 4.4.3).

4 Results

The previous chapter finished the development of this thesis' project: the creation of a small-scale vehicle based on the F1TENTH platform fully capable of autonomous offroad navigation. Following this, the current chapter will now focus on the performance of the developed hard- and software by presenting the results of the research done on the platform.

While certain unified test tracks, autonomous offroad vehicle competitions and challenges exist – which would certainly add interesting datapoints to this work if it would test on them – this thesis will perform no comparative evaluation. The reasoning for this is twofold: a peer comparison of this system is outside the scope of a bachelor's thesis, and even if this work would perform such a comparison, the results would be of limited meaning, owing to the unique status of this project in the field of autonomous offroad vehicles (Section 1.1).

The following sections comprise of the final evaluation of the vehicle's autonomous offroad capabilities, and of the assessment of individual system components, intermediate system stages as well as limitations and other problems derived from the workings of the platform. The following sections will give an overview of the completed system, before evaluating the systems compute and sensor system and moving to its performance for both the perception and the navigation system.

4.1 Completed System Overview

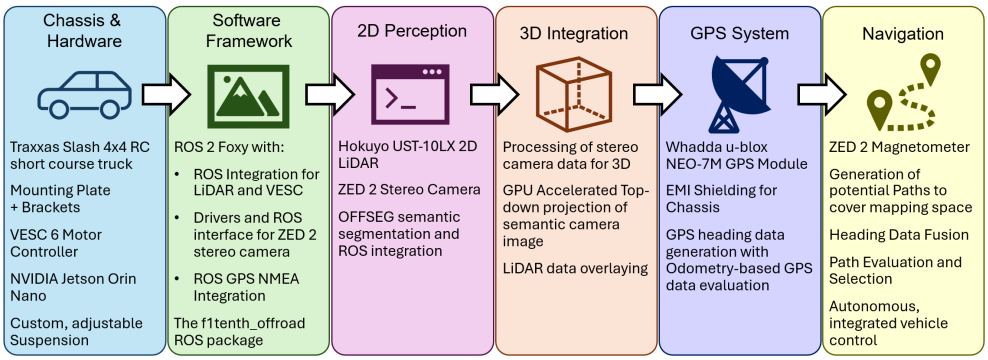


Figure 4.1: Overview of hard- and software components of the finished vehicle platform.

Before analyzing the results and performance of the finished vehicle system, it is worthwhile to gain a complete overview of the final setup, presented in short form Figure 4.1. This is relevant when considering that this vehicle is currently unique in its configuration among AOV, combining small size and low cost with full-size offroad capabilities. Thus, this work considers the vehicle setup a relevant scientific result of the work done in this thesis.

4.1.1 Hardware Overview

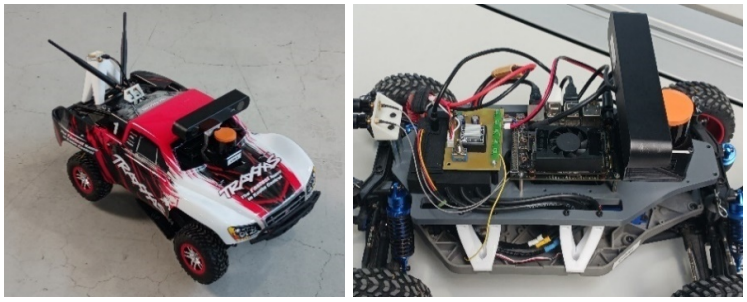


Figure 4.2: On the left, the final, completed vehicle hardware platform with its outer shell. On the right, the main mounting platform with the compute and sensing hardware.

The vehicle developed in this thesis is a first of its kind small-scale fully functional offroad AOV, depicted in Figure 4.2 on the left. It combines the functionality of a full size fully autonomous offroad vehicle on the small footprint, low cost F1TENTH platform (Sections 2.1.4 and 3.1.1). The base of the vehicle is the Traxxas Slash 4x4 short course desert truck, a capable offroad RC car platform, modified with custom, adjustable height shock-and-spring combination units by Monster Hopups (Section 3.6.1). Attached to its chassis with custom designed 3D-printed riser brackets is a mounting platform – depicted in Figure 4.2 on the right – carrying the hardware needed for autonomous offroad operations (Section 3.2.1).

This hardware consists of state-of-the-art sensors and compute solutions, enabling tests previously confined to much larger systems. This vehicle combines a ZED 2 stereo camera with a Hokuyo UST-10LX LiDAR sensor for environmental perception (Sections 2.2.1 and Figure 4.2). These sensors mount to the front of the vehicle on a combined, 3D-printed mounting stalk specially designed for this hardware combination. A Jetson Orin Nano 8GB SoC with a 1 TB Samsung 970 EVO NVMe SSD (Sections 2.3.3 and 3.1.4) mounts to the platform and handles perception and navigation operations, together with a 12V power board and dual WLAN antennae. Lastly, on the outer shell at the back of the vehicle on its 3D printed standoff is the Whadda WPI430 GPS module (Section 2.2.2 and 3.4). Attached to the inside of this plastic shell is Aluminum Foil EMI shielding to improve GPS accuracy (Section 3.4.3).

The VESC 6 motor controller also mounts to the platform and controls the main motor and steering servo while providing odometry (Section 2.1.4 and 2.2.2). The vehicle powers off of a Traxxas 5000 mAh 11.1 V lithium polymer (LiPo) battery. Table 4.1 shows a hardware overview.

Table 4.1: Final hardware setup of this thesis' AOV platform

Chassis	Mechatronics	Systemic Hardware	Sensorics
Traxxas Slash 4x4 Short course desert truck	VESC 6 motor controller	Battery to 12V onboard Power converter	Stereolabs ZED 2 stereo camera with built in odometry and magnetometer
Monster Hopups custom adjustable height suspension system	Traxxas Velineon 3500 3-phase brushless main motor	NVIDIA Jetson Orin Nano with 8 GB RAM	Hokuyo UST-10LX 2D LiDAR sensor
Aluminum Foil EMI Shielding	Traxxas 2075 waterproof steering servo	Dual WLAN Antennae	Whadda WPI430 u-blox NEO-7M GPS module
	Traxxas 5000 mAh 11.1V 3-cell Lithium Polymer Battery		

4.1.2 Sensor Configuration and Software Overview

The physical hardware is operated by ROS 2 (Section 2.3.3), the `f1tenth_system` [82] ROS package (Section 3.2.2) and the new `f1tenth_offroad` [84] ROS software package which this thesis develops (Sections 3.3 and 3.5). The system runs on an Ubuntu 20.04 Linux distribution installed on the Jetson SoC (Section 3.2.2). The platform's sensors' specifications are as follows:

- ZED 2 Camera: 110° horizontal and 70° vertical FOV stereo vision, 672 by 376 pixel resolution at 7.5 FPS nominal setting (see Section 4.2.2 for FPS performance)
- ZED 2 IMU and magnetometer: odometry and magnetic field data at 400 Hz.
- Hokuyo UST-10LX 2D LiDAR: 270° horizontal FOV with 1081 scan points at 40Hz.
- Whadda WPI430 GPS: satellite-based velocity and position information at 1 Hz
- VESC 6 motor controller: odometry data at 50 Hz

These data feeds are provided by either by drivers included with the `f1tenth_system` package (VESC, LiDAR), or by custom ROS driver packages for the ZED 2 stereo camera (`zed_ros_wrapper` [83]) and the GPS (`nmea_navsat_driver` [90]). Then, they feed into the `f1tenth_offroad` package (Sections 3.3, 3.5 and Figure 4.3).

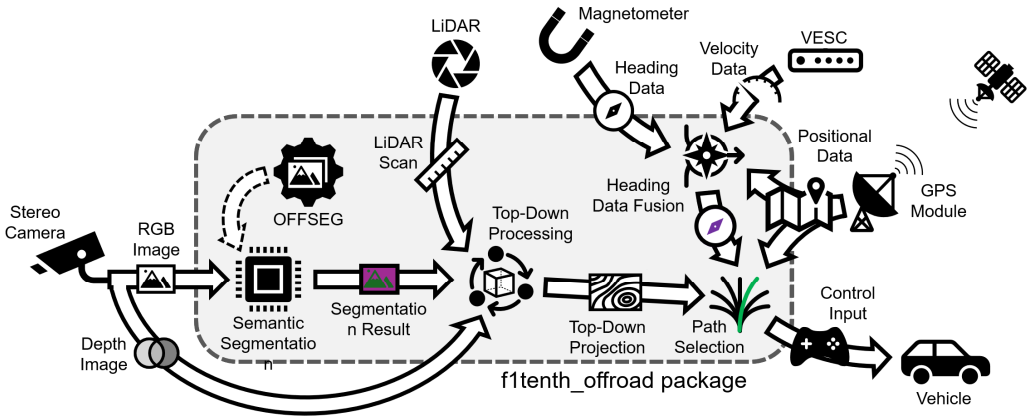


Figure 4.3: Schematic overview of the complete `f1tenth_offroad` ROS package.

First, the system segments incoming camera RGB images into traversable, non-traversable, sky and obstacle regions using the OFFSEG semantic segmentation system (Sections 2.3.1 and 3.3.1). It then uses the corresponding camera-provided depth image information to project the segmentation result of into a top-down map showing the area in front of the vehicle (Sections 2.3.1 and 3.3.2). At this stage, the LiDAR data is overlayed as a red line signifying the detected edges of surrounding objects. Simultaneously, data from the magnetometer, GPS and motor controller fuse to create a reliable heading reading for the robot (Sections 2.3.2 and 3.5.2). The software then uses the processed top-down map with the heading data to select a prospective path for the vehicle. For this, it evaluates a selection of thirteen precomputed, prospective paths – differing by their respective steering input – based on the map regions they pass through (Sections 2.4.1, 3.5.1 and 3.5.3). The algorithm always chooses unobstructed paths over obstructed ones, and of these it chooses the one pointing closest to the robot's global positional goal. Lastly, it transmits the steering value of the selected path to the motor controller, which drives the wheels at -0.5 to 1 m/s until the robot reaches within 5 m of his goal point, whereupon the algorithm assumes that it arrived at the goal and forward motion stops.

4.2 Compute and Sensor System

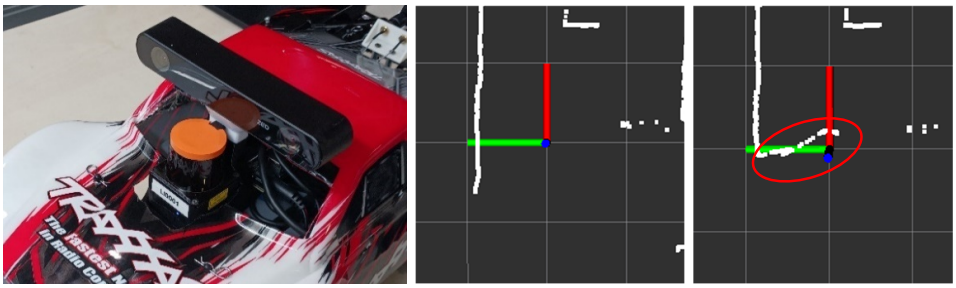
Before assessing the performance characteristics of the software this thesis implements and evaluating the offroad navigation performance of the vehicle, it is worthwhile to assess the sensor and sensor fusion performance individually. The current section provides insight into capabilities of the sensors and sensor systems and the constraints the data processing is facing. Since this project is the first of its kind in terms of offroad autonomous navigational capabilities at this scale, the influences said scale has on the hardware performance and the system as a whole are relevant research results.

This section will evaluate the LiDAR and Camera sensors' performance, before moving to the GPS module and analyzing the power system's capabilities.

4.2.1 LiDAR FOV and Ground Interference

The 2D LiDAR sensors of the type used in this project are a common staple on the F1TENTH vehicles (Section 2.2.3 and 3.2.1), and thus enjoy good ROS system integration, with data reliably and constantly supplied at 40 Hz. However, over the course of this project, this thesis identified two issues concerning LiDAR data: occluded FOV and an effect called 'ground-strikes' for the purposes of this work.

First, this paragraph assesses occluded FOV. Due to the role of the LiDAR sensor as an exteroceptive sensor, it naturally must interface with the environment of the vehicle, and thus requires a through-hole in the outer shell of the vehicle. This mounting location (Figure 4.4 on the left) however proves to be less than ideal. Figure 4.4 pictures LiDAR scan results with and without the outer plastic vehicle chassis shell in the middle and right images. As can is evident from the figure, attaching the shell produces additional data points on the scan, which the software must filter out of the data before applying it to the navigational map. Since a significant amount of the erroneous datapoints are located behind the origin point of the scan in x-direction – red axis, pointing forward in the vehicle frame of reference – they are not relevant to the navigational map. The software however must remove points which are in front of the y-axis (green) from the results based on their distance to the origin, resulting in a minimum scan distance of 20 cm. Sections 4.3.2 and 4.4.1 present the consequences of this.



Without Shell With Shell

Figure 4.4: Left, LiDAR and camera sensor mounting position, middle and right, LiDAR scan data as white points with and without the outer plastic shell, with shell interference circled in red. The intersection point of the axes signifies the scan center point or point of origin.

Secondly, ground-strikes are a product of either the vehicle pitching forward or ascending terrain in front of the vehicle. Since the LiDAR mounts only ca. 15cm off the ground and perpendicular

to the plane of the vehicle, any forward tilt of the chassis exceeding 2° relative to the terrain will cause a ground-strike, were the LiDAR laser rays impact the ground instead of an obstacle. Similarly, any elevation features of the terrain exceeding 15cm in height relative to the vehicle's ground plane – e.g., an upwards slope – will show up on the navigation map. Figure 4.5 illustrates this by showing a tilt-based ground-strike; and Section 4.4.1 analyzes the further consequences of this effect.

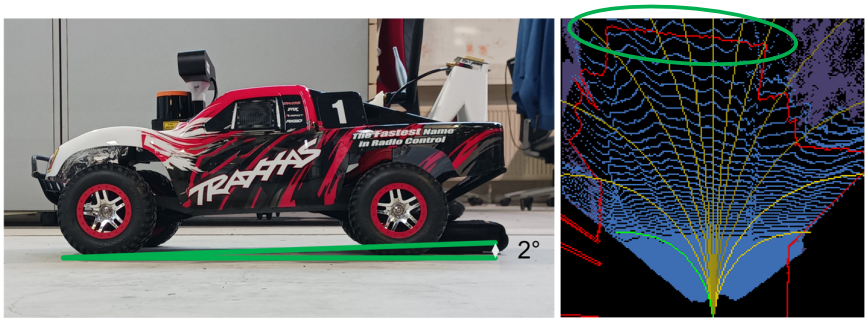
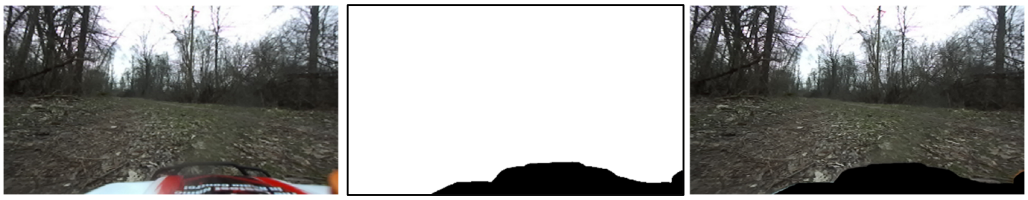


Figure 4.5: Left, the minimum tilt forward to cause a ground strike (barely visible in the tires as the suspension is drooping). Right, a top-down map (Section 3.3.2) with a ground-strike in the LiDAR data, were the ground-strike-related data points in the green circle.

4.2.2 Stereo Camera Performance: FOV, FPS and Depth Quality

After Analyzing the LiDAR sensor's performance, next, this section presents the ZED 2 stereo camera's FOV and occlusions thereof, FPS results in ROS and the `f1tenth_offroad` package for different configurations and the camera's depth quality.

Similar to the LiDAR, due to the camera's mounting position depicted in Figure 4.4 on the left, its FOV is partly occluded by the hood and front bumper of the vehicle as well as the edge of the LiDAR sensor placed before the camera. Thus, a mask blocks out the areas of incoming images occluded by these elements, resulting in blacked out areas that reduce unintended errors of the perception system (Figure 4.6).



Camera Image | Mask | Masked Camera Image

Figure 4.6: Masking pipeline for example incoming camera data, from left to right, with the two input images on the left and in the middle and the resulting output on the right.

Next, this section will cover the camera's framerate and resolution. As Section 4.1.2 presents – with reasons in Section 3.2.3 – the camera records its video feed at a 'VGA' resolution and a target frame rate of 7.5 FPS. Recording at higher resolutions, as done in the beginning of the project, increases data rate rapidly, generating gigabytes of data over continued test runs, with the earliest, 40-minute test run producing 150 Gigabytes (GB) of data. This was unsustainable, and, as Section 4.3.1 shows, unnecessary. At the same time, the ROS system's internal message frequency diagnostic tool never quite reaches the target framerate set in the camera

driver. Table 4.2 shows the test results analyzing camera resolution, target framerate and actual framerate measured in the package, measured over five hundred messages each.

