Robust Localization and Navigation for Autonomous, Military Robots in GNSS-denied Environments

Master Thesis Industrial Design Engineering
Ytsen Boersma

Robust Localization and Navigation for Autonomous, Military Robots in GNSS-denied Environments

General information
DPM 1940
Master thesis
Ytsen Boersma | s1926292

Faculty of Engineering Technology
Department of Design, Production & Management

Study programme
Industrial Design Engineering
Management of Product Development

Educational information
University of Twente
Drienerlolaan 5
7522 NB Enschede

Organisation
Royal Netherlands Army
13th Light Brigade
Robotics & Autonomous Systems Cell
Generaal-majoor De Ruyter van Steveninckkazerne
Eindhovensedijk 42
5688 GN Oirschot

Examination date
July 1, 2022

Examination committee

Chair:	Dr. ir. D. Lutters
Supervisor:	Dr. ir. R.G.J. Damgrave
External member:	Dr. B.P. Sullivan
Organisation supervisor:	First lieutenant ir. A.A. van den Broek

UNIVERSITY OF TWENTE.

Royal Netherlands Army

Acknowledgements

I would like to thank my supervisor Artiom van den Broek and David van Tricht from the Robotics & Autonomous Systems (RAS) Cell for giving me the opportunity to perform my graduation assignment within their unit, and for sharing their organisational and technical advice and knowledge during the project.

Additionally, I would like to thank my supervisor Roy Damgrave and Eric Lutters from the University of Twente for their assistance and advice throughout the project.

Finally, I would like to thank Jeroen Lappenschaar from Avalor AI for his feedback.

Summary

The use of Robotic and Autonomous Systems (RAS) is becoming increasingly important for the Royal Netherlands Army (RNLA), as it allows for the effective deployment of scarce human resources. Additionally, the correct application of RAS can lead to increased personnel safety and aid in achieving battlespace supremacy. Consequently, the Robotics & Autonomous Systems Cell was founded with the goal to design and experiment with autonomous, military robots. Through concept development and experiments, the RAS Cell aims to define a list of requirements for the autonomous platform that can be used to outsource part of the development of RAS projects and products.

Localization and navigation are important aspects of autonomous platforms, as robots can only navigate to a specified location when their own location is known. For military use cases, this functionality must be achieved in environments without a usable Global Navigation Satellite System (GNSS) signal, as this can be jammed or spoofed by hostiles. Additionally, the robot's localization and navigation solution must be capable of dealing with multiple deployment environments, changing (weather) conditions and hardware or software failures. These requirements necessitate a robust localization and navigation solution that can be confidently deployed by the RNLA.

Multiple sensors and localization methods have been researched to determine suitable options for the proposed solution. There two main categories in which these two aspects can be divided: active versus passive sensors, and absolute versus relative localization. Each category contains multiple technologies and algorithms, such as stereo vision cameras, 3D Light Detection and Ranging (LiDAR) sensors, Simultaneous Localization and Mapping (SLAM), celestial navigation and magnetic field-based solutions. Additionally, every sensor-localization method combination has its own, different characteristics.

The aforementioned research indicates that there are multiple sensors and localization methods that are suitable for the autonomous platforms. Many of these solutions allow for accurate localization and navigation in unknown environments. However, whether it is due to a lack of precipitation resistance, the use of active sensors, or a lack of obstacle avoidance capabilities, none of these solutions are robust enough as a standalone solution. This is where adaptive sensor fusion can be applied. By integrating multiple sensors and localization methods, the autonomous platform can use the most suitable technologies which are compatible with the current environment and conditions.

There are multiple types of adaptive sensor fusion that can be implemented: hard sensor fusion, soft sensor fusion, and predetermined sensor fusion. Hard sensor fusion analyses the data from active sensors and disables a sensor when its data falls outside a defined range for certain parameters. Soft sensor fusion corrects deviating sensor data instead of disabling the sensor. This provides a more accurate position estimate than hard sensor fusion, at the cost of higher implementation complexity. Finally, predetermined sensor fusion analyses the environment to determine what sensors and localization methods are usable. This is an efficient approach that is easier to implement than soft sensor fusion.

Similar to the researched sensors and localization methods, none of the adaptive sensor fusion frameworks are a suitable option for standalone operation, as each has its own advantages and disadvantages. Hence, the proposed solution must combine multiple variants to fill in the functionality gaps and increase the solution's robustness.

A proposed solution is designed based on the results from the theoretical research and experiments. This solution combines a custom adaptive sensor fusion framework - which combines predetermined sensor fusion with hard sensor fusion - with a behaviour tree for the sensor fusion management and navigation control. The first step of the proposed solution is to select an initial localization method. This is done based on mission parameters defined by a commander, and the deployment environment variables. The autonomous platform then navigates through a list of defined waypoints using this localization method. While navigating, data from the active sensors is continuously analysed to ensure that each active sensor is still usable. If a persistent sensor error is detected, the behaviour tree will pause the navigation process and select a new, usable localization method. This selection is done based on an updated sensor list that takes into account the unusable sensor. After a successful localization method switch, the behaviour tree will continue the navigation process.

The proposed solution was tested in multiple simulation runs with a TurtleBot3, 2D LiDAR and stereo camera for the autonomous platform and sensor configuration respectively. The simulation results indicate that the adaptive sensor fusion framework is capable of consistently detecting unusable sensor data when it is configured correctly. Furthermore, the behaviour tree successfully pauses the navigation process when a persistent sensor data error is detected. After the localization method switch, the behaviour tree also consistently resumed the waypoint navigation process. As a result, the simulated robot successfully reached the final waypoint in all simulation runs.

Based on these results, it can be concluded that the proposed solution provides a solid foundation for an autonomous, military platform. The flexible and modular setup of the behaviour tree allows the design to grow with Project Sentinel as it progresses and aims for a more capable autonomous platform that can be integrated with multiple, different autonomous platforms and sensor configurations.

There are two main areas that can be explored to improve the proposed solution's functionality in the short-term. First, the behaviour tree's implementation can be enhanced through code optimizations and the integration of live human interactions. Second, localization methods tailored towards difficult environments or challenging conditions can be researched and implemented for increased robustness, allowing the autonomous platform to be deployed in a wider range of scenarios.

Table of Contents

Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
AoI	Area of Interest
APE	Absolute Position Error
BT	Behaviour Tree
CV90	Combat Vehicle 90
DARPA	Defense Advanced Research Projects Agency
EKF	Extended Kalman Filter
FSM	Finite State Machine
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HTN	Hierarchical Tree Network
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
ML	Machine Learning
OP	Observation Post
OR	Optimal Route
PCL	Point Cloud Library
PoI	Point of Interest
RADAR	Radio Detection and Ranging
RAS	Robotics and Autonomous Systems
RNLA	Royal Netherlands Army
ROS	Robotic Operating System
RPE	Relative Position Error
RSO	Resident Space Object
SLAM	Simultaneous Localization and Mapping
SoOp	Signals of Opportunity
SP	Starting Point
TIMU	Timing and Inertial Measurement Unit
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UWB	Ultra-Wideband
UXV	Unmanned Vehicle

1 Introduction

1.1 Project origin

The Royal Netherlands Army (RNLA) has created a special cell that focuses on robots, artificial intelligence (AI) and autonomous systems for military applications [1]. This cell is called the Robotics & Autonomous Systems (RAS) Cell. While the RNLA currently uses remote controlled robots, the goal is to transition to autonomous platforms in the future. This transition should lead to the effective deployment of scarce human resources, increased personnel safety and aid in achieving battlespace supremacy [2]. Through concept development and experiments, the RAS Cell aims to gain knowledge on the advantages and disadvantages of different robots and autonomous systems in military applications. With this knowledge, the RAS Cell can provide an advice on the usability of military RAS, which can aid in the transition to autonomous, military robots.

The RAS Cell is currently working on Project Sentinel, with the goal to define a set of functional requirements for autonomous, military robots. This goal is achieved through the development of a virtual, autonomous platform partially driven by AI. Future versions of this platform should be capable of performing multiple mission types, but the current state aims to execute a reconnaissance mission. For this, the autonomous platform should be capable of detecting, identifying and localizing targets in a specified environment. While the reconnaissance robot is expected to execute a mission autonomously, there will always be a human operator that defines the mission and monitors the autonomous platform remotely. This human-in-the-loop approach is applied for legal purposes, to allow for meaningful human control, and to improve the adoption chance of autonomous, military platforms. The latter is an important topic, as the (social) acceptance of autonomous, military robots is debated [3]. It should therefore be emphasized that the human operator will always supervise the autonomous platform, allowing a human to make choices and intervene when necessary.

Based on the knowledge and experience gained during this project, the RAS Cell can define a set of functional requirements that the autonomous (reconnaissance) platform must meet. These requirements can then be used to outsource part of the development, while developing and maintaining other parts in-house. In conclusion, the end goal of Project Sentinel is to become a smart buyer. To achieve this, the RAS Cell must become a smart specifier, which can be achieved by first becoming a smart developer.

Project Sentinel currently focuses on the execution of reconnaissance mission in a forest environment with a single autonomous robot. More specifically, an Unmanned Ground Vehicle (UGV) is used for the autonomous platform. However, in the future, the platform's capabilities should be expandable in (any combination of) these areas:

1. The number and type of autonomous vehicle (e.g., swarming drones).
2. The military capabilities of the autonomous platform (e.g., weaponization).
3. The type of environments in which the autonomous platform can operate.

Increasing the number and types of autonomous vehicles can be used in human supervised autonomous missions where, for example, a group of 10 UGVs and unmanned aerial vehicles (UAVs) are deployed cooperatively. When combined with additional military capabilities, such as weaponization, this group of autonomous vehicles could be capable of engaging a specified target (area). Expanding the type of environments would allow the aforementioned group of UGVs and UAVs to operate in both urban and natural environments. A deployment example for different environment types is a cooperative human-machine mission executed in an urban environment, where UGVs and soldiers move through a city environment together.

Figure 1 illustrates the autonomous platform's envisioned expandability. The current state of Project Sentinel is shown at the origin of the 3D coordinate system. This location represents a single UGV performing a reconnaissance mission in one environment type. Each axis in the 3D coordinate system represents one of the previously mentioned expandability areas. Every location in the 3D coordinate system represents a possible functionality for autonomous, military robots that the envisioned system must be capable of in the future.