Table 4.2: Test results of camera framerate in ROS and in the f1tenth_offroad package, as well as the estimated data rate for the package FPS reading. VGA resolution is 672 by 376 pixels at 1.01 MB per message, HD720 is 1280 by 720 pixels at 3.69 MB per message.

Setting of FPS @ Resolution	7.5 FPS @ VGA	10 FPS @ VGA	15 FPS @ VGA	7.5 FPS @ HD720	10 FPS @ HD720	15 FPS @ HD720
ROS FPS	5.6 FPS	6.6 FPS	8.2 FPS	2.4 FPS	3.4 FPS	3.8 FPS
Package FPS	7.5 FPS	9.8 FPS	11.3 FPS	7.4 FPS	9.3 FPS	10.8 FPS
Data Rate	7.6 MB/s	9.9 MB/s	11.4 MB/s	27.3 MB/s	34.3 MB/s	39.9 MB/s

The data shows that the ROS utility's frequency measurements are unreliable – at least for large messages – with message size and data rate corresponding to lower FPS readings of the ROS utility. Assuming the package's FPS reading is accurate, the camera does not reach the full 15 FPS for any setting and struggles with the larger messages at 10 FPS. Choosing 7.5 FPS at VGA resolution both satisfies storage requirements (with only ca. 25 GB of data per run) as well as providing enough FPS for the segmentation algorithm (Section 4.3.1 and 4.3.1).

Lastly, the ZED 2 stereo camera generates 3D point cloud data which the software uses to create top-down maps (Sections 3.3.2 and 4.1.2). Its driver provides multiple options for this, under them 'Ultra' and 'Neural'. Initially, the driver utilizes the setting 'Ultra' to alleviate the FPS issues, however it later swaps to 'Neural' because of the much higher quality depth data. Here, the camera driver utilizes a GPU-based neural depth engine to optimize the 3D output, leading to additional issues (Sections 3.6.2 and 4.2.4). Figure 4.7 shows the improved smoothness and continuity of the depth images and subsequent top-down maps when using the 'Neural' setting, which more accurately reflect the environmental circumstances. Generating better maps provides better navigational solutions, and thus the project uses 'Neural' depth.

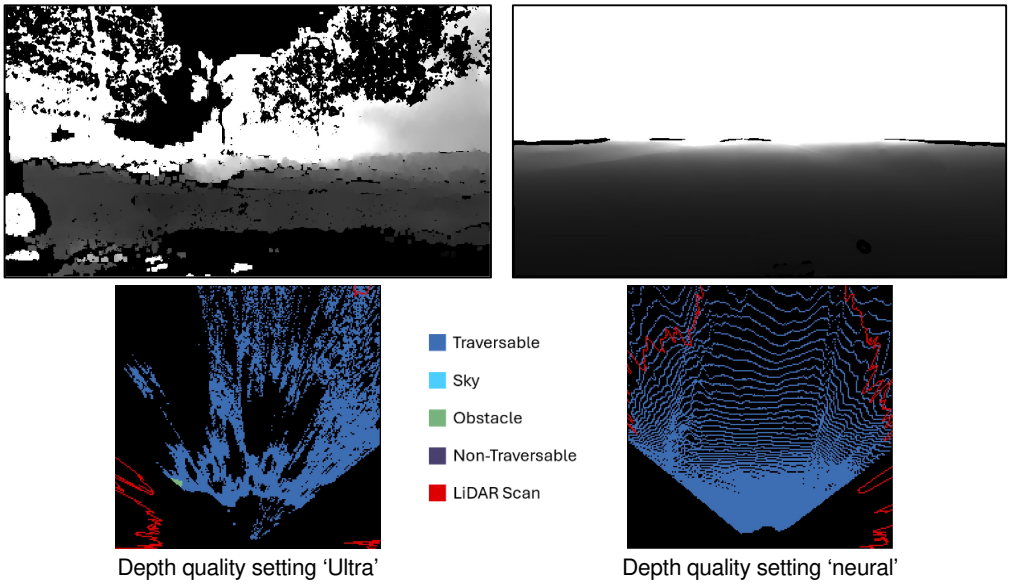


Figure 4.7: Comparison between depth quality settings 'Ultra' and 'Neural', with depth images of the same scene on top with corresponding top-down maps below (Section 3.3.2).

4.2.3 GPS Interference and Accuracy

The integration of the vehicle's GPS system proved uniquely difficult (Section 3.4). During this integration, the vehicle ran through multiple different hardware configurations while trying to increase GPS accuracy and satellite acquisition speed. Figure 4.8 shows data from tests conducted to evaluate these configurations, where the vehicle is either stationary at a known point or moving on a known path, while collecting data over the course of five minutes.

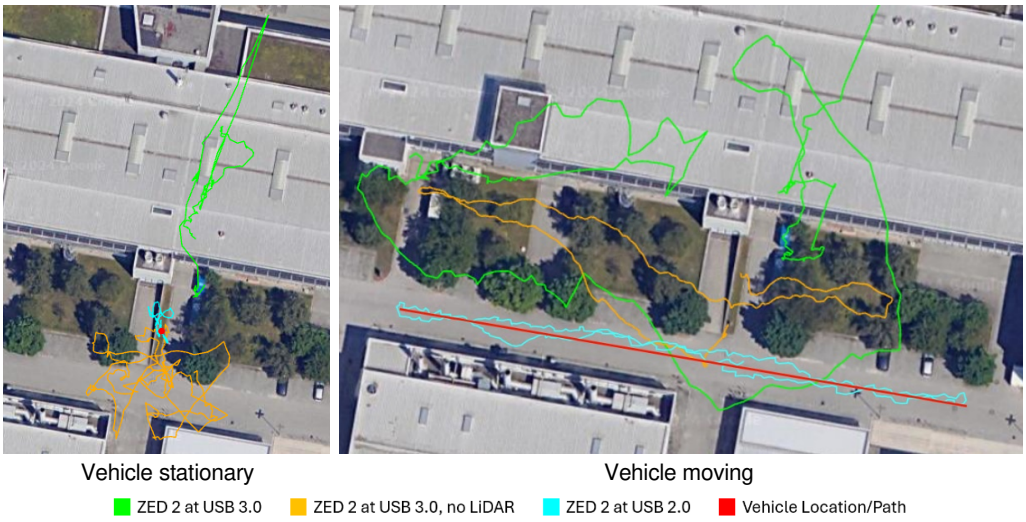


Figure 4.8: Visualization of Vehicle GPS data with multiple hardware configurations while the vehicle is stationary as well as when moving. The vehicle traversed the path for the 'Vehicle moving' data in both directions.

Initially, the ZED 2 camera connects to the Jetson SoC via USB 3.0, and the data from this configuration shows intense deviations from the real position of over 50 m, jumping multiple meters in any direction between GPS fixes. This is insufficient for the purposes of this project, since the navigational algorithm uses the GPS data to steer the vehicle via both heading information and a distance to the target (Section 4.1.2), both of which are unreliable here. The vehicle intends to reach within 5 m of the goal point, which is unattainable with over 50 m deviation, and it needs to use the GPS position for heading data fusion, again unfeasible with jumps over multiple meters in between measurements when moving at 1 m/s.

Next, tests show that disconnecting the ethernet-based LiDAR – while it leads to a small improvement in the data – is still insufficient for navigation. Static deviation is on the order of 20m, while the large deviations in direction of the recorded path in Figure 4.8 show that heading data remains unreliable.

Finally, modifications add shielding to the vehicle's hull and downgrade the camera's USB connection to USB 2.0 while reconnecting the LiDAR sensor. The final measurements show positional accuracies within 5 m and a high degree of the heading's directional accuracy, which improves the GPS system's capabilities enough to facilitate navigation.

4.2.4 Battery and Power System

After discovering power problems (Section 3.6.2), testing analyzes the performance of the power delivery system. In these tests, the system repeatedly runs until it crashes or reaches a runtime

of 45 minutes. Meanwhile, individual test runs contain different modifications to parts of the system to find the cause of the reduced runtime compared to the initial test, which ran for 40 minutes without issues. In these tests (results in Table 4.3) each run repeats with two different batteries to exclude their influence on the test results. In these tests the full `f1tenth_system` with the ZED 2 drivers runs, while spinning the wheels at to simulate operating conditions.

Table 4.3: Power system tests: modified elements, resulting runtime and battery voltage at crash.

Modification	Runtime	Remaining Voltage
'Neural' camera depth quality and 20W Jetson, cooled	16 min	4.03 V
Camera disconnected	Stopped at 45 min	4.00 V
Camera framerate capped at 1.5 FPS	17.3 min	3.98 V
Camera Depth quality set to 'Quality'	Stopped at 45 min	4.00 V
Camera Depth quality set to 'Ultra'	Stopped at 45 min	4.01 V
Jetson Power Level reduced to 10W	Stopped at 45 min	3.85 V
Final test, Jetson at 10W with depth quality 'neural'	Stopped at 1:20 h	3.72 V

Test results indicate that the ZED 2 camera – specifically the 'depth quality' setting in its firmware – impact the runtime and voltage after crash, and cooling has little to no effect on the runtime. When reducing the depth quality from 'Neural' to another setting such as 'Quality' or 'Ultra', the system completes the full 45-minute test, and further research shows that the 'Neural' depth setting utilizes the system's GPU, increasing power draw.

With the Jetson at its top performance setting, the system crashes at around 4.00 V on the battery, because the 12V power distribution board can no longer supply enough power to the Jetson when the battery drops below that voltage. Since the 'Neural' depth setting is highly favored (Section 4.2.2), further tests search for a solution that allows its continuous use. Reducing the Jetson's internal power setting called 'NVPmodel' from the 20W top setting to a lower 10W, the test reaches 45 minutes with depth setting 'Neural' and 3.85V remaining battery voltage. A subsequent endurance test shows the system can now run for over an hour without crashing.

4.3 Perception Performance

After covering the hardware, software, and sensors individually in an isolated manner, this chapter will now concern itself with the combined perception system's performance. Here, the software processes the LiDAR and Camera data streams and combines them before advancing to the navigation system. The perception system (Schematics in Figure 3.7) includes two stages, the OFFSEG semantic segmentation suite and the 3D processing system, and the following sections will analyze their performance.

4.3.1 OFFSEG Semantic Segmentation

This project utilizes the OFFSEG Semantic Segmentation system (Section 2.3.1). Said system transforms the incoming camera feed into information helpful to the navigation algorithm by

segmenting image regions into the groups traversable (with certain subgroups), non-traversable, sky and obstacle. Section 3.1.3 explains the reasoning for choosing this system while Section 3.3.1 covers its integration into the perception system, whereas the current section concerns itself with the overall segmentation accuracy and the compute cost associated with OFFSEG.

Overall Accuracy

OFFSEG is a two-stage segmentation system, that first performs a more general segmentation before moving on to a finer classification of the area detected as traversable. In the testing done in this thesis, the second stage algorithm never seems to work quite right. The examples in Figure 4.9 showcase this: the second stage algorithm provides blotchy results, which almost unanimously either incorrectly or incompletely detect puddles, grass, or mud in the camera image. The specific reasoning for this remains unknown, but it is safe to assume that the OFFSEG algorithm does not favor the camera setup used in this project. With the second stage algorithm in such an unreliable state, compounded with the fact that the algorithm 'jumps' between images – producing wildly different results for consecutive images sporting minimal differences – the perception systems does not use the second stage segmentation. Additionally, its high compute cost further prohibits its implementation, which the next section shows.

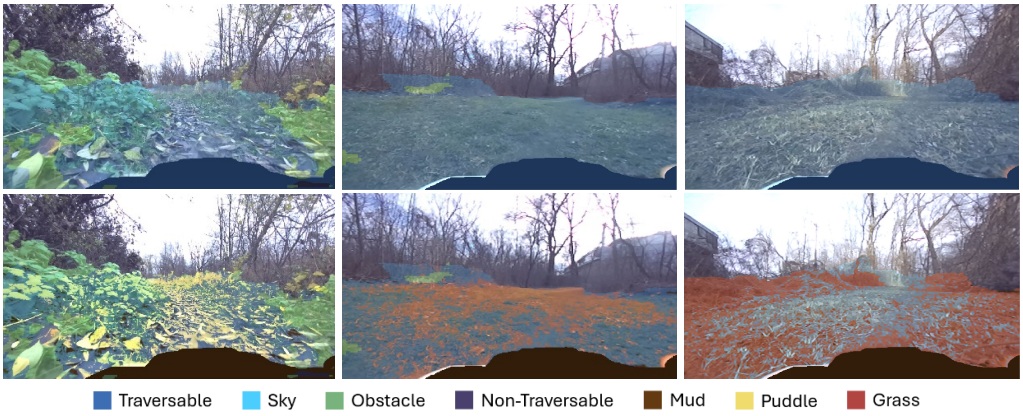


Figure 4.9: Examples of the two stages of the OFFSEG system for comparison. On the top, the single- or first-stage segmentation, on the bottom the two-stage segmentation, both overlayed over their respective camera input.

While the first stage algorithm also is inconsistent – detecting surrounding vegetation or bushes as traversable terrain (Figure 4.9) – it has a much higher overall accuracy and usability compared to the two-stage variant. Also, by restricting the segmentation to the four classes mentioned above, the navigation algorithm simplifies, because it does not have to deal with the additional task of operating on the different advanced ground types.

Additionally, it is worth mentioning that the camera resolution change from HD720 to VGA seems to have no effect on the segmentation accuracy; however, it significantly impacts the perception systems performance, which the following section presents.

Lastly, the masking (Section 4.2.2) positively affects the segmentation system. By blocking out the part of the vehicle's chassis visible in the camera image, the perception system reduces artifacts previously present around the edges of the chassis and the corner of the LiDAR sensor, which would throw off the navigation algorithm's evaluation.

Compute Cost

Table 4.4 shows the respective time required for processing and the resulting FPS – without the 1.5 FPS cap used in operation – for different configurations, where tests record data over 5 minutes. The analysis shows that the only viable option is VGA resolution with the single stage algorithm since vehicle control requires at least 1 FPS to be effective. With this configuration, the segmentation takes 183ms, resulting in an uncapped framerate of 2.25 FPS for the software.

Table 4.4: Segmentation System timing and FPS depending on resolution and stages used.

Resolution	Algorithm used	Segmentation duration	Resulting FPS
VGA (672 x 376)	Single-Stage	183ms	2.25 FPS
VGA (672 x 376)	Two-Stage	2988ms	0.30 FPS
HD720 (1280 x 720)	Single-Stage	630ms	0.53 FPS
HD720 (1280 x 720)	Two-Stage	14317ms	0.07 FPS

When using the two-stage system, segmentation time increases by a factor of ca. 16, which results in a best-case refresh rate of ca. 0.5 FPS, which is unsuitable. Also, utilizing the resolution used at the start of this thesis – HD720 – drastically decreases performance by a factor of more than three, where a single segmentation step with the two-stage algorithm takes more than 14 s. Thus, the system uses the single-stage OFFSEG algorithm at VGA resolution.

4.3.2 3D Processing and top-down Projection

The next step in the Perception pipeline is the generation of a 2D top-down map from the input segmentation data, LiDAR scan and 3D depth information (Section 4.1.2). Table 4.5 details the performance of different implementations of the top-down algorithm used during the development of the final algorithm (Section 3.3.2). The data shows that the GPU-accelerated torch-based version of the top-down algorithm increases performance by a factor of 24, enabling a 156% higher computation frequency for the entire software stack. Also evident from the data is the 14-times increase in processing duration when the system uses the higher initial resolution setting of HD720, further cementing the unviability of this configuration.

Table 4.5: 3D Processing algorithm processing time and resulting FPS depending on configuration.

Configuration	Processing duration	Resulting FPS
For-loops (initial), VGA	682ms	0.88 FPS
`torch`-accelerated on GPU, VGA	28ms	2.25 FPS
`torch`-accelerated on GPU, HD720	389ms	0.52 FPS

The top-down 3D integration mechanism fuses the LiDAR data into the segmentation results, which is vital. Because of the poor accuracy of the OFFSEG segmentation algorithm which the previous section details, the LiDAR data is a critical fallback stopping the vehicle from colliding with unsegmented obstacles such as bushes (Section 4.4). Here, the problem of LiDAR and Camera data synchronization becomes apparent. When operating the fusion algorithm, there are discrepancies between LiDAR and Camera timestamps of between 0.2 s and 1.85 s. The

cause for this is as of the writing of this text unknown but leads to situations such as in the middle of Figure 4.10, where the detected position of the fire hydrant by the camera (dark purple pixels) does not match the LiDAR data (red 'shadow').