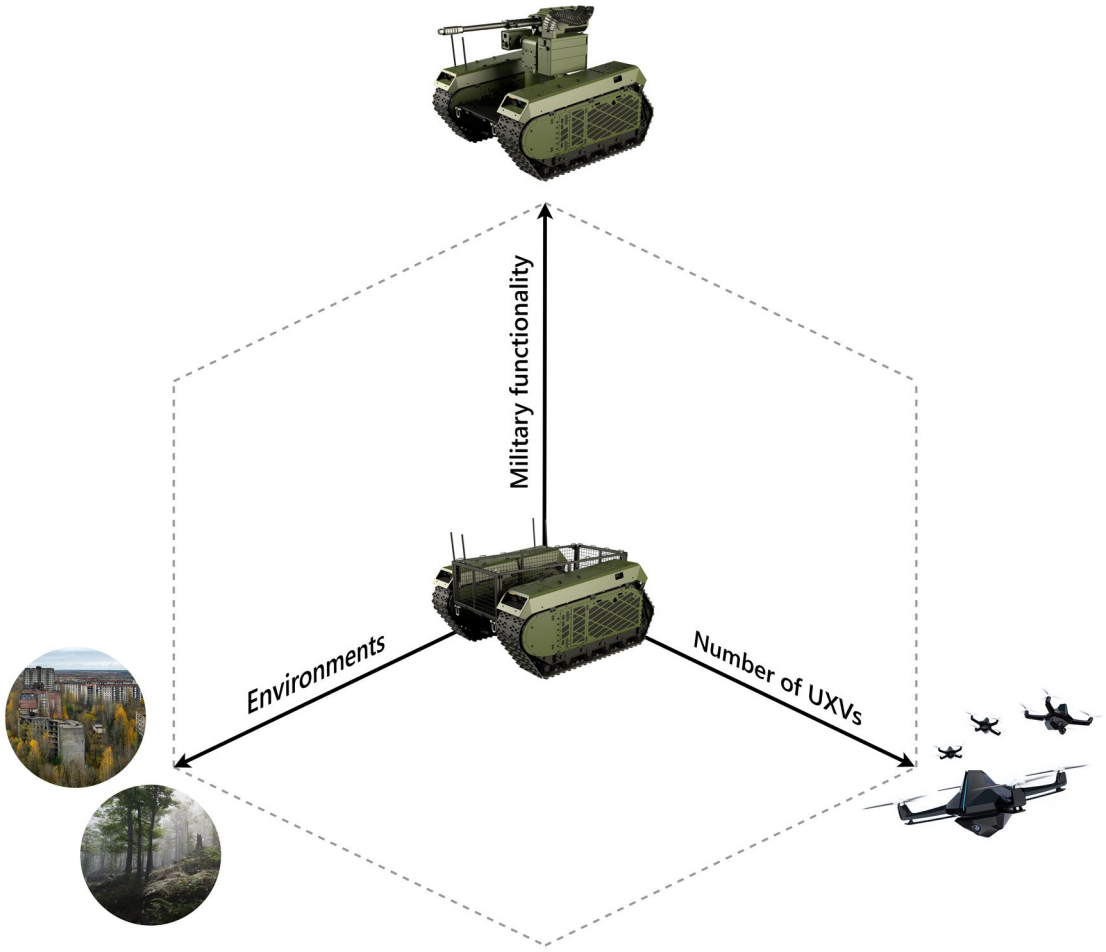


Figure 1: The envisioned expandability of Project Sentinel's autonomous platform.

1.2 Reconnaissance mission example

To provide some context for the deployment of the reconnaissance robot, this section describes the general approach for the execution of a reconnaissance mission. Note that the procedure described below is the current state concept for how a human operator defines a reconnaissance mission. However, due to the ongoing developments within Project Sentinel, this approach could change in the future. For example, future missions could be defined by AI that uses programmed military algorithms and doctrines to create a mission plan. A human operator can then modify or approve this mission plan, maintaining the aforementioned human-in-the-loop approach.

In the first step of the process, a human operator defines the Starting Point (SP) of the reconnaissance robot, an Area of Interest (AoI), and a Point of Interest (PoI). The latter is the observational target (area) for the autonomous platform. Then, an algorithm determines the optimal Observation Post (OP) location, based on the position of the defined PoI. Using the SP and OP, an algorithm then calculates the Optimal Route (OR) for the reconnaissance robot, which is structured in waypoints. These waypoints aim to create traversable trajectories for the autonomous platform. Whether a route is traversable is determined through available geographical information. By following the waypoints, the reconnaissance robot is expected to autonomously navigate from the SP to the OP. During the navigation process, the reconnaissance robot should be able to avoid obstacles and dynamically recalculate its route to the next waypoint when needed. Figure 2 shows a visual representation of the mission setup for a reconnaissance mission.

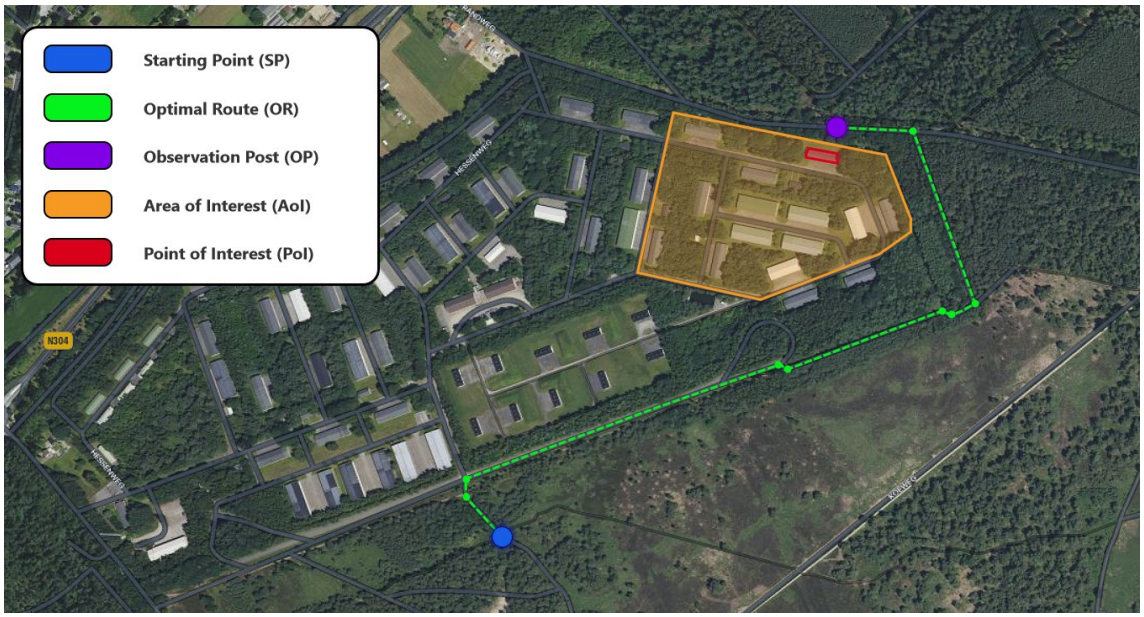


Figure 2: Visual example of the execution setup for an autonomous reconnaissance mission.

1.3 Challenges for autonomous, military reconnaissance robots

There are several challenges that are inherent to the use of autonomous platforms in military use cases that would not be applicable elsewhere. These challenges are defined by the RAS Cell and validated where applicable. The following sections provide an explanation of the challenges and how they build a foundation for the project's structure.

1.3.1 Localization capabilities

Due to the hostile use of Global Navigation Satellite System (GNSS) jamming or spoofing systems [4, 5], the unreliability of GNSS signals in general [6], and the dependency on an external positioning system, it is assumed that there is no usable GNSS signal in the area of operation. Therefore, the autonomous platform should be capable of localization and navigation in GNSS-denied environments. While this assumption improves the platform's robustness, it also results in an increased complexity of the localization system.

Note that the autonomous robot is assumed to operate in GNSS-denied environments, rather than GPS-denied environments. GPS stands for Global Positioning System and it is an American system that is one of the available GNSS solutions. Other examples of GNSS include Europe's Galileo, Russia's GLONASS and China's BeiDou [7]. While GPS is the most common GNSS solution, the autonomous platform's localization and navigation robustness can be improved by assuming that it has no access to any GNSS solution.

Another challenge is presented by the variety of environments that the reconnaissance robot should be able to operate in. These environments can be unpredictable and different from available geographical data. Additionally, certain weather conditions can negatively impact the operation of the localization system. These environmental challenges require a robust localization system for the autonomous platform.

1.3.2 Hardware

The autonomous platform should be able to remain undetected while executing a mission. This provides a challenge for the robot's hardware, as the emitted signals from active sensors could be detected by hostiles, indicating the presence of an autonomous platform and possibly providing a rough location estimate [8, 9]. Therefore, active sensors must be analysed and compared with passive sensors to determine their usability in the field. Subsequently, the proposed solution must implement active sensors accordingly.

1.3.3 Software

To ensure that there is no malicious code or telemetry in the autonomous platform's software, its source code must be accessible to the RAS Cell. This is important for multiple reasons. First, due to the classified and sensitive nature of reconnaissance operations, it is imperative that no data is shared with external entities. Secondly, any data transmitted during a mission can be detected by hostiles, indicating the presence of a possible reconnaissance robot. Another reason not to use black box software is that the RAS Cell must be able to understand and explain the behaviour of the robot. Without access to the source code that controls the robot, this cannot be guaranteed.

Note that the implemented code/software does not have to be open-source. As long as the source code is accessible to the RAS Cell, closed-source or publicly unavailable software can be used too. As mentioned before, part of the autonomous platform's development will be outsourced. If external companies develop custom software for Project Sentinel, they can share the source code with the RAS Cell without publishing or open-sourcing the software.

Finally, for legal reasons, the RAS Cell must be able to understand, explain and justify the behaviour and decisions of the autonomous platform. Therefore, black box systems such as neural networks cannot be used for decision-making processes in the autonomous platform. Consequently, autonomous decisions should be based on mechanisms such as finite state machines (FSMs), behaviour trees (BTs) or hierarchical tree networks (HTNs), in which the origin of a decision can be determined and explained. This requirement of accessible code and understandable behaviour is more important for autonomous platforms than it is for regular vehicles, such as the Combat Vehicle 90 (CV90), as these vehicles do not autonomously make mission-related decisions.