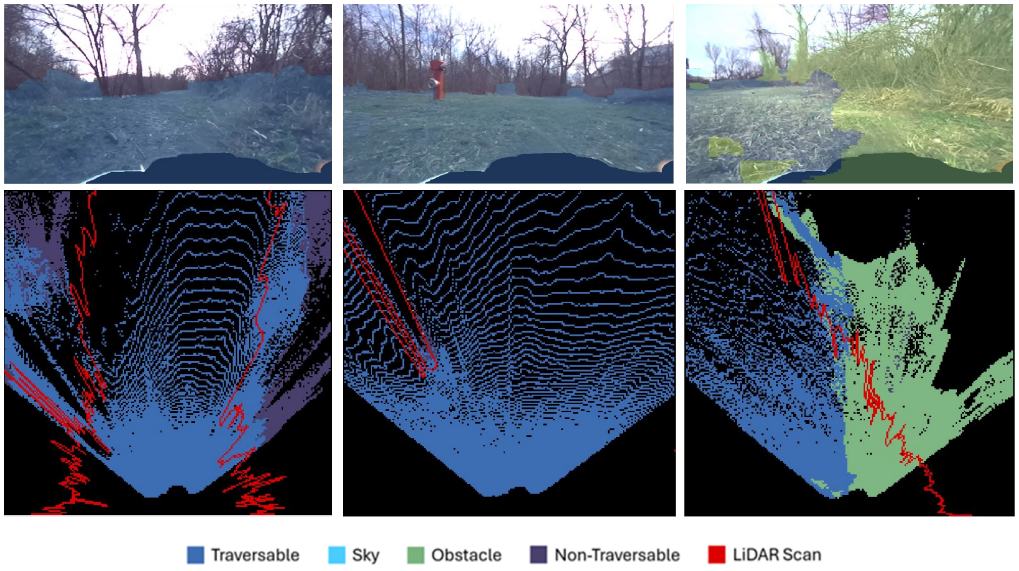


Figure 4.10: Examples of the resulting top-down maps from the 3D integration pipeline, together with respective segmentation results overlayed over their camera inputs.

Figure 4.10 also shows the discrepancies between LiDAR and segmentation data, where the inaccurate segmentation results significantly differ from the LiDAR readings in the top-down maps, e.g., the edges of the impassable/obstacle area not matching the LiDAR scan.

4.4 Navigational Performance in the field

After analyzing the prerequisites and other systems providing data for the task of autonomous offroad navigation, this section will now present the performance and test results from the navigational system. This system is the nexus of this thesis' project, combining all prior data streams (Figure 4.3) to find the most optimal path through the environment which its sensor systems perceive. The navigation system which this project uses bases itself on a trajectory navigation approach where prospective paths are tentacles (Section 2.4.1) which this thesis determines to be a suitable basis for offroad navigation in this project (Section 3.1.3).

This section will cover the evaluation of the navigational map, associated problems, and the performance of the chosen systems, before moving on to the evaluation of the implemented heading fusion algorithm and then giving examples and data of the actual in-the-field performance of the vehicle.

4.4.1 Perception Ramifications for the Navigational Map

The navigation process presented in this thesis relies on the top-down map which the previous stage of the software provides via the fusion of semantic segmentation and LiDAR data into a classification of the terrain in front the vehicle (Section 3.3, evaluated in Section 4.3). This map

contains data from the camera in the form of segmentation results which the perception transforms into a top-down map, where the colors of different areas signify the respective detected terrain type. Additionally, the algorithm overlays the LiDAR data on the map in the form of a red line defining the outlines of detected obstacles or terrain. When performing evaluation (Section 3.5.3), the algorithm utilizes this data to decide which of the prospective paths (Section 3.5.1) to follow. However, issues with the perception system (Sections 4.2.1 and 4.3.1) heavily influence this decision, leading to unintended navigational complications.

Firstly, this paragraph covers the LiDAR sensor's ground strikes and limited minimum sensing distance. Section 4.2.1 describes these effects in detail, and the leftmost example in Figure 4.11 shows an example of the ground-strike effect together with its influence on the navigational system. Even though the area ahead of the vehicle is clear of obstacles – and the segmentation recognizes this – a ground-strike occurs, causing the navigation algorithm to steer to the right, despite the goal laying straight ahead. Since a tilt of 2° – which is common when driving in offroad terrain – is sufficient to cause a ground strike they are a significant issue for navigation.

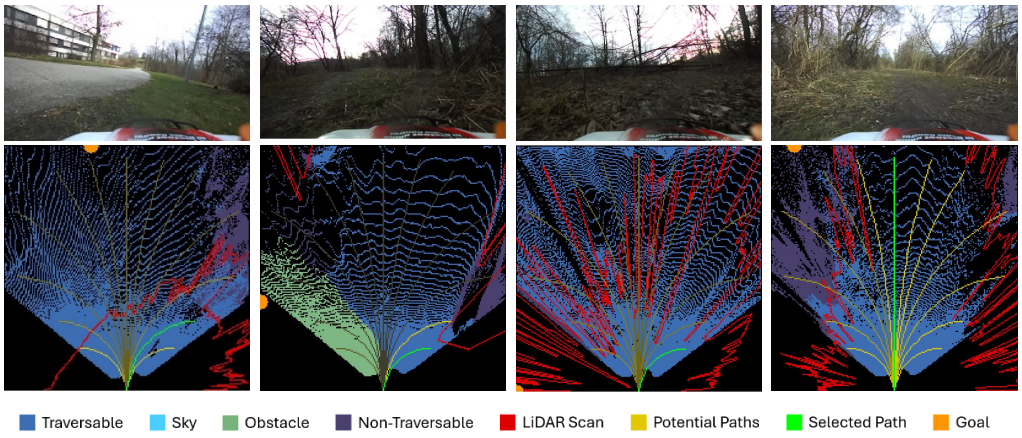


Figure 4.11: Examples of navigational results with their corresponding camera images. The saturation of the paths indicates their evaluation result, with better results indicated by brighter yellow. The orange dot on the edge of the map is the direction to the goal.

Because of the inaccuracies in the segmentation algorithm (Section 4.3.1), mischaracterizations as in Figure 4.11's second from the left example can cause navigational inaccuracies. Here, the segmentation marks a large, empty, traversable space of the map as an obstacle, which prevents the navigation from making the left turn which would achieve the optimal target trajectory.

Another problem occurs with grass or small sticks tall enough to reach into the LiDAR's scanning plane, but which the vehicle would easily traverse. The example second from the right in Figure 4.11 shows this issue, where an empty, traversable space is occluded by noisy LiDAR data due to grass and small sticks, again causing a deviation from the optimal navigation result.

The issues presented above however are mostly temporary, meaning that the algorithm corrects itself in the next computation cycle after initiating a non-critical maneuver. In most other non-edge cases, the algorithm works well, as Figure 4.11 shows in the right-most example.

Other errors are less recoverable. Since the LiDAR has an occlusion-mandated minimum sensing distance of 20cm (Section 4.2.1), it cannot detect any obstacle closer than this. If such an obstacle slips through the error-prone segmentation – thus escaping detection – it can cause a collision. A similar issue presents in the form of bodies of water, which the algorithm can only detect as reflections of the sky and classifying it as such. If this does not occur, the vehicle may

classify water as traversable terrain, which would lead to the loss of the vehicle. This prompts the operator to use extreme caution when operating the vehicle near bodies of water and can necessitate interventions in the autonomous operation.

4.4.2 Heading System

This project introduced a heading data fusion system designed to combat the difficulties with the ZED 2 stereo camera's integrated magnetometer (Section 3.5.2). Here, the algorithm fuses additional data from the GPS with the magnetometer readings to obtain more accurate heading information, which enables the navigational system to more accurately determine the direction of current travel, which is relevant to path selection (Section 4.1.2). Additionally, data from the vehicle's IMU provides a GPS influence factor, which negates said sensors influence on the heading data at standstill, which proves highly effective at eliminating heading data corruption at standstill since the GPS heading measurement there would jump around wildly (Figure 3.16).

Closer evaluation of the accuracy of this fusion algorithm would require a more accurate heading system as a reference, which unfortunately is unavailable in the scope of this work (Section 3.1.2). Close examination of the data recorded in the field leads to the suggestion that the system is accurate to within ca. 15° of true heading while moving and taking full advantage of its heading data fusion system; at standstill, this deviation is approximately 30°.

4.4.3 In-the-field Performance

Finally, this section presents the vehicle's performance in the field, traversing offroad terrain autonomously towards either a single or a series of GPS waypoints. For this evaluation – and all in-the-field tests conducted in this thesis – this thesis utilizes a roughly 1km round-trip track of public paths and overland driving near the TUM's Garching campus' physics building (Figure 4.12). The project chooses this track because it provides a representative mix of unmarked roads, forest vehicle and foot paths, as well as traversable offroad terrain.

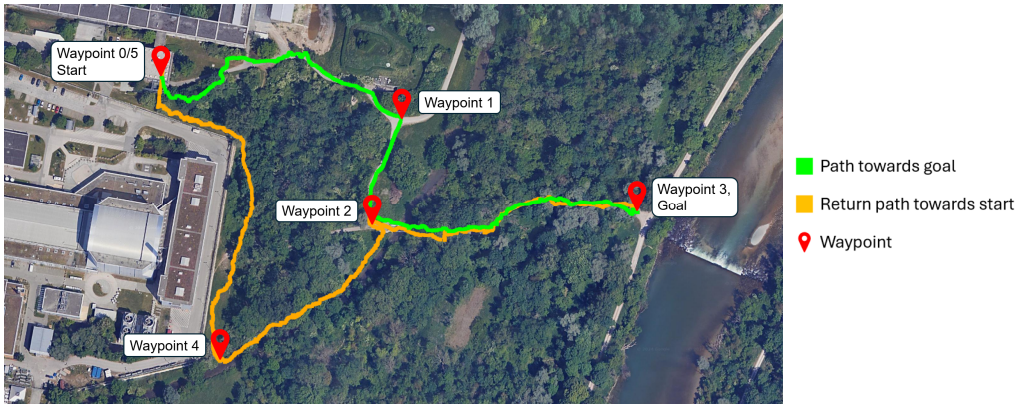


Figure 4.12: Offroad testing track used in this thesis with a round trip length of around 1km. The track here represents GPS data from one of the final tests.

During the tests, the vehicle traverses the terrain autonomously with as few operator interventions as possible, until it reaches the end point, where the vehicle stops, and either receives new waypoints after a reboot or the test ends. While the vehicle completed a significant portion of the

journeys autonomously while dodging obstacles and following established paths without collisions, the tests reveal several issues.

Two remaining issues appear with the software stack, which this thesis only manages to partly alleviate within the scope of this thesis. Firstly, the vehicle routinely steers too close to obstacles and non-traversable terrain, despite obviously detecting it, hence why it steers at all. Said steering problem is a configuration issue within the navigational system, which Section 3.6.4 partly fixes by slowing the vehicle for the tighter turns, thus producing a smaller turning radius which increases the distance to obstacles slightly. While this reduces the number of collisions due to obstacle proximity in turning, it does not entirely remedy the problem.

Secondly, the current iteration of the software limits itself to only 1.5 FPS to guarantee resource availability to the rest of the software running on the vehicle. This, however, leads to a minimum 666 ms pause between individual steering commands – sometimes more due to segmentation complexity – which is a non-issue if the issued steering angle is small, however, even while moving at 1 m/s and later 0.5 m/s (Section 3.6.4), the vehicle can collide with previously undetected terrain at maximum steering during this interval.

Finally, the lacking performance of the OFFSEG software stack (Section 4.3.1) and the 20cm minimum detection distance of the LiDAR (Section 4.2.1) can lead to collisions when obstacles are close to the vehicle and remain unsegmented. Section 4.4.1 additionally mentions the current system's problems with the detection of water. Together, these four issues account for a total of eleven interventions in the autonomous driving algorithms needed during the initial 1 km test where the operator temporarily suspends the system's control during recovery. As the completion of the course took 43 minutes, this amounts to one intervention every 3.9 minutes, and Table 4.6 shows this initial test data grouped by the different failure types.

Table 4.6: Navigation system errors during the field tests grouped by type.

Test	Steering obstacle proximity	Collision during 666 ms pause	Segmentation or LiDAR failure	Imminent water contact	Total Errors
Initial	2	3	5	1	11
Final	1	1	6	0	8

After additional modifications (Section 3.6.4), the vehicle produced a better test result in the final test (Table 4.6). Here, the system achieves an overall reduction of ca. 25% in total navigation system errors before recovery, notably in the classes of steering to close to obstacles and collisions due to the compute pause. Next, the recovery mechanism (Section 3.6.4) manages to recover five of the eight navigation system errors (Table 4.7), resulting in 62.5% of total errors recovered, and 71.4% of attempted recoveries succeeding. In one case, the system does not initiate recovery after a collision due to the LiDAR minimum sensing distance combined with a segmentation failure causing the rammed obstacle to escape detection, while the vehicle did not manage to resume operation after recovery initiation in two other cases. Thus, the recovery system and improved navigation reduce the final failure count from 11 to 3, or by 72.8%, with Section 5.1.3 presenting evaluation of these results.

Table 4.7: Recovery attempts, failures, and successes during the final test.

Total Errors	Recovery attempts	Recovery success	Recovery failure	Final failures
8	7	5	2	3

5 Discussion

This thesis originally set out with the goal of developing a fully functional, autonomous offroad vehicle, built on the F1TENTH platform and incorporating the knowledge from and capabilities of full-size offroad AV. In the preceding chapters, this work first showed and analyzed said platforms, knowledge and capabilities (Chapter 2), before the thesis described the developmental process (Chapter 3) and presented the results of these efforts (Chapter 4). What remains now is the evaluation and discussion of these final results, how they fit into the existing landscape of autonomous offroad vehicles as well as a final verdict on the goal this thesis originally set out to reach.

5.1 Evaluation of the in-the-field Performance

Before moving on to higher level comparisons and concluding analysis this section will first selectively evaluate the results of Chapter 4, analyzing the influencing factors between the individual systems, the overarching performance as well as the validity of the presented results. This provides a more nuanced knowledge basis needed for functionally assessing this thesis in front of the backdrop of other established systems and evaluating the goals given when initially starting project.

5.1.1 Compute Cost and FPS

First, this section will provide an overarching discussion of the `f1tenth_offroad` ROS package's computational performance during operation in the field (Figure 5.1), since this package is the most significant result of this thesis.

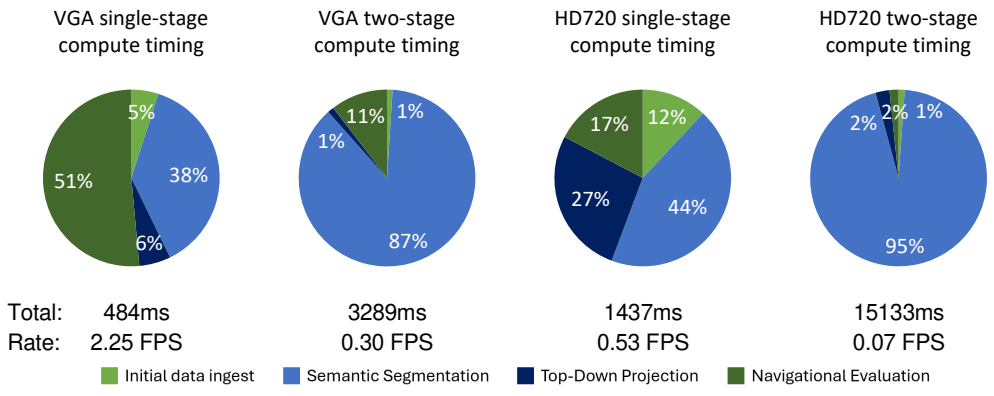


Figure 5.1: Relative time taken by the respective steps in the `f1tenth_offroad` ROS package developed for this project, with total computation cycle time and resulting FPS.

The presented diagrams reference data which Section 4.3 presents. Here, the left-most pie chart represents the setup used during operation in the field, where it is evident that semantic segmentation and the navigational evaluation represent the largest chunks of computational cost, specifically with the latter consuming more than half of the time each cycle. The initial data ingest and the highly optimized top-down projection (Sections 3.6.3 and 4.3.2) together account for ca. 10% of the computation time. As such, any further optimization, which Section 6.2 expands upon, should first focus on the navigational evaluation, since the semantic segmentation is governed by OFFSEG (Section 2.3.1), which already is optimized.

Moving towards the non-used two-stage algorithm, the picture changes dramatically. In Figure 5.1's second from the left diagram, it is immediately obvious that the semantic segmentation governs the computational time requirements. If the vehicle should use this system in the future, it requires a reduction of computational cost (Section 6.2).

Lastly, the right two diagrams in Figure 5.1 reference data from tests which utilize the higher HD720 resolution, which may find future use to further improve segmentation performance (Section 6.2). In both cases, segmentation dominates the computational cost, although single-stage segmentation much more evenly distributes the computational cost between the individual steps. This leads to the conclusion that segmentation optimization might be insufficient here, and performance instead hinges on the NVIDIA Jetson Nano.

5.1.2 Assessment and Validity of the presented Results

Finally, a validity assessment for the results presented in this thesis is in order. This project was set up to bridge the gap between small-scale, inexpensive AV such as the F1TENTH platform and large and expensive full-scale offroad AV such as Stanley (Section 1.1). It is a project governed by the spirit of rapid prototyping in both software and hardware, neglecting the steps of precise optimization and in-depth analysis for all but a few elements.

As such, the results which this work presents are specific to this project, governed by the framework of working in a solution space previously unoccupied. There are no examples available for fine comparison of this system to another solution on the same scale, and comparisons with other platforms, larger AOV or non-offroad small-scale vehicles – governed by different framework conditions – can only ever by conditional. Most of the systems used on the vehicle are either adapted – featuring custom modifications for this specific platform – or are developed from scratch (Section 4.1), which lays the groundwork for the further analysis. This thesis assessed many distinct aspects of a rapidly developed system which means that – considering the scope of testing possible in a bachelor's thesis – future research on this topic should consider the results of this work with caution concerning measurement accuracy or sample size.

Results detailing the LiDAR sensor's FOV and ground-strike issues (Section 4.2.1) are heavily dependent on the sensors specific integration on this project. Another choice of LiDAR sensor or placement on the vehicle would skew or invalidate this work's results, which shows their conditional validity. Similarly, the camera system's results (Section 4.2.2) also hinge on camera placement, as well as on the chosen ROS infrastructure and hardware, which are conditional to the presented data. Finally, the performance metrics introduced in this thesis lack broader testing, which could provide information about root causes. As but one system of a larger project however, said testing is outside of the scope of this work.

The GPS system presents a major problem for this project, with many unique challenges again specific to this very platform (Section 4.2.3), similar to the magnetometer problems and resulting

fusion (Sections 3.5.2 and 4.4.2). With a different combination of sensors or other hardware, the problems faced here may be nonexistent in other projects, which applies as well to the results of the power system tests (Section 4.2.4).

Finally, the validity of the performance test results for the integrated software developed here – the `f1tenth_offroad` ROS package – depends strongly on the validity of the results which the preceding paragraphs present. Also, the creators of the OFFSEG segmentation suite neither specifically developed nor did this work retrain their system for the application on a small-scale vehicle, relativizing the results of Section 4.3.1.

It is then obvious that any analysis of the final navigation system not only depends on the quality of the rapidly developed navigation system itself, which ran through few tests – again, the scope of a bachelor's thesis governs the amount of testing possible – but also on the validity of the results of the prior systems feeding into it. Since the prior paragraphs detail the issues with said validity, the navigation system's results and performance rely on the myriad, specific, conditional, and unique choices made in and for this project. Additionally, it is worth noting that the data which Section 4.4.3' Table 4.6 provides – and on which much of the further analysis of this thesis bases itself – remains an approximation, since the exact cause of a collision or vehicle failure is sometimes difficult to evaluate.

5.1.3 Overarching System Performance as the sum of its Parts

This thesis' vehicle platform is a complex system, which utilizes many different algorithms, sensors, and methods to achieve the task of autonomous offroad navigation. The results of individual system studies and tests are available in Sections 4.2, 4.3 and 4.4.2, and Sections 5.1.1 and 5.1.2 assesses the system's computational performance and the ramifications and validity of the prior results respectively. However, as previous section discusses, these results are difficult to contextualize and evaluate outside of the framework of this project. As such, the true results and contributions of this work lie in the completed system as the combination of its subsystems and components, which this section will evaluate.

The capstone and ultimate metric of this hard- and software system is the navigational performance, which is the part of the vehicle's system where all the other data streams and components culminate, analogous to Figure 5.2.

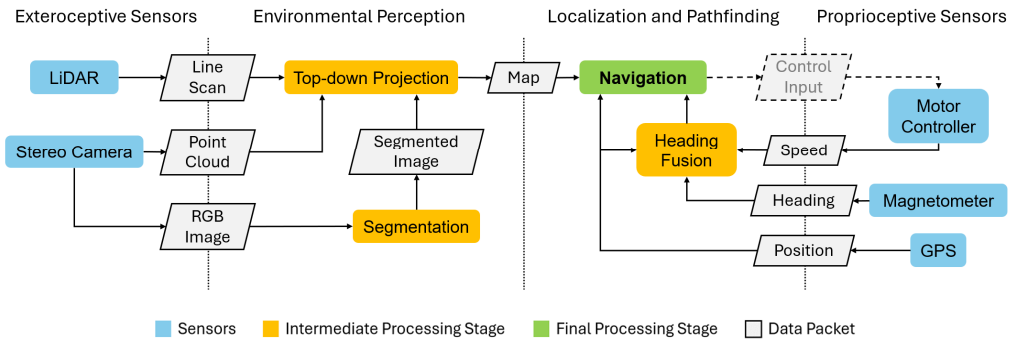


Figure 5.2: Hierarchical overview of the information flow in the `f1tenth_offroad` package.

Starting on the environmental perception side, Section 4.3 presents said system's performance in depth, with Section 4.4.1 additionally describing its effects on the navigational system. While both the LiDAR and the semantic segmentation occasionally produce incorrect results, the

navigational system utilizing their data compensates for this using a worst-case approach. Here, the closest obstacle always takes precedence, providing a measure in which both sensors can compensate and correct for non-detections of the other, preventing collisions. This however means that any false positive obstacle detection effectively causes the vehicle to take a non-optimal path, though when weighed against the possibility of a collision or getting the vehicle stuck, the algorithm can much more easily compensate for an unnecessary course change. While both systems complement each other well in this case, Section 4.4.1 presents issues causing failures which indeed result in collisions or the vehicle getting stuck. These perception failures are the greatest remaining problem for the perception system (Section 6.2) causing more than 50% of the errors in the autonomous driving protocol while testing (Section 4.4.3). The lower vantage point of the chassis when compared to the full-size solutions for its creators developed the OFFSEG semantic segmentation algorithm – which this project uses – is likely a cause for the relatively poor performance of the segmentation, which further work may improve (Section 6.2)

This thesis improved the positioning system – analyzed in Section 4.2.3 –to a point where it could reliably route the vehicle within the required 5 m radius of the goal point. This however had severe ramifications for the ZED 2 stereo camera (Section 3.4.3), which impacts the further usage of its remaining performance potential. While the improved resolution has not shown any advantage for the single stage segmentation (Section 4.3.1), it may impact the future usage of the second stage segmentation in the future (Section 6.2).

After the integration of the GPS system, magnetometer and odometry based heading fusion system, evaluated in Section 4.4.2, the vehicle could reliably drive into the correct goal direction and reach its target. While the system remains far from perfect – with an assumed 15° deviation from the true heading direction – it is within non-critical ranges for this project. Since this project does not expressly intend on finding the absolute fastest path to goal, a path with slight curves and occasional deviations from the straight line is acceptable. This is because the navigational system will automatically self-correct once the distance to the goal shrinks, making angular deviations larger, thus naturally increasing the steering input, allowing the vehicle to reach its goal.

Finally, the navigational system. One of the largest remaining issues addressed in Section 4.4.3 is the large pause interval of 660 ms between control inputs, which internal 1.5 FPS throttling causes. While the algorithm may theoretically run faster (Section 4.3.1) this is has adverse effects on the rest of the system due to resource unavailability which causes occasional crashes, however exact tests towards this are outside the scope of this thesis.

Overall, when considering the difficulties it must deal with, the navigational algorithm performs surprisingly well. Even with imperfect and error-prone data streams from all sources, the algorithm manages to effectively compensate for most of the issues and still steer the vehicle to its goal reliably. The recovery system (Section 3.6.4) especially increases the system's overall capability, allowing it to compensate for crashes and collisions caused by occasional lapses in the other systems. While this project neglects more complex navigational systems (Section 2.4.2) according to reasoning in Section 3.1.3, the chosen system of trajectory-based tentacle navigation proved reliable and effective at momentary, short range navigation.

Lastly, it is important to contextualize the software package – which has been the focus of this section thus far – in front of the background of the vehicle's mechanical basis. This mechanical offroad performance of the vehicle governs the effectiveness of any and all control inputs from the software. The chosen basis here, the Traxxas Slash 4x4 with additional suspension upgrades (Section 4.1.1) proved an effective platform, with no interventions necessary because of

the inadequacy of the chassis. The arrangement of sensors – while evaluated as suboptimal (Sections 4.2.1 and 4.2.2) – was largely nonconsequential for the vehicle as a whole, with the other factors governing performance (Section 4.4.3).

5.2 Comparison with existing Systems

After the previous section discussed the specific results of this thesis' project, the individual components, and their interactions as well as their validity, it is now important to compare the developed system with other examples and platform available in the space of autonomous AV. This is especially important when considering that the primary goal of this project is to create an in-between solution bridging the gap between full-size offroad AV and the smaller, cheaper RV scale AV. Thus, this section will present this comparison between both full-size and the smaller scale vehicles to then assess in the last section of this chapter whether or not this thesis reached its original goal. This comparison will include both hard- and software, of which overviews are available in Sections 4.1.1 and 4.1.2 respectively. Section 1.1 and specifically Figure 1.1 outline the aspects this system tries to gain from the respective platforms.

5.2.1 Full-size Systems

Chapter 2 presents many previous, current, on- and offroad full-size AV systems as examples for the categories of mechatronics and chassis, sensorics, perception as well as pathfinding and navigation. Since this thesis aims to miniaturize the approach to a full-size offroad autonomous system, the current section will analyze the developed solution in the light of these four categories and compare it with the presented full-size solutions such as EDGAR (Figure 5.3).

Figure 5.3: The F1TENTH-platfrom-based vehicle developed in this thesis in front of the TUM's EDGAR vehicle which Appendix C presents.

Firstly, this section will assess the most obvious difference between full-size solutions and this thesis' RC-car-based variant, which is immediately obvious from Figure 5.3: the chassis. While it is naturally not comparable in scale – since that is part of the problem statement of this project (Section 1) – it still features all the important elements which Section 3.1.1 extracts from Section 2.1.1, such as lockable differentials, all-wheel drive and a robust and capable suspension. Additionally, when compared to a full-scale solution, the developed platform is much more robust in general, surviving falls, crashes and collisions that would impart severe damage to a full-scale

vehicle. The used Traxxas Slash 4x4 chassis and motor system are capable for their size and scale, traversing vegetation and deep ruts which may cause problems for larger platforms. Thus, this thesis concludes that while the developed mechanical vehicle obviously cannot stack up to a dedicated offroad ATV used as a basis for example in the AVIDOR-2004 vehicle (Section 2.1.1) it is a nearly ideally scaling image of its larger counterparts. This should provide good scalability for future research and allows the rest of the system to adopt methods and techniques from larger vehicles without intensive adaptation.

Next, this section will cover the sensorics used on the vehicle. Sections 2.2 and 3.1.2 respectively present and analyze the sensor system currently used on larger research vehicles. Now, while the system incorporates many of the sensor devices used on larger vehicles in the category of exteroceptive sensors (Section 2.2.1), such as a stereo camera or 2D LiDAR, this project has to adapt to its scale. Because of the limited available compute resources, similarly limited availability of funds for more advanced systems and likewise limited space on the vehicle, the developed system uses far less sensory equipment as compared to larger vehicles. Multiple 2D LiDAR scanners on Stanley or RASCAL, multiple camera systems on RAVON or the 3-layered camera system with surround capabilities which EDGAR uses (Appendix C) all dwarf the singular stereo camera and 2D LiDAR on this project. While this obviously presents a large discrepancy in capability, it also shows that results are obtainable with far less resources. This simultaneously allows the vehicle to still profit from advanced sensor processing and perception techniques presented in the following paragraphs, while cutting down on complexity and overhead of fusion systems, which makes the software lighter and more efficient.

Regarding Section 2.2.2's proprioceptive sensors, this project uses a similar set of sensors as full-size AOV: IMU/odometry, GPS, magnetometer, and encoders. However, obvious from Sections 4.2.3 and 4.4.2, the accuracy of our system cannot match capabilities which full-scale vehicles require (Section 2.2.2). While this diminishes capabilities for research on and implementation of more localization-dependent systems (Section 2.4.2), the success of this system also shows that such inaccuracies may not immediately crush any attempt at it. This opens the door for more research into the specific causes and consequences for this system's issues and may provide interesting material for future researchers who deal with similar issues.

When assessing perception in the field of environmental perception (Section 2.3.1), this thesis' system proves to be an excellent testbed for current technologies. With capable hardware and software frameworks, which are virtually analogous to current systems used by real, full-size vehicles (Sections 2.3.3 and 3.1.4), this projects vehicle implements modern techniques such as semantic segmentation, 3D top-down mapping and Camera-LiDAR-fusion. While the system again falls short of the performance possible in larger systems with more capable hard- and software, when it comes to perception, the developed system falls much closer to actual vehicles, allowing a direct, full adaptation of research areas currently in development. The localization side of perception (Section 2.3.2) however shows a different image, which the lacking proprioceptive sensor performance discussed in the prior paragraph causes. Advanced localization techniques and potential use-cases in navigation (Section 2.4.2) are thus out of reach on this platform.

Finally, the goal of every autonomous vehicle system: navigation (Sections 2.4 and 3.1.3). This project's navigation algorithm bases itself on a trajectory navigation system using tentacles (Section 2.4.1), which the vehicle directly adapts from full-size vehicles such as the MuCAR-3 AOV, once again showing the scalability of this system implementation. The previous paragraphs already explain why a more advanced navigational system (Section 2.4.2) which looks over the horizon of local pathfinding is not implementable here: poor localization and lacking

computational capability. However, since some AOV examples still use local pathfinding, multi-level planning is not a necessity, which the developed system's in-the-field performance proves.

5.2.2 F1TENTH Platform

Firstly, while it integrates software and concepts from the full-scale systems the previous section compares it with, this project builds on the F1TENTH platform, from which it also intends to inherit certain features. Relevant to this comparison here are the ease of use and greater simplicity, inexpensive hardware, better availability at research institutions.

Secondly, this thesis stipulates that the usage of the F1TENTH platform coincides with a greater ease of use when compared to a full-size solution. Development and testing are fast, easy, and work well with rapid prototyping techniques. Larger vehicles would require dedicated test tracks, complicated control systems to cover a wider range of scenarios and teams of researchers to successfully develop and operate them. In contrast to this, the presented work proves that a single individual can achieve autonomous navigation on the F1TENTH platform in the scope of a bachelor's thesis, with significantly reduced complexity in the systems of the vehicle.

This transitions directly into the next aspect, inexpensive hardware. Where larger vehicles require entire sensor arrays of expensive hardware to cover a sufficient area far enough into the distance to be viable for AV development, this thesis only requires single 2D LiDAR, stereo camera, and GPS module, again, simplifying the entire approach. Additionally, compared to the procurement of an entire full-size car or other vehicle, an RC car is a small investment, even when considering the cost of compute hardware, motor controller and power delivery systems.

Lastly, Section 2.1.4 shows that F1TENTH platforms are already available at automotive and mobility departments in research institutions around the world. Combined with the fact that the only actual modifications to the existing platform requiring extra hardware were the GPS sensor and custom spring set (which together come in at under 100 $), it is obvious that research with a similar project is easily attainable for any researcher with access to an F1TENTH platform.

5.2.3 Final Verdict on this thesis' goals

This chapter will now conclude with a final verdict on this thesis' goals as set out in Section 1.1, specifically Figure 1.1. Analogous to this, Figure 5.4 shows the achieved elements the goal.

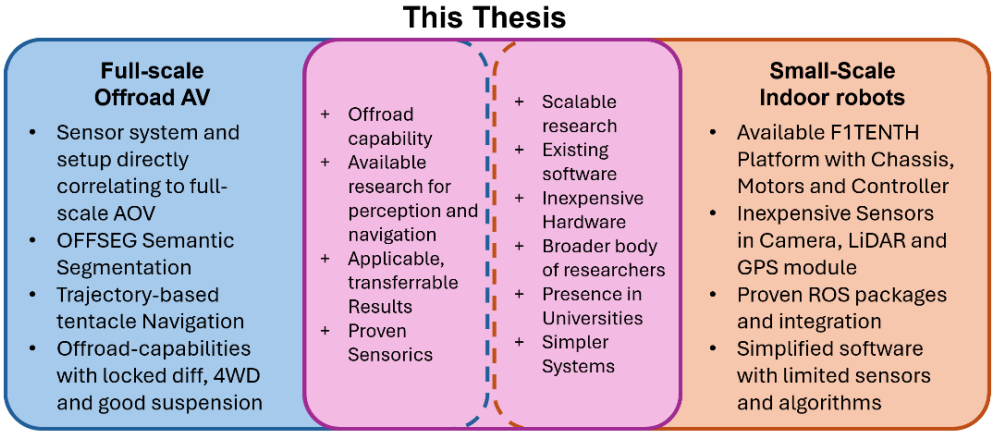


Figure 5.4: Final assessment of this thesis' results, analogous to Figure 1.1

6 Conclusion

Over the course of this thesis – written at the Technical University of Munich's Autonomous Vehicle Systems department – this project transformed a common F1TENTH-platform set up for indoor experiments into a capable, autonomous offroad vehicle. The developmental process integrated many lessons-learnt, findings, algorithms, and technologies from successful full-size vehicles on a miniature scale, which never existed before in the field of AOV.