1.4 Problem statement

The current state of Project Sentinel comprises of a simulated robot in both the Robotic Operating System (ROS) and Unreal Engine that can navigate through waypoints with obstacle avoidance based on LiDAR data. The RAS Cell uses simulation environments instead of a physical robot, as this allows for faster iterations in the early phases of the project. Besides the simulated robot, several algorithms have been created that utilize multiple data sources to calculate the aforementioned OP(s) and OR.

To improve the usability of the current state for future deployment, a robust localization and navigation system must be developed that can be deployed in GNSS-denied environments. Additionally, the system must be able to avoid obstacles during navigation and subsequently recalculate its path to the nearest waypoint. Furthermore, the envisioned system is expected to be deployable anywhere, at any time, and on any platform type (e.g., UGVs and UAVs). The latter requires a solution that can operate for prolonged periods of time (more than 24 hours) and at relatively high vehicle speeds (more than 160 km/h) [10, 11]. Finally, the system must be capable of detecting and bypassing sensor failures or other issues that result in unusable sensor data. These problems must be solved while also overcoming the previously described challenges.

Solving the aforementioned challenges and problems is important for several reasons. First, robustness is an important factor for military equipment, which requires redundancy and self-reliance. Second, the RAS Cell can gain knowledge and derive requirements from a proposed solution, which can be used to outsource parts of the development process. Finally, demonstrating new functionalities and capabilities for the autonomous platform is crucial to maintain Project Sentinel, as presented deliverables can be used to get approval for more project resources.

This thesis attempts to solve the challenges and problems defined in section 1.3 and 1.4 respectively, with the result of creating value for Project Sentinel. To achieve this goal, research and validation experiments will be conducted to design a proposed solution. Then, this proposed solution will be tested in simulations to evaluate its usability and robustness. Based on this structure, the following contributions will be delivered:

- Research existing solutions and state-of-the-art methods for GNSS-denied localization systems (including the relevant sensors) and provide an overview of the research results.
- Perform validation experiments to confirm relevant outcomes of the theoretical research and provide an evaluation of the results.
- Research the state-of-the-art methods for adaptive sensor fusion frameworks and provide an overview of the results.
- Define a set of underpinned requirements that the proposed solution must meet, and list the boundaries applicable to the proposed solution.
- Design and present a proposed solution that is based on the outcomes of the theoretical research, with an explanation of the working principles. Furthermore, underpinnings for each aspect of the proposed solution are provided.
- Create a prototype of the proposed solution and perform testing simulations for analysis. Furthermore, the results of these simulations are presented to evaluate the proposed solution's functionality.
- Provide a set of underpinned recommendations for the proposed solution to enhance its robustness and usability for future revisions.

1.5 Mission-oriented command

The RNLA follows the concept of mission-oriented command when giving orders [12]. This concept is centred around the intent of the commander's order. As a result, people executing the command should focus on the envisioned goal and effect of the order, not on the means of how the order should be performed. Therefore, people executing a command are given the freedom in how they execute an order as effectively and efficiently as possible, as long as the commander's intent is achieved. In other words, a commander will communicate *what* he wants to achieve, and the people executing the command will determine *how* the intent is achieved.

The concept of mission-oriented command is also applied to the execution of this project. An end goal is described by the RAS Cell, but an exhaustive list of strict requirements is not provided. Instead, a thorough analysis should result in an overview that presents the most effective and efficient solution to achieve the desired intent.

1.6 Structure

The structure of this thesis is outlined in the sections below, which provide a brief overview of the individual chapters and their contents. In addition, all chapters start with an introduction that provide a more detailed overview of the chapter's contents.

Chapter 2 - Sensors and localization methods
This chapter provides an overview of different sensor types and state-of-the-art methods for localization in GNSS-denied environments. Experiments are conducted to validate research outcomes with little supporting evidence. A list of conclusions is provided to summarize the research. Finally, future technologies for improved GNSS-denied localization will be discussed.

Chapter 3 - Adaptive sensor fusion
Based on the outcome of the previous chapter, three different types of adaptive sensor fusion frameworks are reviewed in this chapter. For each approach, both the advantages and disadvantages are discussed.

Chapter 4 - Requirements and boundaries
This chapter defines an underpinned list of requirements for the proposed solution, as well as several boundaries regarding the autonomous platform's deployment. These are used to define the functionality requirements of the autonomous platform and to limit the scope of the assignment, respectively.

Chapter 5 - Proposed solution
A proposed solution is defined based on the theoretical research and the previously defined list of requirements. This chapter presents the proposed solution and provides an explanation for each of its components. This explanation covers the working principle and reasoning behind each part.

Chapter 6 - Simulation testing
The proposed solution is tested in a simulation environment for accuracy, reliability and viability. First, the simulation setup and results will be discussed. Then, based on these results, a conclusion is provided with a list of recommendations for future improvements.

Chapter 7 - Future expandability
This chapter provides an overview of how the proposed solution can be applied to the envisioned use cases of Project Sentinel. Furthermore, where applicable, explanations are given on how these different deployment scenarios can be exploited to improve the proposed solution's performance and usability.

Chapter 8 - Conclusion
The results from the proposed solution simulations are compared to the original problem statement for this project. Based on the results from this comparison, a conclusion on the usability and quality of the proposed localization system is provided.

Chapter 9 - Discussion and recommendations
In this chapter, the proposed solution is compared to the theoretical research. Moreover, the assumptions made in this thesis are reviewed and further substantiated. Next, non-military use cases for the proposed solution are briefly discussed. Finally, multiple recommendations for future research topics are provided. These topics are aimed at improving the autonomous platform's robustness and usability.

2 Sensors and localization methods

Theoretical research is performed to create a foundation of knowledge for the proposed solution. In this chapter, both existing solutions and state-of-the-art methods for GNSS-denied localization are reviewed. Additionally, experiments are conducted to validate certain outcomes of the theoretical research. Based on the theoretical research and the experiments, a research summary will be provided. Finally, this chapter will provide a section discussing the future technologies for GNSS-denied localization technologies.

2.1 Existing solutions and state-of-the-art methods

Besides existing (commercial) solutions for GNSS-denied localization and navigation, state-of-the-art methods for are also researched and developed in the academic world. The latter might not all be proven, but they can indicate a pattern or possible direction for capable GNSS-denied localization and navigation technologies. This theoretical research is performed to get an overview of possible solutions and their characteristics, which can be used when designing a proposed solution. The upcoming sections discuss the following technologies in order: magnetic field-based solutions, celestial navigation, LiDAR and Radio Detection and Ranging (RADAR) solutions, vision-based solutions, Signals of Opportunity, ad hoc and collaborative solutions, and finally sensor fusion.

2.1.1 Magnetic field-based solutions

The earth's magnetic field can be used for absolute localization and navigation in GNSS-denied environments. The first step in this process is to generate a magnetic field map of the environment, which can be done with magnetometers. Then, an autonomous platform can measure the local magnetic field variations using the same sensor type and compare these with the known map to estimate an absolute position [13].

This concept is used in the AFRL Agile Pod [14], which is based on Magnetic Anomaly Navigation technology. During multiple real-life experiments, this solution achieved a position accuracy of approximately 40 meters with a good map of the environment's magnetic field. Honeywell also offers a magnetic field-based solution, which was successfully tested with an aircraft, though no further data was provided [15].

Raquet et al. [16] used a Honeywell 3-axis magnetometer to create a magnetic field map of an area, which could then be used to localize a ground vehicle with a position error between 100 and 200 meters. For these tests, the mapping, localization and navigation were performed with the same sensor.

The main disadvantage of these solutions is that it requires a magnetic field map of the deployment area. Once such a map is available though, magnetic field-based solutions provide a passive, absolute localization method. Advantages of magnetic field-based solutions include: being globally deployable; operating on both ground and airborne vehicles; operating regardless the time of day or the weather conditions; requires no infrastructure; it is invulnerable to spoofing or interference; and the sensors used for localization and navigation are passive, thus being undetectable by hostiles [14].

2.1.2 Celestial navigation

There are two main types of celestial navigation: star tracking and sky polarization analysis. Both methods use a passive sensor to either track stars and other objects in space, or to measure the local sky polarization.

Honeywell produces a celestial navigation system that detects and tracks stars and other Resident Space Objects (RSOs) to determine a platform's absolute location [17]. Advantages of this technology include: being immune to jamming or spoofing; using passive sensors; and being capable of tracking stars both during the daytime and at night, even with moderate cloud coverage [18]. According to Honeywell's specifications, their celestial navigation system can achieve a position accuracy of 30 meters CEP50 (i.e., 50% of the position estimates are within 30 meters of the true position).

Wei et al. [19] used a star sensor in conjunction with an inclinometer to measure a vehicle's rotation. Combining the data from these two sensors resulted in a position accuracy between 20 and 70 meters during night-time experiments. Similarly, Ning and Fang [20] presented a method that merges star sensor data with an Inertial Measurement Unit (IMU), resulting in position errors of approximately 60 meters. Since both methods rely on star sensors, the position accuracy can vary depending on the environment temperature, sky refraction, weather conditions and any other sky obstructions.

Polaris Sensor Technologies, Inc. offers SkyPASS, a system that is capable of determining a platform's heading, roll and pitch through sky polarization and sun tracking [21]. Sky polarization is a process that takes place when unpolarized sunlight hits the earth's atmosphere, creating a global polarization pattern [22]. The SkyPASS azimuth sensing system is capable of measuring the local sky polarization, which can be compared to the global polarization pattern. When applicable, the system is also capable of sun tracking for sensor redundancy and improved accuracy. According to Polaris Sensor Technologies, Inc., the SkyPASS system is capable of achieving a 0.1 degree heading accuracy and a 0.2 degree roll and pitch accuracy.

Zhang and colleagues [23] demonstrated the combination of a custom sky polarization sensor and an IMU to achieve a position accuracy of approximately 50 kilometres. It should be noted that the aforementioned position accuracy was achieved using a custom sky polarization sensor instead of an existing, commercial product. Additionally, the vibrations from the testing platform impacted the accuracy of the recorded IMU data. Therefore, the achieved position accuracy is limited by the imperfect test setup. The main disadvantage of this localization method is that the position accuracy varies depending on the time of day at which the sky polarization measurements are taken, the presence of sky obstructions, and the temperature of the environment [24].

The advantages of using sky polarization include: it is undetectable due to the use of passive sensors; the technology is immune to spoofing or jamming; the position estimate is driftless; the system is not affected by magnetic disturbances; and it works with civil twilight, making it deployable in urban environments.