Because it is neither a new design nor a direct copy, the finished vehicle depicted in Figure 6.1 is a synthesis of knowledge and research spanning many quadrants of current and past autonomous vehicle technology. As such, the work done here is not in its elements revolutionary, but innovative in its combination of these known components into a new form.

Figure 6.1: Final version of the modified F1TENTH platform developed in this thesis.

The vehicle developed within this work started as a standard F1TENTH platform with an additional ZED 2 stereo camera. After developing custom mounts and brackets for initial tests, software development began. Here, the development of the `f1tenth_offroad` package begun with the integration of the OFFSEG semantic segmentation suite, before moving on to the fusion of this segmentation, the LiDAR scan data, and the camera 3D data into a top-down map. After this, the vehicle added a GPS positioning system and weathered the complications associated with it. Navigation was the next goal, which the vehicle achieves via the application of a trajectory-based tentacle evaluation system. Lastly, the vehicle underwent modifications to the chassis for offroad driving, as well as to the software, for performance improvement and power reduction.

After the initial analysis of the state-of-the-art, the presentation of the method and results as well as the discussion thereof, the current chapter will conclude this thesis with a final assessment of the contributions made, possible objectives for further research and closing remarks.

6.1 Contributions of this Work…

This project's main contribution is the developed vehicle, which spans the gap between full-size offroad autonomous vehicles and small-scale research and teaching AV. Additionally, as Section 5.1 extensively explains, the primary value of this work is not in the novelty of the utilized systems and components, but in the manner of their combination. This thesis' outcome is a truly unique, first-of-its-kind small-scale F1TENTH platform vehicle incorporating the capabilities of full-size AOV, which of course uses many existing elements from both of its 'parents', combining them in a novel way. As such, the contributions of this work depend on the point of view, or which of his 'parents' assesses its differences, in a manner of speaking. The current section will present this thesis' contributions in an equivalent manner.

6.1.1 … to the Research on the F1TENTH-platform

When compared with existing F1TENTH-platform research, this work presents the following contributions:

- Development of an offroad-suitable hardware platform, with detailed power and performance measurements for individual components and systems
- GPS sensor usage on a F1TENTH-platform vehicle, with results and analysis of interfering factors and performance for future use
- Integration of high-performance semantic segmentation specific for offroad navigation on the F1TENTH's onboard hardware
- Magnetometer, crude GPS-based heading and odometry fusion algorithm
- Trajectory-based tentacle navigation system using segmentation-LiDAR-stereo-fusion and magnetometer-GPS-odometry-fusion data
- Creation of the open-source `f1tenth_offroad` ROS package [84], which combines all aspects of perception and navigation, making them available for any and all researchers interested in autonomous offroad driving

6.1.2 … to the Field of Autonomous Offroad Driving

When considering the field of full-size autonomous offroad driving, the developed platform contributes the following unique aspects:

- Potential availability of autonomous offroad driving research platforms to any institution already supporting F1TENTH-based projects
- Simplification of the AOV system to allow for focus on specific element of research elements without significant overhead
- Inexpensive hardware setup that is widely available and easy to set up, lowering the initial hurdle for AOV research
- Scalable system that allows both the straight-forward application of full-size technology or sensors and provides good applicability of the results gathered
- Risk reduction through the usage of a much smaller, cheaper vehicle

6.2 Possible Objectives for further Development

Though this thesis answered the questions asked at the beginning and the project currently remains in a completed stage, many open questions and tasks remain. Considering the amount of potential further developments and research areas, this section will list them in order of their perceived importance without in-depth explanations as to not exceed the scope of this chapter.

1. The crashes apparent when the vehicle exceeds the 1.5 FPS limit require investigation, since higher computation rates could improve performance significantly.
2. To improve overall system performance, further modifications should improve either the navigational computations or the segmentation performance for higher FPS rates (Section 5.1.1) or to remedy the current 1.5 FPS limit.
3. If none of this is possible, the navigation algorithm should be decoupled from the perception pipeline such that control inputs can happen at a higher frequency to alleviate problems currently present (Section 5.1.3).
4. Retraining of the segmentation model with a custom dataset of camera images recorded on the vehicle could improve the accuracy of the semantic segmentation.
5. If this is insufficient, it may be viable to increase the camera resolution from VGA to HD720 to increase segmentation retraining results.
6. Either a time-average or an odometry-based filtering approach could mediate LiDAR ground strikes, which currently influence the navigational performance.
7. In the same vein, a 3D LiDAR could bring additional perceptive qualities (Appendix D) while at the same time improving 3D projection accuracy.
8. Retraining the two-stage segmentation algorithm might provide better results, which could then improve navigation quality via the knowledge of specific terrain types and puddles.
9. With a more robust localization system (via DGPS, odometry fusion or other means), the robot could implement SLAM or other multilevel navigation approaches (Section 2.4.2), which would increase its navigational capabilities.

6.3 Closing remarks

The author of this thesis greatly appreciates the opportunity to work on such an inter-disciplinary and interesting project. While the system's components themselves are hardly an innovative leap forward on their own, their unique and novel combination on this vehicle proves that the field of offroad AV does not only permit expensive and singular prestige projects.

While the primary goal of this work was to show that the creation of such a small-scale AOV vehicle is possible, and the intended consequence of this is primarily the enabling of other research based on this platform, other exiting possibilities remain. The vehicle in and of itself may prove useful beyond the application of a research testbed, for autonomous mapping, search and rescue or fire prevention, many applications are conceivable for a small, autonomous offroad system.

List of Figures

List of Tables

Bibliography

[1] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, "Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review," *Sensors*, early access. doi: 10.3390/s20154220.

[2] J. Kocic, N. Jovicic, and V. Drndarevic, "Sensors and Sensor Fusion in Autonomous Vehicles," in *2018 26th Telecommunications Forum (TELFOR): Proceedings of papers : Belgrade, Serbia, November, 20-21, 2018 = XXVI Telekomunikacioni forum TELFOR 2018 : zbornik radova*, Belgrade, 2018, pp. 420–425, doi: 10.1109/TELFOR.2018.8612054.

[3] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review," *Sensors*, early access. doi: 10.3390/s21062140.

[4] H. A. Ignatious, H.-E. Sayed, and M. Khan, "An overview of sensors in Autonomous Vehicles," *Procedia Computer Science*, vol. 198, pp. 736–741, 2022. doi: 10.1016/j.procs.2021.12.315. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921025540

[5] K. Berns, K.-D. Kuhnert, and C. Armbrust, "Off-road Robotics—An Overview," *Künstl Intell*, vol. 25, no. 2, pp. 109–116, 2011, doi: 10.1007/s13218-011-0100-4.

[6] Amirreza Shaban, Xiangyun Meng, JoonHo Lee, Byron Boots, and Dieter Fox, "Semantic Terrain Classification for Off-Road Autonomous Driving," in *Proceedings of the 5th Conference on Robot Learning*, vol. 164, Aleksandra Faust, David Hsu, and Gerhard Neumann, Eds., Proceedings of Machine Learning Research: PMLR, 2022, 619--629. [Online]. Available: https://proceedings.mlr.press/v164/shaban22a.html

[7] R. Behringer, "The DARPA grand challenge - autonomous ground vehicles in the desert," *IFAC Proceedings Volumes*, vol. 37, no. 8, pp. 904–909, 2004. doi: 10.1016/S1474-6670(17)32095-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017320955

[8] D. Khalatyan, "Towards Modularity and Reconfigurability for Robotic Off-Highway Equipment: A Proof-of-Concept Prototype: Towards Modularity and Reconfigurability for Robotic Off-Highway Equipment: A Proof-of-Concept Prototype," [Online]. Available: https://qspace.library.queensu.ca/items/4787bda8-ccb5-4445-8878-28c7283a7e95

[9] Robert Hudjakov, "Long-Range Navigation for Unmanned Off-Road Ground Vehicle," Faculty of Mechanical Engineering, TALLINN UNIVERSITY OF TECHNOLOGY, 2013. [Online]. Available: https://www.researchgate.net/publication/281432404_Long-Range_Navigation_for_Unmanned_Off-Road_Ground_Vehicle

[10] T. Bailey, "Mobile Robot Localisation and Mapping in Extensive Outdoor Environments," 2002. [Online]. Available: https://www.semanticscholar.org/paper/Mobile-Robot-Localisation-and-Mapping-in-Extensive-Bailey/b9fc38a6ce41c8709377f6a5974a66b59de9487c

[11] A. Agnihotri, M. O'Kelly, R. Mangharam, and H. Abbas, "Teaching Autonomous Systems at 1/10th-scale," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, New York, NY, USA, 2020, doi: 10.1145/3328778.3366796.

[12] B. Li *et al.,* "Toward Fair and Thrilling Autonomous Racing: Governance Rules and Performance Metrics for the Autonomous One," *IEEE Trans. Intell. Veh.*, vol. 8, no. 8, pp. 3974–3982, 2023, doi: 10.1109/TIV.2023.3298914.

[13] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning," *Proceedings of Machine Learning Research*, vol. 123, 2020. [Online]. Available: https://par.nsf.gov/biblio/10221872

[14] S. Thrun *et al.,* "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006, doi: 10.1002/rob.20147.

[15] R. Behringer *et al.,* "RASCAL - an autonomous ground vehicle for desert driving in the DARPA grand challenge 2005," in *2005 IEEE Intelligent Transportation Systems Conference (ITSC): Vienna, Austria, 13 - 16 September 2005 ; [8th International Conference on Intelligent Transportation Systems*, Vienna, Austria, 2005, pp. 644–649, doi: 10.1109/ITSC.2005.1520123.

[16] R. Behringer *et al.,* "The DARPA grand challenge - development of an autonomous vehicle," in *2004 IEEE Intelligent Vehicles Symposium: Parma, Italy, June 14 - 17, 2004*, Parma, Italy, 2004, pp. 226–231, doi: 10.1109/IVS.2004.1336386.

[17] G. Seetharaman, A. Lakhotia, and E. P. Blasch, "Unmanned vehicles come of age: The DARPA grand challenge," *Computer*, vol. 39, no. 12, pp. 26–29, 2006, doi: 10.1109/MC.2006.447.

[18] J. van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 384–406, 2018. doi: 10.1016/j.trc.2018.02.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/s0968090x18302134

[19] *Off-road vehicle engineering principles*. St. Joseph, Mich.: American Society of Agricultural Engineers, 2006.

[20] Bill McBride, Raul Longoria, and Eric Krotkov, *Measurement and prediction of the off-road mobility of small, robotic ground vehicles*, 2003. [Online]. Available: https://www.researchgate.net/profile/raul-longoria-2/publication/246790618_measurement_and_prediction_of_the_off-road_mobility_of_small_robotic_ground_vehicles

[21] J. Folkesson and H. Christensen, "SIFT Based Graphical SLAM on a Packbot," *Field and Service Robotics*, vol. 42, pp. 317–328, 2008. doi: 10.1007/978-3-540-75404-6_30. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-75404-6_30

[22] Chris Prahacs, Aaron Saunders, Matthew K Smith, Dave Mcmordie, and Martin Buehler, "Towards legged amphibious mobile robotics," in 2004. [Online]. Available: https://www.researchgate.net/publication/228869003_Towards_legged_amphibious_mobile_robotics

[23] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A Simple and Highly Mobile Hexapod Robot," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001, doi: 10.1177/02783640122067570.

[24] B. Siciliano, C. Laschi, and O. Khatib, Eds. *Experimental Robotics* (Springer Proceedings in Advanced Robotics). Cham: Springer International Publishing, 2021.

[25] N. D. Wallace, H. Kong, A. J. Hill, and S. Sukkarieh, "Motion Cost Characterisation of an Omnidirectional WMR on Uneven Terrains," *IFAC-PapersOnLine*, vol. 52, no. 22, pp. 31–36, 2019. doi: 10.1016/j.ifacol.2019.11.043. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319309784

[26] R. Thakker *et al.,* "Autonomous Off-Road Navigation over Extreme Terrains with Perceptually-Challenging Conditions," in *Experimental Robotics* (Springer Proceedings in Advanced Robotics), B. Siciliano, C. Laschi, and O. Khatib, Eds., Cham: Springer International Publishing, 2021, pp. 161–173.

[27] "ODIN - OE Data Integration Network." Accessed: Jan. 26, 2024. [Online]. Available: https://odin.tradoc.army.mil/WEG/Asset/Telemax_PRO_German_Tracked_Unmanned_Ground_Vehicle_(UGV)

[28] Traxxas.com. "Traxxas X-Maxx | RC Monster Truck." Accessed: Jan. 26, 2024. [Online]. Available: https://traxxas.com/products/landing/x-maxx/

[29] NASA/JPL-Caltech/MSSS. "6037 MSL Banner." Accessed: Jan. 26, 2024. [Online]. Available: https://mars.nasa.gov/msl/home/

[30] A. Rankin, M. Maimone, J. Biesiadecki, N. Patel, D. Levine, and O. Toupet, "Driving Curiosity: Mars Rover Mobility Trends During the First Seven Years," in *2020 IEEE Aerospace Conference: Yellowstone Conference Center, Big Sky, Montana, March 7-14, 2020*, Big Sky, MT, USA, 2020, pp. 1–19, doi: 10.1109/AERO47225.2020.9172469.

[31] "Slash 4X4 VXL TSM | 4X4 Brushless RC Truck | Traxxas." Accessed: Feb. 10, 2024. [Online]. Available: https://traxxas.com/products/models/electric/slash-4x4-tsm?t=features

[32] "Build." Accessed: Feb. 10, 2024. [Online]. Available: https://f1tenth.org/build

[33] T. Hattori, X. Huo, S. Maldonado, P. Napier, W. Swist, and S. Anderson, "Autonomous Miniature Car for Room Exploration and Object Search," *1546-2188*, 2023. [Online]. Available: https://repository.arizona.edu/handle/10150/670501

[34] "F1TENTH." Accessed: Feb. 10, 2024. [Online]. Available: https://f1tenth.org/

[35] S. Campbell *et al.,* "Sensor Technology in Autonomous Vehicles : A review," in *29th Irish Signals and Systems Conference (ISSC): Belfast, UK, June 21-22, 2018*, Belfast, 2018, pp. 1–4, doi: 10.1109/ISSC.2018.8585340.

[36] "What is LiDAR and How Does it Work? | Synopsys." Accessed: Feb. 11, 2024. [Online]. Available: https://www.synopsys.com/glossary/what-is-lidar.html

[37] N. Li *et al.,* "A Progress Review on Solid-State LiDAR and Nanophotonics-Based LiDAR Sensors," *Laser & Photonics Reviews*, vol. 16, no. 11, 2022, Art. no. 2100511. doi: 10.1002/lpor.202100511. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/lpor.202100511

[38] Smithsonian Institution. ""Stanley" Robot Car | Smithsonian Institution." Accessed: Feb. 11, 2024. [Online]. Available: https://www.si.edu/object/nmah_1377824

[39] "Elrob 2009 TeamInformation Kaiserslautern." Accessed: Feb. 11, 2024. [Online]. Available: https://www.elrob.org/files/elrob2009/

[40] C. Armbrust *et al.,* "RAVON: The robust autonomous vehicle for off-road navigation," in *Using Robots in Hazardous Environments*, Elsevier, 2011, pp. 353–396.

[41] Harald Weber, "LiDAR sensor functionality and variants: SICK AG WHITEPAPER," SICK AG, Waldkirch, Germany, 2018. Accessed: Dec. 2, 2024. [Online]. Available: https://cdn.sick.com/media/docs/3/63/963/whitepaper_lidar_en_im0079963.pdf

[42] HOKUYO AUTOMATIC CO., LTD. "UST-10/20LX | Products List | Scanning Rangefinder | Distance Data Output | UST-10/20LX | HOKUYO AUTOMATIC CO., LTD." Accessed: Feb. 12, 2024. [Online]. Available: https://www.hokuyo-aut.jp/search/single.php?serial=167

[43] M. Himmelsbach *et al.,* "Team MuCAR-3 at C-ELROB 2009," in *Proceedings of 1st Workshop on Field Robotics, Civilian European Land Robot Trial 2009*, 2009.

[44] M. Yahiaoui *et al.,* "FisheyeMODNet: Moving Object detection on Surround-view Cameras for Autonomous Driving," Aug. 2019. [Online]. Available: http://arxiv.org/pdf/1908.11789.pdf

[45] G. Reina, A. Milella, and R. Worst, "LIDAR and stereo combination for traversability assessment of off-road robotic vehicles," *Robotica*, vol. 34, no. 12, pp. 2823–2841, 2016, doi: 10.1017/S0263574715000442.

[46] Mars.nasa.gov. "Blog: Mars Perseverance Rover Mission - NASA." Accessed: Feb. 20, 2024. [Online]. Available: https://mars.nasa.gov/mars2020/mission/status/

[47] J. N. Maki *et al.,* "The Mars 2020 Engineering Cameras and Microphone on the Perseverance Rover: A Next-Generation Imaging System for Mars Exploration," *Space Sci Rev*, early access. doi: 10.1007/s11214-020-00765-9.

[48] K. S. Chong and L. Kleeman, "Accurate odometry and error modelling for a mobile robot," in *Proceedings / 1997 IEEE International Conference on Robotics and Automation: April 20 - 25, 1997, Albuquerque, NM*, Albuquerque, NM, USA, 19XX-, pp. 2783–2788, doi: 10.1109/ROBOT.1997.606708.

[49] M. BROSSARD and S. BONNABEL, "Learning Wheel Odometry and IMU Errors for Localization," in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, 2019, pp. 291–297, doi: 10.1109/ICRA.2019.8794237.

[50] GitHub. "GitHub - vedderb/bldc: The VESC motor control firmware." Accessed: Feb. 23, 2024. [Online]. Available: https://github.com/vedderb/bldc/

[51] Moafipoor, Shahram, Dorota A. Grejner-Brzezinska, C. K. Toth, Ed., *Adaptive calibration of a magnetometer compass for a personal navigation system*, 2007.

[52] C. J. Hegarty and E. Chatre, "Evolution of the Global Navigation SatelliteSystem (GNSS)," *Proc. IEEE*, vol. 96, no. 12, pp. 1902–1917, 2008, doi: 10.1109/JPROC.2008.2006090.

[53] X. Li *et al.,* "Accuracy and reliability of multi-GNSS real-time precise positioning: GPS, GLONASS, BeiDou, and Galileo," *J Geod*, vol. 89, no. 6, pp. 607–635, 2015. doi: 10.1007/s00190-015-0802-8. [Online]. Available: https://link.springer.com/article/10.1007/s00190-015-0802-8

[54] P. Karle *et al.,* "EDGAR: An Autonomous Driving Research Platform -- From Feature Development to Real-World Application," Sep. 2023.

[55] M. Matosevic, Z. Salcic, and S. Berber, "A Comparison of Accuracy Using a GPS and a Low-Cost DGPS," *IEEE Trans. Instrum. Meas.*, vol. 55, no. 5, pp. 1677–1683, 2006, doi: 10.1109/TIM.2006.880918.

[56] S. Pendleton *et al.,* "Perception, Planning, Control, and Coordination for Autonomous Vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017, doi: 10.3390/machines5010006.

[57] W. Shi, M. B. Alawieh, X. Li, and H. Yu, "Algorithm and hardware implementation for visual perception system in autonomous vehicle: A survey," *Integration*, vol. 59, pp. 148–156, 2017. doi: 10.1016/j.vlsi.2017.07.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926017303218

[58] Z. Xiang and U. Ozguner, "Environmental perception and multi-sensor data fusion for off-road autonomous vehicles," in *2005 IEEE Intelligent Transportation Systems Conference (ITSC): Vienna, Austria, 13 - 16 September 2005 ; [8th International Conference on Intelligent Transportation Systems*, Vienna, Austria, 2005, pp. 584–589, doi: 10.1109/ITSC.2005.1520113.

[59] D. Chen, M. Zhuang, X. Zhong, W. Wu, and Q. Liu, "RSPMP: real-time semantic perception and motion planning for autonomous navigation of unmanned ground vehicle in off-road environments," *Appl Intell*, vol. 53, no. 5, pp. 4979–4995, 2022. doi: 10.1007/s10489-022-03283-z. [Online]. Available: https://link.springer.com/article/10.1007/s10489-022-03283-z

[60] B. Forkel, J. Kallwies, and H. -J. Wuensche, "Probabilistic Terrain Estimation for Autonomous Off-Road Driving," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13864–13870, doi: 10.1109/ICRA48506.2021.9561689.

[61] K. Viswanath, K. Singh, P. Jiang, S. P. B., and S. Saripalli, "OFFSEG: A Semantic Segmentation Framework For Off-Road Driving," 2021, doi: 10.48550/arXiv.2103.12417.

[62] D. Maturana, P.-W. Chou, M. Uenoyama, and S. Scherer, "Real-Time Semantic Mapping for Autonomous Off-Road Navigation," in *Field and Service Robotics* (Springer Proceedings in Advanced Robotics), M. Hutter and R. Siegwart, Eds., Cham: Springer International Publishing, 2018, pp. 335–350.

[63] F. Elander, "Semantic segmentation of off-road scenery on embedded hardware using transfer learning," Master of Science, Industrial Engineering and Management, Royal Institute of Technology, Stockholm, 2021.

[64] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation," Apr. 2020. [Online]. Available: http://arxiv.org/pdf/2004.02147

[65] Q.-C. Nguyen, M.-T. Pham, D.-D. Phan, and D.-L. Vu, "Efficient Multi-Organ Segmentation Using HRNet And OCRNet," in *2022 RIVF International Conference on Computing and Communication Technologies (RIVF): RIVF 2022 : December 20-22, 2022, Ho Chi Minh City, Vietnam : proceedings*, Ho Chi Minh City, Vietnam, V. N. Q. Bao, Ed., 2022, pp. 542–547, doi: 10.1109/RIVF55975.2022.10013867.

[66] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous

Vehicle Applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, 2018, doi: 10.1109/JIOT.2018.2812300.

[67] R. Ren, H. Fu, H. Xue, X. Li, X. Hu, and M. Wu, "LiDAR-based robust localization for field autonomous vehicles in off-road environments," *Journal of Field Robotics*, vol. 38, no. 8, pp. 1059–1077, 2021, doi: 10.1002/rob.22031.

[68] T. Luettel, M. Himmelsbach, F. v. Hundelshausen, M. Manz, A. Mueller, H.-J. Wuensche. All: Autonomous Systems Technology (TAS), UniBw Muenchen, Ed., *Autonomous Offroad Navigation Under Poor GPS Conditions*, 2009.

[69] Wikipedia. "Stanley (vehicle)." Accessed: Mar. 28, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Stanley_(vehicle)&oldid=1197125325

[70] NVIDIA. "Eingebettete Systeme von NVIDIA für autonome Maschinen der nächsten Generation." Accessed: Mar. 28, 2024. [Online]. Available: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/

[71] B. Zhou *et al.,* "An autonomous navigation approach for unmanned vehicle in outdoor unstructured terrain with dynamic and negative obstacles," *Robotica*, vol. 40, no. 8, pp. 2831–2854, 2022. doi: 10.1017/S0263574721001983. [Online]. Available: https://www.cambridge.org/core/journals/robotica/article/an-autonomous-navigation-approach-for-unmanned-vehicle-in-outdoor-unstructured-terrain-with-dynamic-and-negative-obstacles/1E8CB785AE85F33F7C87D018ACBAB50D

[72] A. A. Suzen, B. Duman, and B. Sen, "Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN," in *HORA 2020: 2nd International Congress on Human-Computer Interaction, Optimization and Robotic Applications : June 26-27, 2020, Turkey : proceedings*, Ankara, Turkey, 2020, pp. 1–5, doi: 10.1109/HORA49412.2020.9152915.

[73] "ROS: Home." Accessed: Mar. 28, 2024. [Online]. Available: https://www.ros.org/

[74] A. Hussein, P. Marin-Plaza, D. Martin, A. de La Escalera, and J. M. Armingol, "Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, Gotenburg, Sweden, 2016, pp. 104–109.

[75] K. Konolige *et al.,* "Mapping, navigation, and learning for off-road traversal," *Journal of Field Robotics*, vol. 26, no. 1, pp. 88–113, 2009, doi: 10.1002/rob.20271.

[76] S. Campbell, N. O'Mahony, A. Carvalho, L. Krpalkova, D. Riordan, and J. Walsh, "Path Planning Techniques for Mobile Robots A Review," in *2020 6th International Conference on Mechatronics and Robotics Engineering: ICMRE 2020 : Barcelona, Spain, February 12-15, 2020*, Barcelona, Spain, 2020, pp. 12–16, doi: 10.1109/ICMRE49073.2020.9065187.

[77] A. Y. Kapi, M. S. Sunar, and Z. A. Algfoor, "Summary of Pathfinding in Off-Road Environment," in *2020 6th International Conference on Interactive Digital Media (ICIDM)*, Bandung, Indonesia, 2020, pp. 1–4, doi: 10.1109/ICIDM51048.2020.9339639.

[78] K.-D. Kuhnert, "Software architecture of the Autonomous Mobile Outdoor Robot AMOR," in *2008 IEEE Intelligent Vehicles Symposium: Eindhoven, Netherlands, 4 - 6 June 2008*, Eindhoven, Netherlands, 2008, pp. 889–894, doi: 10.1109/IVS.2008.4621234.

[79] M. Himmelsbach, T. Luettel, F. Hecker, F. v. Hundelshausen, and H.-J. Wuensche, "Autonomous Off-Road Navigation for MuCAR-3," *Künstl Intell*, vol. 25, no. 2, pp. 145–149, 2011.

[80] A. Cherubini, F. Spindler, and F. Chaumette, "A new tentacles-based technique for avoiding obstacles during visual navigation," in *2012 IEEE International Conference on Robotics and Automation (ICRA 2012): St. Paul, Minnesota, USA, 14 - 18 May 2012*, St Paul, MN, USA, 2012, pp. 4850–4855, doi: 10.1109/ICRA.2012.6224584.

[81] Raia Hadsell, Ayse Erkan, Pierre Sermanet, Jan Ben, and Yann LeCun, "A Multi-Range Vision Strategy for Autonomous Offroad Navigation," in 2007. [Online]. Available: https://www.researchgate.net/publication/216792726_A_Multi-Range_Vision_Strategy_for_Autonomous_Offroad_Navigation

[82] "GitHub - f1tenth/f1tenth_system: Drivers and system level code for the F1TENTH vehicles." Accessed: Apr. 10, 2024. [Online]. Available: https://github.com/f1tenth/f1tenth_system

[83] GitHub. "GitHub - stereolabs/zed-ros-wrapper: ROS wrapper for the ZED SDK." Accessed: Apr. 10, 2024.

[84] GitHub. "TUM-AVS/f1tenth_offroad." Accessed: Apr. 11, 2024. [Online]. Available: https://github.com/TUM-AVS/f1tenth_offroad

[85] Stevewhims. "Aktivieren von NVIDIA CUDA unter WSL 2." Accessed: Apr. 14, 2024. [Online]. Available: https://learn.microsoft.com/de-de/windows/ai/directml/gpu-cuda-in-wsl

[86] GitHub. "kasiv008/OFFSEG: Official Implementation of OFFSEG: A Semantic Segmentation framework for Off-Road Driving." Accessed: Apr. 11, 2024. [Online]. Available: https://github.com/kasiv008/OFFSEG

[87] "kmcuda/src at master · src-d/kmcuda." Accessed: Apr. 11, 2024. [Online]. Available: https://github.com/src-d/kmcuda/tree/master/src

[88] GitHub. "image_pipeline/depth_image_proc/src/nodelets/point_cloud_xyz.cpp at master · mjgarcia/image_pipeline." Accessed: Apr. 11, 2024. [Online]. Available: https://github.com/mjgarcia/image_pipeline/blob/master/depth_image_proc/src/nodelets/point_cloud_xyz.cpp

[89] "ros/src/detector/scripts/pipeline.py at master · bostondiditeam/ros." Accessed: Apr. 11, 2024. [Online]. Available: https://github.com/bostondiditeam/ros/blob/master/src/detector/scripts/pipeline.py

[90] GitHub. "GitHub - ros-drivers/nmea_navsat_driver: ROS package containing drivers for NMEA devices that can output satellite navigation data (e.g. GPS or GLONASS)." Accessed: Apr. 15, 2024. [Online]. Available: https://github.com/ros-drivers/nmea_navsat_driver

[91] H. Marsiske, "Elrob: Leistungstest für Roboter - Motivationsschub für die Forscher," *heise online*, 15 Jun., 2009. Accessed: Apr. 4, 2024. [Online]. Available: https://www.heise.de/news/Elrob-Leistungstest-fuer-Roboter-Motivationsschub-fuer-die-Forscher-181837.html

[92] H. Marsiske, "ELROB 2008: Nur zwei kamen durch," *heise online*, 07 Jan., 2008. Accessed: Feb. 11, 2024. [Online]. Available: https://www.heise.de/news/ELROB-2008-Nur-zwei-kamen-durch-182649.html

[93] F. E. Schneider, D. Wildermuth, and H.-L. Wolf, "ELROB and EURATHLON: Improving search & rescue robotics through real-world robot competitions," in *2015 10th International Workshop on Robot Motion and Control (RoMoCo 2015): Poznań, Poland, 6-8 July 2015*, Poznan, Poland, 2015, pp. 118–123, doi: 10.1109/RoMoCo.2015.7219722.

[94] B. Zhou *et al.,* "The Mars rover subsurface penetrating radar onboard China's Mars 2020 mission," *Earth and Planetary Physics*, vol. 4, no. 4, pp. 1–10, 2020, doi: 10.26464/epp2020054.

[95] W. Xu, C. Yan, W. Jia, X. Ji, and J. Liu, "Analyzing and Enhancing the Security of Ultrasonic Sensors for Autonomous Vehicles," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5015–5029, 2018, doi: 10.1109/JIOT.2018.2867917.

[96] M. N. Mubarak, "Outdoor obstacle detection using ultrasonic sensors for an autonomous vehicle ensuring safe operations," [Online]. Available: https://trepo.tuni.fi/handle/123456789/21422

[97] W. C. Lin and H. R. Tsui, "On dual ultrasound sensor technique for unmanned vehicles," *Automation in Construction*, vol. 1, no. 2, pp. 153–165, 1992. doi: 10.1016/0926-5805(92)90005-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0926580592900055

[98] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: a survey," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 315–329, 2020, doi: 10.1109/JAS.2020.1003021.

[99] P. Neigel, J. Rambach, and D. Stricker, "OFFSED: Off-Road Semantic Segmentation Dataset," in *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Online Streaming, --- Select a Country ---, 2021, pp. 552–557.

[100] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE transactions on neural networks and learning systems*, early access. doi: 10.1109/TNNLS.2021.3084827.

[101] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Nov. 2015. [Online]. Available: http://arxiv.org/pdf/1511.08458.pdf

[102] K. Kumar and G. S. M. Thakur, "Advanced Applications of Neural Networks and Artificial Intelligence: A Review," *IJITCS*, vol. 4, no. 6, pp. 57–68, 2012, doi: 10.5815/ijitcs.2012.06.08.

[103] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Muller, "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications," *Proc. IEEE*, vol. 109, no. 3, pp. 247–278, 2021, doi: 10.1109/JPROC.2021.3060483.

[104] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proceedings of 2017 International Conference on Engineering & Technology (ICET'2017): Akdeniz University, Antalya, Turkey, 21-23 August 2017*, Antalya, 2017, pp. 1–6, doi: 10.1109/ICEngTechnol.2017.8308186.

[105] H. Stoll, P. Zimmer, F. Hartmann, and E. Sax, "GPS-independent localization for off-road vehicles using ultra-wideband (UWB)," in *IEEE ITSC 2017: 20th International Conference on Intelligent Transportation Systems : Mielparque Yokohama in Yokohama,*

Kanagawa, Japan, October 16-19, 2017, Yokohama, 2017, pp. 1–6, doi: 10.1109/ITSC.2017.8317763.

[106] H. Mousazadeh, "A technical review on navigation systems of agricultural autonomous off-road vehicles," *Journal of Terramechanics*, vol. 50, no. 3, pp. 211–232, 2013. doi: 10.1016/j.jterra.2013.03.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022489813000220

[107] A. Hussein, H. Mostafa, M. Badrel-din, O. Sultan, and A. Khamis, "Metaheuristic optimization approach to mobile robot path planning," in *2012 International Conference on Engineering and Technology (ICET 2012): New Cairo City], Cairo, Egypt, 10 - 11 October 2012*, Cairo, Egypt, 2012, pp. 1–6, doi: 10.1109/ICEngTechnol.2012.6396150.