2.1.3 LiDAR and RADAR solutions

Light Detection and Ranging, also known as LiDAR, uses the reflected light from emitted infrared (IR) laser beams to determine the distance to a point in space. By performing this measurement for multiple points in space, a 2D or 3D scan of the environment can be generated. This technology can be used for different localization methods such as Simultaneous Localization and Mapping (SLAM) algorithms [25] and odometry [26].

Exyn Technologies and Near Earth Autonomy use 3D LiDAR sensors for localization and navigation in GNSS-denied environments. Both companies use the point cloud data from the LiDAR sensor as input data for a SLAM algorithm [27, 28]. Benefits of the 3D LiDAR SLAM technology include driftless localization within a generated map, and loop closure for previously scanned locations. Loop closure is a concept in robotics where an autonomous platform recognizes a current location as a previously visited location, based on the current sensor data and the previously generated environment map. This knowledge is then used to correct the position estimate of the robot. These advantages, combined with the accurate LiDAR data, allow for a position error of at most 10 centimetres [29]. The downside of using 3D LiDAR sensors is that it has high storage requirements. According to Exyn Technologies, approximately 3GB of storage is required for each minute of collected point cloud data [29].

Raw LiDAR point cloud data can be converted to a semantic segmentation map of the environment. Chen et al. [30] implemented this technology to augment a LiDAR-based SLAM algorithm. Their solution achieves a position error of up to approximately 2% for each 100 meters travelled. Similarly, Wang et al. [31] utilize semantic segmentation to simplify a point cloud map generated by a LiDAR-based SLAM algorithm. When a vehicle travels through a previously mapped environment, the raw LiDAR data is converted to semantic information which can then be compared to the existing semantic map.

LiDAR data can also be used to compare the environment with OpenStreetMap data. Suger and Burgard [32] use a 3D LiDAR sensor and semantic segmentation to create a semantic map of the environment. This map is then compared to the terrain classification data from OpenStreetMap, which results in an approximate position error of 1.5 meters. The position accuracy can vary depending on the quality of the semantic segmentation process. Wang and colleagues [33] extracted line features from 3D LiDAR data and compared these to the footprints of buildings from OpenStreetMap. This method is capable of an approximate position accuracy of 1.0 meters in urban environments.

A similar type of sensor is RADAR. Instead of using infrared laser beams, it emits radio waves that are reflected by objects. The distance, velocity and angle of objects can be determined by analyzing the difference between the emitted wave and the reflected wave. Honeywell produces a RADAR system that can be used for autonomous platforms in GNSS-denied environments [34]. Their technology uses radio waves that are invulnerable to weather conditions such as rain, fog, dust and snow. Position errors of this method are approximately 2% of the distance travelled.

Quist and colleagues [35] used RADAR data with a recursive-RANSAC algorithm for radar odometry. Their solution achieved a relative position accuracy of approximately 3.4% of the distance travelled. Similarly, Scannapieco et al. [36] used RADAR odometry to achieve an approximate position error of 3.0 meters while maintaining real-time performance.

2.1.4 Vision-based solutions

Matching ground imagery with aerial/satellite images can be used to determine a platforms absolute position. For this solution, the ground imagery can be recorded with both optical and infrared cameras. Honeywell, Maxar and Lockheed Martin all produce vision-based localization and navigation systems for airborne platforms in GNSS-denied environments that are based on this technology [37, 38, 39]. Based on real-world experiments performed by Honeywell and Maxar, this solution is capable of achieving a position accuracy of 10 meters CEP50.

Viswanathan et al. [40] used panorama images captured by a UGV for localization. These panoramas are altered to emulate a top-down perspective, after which they are compared to satellite imagery from Google Maps (and Google Street View when possible). This method results in a position accuracy of approximately 10 meters. Patel [41] combined the aforementioned technology with IMU data for attitude estimations, which leads to an approximate position accuracy of 4 meters. Additionally, this method achieved real-time performance running on an NVIDIA Jetson TX2 module. Similarly, Floros et al. [42] combined OpenStreetMap data with vision data and visual odometry to achieve a position accuracy of approximately 5.2 meters in urban environments.

The advantages of ground-to-satellite imagery matching include: immune to spoofing or jamming, based on passive sensors, and it can be applied in different weather conditions. Disadvantages of the method include: not usable in low-light environments, and not functional when there are sky obstructions, such as a tree canopy.

Another vision-based solution is demonstrated by SRI International, who used cameras for a visual SLAM algorithm. Their solution uses vision data to detect landmarks, which are then mapped in a 3D map of the environment [43]. Similar to the LiDAR-based SLAM algorithms, the vision-based SLAM method is capable of detecting loop closures for improved accuracy. According to SRI International, their solution is capable of creating 3D maps of the environment with centimetre level accuracy [44].

Fan et al. [45] extended a visual SLAM algorithm with semantic segmentation capabilities. The semantic segmentation process allows for dynamic objects to be filtered from the localization process. The resulting scan only includes static objects, which improves the algorithm's position accuracy. The proposed method achieves a position accuracy of approximately 1.0 meter in an indoor environment. Similarly, Xiao and colleagues [46] used a semantic visual SLAM algorithm for outdoor experiments, which resulted in a maximum position error of 30 meters. When loop closure is detected, the position accuracy can be significantly increased to approximately 2.0 meters.

Visual odometry is a basic vision-based localization method that identifies and tracks features across multiple camera frames [47]. By analyzing the different position of corresponding features across frames, the platform's updated position can be estimated. This method provides a relative position that will drift over time, similar to inertial odometry and wheel odometry. Consequently, the position estimate will become less accurate over time. Hence, visual odometry itself is not suitable for accurate navigation.

Yao et al. [48] used an RGB-D camera for the visual odometry camera, achieving a maximum position error of approximately 1.0 meter with real-time performance. Badshah and colleagues [49] improved the stock visual odometry method with a modified normalized phase correlation, which improves the algorithm's localization performance in environments with less textures. The presented method achieved a maximum position error of approximately 9 meters, where the biggest part of the position error is due to GPS location jumps (GPS data was used to measure the ground truth as comparison).

As mentioned before, the main disadvantage of visual odometry is the drift rate that leads to an increased position error over time. Additionally, visual odometry algorithms perform worse in dark environments and in environments with moving objects. The main benefits are that it is immune to jamming and spoofing, and that it is undetectable due to the exclusive use of passive sensors.

2.1.5 Signals of Opportunity

In areas where GNSS signals are not usable, other radio frequency signals can still be used for localization and navigation. This technique is called Signals of Opportunity (SoOP), and it uses existing radio frequency waves from satellites, cellular towers, Wi-Fi transmitters, television communication towers and other sources. BAE Systems has developed NAVSOP: Navigation via Signals of Opportunity, which is based on the SoOP technology [50]. Specifications or validation data are not provided.

Souli et al. [51] used SoOP with multiple frequency bands, resulting in a position error of approximately 150 meters. Similarly, Ismail and colleagues [52] used LTE cellular towers and wheel odometry to determine a platform's position. The achieved position error was approximately 13 meters over a trajectory of approximately 4,900 meters. Singh and Sujit [53] also used cellular towers with a 100-meter communication range, which resulted in a position accuracy of 50 meters. The proposed solution combined dead reckoning with the cellular SoOP. Page and Wickramarathne [54] used 4G LTE transmitters with a range of one kilometre. With the transmitters located 500 meters apart, a position error of 50 meters was achieved.

Note that SoOP can provide a relative or absolute position estimate, depending on whether the location of the used radio frequency transmitters is known. When these positions are unknown, SoOP will provide a relative position, since the algorithm has to estimate the position of both the transmitters and the autonomous platform itself. Subsequently, the relative position estimate obtained from SoOP is less accurate than the absolute position estimate.

The main benefit of SoOP is that it requires no additional infrastructure, since it uses existing sources for the localization and navigation process. Consequently, the autonomous platform only needs a passive receiver capable of detecting radio waves with a variety of frequencies [55]. Due to the passive receiver, SoOP localization systems cannot be detected by hostiles. The main drawback of SoOP is that there is no guaranteed position accuracy, due to the variety of available signals at different locations. Furthermore, the quality and general availability of these radio frequency signals is not guaranteed, which can result in varying position errors [56].

2.1.6 Ad hoc and collaborative localization

Creating an alternative positioning network to GNSS with a private infrastructure and a network of multiple devices, or nodes, is known as ad hoc localization or collaborative localization. Robotic Research's WarLoc solution is based on this technology, and it aims at providing position data to military personnel in GNSS-denied environments [57]. By combining the communication signals from multiple users with IMU data, WarLoc is able to provide a relative position of all users in the network [58].

Similarly, TRX Systems, Inc. has developed an ad hoc localization solution known as NEON. This system combines IMU data with Ultra-Wideband (UWB) signals to communicate between different users in the ad hoc network [59]. The system's position accuracy can be improved by deploying UWB beacons with a known position in the environment [60]. Deploying UWB beacons during operation can also be used to extend the system's localization range.

Another ad hoc localization method is based on a support node outside a GNSS-denied environment that communicates with a vehicle inside the GNSS-denied environment. Russell et al. [61] applied this method with two drones, where one drone is within a GNSS-denied environment. The second drone is located outside the GNSS-denied environment, allowing it to determine its absolute position. By communicating between the two drones and using Direction of Arrival (DOA) measurements, the drone in the GNSS-denied environment is capable of absolute localization with a position error up to approximately 500 meters. This accuracy was achieved with a distance of 1.38 kilometres between the two drones. Similarly, Zhang and colleagues [62] used a reference device with a known location for collaborative localization, achieving a position error of approximately 2 meters in an indoor experiment.

A comparable localization method is presented by Ma and colleagues [63], where three transmitters with known locations are used to localize a platform in a GNSS-denied environment. This ad hoc method achieved a position error of approximately 20 meters with an average distance between the transmitters of 150 kilometres. For this solution, the transmitters can be placed within the GNSS-denied environment, as long as their exact location is known.

There are several advantages and disadvantages for ad hoc or collaborative localization [64]. The main advantage is that the aforementioned localization methods can be used under any weather conditions and during any time of day, since the localization process is based on radio frequency signals. The disadvantages of ad hoc or collaborative localization include: active radio emitters are used to communicate between the different nodes in the network, which increases the detectability of the system; and a private infrastructure is required, which makes the solution less flexible and more cumbersome to setup. While the required infrastructure is a disadvantage for when this localization method is used with a single autonomous platform, it can be advantageous in the future where multiple autonomous robots are cooperatively executing a mission.

2.1.7 Sensor fusion

Sensor fusion is a concept where multiple sensors are combined to perform a certain (set of) task(s). Using multiple sensors on autonomous platforms can increase the accuracy of localization and navigation, provide redundancy, and allow the system to operate in different environments and scenarios. An example of sensor fusion is the autonomous shuttle from Unmanned Systems Lab. This platform uses a 3D LiDAR sensor, a stereo camera, an IMU and a GPS sensor [65]. With this sensor suite, the autonomous shuttle is capable of obstacle detection, object classification, localization and navigation with a global positioning accuracy of approximately 5.0 centimetres.

Similarly, Starship delivery robots combine RADAR, GPS, cameras and ultrasonic sensors to perform edge detection, obstacle avoidance, object classification and dynamic path planning [66]. Like the previously discussed autonomous shuttle, the disadvantage of Starship's robots is that they still require a GPS signal for localization and navigation. Another disadvantage is the increased computational requirements, as there is more sensor data that needs to be processed.

Oshkosh Defense has developed a sensor fusion platform that is capable of GPS-denied localization and navigation for 10 kilometres [67]. The system is equipped with a high-definition LiDAR sensor, 12 short range RADAR sensors, three long range RADAR sensors, a wide dynamic range camera, four situational awareness cameras, and a short-wave infrared camera [68]. These sensors allow the system to operate in all weather conditions and at all times of day. The position accuracy is unknown, nor is the reason behind the 10 kilometre distance limit.

Wang and colleagues [69] presented a sensor fusion strategy combining a LiDAR-based SLAM algorithm with vision-based optical flow measurements and IMU data. This combination allows for accurate indoor navigation with low computational requirements. Similarly, Shi et al. [70] demonstrated the combination of a panoramic camera with a 2D LiDAR to achieve a position accuracy of approximately 0.2 meters over a 580-meter trajectory. This solution uses the LiDAR data to improve the scale accuracy of the vision data and for loop closure detection. The latter can be used to reduce position errors as previously visited locations are revisited.

Another sensor fusion example that combines LiDAR data with vision data is presented by Viswanathan et al. [71]. The presented solution creates a semantic segmentation map of the environment using LiDAR and vision data, which can then be compared to satellite imagery for localization. During experiments, this method achieved a position accuracy of approximately 5.4 meters. Maturana and colleagues [72] presented a similar sensor fusion method where LiDAR and vision data are combined for semantic mapping. Despite not providing any position accuracy data, the researchers did state that their method was capable of determining optimal routes through off-road terrain and detecting loop closures while navigating.

Mostafa and colleagues [73] presented a combination of visual odometry, RADAR odometry, and IMU data capable of achieving a position error of approximately 5 meters. The presented method can be performed in real-time on a drone's onboard computer, indicated low computational requirements. Additionally, the visual odometry and RADAR odometry algorithms can also be used in a standalone setup, in case of sensor failure.

2.2 Active sensor detectability

As mentioned in section 1.3.2, using active sensors in hostile environments can be a risk. The signals emitted by active sensors can be detected by hostiles, which could lead to the detection of the autonomous platform itself [74]. Detection should be avoided at all cost, as it would compromise the (reconnaissance) mission and most likely lead to the loss of the autonomous platform. This section analyses the detectability of different active sensors that are commonly used in the solutions presented in the previous sections. Both theoretical research and experiments are performed to determine the detection risk. This risk is expressed as a binary chance of being detected by hostiles at the autonomous platform's envisioned reconnaissance distance (approximately 1,000 to 2,000 meters). In other words: an active sensor will either be defined as detectable (i.e., high detection risk) or undetectable (i.e., low detection risk) at reconnaissance distances.

2.2.1 LiDAR detectability

A common sensor type used in SLAM-based localization methods is LiDAR. These sensors emit infrared lasers in a 2D or 3D pattern and measure the reflected light to determine the distance to objects in the sensor's surroundings. These measurements can then be visualized in a 2D or 3D point cloud, and used in a SLAM algorithm. However, the emitted laser beams can also be detected by hostiles.

Common LiDAR sensors use infrared laser beams with a wavelength of approximately 900 nm [75]. These laser beams can be detected by police LiDAR detectors, as these are sensitive to laser beams with a similar wavelength [76]. Similarly, there are laser warning systems for military applications that are capable of detecting IR lasers from laser-guided missile systems [77, 78]. The sensors in these warning systems are capable of detecting (IR) laser beams with a wavelength between 400 nm and 1600 nm [79], which includes the wavelength emitted by LiDAR sensors.

Visually, LiDAR sensors can be detected at night using image intensification night vision equipment. Ford published a video in which multiple 3D LiDAR sensors are mounted to an autonomous vehicle that drives around a track at night [80]. When looking through the night vision goggles, both the source of the infrared laser beams and the resulting scanning pattern were clearly visible to the user (Figure 3). Because the emitted pattern itself is visible with night vision equipment, the LiDAR sensor can be spotted even without a direct line of sight. Note that the 3D LiDAR scans the environment using a rotational scanning cycle. This cycle is performed multiple times per second, which results in a nearly continuous visibility of the LiDAR sensor and the emitted pattern.

Figure 3: LiDAR visibility in a screenshot taken from Ford's video showcasing autonomous driving at night.

While the night-time visibility of LiDAR sensors is a commonly showcased characteristic, the daytime visibility is not discussed. Given that the proposed solution should be able to operate during the daytime, it is important to analyse this characteristic for undetected operation. Hence, to analyse the visibility of LiDAR sensors during the daytime, an Eken H9R action camera was modified by removing the infrared filter from the lens. This allows the camera to detect and record both visible and infrared light. A Milrem THeMIS equipped with two Velodyne Puck 3D LiDAR sensors was recorded using this camera. Figure 4 displays the visibility of the LiDAR sensor during the daytime when using the modified Eken H9R. More images of this experiment can be found in Appendix A.

Figure 4: Daylight visibility of a Velodyne Puck 3D LiDAR as recorded by the modified Eken H9R.

During the experiment, the LiDAR sensors were only visible from short distances (up to approximately 8 meters). Note that only the source of the emitted laser beams was visible, not the emitted pattern. Additionally, due to the rotational scanning procedure of the LiDAR, the sensor is not continuously visible, but only for short periods of time. This reduces the detectability of the LiDAR sensor, both during the daytime and at night.

Based on the experiment and online videos, it is assumed that LiDAR sensors are not visibly detectable during the daytime at reconnaissance distances. However, due to the lack of ambient light and the hostile use of night vision equipment, LiDAR sensors are assumed to be visibly detectable at night, even at reconnaissance distances.

2.2.2 RADAR detectability

Another active sensor type used in autonomous vehicles is RADAR. Its functionality is similar to a LiDAR sensor, although it emits radio waves instead of infrared laser beams to scan its surroundings. RADAR can be used with different signal wavelengths, as well as different power levels for the emitter [81]. Despite being invisible, emitted RADAR signals can still be detected. One example is found in fighter jets, where a fire control RADAR is used to lock-on to targets [82]. The targeted aircraft can detect the fire control RADAR signal due to its characteristics, indicating that a hostile fighter jet is within firing range. The common wavelength for fire control RADAR signals lies in the X (8 to 12 GHz) radio frequency band [83]. Police RADAR detectors are another example of how RADAR signals can be detected. Similar to the police LiDAR detector, these devices can be used to detect speed measurements taken by law enforcement. Typical police RADAR detectors are able to detect different wavelength signals in the X (8 to 12 GHz), K_a (12 to 18GHz) and K (18 to 27 GHz) radio frequency bands [83, 84].

2.2.3 Depth camera detectability

Certain depth cameras emit an IR pattern and analyse its deformation to increase the accuracy of the depth scan. This technology is also known as structured light, and it is used in depth cameras such as the Intel RealSense and the Microsoft Xbox Kinect sensor. While improving scan accuracy, structured light also increases the sensor's detectability.

The modified Eken H9R camera from the LiDAR experiment was used to analyse the visibility of the emitted infrared pattern. For the depth camera, an Intel RealSense D435i was used. The structured light pattern was recorded during the daytime from multiple distances and angles to determine the sensor's detectability. Additionally, the experiment is conducted in a forest environment to emulate a realistic mission scenario.

At ~3 meters, the emitted infrared light is clearly visible when the Eken H9R is pointed directly at the depth camera's infrared emitter. Increasing the distance to ~7 meters results in a similar detectability. At approximately 12 meters, the emitted infrared light is still visible, albeit less noticeable (Figure 5). When the Eken H9R is not directly pointed at the depth camera, the emitted IR light is not visible, regardless of the distance. Moreover, the emitted pattern is invisible to the infrared camera, regardless of the distance or angle. The raw images taken from all three distances can be found in Appendix B.

Another factor that influences the detectability is the presence of natural obstructions. These can block the depth's camera emitted infrared light, even if the infrared camera is pointed directly at the depth camera. Therefore, natural environments with vegetation decrease the detectability of infrared-aided depth cameras.

Figure 5: Image captured of the Intel RealSense D435i's emitted infrared light at approximately 12 meters.

In addition to the experiment, there are multiple online videos showcasing the visibility of the emitted infrared pattern in dark environments [85, 86]. These videos indicate that the detectability is significantly higher at night, even at longer distances and different angles. An example of this can be seen in Figure 6, where the emitted pattern from an Xbox 360 Kinect is recorded in a dark room using an infrared camera.

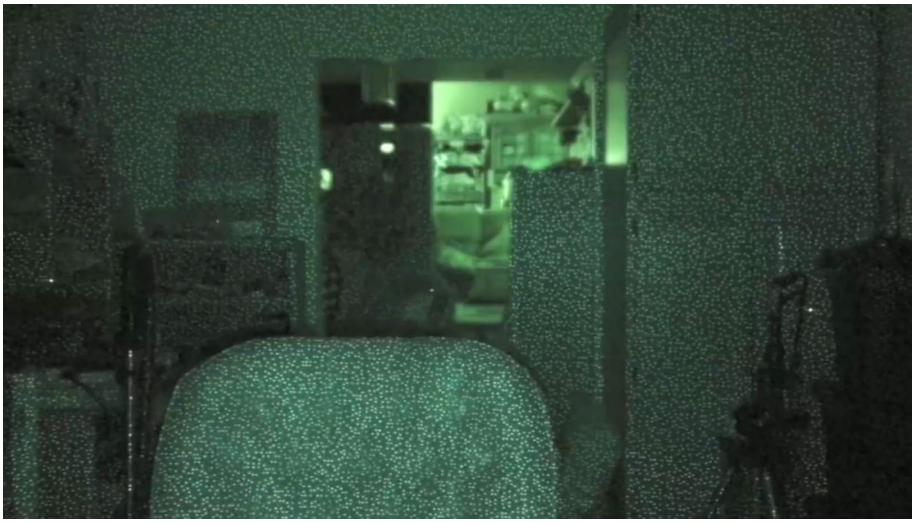


Figure 6: Visibility of the infrared dot pattern emitted by an Xbox 360 Kinect in a dark room.