[108] H. Rastgoftar, B. Zhang, and E. M. Atkins, "A Data-Driven Approach for Autonomous Motion Planning and Control in Off-Road Driving Scenarios," in *2018 Annual American Control Conference (ACC): June 27-29, 2018, Wisconsin Center, Milwaukee, USA*, Milwaukee, WI, 2018, pp. 5876–5883, doi: 10.23919/ACC.2018.8431069.

[109] J. Silveira, K. Cabral, S. Givigi, and J. A. Marshall, "Real-Time Fast Marching Tree for Mobile Robot Motion Planning in Dynamic Environments," in *ICRA 2023: Conference proceedings : 29th May-2nd June 2023, ExCeL London : IEEE International Conference on Robotics and Automation*, London, United Kingdom, M. O'Malley, Ed., 2023, pp. 7837–7843, doi: 10.1109/ICRA48891.2023.10160595.

Appendix

Appendix A ELROB Vehicle's Chassis Setups

Figure A.1: Notable ELROB AOVs. On the left, MuCAR-3 of the UniBw [91], and an the right the RAVON vehicle of the University of Kaiserslautern [92]

The ELROB offers both military and civilian challenges, which allow teams to choose from a set of predefined scenarios to develop systems for and compete in. These range from "different kinds of reconnaissance and surveillance missions combined with the detection of special objects, or transportation, which can be carried out with a single vehicle or in form of a convoy with at least two vehicles" [93, p. 119].

Firstly, competing in the 2008 ELROB, is RAVON, the Robust Autonomous Vehicle for Offroad Navigation (Figure A.1, left), developed by the University of Kaiserslautern in Hannover, Germany. While it failed to finish the 2008 Offroad Course, it completed a majority of it, with only a modified tracked military vehicle covering more of the track [92]. The platform used for the RAVON robot is a robuCAR TT system manufactured by Robosoft, of which the researchers altered several components to comply with the high requirements for offroad driving, such as reinforcing the shock absorbers and transverse links. Having independent axles allows the platform to make sharp turns and – by steering in parallel directions – drift away from lateral obstacles without changing direction, which leads the Kaiserslautern team to theorize that this increased mobility can drastically increase its capabilities to maneuver in obstacle rich environments. RAVON utilizes four independent 1.9 kW motors driving heavy Hankook offroad wheels, allowing the robot to reach a maximum velocity of 3 m/s and climbing up to 100 percent slopes, and with its eight 55Ah OPTIMA Batteries, it can travel for up to 4 hours [40].

Also highlighted here are the efforts of the MuCAR-3 Team, based at the University of the Bundeswehr (German Armed Forces) in Neubiberg, Germany, who competed in the 2009 ELROB Installment. In direct analogy to the Stanford University's "Stanley" vehicle used a VW Touareg as the basis of their Platform (Figure A.1, right), likely with similar considerations, maybe inspiration taken from the Stanford Team. In addition to modifications the Stanford Team mentioned three to five years earlier, the MuCAR-3 AOV was additionally with a high-power generator, to power more advanced Computing hardware as compared to earlier approaches [43]. As the Touareg provides low level systems for electronic control of throttle and brake, the only needed electric actuators interface with the original Steering Wheel shaft, the parking brake as well as the gear shift lever [43].

Appendix B Other Exteroceptive Sensors used on AV

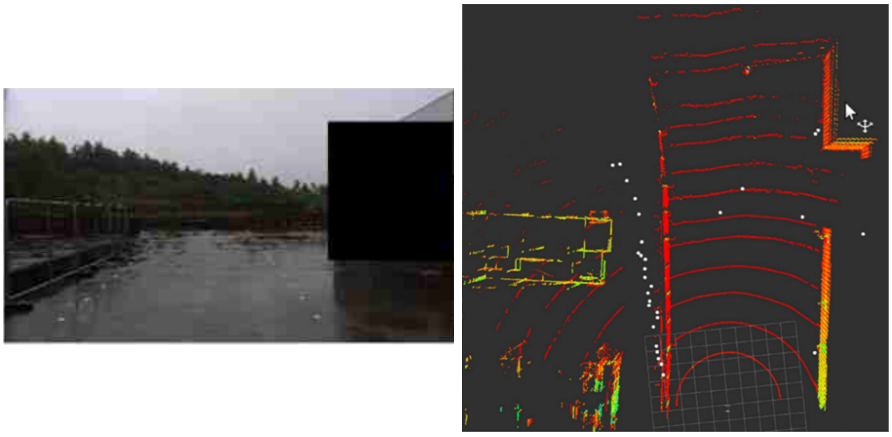


Figure B.1: On the left: forward facing camera image of an AV [3]. On the right: Combined top-down sensor image from a Radar and LIDAR sensor viewing the same scene as the camera. Colored points represent data from the LIDAR sensor, while white points represent Radar data [3].

Radar Sensors

"Radio Detection and Ranging, or Radar, was first established before World War II and operated on the principle of radiating electromagnetic (EM) waves within the area of interest and receiving the scattered waves (or reflections) of targets for further signal processing and establishing range information about the targets" [3, p. 11]. Utilizing the Doppler property of EM Waves, a radar can additionally determine the relative speed of objects within the scan range directly, without the need for post processing [2]. Radars are not affected by adverse weather conditions or lighting, and can detect objects at distances up to 200m away [2]. Problems of Radar sensors include poor resolution (especially in the vertical direction), less data points, slow processing speeds, and false detections of metal and/or stationary objects [2, 3, 35]. An example of this can is visible in Figure B.1, which shows combined LIDAR and radar data on the right. The Radar sensor delivers less data overall, with some false positives (white points) in the middle of the image [3].

Still, radar sensors are employed in some AV [3] to detect both moving and stationary obstacles, and some vehicles might utilize them in mapping workloads. It is however limited in object detection tasks by its low resolution, which is why radar data is often fused with LIDAR or camera data in order to get more reliable information [3]. Radars mainly operate at 24 GHz or at higher frequencies around 79 GHz; though the 79 GHz Sensors have a higher resolution of range, velocity and angle and have thus become more favored as time goes on [3].

The presented examples use radar, although less than LIDAR or cameras. RASCAL uses a radar system for the purpose of detecting metal objects like fences, which might be hard to pick up on other sensor systems [15], the AVIDOR-2004 vehicle of their sister team also employs a radar in the mm-wave range [7]. Stanford's Stanley vehicle uses two forward facing 24 GHz

radar sensors, depicted in Figure 2.8, for far field detection up to 200m [14]. Radar sensors are omitted in RAVON [40] and MuCAR-3 [43], and from literature analyzed here, it seems that while radar is in use on certain Mars rovers, it seems to be for more scientific purposes and not for pathfinding or perception [94]. Radar sensors are omitted in all of the smaller vehicles which provide sensor stacks, such as the F1TENTH platforms [11, 13, 33], all of the vehicles tested and analyzed in [26] and the University of Sydney's Swagbot [24].

Ultrasonic Sensors

"An Ultrasonic sensor is a device that uses sound waves to measure the distance to an object" [35, p. 2]. Similar to LIDAR sensors, by measuring the TOF for a sent out pulse of ultrasound returning after reflecting off of a surface and knowing the velocity of the wave, it is possible to calculate the distance to said surface [35, 95]. They have found application in obstacle avoidance and path planning and are robust, cheap, reliable in adverse conditions and excel in the precise measurement of small distances [35]. Their downsides include difficulties with the detection of angled surfaces, a limited range – about 6m [16, 96] – and susceptibility to shocks, vibrations and external disturbances [35, 96, 97]. Other limitations lie in their slow response time and long signal travel time, which limits the frequency of measurements, as well as their large area of detection, which effectively confines their use to 1D distance measurements [96].

In AVs, only two of the presented examples mention the use of ultrasound sensors, SciAutonics' AVIDOR-2004 [7] and RASCAL [16], the latter of which employed them for redundancy in short range obstacle detection or emergency situations, in which longer range sensors may not work correctly. When this fallback is necessary, the vehicle is then limited to slow speeds owing to the ultrasound sensors short range; however, as this system is a fallback only used in emergencies, this limitation was found not impact vehicle performance by researches in [16].

Appendix C Additional Information on Sensor Configurations

Figure C.1: TUM's EDGAR AV, an example of a modern AV platform.

EDGAR (Figure C.1) is based on a Volkswagen T7 Multivan eHybrid [54], the high roof of which introduced limitations for the sensor roof rack (Figure C.2), necessitating trade-offs between long- and short-range sensors. The vehicle also enables the researchers to use built-in sensors such as ultrasound and radar sensors and control the vehicle via built in actuators for assisted driving functionality.

For the exteroceptive sensors, EDGAR uses six mid-range cameras to provide a 360° view around the vehicle, placed and equipped to minimize blind spots. In addition to this, two long-range cameras are placed at the front of the vehicle to facilitate long range, stereo vision, with two additional short range, active IR depth cameras placed on the roof rack (Figure C.2). EDGAR's LiDAR complement utilizes four 3D units, two rotating sensors placed on the front corners of the roof rack, suppling 360° coverage around the vehicle, and two long range solid-state LiDAR for front and rear long-range detections (Figure C.2). The vehicle also employs 6 Radar sensors, which alternate between near- and far-field scanning patterns. Lastly, EDGAR uses four microphones at the corners of the vehicle. This is done to locate and detect emergency vehicle sirens, provide blind spot detection and to generate datapoints for road surface estimation [54]. Figure C.2 shows EDGARs exteroceptive sensors as well as the sensor combinations of the previous examples.

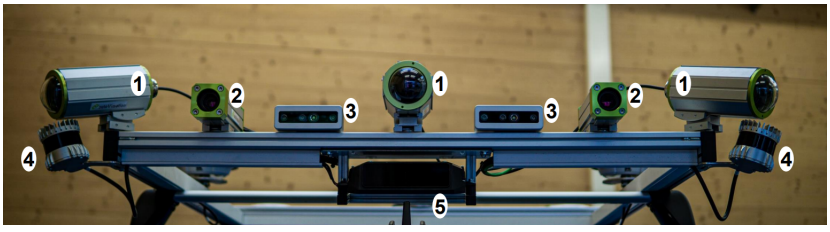


Figure C.2: EDGAR's sensor roof rack. Annotations 1, 2 and 3 are medium-, long- and short-range camera respectively, with 4 and 5 denoting medium- and short-range LiDAR respectively, adapted from [54]

Table C.1: Exteroceptive sensors in the presented AV examples, ordered by appearance in this paper. Checkmarks indicate sensor presence, while sensors not mentioned in the corresponding reference have their field left empty. Checkmarks centered in subcolumns (e.g., between 2D and 3D LiDAR) indicate presence without information about subtypes.

	LiDAR		Camera			Ultrasonic
Vehicle Names	2D	3D	mono	stereo	Radar	Sensors
Stanley [14]	✓		✓		✓	
RAVON [40]	✓	✓		✓		
RASCAL [15]	✓			✓	✓	✓
AVIDOR-2004 [16]	✓				✓	✓
MuCAR-3 [43]		✓	✓			
Tracked and Ackermann [26]	✓		✓		✓	
Mars Rovers [47]			✓	✓		
F1TENTH in [11]				✓		
F1TENTH in [33]	✓					
F1TENTH in [13]	✓			✓		
EDGAR [54]		✓	✓	✓	✓	✓

Proprioceptive Sensors on EDGAR include a combined GPS-IMU system, taking advantage of both DGPS to achieve 2cm of GPS accuracy and a dual-antenna setup to measure heading via GPS, even at a standstill. The IMU is only used during a GPS outage, to bring the vehicle to a safe stop [54]. Table C.2 contains a comparison of proprioceptive sensors of the presented AVs.

Table C.2: Interoceptive sensors used in the presented AV examples, ordered by appearance in this paper. Black checkmarks indicate presence, grey checkmarks indicate situations where research does not explicitly mention sensor presence, but where it is highly likely, with fields for all other non-mentioned sensors left blank.

		IMS		GPS	
Vehicle Names	Encoders	IMU	Magnetometer	position	heading
Stanley [14]	✓	✓		✓	✓
RAVON [40]	✓	✓	✓	✓	
RASCAL [15]	✓	✓	✓	✓	
AVIDOR-2004 [16]	✓	✓		✓	
MuCAR-3 [43]	✓	✓		✓	✓
Tracked and Ackermann [26]	✓	✓			
F1TENTH [11, 13, 33]	✓	✓			
EDGAR [54]	✓	✓		✓	✓

Appendix D Expanded Perception Preliminaries and Techniques

This chapter of the appendix will present the background and extended knowledge required to fully understand and contextualize the perception algorithms presented in this work.

Strictly Calculative Approaches to Camera Perception

One directly calculative approach is pursued by researchers in [15] with their previously presented RASCAL vehicle. By dividing each camera feed of their stereo setup into multiple Regions of Interest (ROIs), median-filtering, applying a linear gradient filter and using a Hough transformation their vision algorithm extracts line segments of road borders from the camera feed (Figure D.1 on the left). The algorithm then forwards these, along with obstacles which the stereo camera detects via the maximally stable extremal regions approach, to the path planning and control systems.

Another approach is presented in [43] with the MuCAR-3 vehicle, which has been an example in previous sections. Here, the researchers detect the road surface by assessing and normalizing the saturation value of each pixel in the image in the HSI color format – denoted by $\mathrm{s_w}(x,y)$ – according to formula (C.1),

$$\mathrm{s_w}(x,y) = \begin{cases} 0 & \mathrm{s_w}(x,y) \leq \mu_\mathrm{s} \\ \dfrac{255(\mathrm{s_w}(x,y) - \mu_\mathrm{s})}{s_\mathrm{off}} & \mu_\mathrm{s} < \mathrm{s_w}(x,y) < (\mu_\mathrm{s} + s_\mathrm{off}) \\ 255 & \mathrm{s_w}(x,y) \geq (\mu_\mathrm{s} + s_\mathrm{off}) \end{cases} \tag{C.1}$$

where μ_s is a temporal low-pass filtered mean saturation value serving as the lower cutoff, and s_off is the upper cut-off value for the normalization. The system can then use this further via path planning algorithms, which Figure D.1 displays on the right, a algorithm overlays colored lines over the black-and-white perception result indicating the drivability of selected paths.

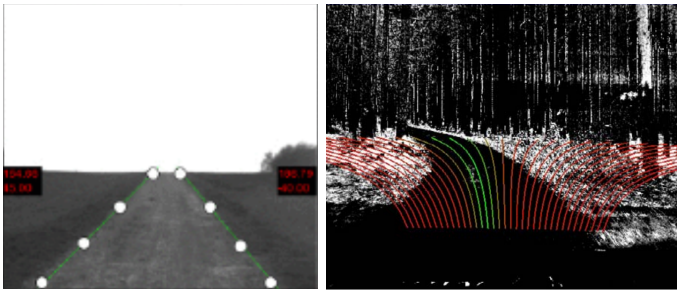


Figure D.1: Left, an image from RASCAL's right camera, which was analyzed and annotated with road boarders according to its perception algorithm [15], on the right an image from MuCAR-3's perception pipeline, showcasing the result of its saturation based analysis algorithm, where black indicates traversability, in conjunction with an early drivability analysis result in the form of colored paths [43]

Artificial Intelligence Basics

Recent Advancements in AI and machine learning have enabled the deployment of these technologies into the realm of Autonomous Driving [56, 57, 98], and can apply to the topic of semantic segmentation: “Current state-of-the-art approaches to semantic full image segmentation mainly rely on convolutional neural networks that take in a monocular RGB-image as input and produce a class label for every pixel in the image” [99, p. 552]. It has been demonstrated that utilizing convolutional neural networks (CNNs) for this task yields systems that are robust to scaling, while at the same time outperforming other state-of-the-art methods in literature, which rely on hand-crafted features and their relations in color and texture [57], and can rival human perception accuracy in some tasks [98].

A convolutional neural network is a special type of artificial neural network [100, 101], which in turn represents a generalized, mathematically modeled version of a biological nervous system [101, 102]. An artificial neural network is a collection of neurons – modeled as nodes in a graph – which receive and forward information along the directed nodes of said graph according to an activation function, which describes how the output value derives from the input values. By organizing these neurons into the simplest form of an artificial neural network, which contains three layers and is depicted in Figure D.2 on the left, we can derive output variables from a set of input variable [102]. This is done by iterating though the linear transformations formed by the activation functions of the neurons in each layer, successively applying these functions until we arrive at the output layer [102, 103]. More advanced, or ‘deep’ neural networks, employ multiple hidden layers between input and output, which enhances their prediction power and accuracy at the cost of increased computational complexity and less reliable estimations, and have proven themselves as powerful tools in computer vision [103]. Convolutional neural networks are designed to work primarily with images which otherwise would be too complex to efficiently analyze them with regular artificial neural networks [101, 104]. To this effect, they introduce three advantages over general artificial neural networks, reducing complexity as well as time required to ‘train’ a network [100, 104], e.g. adapting the activation functions to achieve the desired outputs [101, 102]:

10. Local connections: Neurons do not connect to all neurons of the previous layer, only to a small area, e.g., 3x3 pixels.
11. Weight sharing: a group of neurons may share the same computational weights or activation function.
12. Down-/subsampling: reduction of the number of datapoints while retaining useful information in the form of a pooling layer.