Based on the experiment and online videos, it is assumed that infrared-aided depth cameras are not visibly detectable during the daytime at reconnaissance distances. However, it is assumed that infrared-aided depth cameras are visibly detectable at night, even at reconnaissance distances.

2.3 Precipitation resistance of vision-based localization methods

Multiple sources from the state-of-the-art methods stated that vision-based localization methods are susceptible to reduced performance or complete failure when used during precipitation. However, none of these sources verified the negative effect of precipitation on vision-based localization methods with proper A/B testing. The reason for this could be the difficulty of controlling all variables throughout multiple real-world experiments. Therefore, the goal of this section is to analyse the robustness of vision-based localization methods during rain and other types of precipitation, using simulations for consistent A/B testing. Multiple simulations with two different vision-based algorithms were performed in MathWorks MATLAB. The Simulink package is used to combine the aforementioned algorithms with a 3D environment rendered in Unreal Engine 4.

The first set of simulations uses a visual SLAM algorithm based on feature extraction to calculate an estimated position of a drone as it navigates through a city environment. This position estimate is the evaluated performance metric for the first set of simulations. The second simulation type uses a semantic segmentation algorithm for object detection, classification, and object masking of monocular camera data captured from a car moving through a city environment. This second set of simulation runs is not focused on navigation accuracy, but on the performance of the semantic segmentation process (i.e., are objects correctly identified and masked under different weather conditions).

The cloud density, fog density, and rain density settings in the Unreal Engine environment were gradually increased throughout the simulations to determine an approximate precipitation resistance of both algorithm types. Modifying these settings also impacts the lighting conditions of the scene. The weather parameters were the only variables that changed between the different simulation runs. This ensures that observed differences are the result of the different weather conditions. While the Unreal Engine environment cannot achieve the same level of realism achieved in real-world experiments [87], it does provide a consistent simulation environment where the testing variables can be controlled. An example of the realism gap is the presence of raindrops on a camera lens, which can distort or obfuscate (part of) the image. This will not occur in the simulated environment, while it is a valid concern in real-world applications.

The hypothesis for these simulations is that semantic segmentation algorithms are more resistant to precipitation than feature extraction algorithms. This hypothesis is based on the academic research, the different working principles of the two algorithm types, and the current application of vision-based algorithms in autonomous vehicles.

2.3.1 Feature extraction simulations

The visual SLAM simulations are based on MATLAB's built-in 'Stereo Visual SLAM for UAV Navigation in 3D Simulation' example. In this simulation, a drone manoeuvres around two city blocks after which it ends near the starting position. The trajectory length for this simulation is approximately 550 meters. The drone's position estimate is determined using a stereo camera and a visual SLAM algorithm based on ORB-SLAM [88].

The first set of simulations emulates a light drizzle or fog, where the general visibility is reduced without there being visible, distinct raindrops. A second set of simulations is used to emulate varying raining conditions, from light rain to heavy rain. The results of the visual SLAM simulations can be seen in Appendix C. Here, two figures are presented for each simulation. The results from one of the simulation runs is shown in Figure 7.

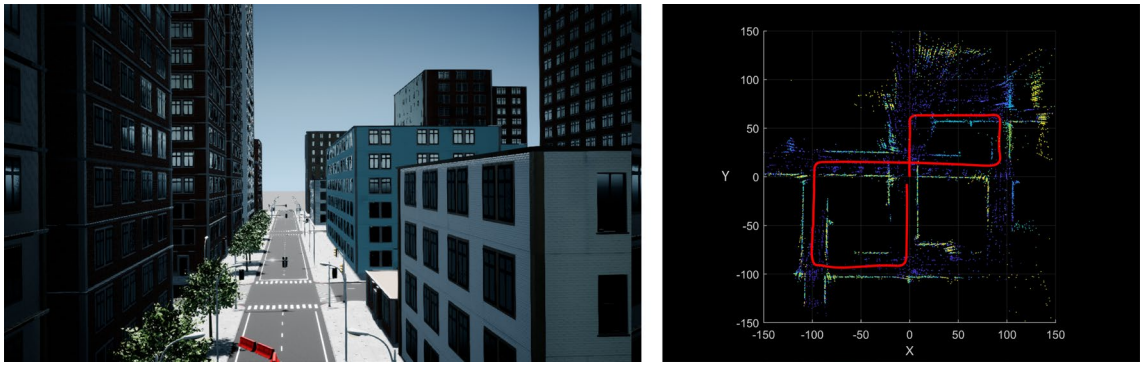


Figure 7: Results from one of the visual SLAM simulations without visible raindrops.

The first image (seen on the left) is a raw image capture from the drone's stereo camera. This image can be used to compare the weather conditions between the different simulation runs. The second image (seen on the right) shows a graph with the estimated position of the drone. This graph can be used to evaluate the performance of the visual SLAM algorithm in the different weather conditions. The results from the light drizzle/fog simulations indicate that the feature extraction algorithm is still capable of estimating the drone's position under these conditions, albeit with reduced accuracy. Additionally, the visual SLAM algorithm did not suffer from a noticeable increase in computation times in the light drizzle/light fog circumstances.

Setting the rain density to 10% was enough for the feature extraction algorithm to fail. In this scenario, the algorithm was unable to match a sufficient number of features between frames while maintaining real-time performance. This failure occurred after approximately 170 meters. Increasing the rain density to 30% resulted in the visual SLAM algorithm failing almost instantly after starting the simulation.

2.3.2 Semantic segmentation simulations

The semantic segmentation simulations are based on MATLAB's built-in 'Depth and Semantic Segmentation Visualization Using Unreal Engine Simulation' example. In this simulation, a car partially drives around one city block with an unknown trajectory length. While driving around, the monocular camera captures data which is used for the semantic segmentation algorithm.

Similar to the visual SLAM methodology, the first set of simulations emulates weather conditions similar to a light drizzle or fog, where the general visibility is reduced without there being visible, distinct raindrops. Next, a set of simulations was performed that emulated varying raining conditions, from light rain to heavy rain.

The full results of the semantic segmentation simulations can be seen in Appendix D. Two figures are presented for each simulation run. The results from one of the simulation runs is shown in Figure 8. The first image (seen on the left) is a raw image capture from the car's monocular camera. This image can be used to compare the weather conditions between the different simulation runs. The second image (seen on the right) shows the corresponding image with coloured masks for detected object types. This image can be used to evaluate the performance of the semantic segmentation algorithm in the different weather conditions.

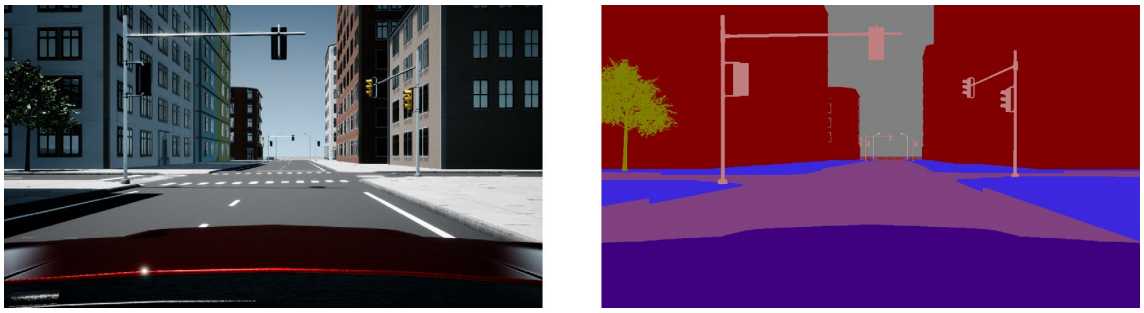


Figure 8: Results from one of the semantic segmentation simulations without visible raindrops.

The results from the first set of simulations indicate that the object detection, classification and masking performance of the semantic segmentation algorithm is not impacted by a light drizzle or a light fog. Both the semantic segmentation performance and the computation times were unaffected by the cloud and fog density settings, allowing the simulations to run with real-time performance.

Enabling rain did not reduce the performance of the semantic segmentation process, although it did result in marginally higher computation times. This effect cannot be deduced from the images, but it was noticeable while running the simulations. Despite this increase in computation times, the algorithm was capable of running the simulation with real-time performance.

2.3.3 Precipitation resistance conclusion

Based on the results from the simulations, it is assumed that visual SLAM and other visual feature extraction-based algorithms cannot be used when there are visible, distinct raindrops. During light drizzle/fog conditions, visual feature extraction-based methods can still function, although their accuracy will be reduced. Therefore, under these circumstances, the use of sensor fusion or other sensor types are advised.

Using semantic segmentation algorithms based on visual data can be applied in varying weather conditions, even with heavy precipitation. It is assumed that the accuracy of these algorithms will not decrease under the aforementioned circumstances, although their computation times are expected to increase.

These two conclusions are in accordance with the hypothesis for these experiments, which stated that "*semantic segmentation algorithms are more resistant to precipitation than feature extraction algorithms*".

2.4 Research conclusion

To summarize the research on sensors and localization methods, two tables have been created that can be used in the decision-making process for the proposed solution. The first table contains a list of verified state-of-the-art methods and contains all localization methods with their used sensor(s), average position error, and other notes (Appendix E). The second table combines the data from both existing solutions and state-of-the-art methods. It contains the advantages, disadvantages, and the Absolute Position Error (APE) for the main localization method categories (Table 1).

Categories	Main advantages	Main disadvantages	APE (m)
Magnetic field-based solutions	• Provides a drift-free, absolute position • Not impacted by the environment • Uses passive sensors	• Requires a magnetic field map of the deployment area	50
Celestial navigation	• Provides a drift-free, absolute position • Uses passive sensors • Immune to hostile interference	• Negatively impacted by sky obstructions and other environmental conditions • Not guaranteed to function during the daytime	30
LiDAR and RADAR solutions	• Drift-free position accuracy • Loop closure capabilities • Not impacted by the weather • Operates at all times of day • Can be used to create a 3D map of the environment	• Uses active sensors • Requires relatively high computational power • High storage requirements • Vulnerable to hostile interference	1.0
Vision-based solutions	• Uses passive sensors • Loop closure capabilities • Can be used to create a 3D map of the environment • Ground-to-satellite imagery matching can provide a drift-free absolute position	• Vulnerable to hostile interference • Not usable in dark environments • Negatively impacted by precipitation and certain environmental conditions	1.0
Signals of Opportunity	• Drift-free position accuracy • Uses passive sensors • Not impacted by the weather • Operates at all times of day	• Usable signals are not guaranteed without a private infrastructure • Varying position accuracy	50
Ad hoc and collaborative localization	• Not impacted by the weather • Operates at all times of day	• Requires an infrastructure • Based on active sensors	20
Sensor fusion	• Sensor data redundancy • Can be used to create a 3D map of the environment	• Partially uses active sensors • Partially vulnerable to hostile interference	0.5

Table 1: The main localization methods and their characteristics.

Additionally, the following main points have been learned during the research:

- There is no single sensor-localization method combination that can be used to achieve GNSS-like position accuracy in any environment or under any conditions.
- Active sensors, such as LiDAR and RADAR, should be implemented with care, as their emitted signals are proven to be detectable with the right equipment.
- Relative localization methods will always contain a drift in the estimated position, leading to an increasingly large position error over time.
- Detecting loop closures can result in lower position errors, but their detection is not guaranteed. For example, when using cameras, loop closures can remain undetected when approaching the same location from different directions. Furthermore, varying weather conditions can result in a different image, which can also lead to undetected loop closures. Finally, physical environment changes between visits can result in undetected loop closures.

2.5 Future technologies

While this project focuses on the implementation of existing solutions and state-of-the-art methods for localization in GNSS-denied environments, future technologies are constantly being developed and researched. If these technologies improve upon existing solutions, then they could be implemented in a future version of Project Sentinel. Hence, the following sections discuss technologies that are currently being researched and that could lead to improved GNSS-denied localization performance in the future.

Quantum-based IMU

The data generated by existing IMUs contain a drift that can lead to an increasing localization error over time. The Quantum-Assisted Sensing and Readout programme of the Defense Advanced Research Projects Agency (DARPA) is researching quantum-based IMUs that provide drift-free inertial measurements as a result of atomic properties [89]. Consequently, IMU-based localization methods can experience an increase in localization accuracy, particularly over longer distances.

Combined timing and IMU chip

In addition to the quantum-based IMUs, DARPA is researching the use of a single-chip Timing and Inertial Measurement Unit (TIMU) [90]. These TIMUs contain a clock, three gyroscopes and three accelerometers in a single chip. As a result of this combination, a TIMU should be able to provide better inertial and timing accuracy than existing IMUs, while requiring less power. Similar to the quantum-based IMUs, this solution should reduce the localization error of the system.

Micro-scale gyroscopes

Finally, Micro-Scale Rate-Integrating Gyroscopes are part of DARPA's current research. The goal is to develop micro-scale, high-performance gyroscopes for accurate inertial navigation [90]. Once the technology is developed, it could be implemented into existing IMUs. This would lead to better angular measurements from the IMU, resulting in more accurate localization in scenarios where vehicle rotations are common.

3 Adaptive sensor fusion

Based on the research on sensors and localization methods, it can be concluded that there are no single sensor-localization method combinations that can accurately and robustly localize a robot in GNSS-denied environments. Therefore, sensor fusion must be utilized to fill in the functionality gaps of certain sensors and localization methods. However, to manage varying environments, new mission parameters and sensor failures, the implemented sensor fusion framework must be capable of dynamically changing its configuration. Therefore, this chapter provides an overview of the state-of-the-art solutions for adaptive sensor fusion frameworks, which can be used in a later stage to design an adaptive sensor fusion framework for the reconnaissance robot. Hence, the goal of this research is to identify usable (parts of) adaptive sensor fusion frameworks for Project Sentinel's use case that can be (partially) implemented in the proposed solution.

3.1 Hard sensor fusion

Kobayashi and colleagues [91] presented a sensor fusion framework that combines sensor characteristics with the data produced by each sensor to determine what sensor (data) should be included in the sensor fusion process. If the sensor data falls outside a specified range, or does not meet the expected characteristics, then this sensor is completely ignored in the sensor fusion process. Due to the binary nature of this model, it is also referred to as 'hard sensor fusion'. Similarly, Cohen and Edan [92] presented a rule-based framework where the most reliable sensors are selected in combination with the simplest sensor fusion algorithm. This model uses the data from all sensors as an input for the localization method and sensor selection steps, in which the active sensor(s) and localization method will be determined (Figure 9).

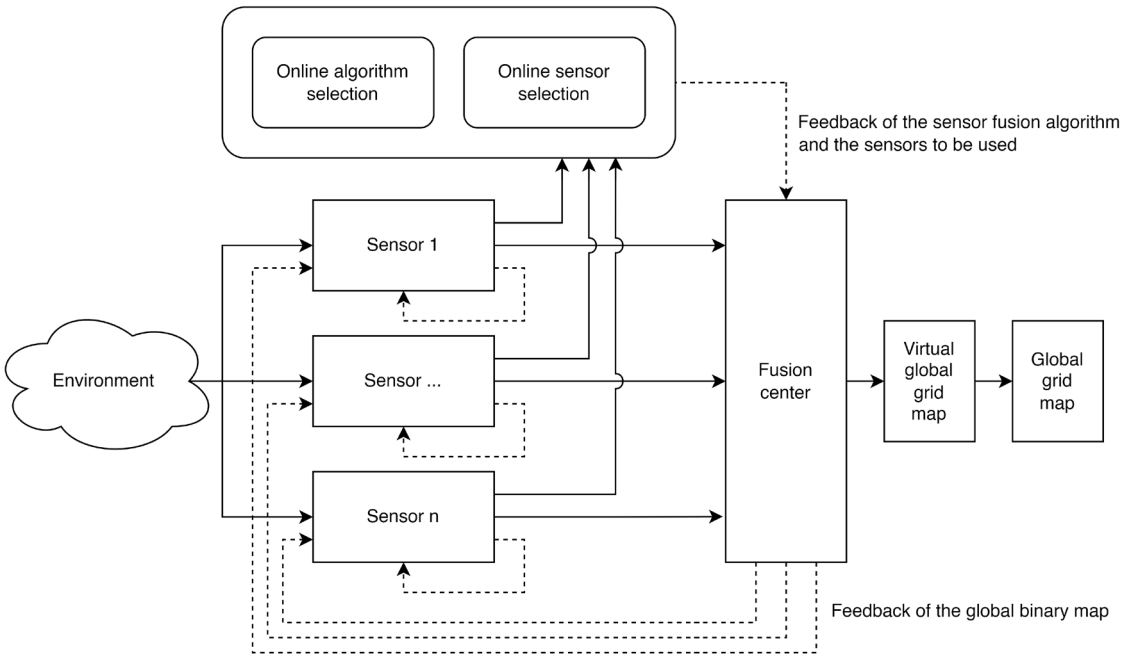


Figure 9: Hard sensor fusion model with the algorithm and sensor selection steps (adapted from [92]).

Hard sensor fusion's main advantage is the efficiency of the sensor monitoring and fusion process. Fully disabling a sensor results in lower power consumption and computational requirements, which is beneficial for autonomous platforms. Another advantage is the implementation simplicity: the robot controller only needs to monitor sensor data and disable a sensor when its data is unusable. The disadvantage of this approach is that there is less sensor redundancy, which can reduce the accuracy of the estimated position.

3.2 Soft sensor fusion

Soft sensor fusion frameworks use all sensor data to determine the optimal fusion strategy, similar to the hard sensor fusion frameworks. However, instead of either using a sensor or not using a sensor, the soft sensor fusion frameworks apply weights to the data generated by each sensor (type) [93]. While navigating, each sensor's weight is modified depending on the difference between the estimated position and the sensor's data. Lee et al. [94] presented a soft sensor fusion framework that combines multiple sensors and continuously determines the errors and biases for each sensor (Figure 10). With this corrected sensor data, a more accurate estimated position can be determined.

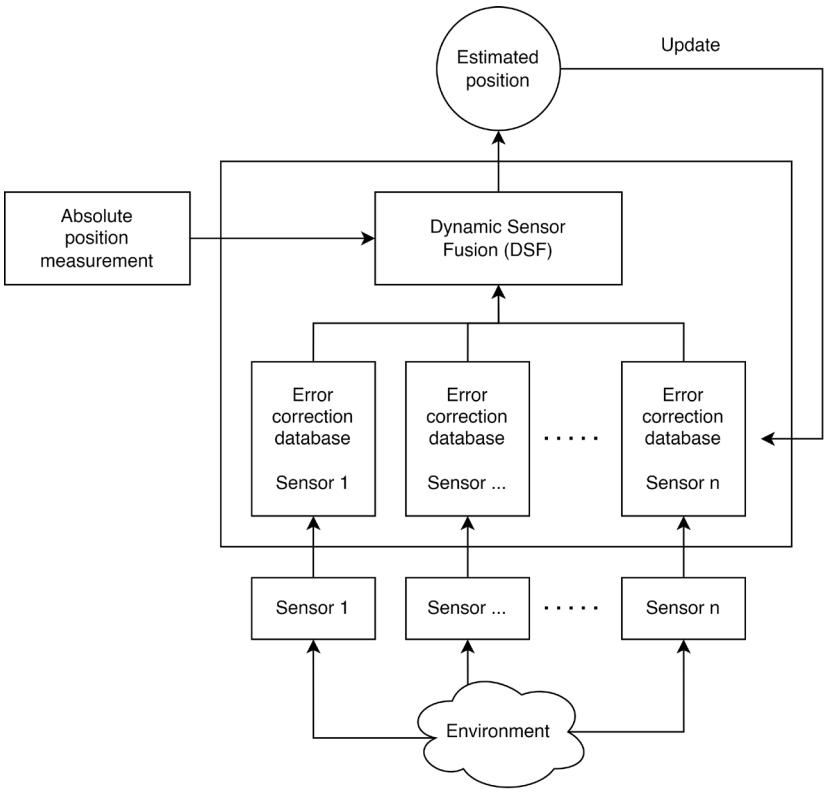


Figure 10: Soft sensor fusion framework based on error and bias correction (adapted from [94]).