Figure D.2 shows an example of a CNN on the right. As its creators designed it to work with images as input, the first layers are two-dimensional, square fields of neurons.

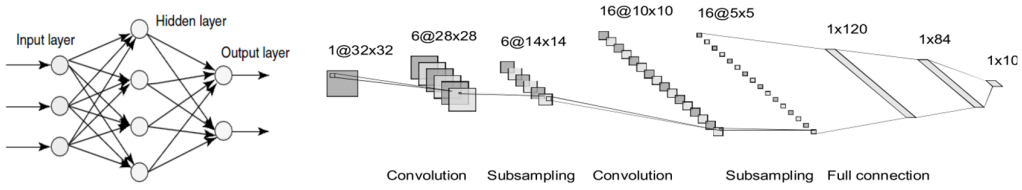


Figure D.2: Left, a visual representation of a simple artificial neural network containing an input, hidden, and output layer, fully connected between layers, in the form of a directed graph [102]. Right, the architecture of LeNet-5, a convolutional neural network

designed for reading handwritten text, with two convolutional layers (which utilize local connections and weight sharing), two pooling layers and three fully connected layers [100].

To ‘train’ the network, making the outputs display the desired values for a given input, neural networks self-optimize through machine learning. This can happen through two different paradigms [101, 102]:

- Unsupervised Learning: Users provide training data without preassigned, expected outputs, instead, the neural network attempts to cluster the input according to self-detected patterns into different classes.
- Supervised Learning: Users provide the neural network with example inputs and their respective expected outputs, which it then uses to self-optimize – ‘train’ – to reduce the overall output error and provide the correct results. Researchers use this in image-focused approaches, which are relevant here.

Further expanding the capabilities of CNNs are Fully Convolutional Networks (FCNs) [100], which build upon the idea by introducing “deconvolution” layers, which up-sample to increase output resolution – if the output is an image – as well as skip layers, which connect early layers to up-sampled layers to preserve high-frequency detail [62].

There exist many more approaches for vehicle, pedestrian and lane detection, which use more complicated perception approaches, often combining machine learning and AI with more traditionally calculative algorithms [18, 56, 57], but since these are not deemed relevant here for the task of perception for offroad autonomous locomotion, this thesis will not present them here.

LiDAR Perception

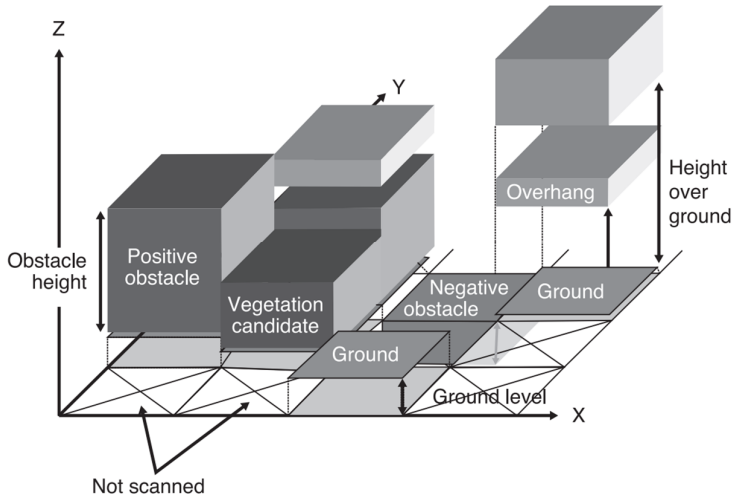


Figure D.3: Height-Based LiDAR Classification categories of RAVON [40]

Stanley, one of the earliest offroad AVs, uses a mainly LiDAR based perception system [14]. Here, the algorithm transforms LiDAR Data into a 2D top-down grid, and then analyzed based on the vertical distance between points. When measurements between vertical points in a certain

grid space exceed a critical threshold value, the algorithm detects an obstacle. When it finds no such points, but at least one point falls into a grid cell, it classifies terrain as drivable, while with no points in a cell cause the algorithm to assume that it is unknown. Similar systems are employed in later offroad AVs such as MuCar-3 [43], who reference the work done in [14], and RAVON, who additionally classify overhangs, vegetation and negative obstacles [40], shown in Figure D.3. However, while effective at standstill, in motion, such systems proved unreliable, since small pose estimation errors related to the vehicles orientation could throw off readings, resulting wrong perception results, subsequently forcing the vehicle off the road [14]. While a system could combat this by better pose estimation systems, these proved too expensive for the researchers of Stanford University and their Stanley vehicle. Instead, testing showed that time between measurements links to measurement errors, and by applying a time-based confidence value to measurements, results can improve. Lastly, the researchers apply a learning-based parameter tuning approach like a behavioral copying in AI. A human drives the vehicle over only traversable terrain, while an algorithm optimizes the thresholds and confidence values to classify the driven-over terrain as traversable in the perception algorithm. Combining these two systems, error rates dropped from 12.6% to 0.002% [14].

Newer work on the topic of LiDAR perception includes work done by a team of the UniBw in Munich, who focus on the – drivable – terrain itself rather than classifying obstacles [60]. Here, by accumulating the LiDAR data over time via an information filter, and subsequently applying a special smoothing operation, they decouple terrain estimation and obstacle detection, resulting in accurate height and slope information, while at the same time allowing the algorithm to interpolate to areas which are not visible to the sensors, further expanding capability [60].

Another recent approach is presented by researchers of the University of Washington in [6], where a special CNN called Bird's Eye View Network (BEVNet) predicts terrain classes based on LiDAR input. This network takes LiDAR data discretized into 3D pixels as input – so called voxels – and directly generates a 2D top-down traversability map, classifying terrain into free, low-cost, medium-cost, and lethal. This approach yielded more reliable results especially in offroad terrain, for example on the RELLIS-3D dataset, where it outperformed any other tested solutions with a large gap [6].

Historic Sensor Fusion Approaches

The first example of sensor fusion on offroad AV presented here comes from one of early DARPA Grand Challenge vehicles, Stanley of Stanford University [14]. Developed to combat range limitations with the existing LiDAR road/obstacle detection system, Stanley uses a long-range color camera to 'extend' the range of the LiDAR system. It does this by identifying a quadrilateral of drivable terrain using the LiDAR system, mapping it into the camera image, and detecting the color of the drivable terrain using Gaussian functions. These then merge into an internal, dynamic database of drivable terrain colors – thus integrating elements of early machine learning – which are subsequently used to classify a camera image into drivable and non-drivable terrain based on the color of the surface. The algorithm only keeps drivable areas which connect to the initial quadrilateral, and all other surfaces are set to non-drivable. This Camera-LiDAR fusion system extends the range of the system from 22m up to 70m, enabling higher driving speeds [14].

Another approach to LiDAR-Camera fusion is shown in [43], on the MuCAR-3 vehicle. Here, an algorithm performs traversability analysis on combined factors from both camera and LiDAR, where the latter provides overall binary drivability assessment together with a 'clearness' and a

'flatness' factor, while the camera provides an additional factor based on the calculative approach shown previously.

Next, utilizing a stereo camera and a 3D LiDAR system, researchers in [45] combine classification data from the two systems into a unified perception model. Both sensors utilize a similar system related to basic perception algorithms already presented, using Gaussian functions and comparative algorithms to assess whether certain features belong to the 'ground' class. After these algorithms run, they generate two separate maps. These then combine using data generated from ground truth datasets, where they calculate weights for the detections based on the respective accuracy of both classifier systems and the two classes. This showed a notable improvement from an accuracy of 95.5% and 95.1% for LiDAR and camera respectively to 96.5% combined. A similar approach to sensor fusion is proposed in [74], where the selection between the two separate maps is based on simply a confidence value provided by the classification algorithms.

The "sensor fusion" term may also apply to the integration of Radar and Ultrasound sensors, exemplary shown on the RAVON vehicle in [15]. Here, researchers use Radar sensors to detect metal wire fences, which are hard to detect with other sensors, and ultrasound for low-range, poor visibility situations. These then merge with other obstacle/environment data – e.g., from a Camera and LiDAR – in a unified coordinate frame, and subsequently passed on to navigation and pathfinding routines.

Local localization and SLAM

SLAM systems usually use grid-based maps to record occupancy of obstacles or other terrain elements, where the map is filled in an iterative, live process while the robot is moving through the environment [10, 18]. According to [18], this process is realized as follows:

1. The robot uses exteroceptive sensor to gain knowledge about its environment, e.g., location of obstacles or terrain features.
2. Data transfers into the grid depending on the sensor data, e.g., increasing or decreasing the likelihood of obstacles.
3. Using distance-to-landmark information or odometry, the robot localizes itself on the map.
4. The robot moves somewhere else in the space of the map.
5. Repeat the process until the map is complete or the robot reaches the goal.

The data stored in the grid can vary, and include visual terrain imagery, as well as obstacle probabilities, terrain slopes or predicted movement costs [10, 18, 59, 60]. Challenges here include stationary vs. dynamic obstacles and higher vehicle speeds [18]. Specifically in off-road environments, additional complications include reliable data acquisition when the vehicle position is uncertain, complex geometries of obstacles and terrain, decluttering of the long-term map, large computational overhead with long-duration, complex maps and the reliable detection of recurring elements and looping paths [10].

To combat this, the following approaches to local localization have been developed by researchers: Stanley, Stanford University's entry to the DARPA Grand Challenge, used an IMU-system based mainly on accelerometers and gyroscopes to gain information about vehicle orientation in order to facilitate SLAM for short-range navigation purposes [14], and the RAVON offroad AV

expands on this by additionally including wheel odometry – as does the MuCAR-3 vehicle [68] – and a magnetic field sensor [40]. Researchers from the Karlsruhe Institute of Technology have proposed a system in [105] that, in addition to odometry and IMU measurements, also includes a network of ultra-wideband (UWB) antennas to provide position information via TOF measurements. Other local localization approaches include the usage of highly accurate RTK-GPS [106], and, more recently, Camera, LiDAR, Radar and Ultrasonic based localization systems [1, 67].

One Camera-based localization approach a two-stage approach, where an algorithm first performs rough position orientation through orientation histograms, then, it achieves fine localization through a landmark map, which achieved up to 75cm accuracy, but was susceptible to changes in illumination and the angle of the camera system. An alternative approach to camera based systems utilizes visual odometry, where a stereo camera is used to track feature points in the environment over multiple frames, thus enabling the calculation of relative position changes [66]. Such systems, which utilize machine vision, are complex and computationally expensive [106], but can be low cost alternatives that perform reasonably well [66].

LiDAR and Radar based techniques work in similar ways, through tracking of landmark points in the sensor data like SLAM techniques. In Radar, due to the low number of detection points, errors can still reach into the meters, although some solutions with ground penetrating radars that utilize map knowledge achieved accuracy results in the centimeter range. LiDAR systems provide significantly more data, and as such, are much more viable for the task of localization, routinely achieving accuracies in the centimeter range. While they can provide accurate and robust information, their drawbacks include high power draw and complex computations, as well as high implementation costs [66].

At this point, it is also worth noting that some researchers propose systems in which the need for precise, local localization is mitigated, substituting the mediated perception approach for a behavior reflex approach, which provides end-to-end driving solutions that don't rely on extensive SLAM techniques [1].

Appendix E Global Pathfinding

Numerous different approaches for global pathfinding in offroad navigation have been developed, tested and evaluated by researchers, namely classic path planning algorithms – such as Dijkstra's or the A* algorithm – and improvements thereof, which introduce adaptivity and other improvements [77]. All of these system operate on some variant of map-type data which is previously known [5].

Classic Path planning algorithms

Classic Path planning algorithms for the task of autonomous offroad navigation encompass the A*, D* and Dijkstra's algorithm [56, 77]. These methods apply as a graph search, with a graph network that reflects drivable roads and paths, their connections and may also include other information such as motion cost and slopes which then constitute the 'weight' of individual graph edges. These networks can be either manually generated using maps, may be computer calculated based on vehicle movement or through computer vision techniques [56].

Examples of experiments using these algorithms encompass work done by researchers in [78] and a comparative study in [77], where the latter especially shows that the base level algorithms are unsuitable for offroad navigation, since they are incapable of accounting for the movement cost of steep slopes, uneven maps, instability of data and power limits [77]. Expanding on this, papers [77], [56] and [75] have found the A*, Dijkstra's and D* algorithm respectively to be insufficient for the task of autonomous offroad navigation.

Improved Algorithms

Modifying the A*, D* and Dijkstra's algorithm can lead to viability in offroad tasksets, as analyzed in [77]. By including terrain slope into the calculations, it is possible to modify the algorithms in order obtain viable paths, however most studies lack tests for real time capability. Other improvements include a modified A* algorithm that utilizes rays rather than grid cells on a map in [81], which has been tested in the field and has proven to be viable. Other notable improved classic algorithms are an 3D anthill colony algorithm, which, while faster, did not find a shorter path compared to other solutions, and an adaptive Dijkstra's and A* algorithm, which also showed to reduce processing time while not addressing other issues previously mentioned [77].

Another improvement for global path planning includes trajectory-based optimization [107]. Here, the algorithm selects a random path through the environment at first, and then iteratively optimizes it. One of the proposed algorithms for this is Tabu Search, which uses local optimization and the acceptance of non-improving solutions over endless optimization attempts to facilitate an increase in computational speed. Another proposed algorithm is Simulated Annealing. Inspired by the physical process of annealing, it is easy to implement, able to solve complex, nonlinear problems without entrapment in optimization loops, however, it also possesses many variables that require tuning. Lastly, a Genetic Algorithm, based on the principles of natural selection, in which advantageous solutions with unique advantages progress in the iterative process, can apply to the task of global pathfinding. While it is highly capable, it also cannot guarantee finding the optimal path. In testing done in [107], SA is proven to be the fastest of these three solutions, while the genetic algorithm arrives at a near optimal solution faster than any other algorithm.

Two more global navigation algorithms are presented in [76], which are the potential field method and the cell decomposition method. The former treats the robot as a particle, which moves in an artificial potential field constituted by influences from obstacles as repulsive forces and the destination point as an attractive force. The cell decomposition method breaks the robot's surroundings up into cells, which it then populates with obstacles and traverses with the goal of reaching the robots target point. While both of these methods have proven viable, they are still limited in real-world application due to high computational requirements and inability to perform in dynamic situations [76].

Some researchers introduce custom solutions to the global pathfinding problem. In [75], after determining that a D* algorithm is insufficient, researchers implement a custom gradient planner, which operates on a cost map by simulating wavefronts over the cell space of the map. Work done in [108] implements a global planner based on dynamic programming, and researchers in [71] implement another custom solution that values terrain reliability, terrain flatness and slope analysis. Finishing up, research in [109] introduces a Real-time fast marching tree analysis for the task of mobile robot navigation. As its creators designed it to combat complex and dynamic obstacles, it combines the advantages of multiple algorithms, by first providing a suboptimal, local path to start robot navigation while simultaneously searching for a more optimal path in the background and reintegrating new data about the environment. In comparison with other algorithms, this approach proved to be effective, realizing a faster time to target even though initial path computation may provide a suboptimal result [109].

AI-based navigation approaches

In comparison to the approaches presented above, AI-based navigation approaches have potential for more human-like learning in a heuristic approach [76, 106]. The following section will briefly present an overview of the different approaches, which include artificial neural networks, fuzzy logic, and genetic algorithms.

Firstly, using artificial neural networks for navigation has the main advantage of being able to deal well with a variety of factors and sensors, however drawbacks include large amounts of training data, often by driving the vehicle manually first, and error minimization between computed data and required output [76, 106]. Some researchers propose hybrid approaches, where other algorithms are introduced to smooth the generated paths with some success [76].

Next, approaches using fuzzy logic employ semi-states which operate between the classical true and false of Boolean logic and seen extensive use in the field of robot path planning. The main advantage of an approach like this is the heuristic capability of the controller, being able to infer knowledge and output from sensor data even through uncertainties. It's shown by researchers that this can increase the capabilities of an navigation system with respect to dynamic obstacles [76].

Lastly, concept of natural selection in biology inspires the genetic algorithm, which is one of the most viable approaches to generating solutions to pathfinding tasks. By starting with (often random) solutions to the problem, the genetic algorithm will then iteratively optimize, first by generating a population of paths derived from the input using mutation and crossovers, evaluating the solutions by some metrics, e.g. length of paths, travel costs or distance to prospective obstacles, selecting the best fit solutions, and then starting again [76, 106, 107]. In tests done in [107] it has been shown to arrive at a near optimal solution faster than other approaches, while taking slightly longer to reach the optimum, which it is not guaranteed to find [107].