By correcting the errors and biases in sensor data, instead of disabling a sensor entirely, soft sensor fusion can provide a higher level of redundancy and better position accuracy compared to hard sensor fusion. However, this does come at the cost of reduced efficiency, due to the continuous corrections that have to be applied to the active sensors. Subsequently, soft sensor fusion requires more computational power and it has a higher power consumption compared to the hard sensor fusion frameworks.

Hamadi and colleagues [95] demonstrated a similar model that combines multiple sensor data into two intermediate sensor fusion blocks. The estimated positions from these two sensor fusion blocks are then compared and evaluated against the raw individual sensor data. Based on this, each sensor's output data can be corrected to account for any error or bias. This correction improves the accuracy of the final estimated position.

Additionally, the presented framework is capable of detecting hardware or software errors in either the sensors or sensor fusion blocks. If an error occurs, the adaptive sensor fusion framework can take appropriate measures and disable the malfunctioning sensor or sensor fusion block. This improves the redundancy of the system, but it reduces the efficiency due to the need for duplicate sensor configurations. Figure 11 illustrates this redundant soft sensor fusion framework.

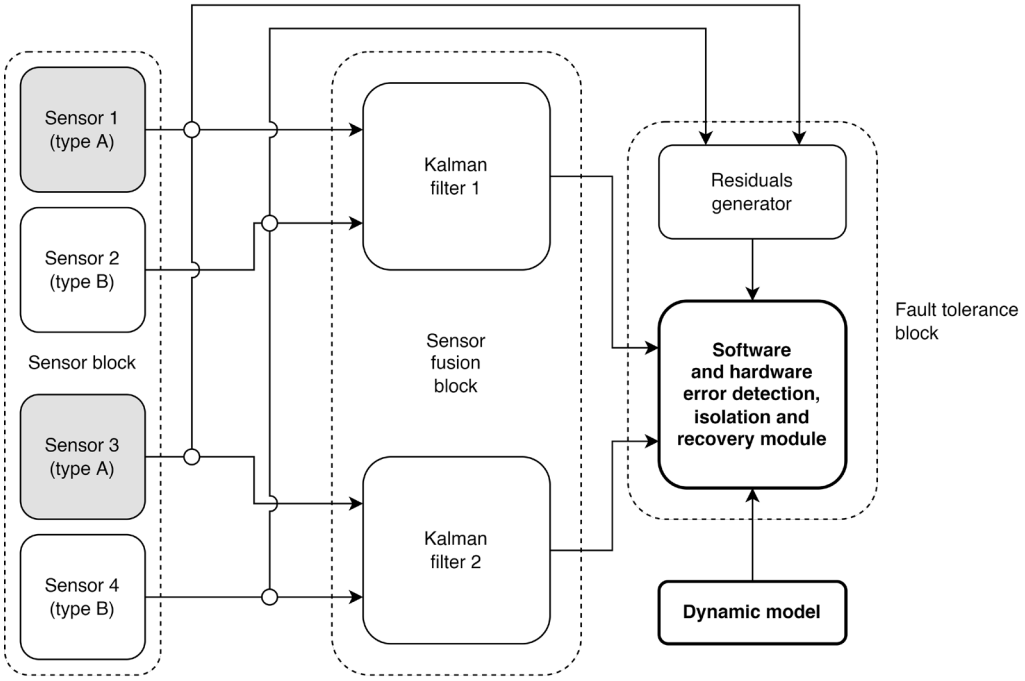


Figure 11: Adaptive sensor fusion with software and hardware failure detection (adapted from [95]).

A similar strategy is implemented in ArduPilot, an open-source software package designed to control various autonomous platforms. In this software, multiple extended Kalman filters (EKFs) are created using different configurations of sensors. This requires multiple sensors and sensor types [96]. Each EKF is defined as a "lane" and will always be active. However, the position estimate of only one of the lanes will be used for navigation.

Determining what lane to use is done using a relative error accumulation algorithm. This algorithm measures the performance of each lane and calculates an error score of the non-primary lanes that is based on the performance of the primary lane [97]. While navigating, a lane switching algorithm determines what lane to use as the primary lane, based on the relative error and a defined threshold value (Figure 12). ArduPilot's lane switching approach results in better position accuracy at the cost of a higher energy consumption and increased computational requirements.

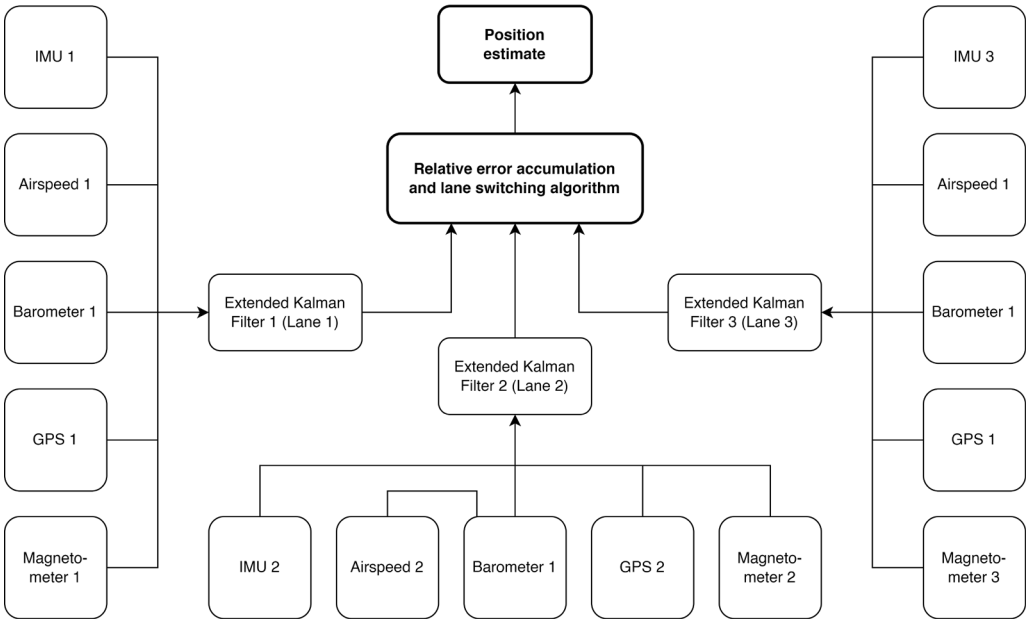


Figure 12: ArduPilot's EKF lane switching approach for multiple sensors.

3.3 Predetermined sensor fusion

A different approach is presented by Silva et al. [98], who utilized a camera in conjunction with a neural network to classify a robot's environment. If the neural network estimated the environment would lead to wheel slip, the adaptive sensor fusion framework would only use visual odometry for localization. On the other hand, if the environment was assumed to provide sufficient grip for the wheels, then wheel odometry and visual odometry would be combined in the sensor fusion framework for localization. Figure 13 shows the described adaptive sensor fusion framework.

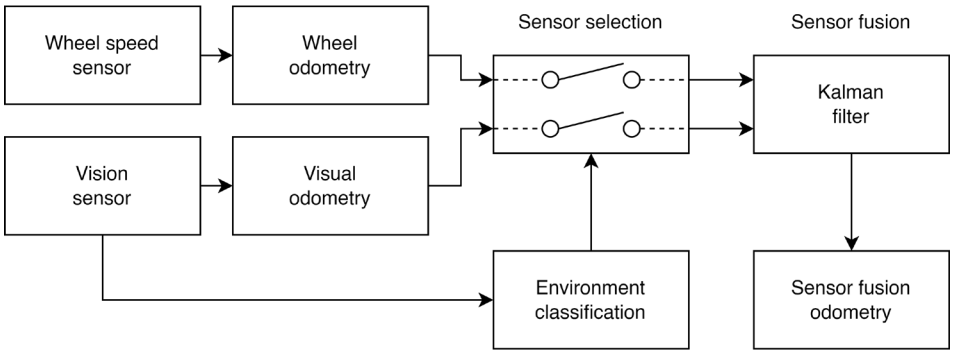


Figure 13: Predetermined adaptive sensor fusion network based on a neural network (adapted from [98]).

This predetermined sensor fusion approach uses environment data to determine in advance what sensors should be used in the sensor fusion framework. Applying this method can be more efficient compared to the previously described methods, as it does not require all sensors to be active to determine what sensors to use. This advantage will increase as the number of sensors increases. The disadvantage of this approach is that it can yield lower position accuracies than soft sensor fusion. This is the result of not utilizing suboptimal sensor data and correcting this signal for better accuracy.

4 Requirements and boundaries

With the knowledge about sensors, localization methods and adaptive sensor fusion frameworks, the requirements for the proposed solution can be defined. In accordance with the concept of mission-oriented command (section 1.5), an underpinned list of minimal requirements is defined. This list is based on the challenges and problems related to autonomous, military robots; the envisioned functionality of the autonomous platform; and the results from the theoretical research. The list of requirements is structured according to the theory of the KANO model [99].

Basic requirements

- The proposed solution must be undetectable by hostiles.
    - As mentioned in section 1.3.2, it is important that the autonomous robot remains undetected at all times, as this can give away the robot's position, possibly compromising the mission.
- The proposed solution must be compatible with, or adaptable to ROS.
    - Designing the proposed solution around ROS allows it to be efficiently adapted to other ROS-based platforms, despite any hardware differences. This matches the RAS Cell's need for expandability to other autonomous platform types.
- The proposed solution must be able to autonomously avoid obstacles.
    - Although the planned waypoints aim to create a traversable route, the autonomous platform must be able to avoid unexpected obstacles to increase the system's robustness.
- The proposed solution must be functional in GNSS-denied environments.
    - As mentioned in section 1.3.1, the proposed solution must be functional in GNSS-denied environments, as there are several causes that could lead to unusable GNSS signals in the deployment environment.
- The proposed solution must be able to operate in diverse natural and urban environments, and under diverse and changing conditions (e.g., rain).
    - While Project Sentinel currently focuses one a single environment type, future revisions must be able to operate in different environments and under challenging, varying conditions.
- The proposed solution must be able to operate autonomously in unknown areas without first mapping the environment in which it will be deployed.
    - Some localization methods require knowledge of the deployment area in advance, to successfully operate. To ensure that the proposed solution is self-reliant, it should be able to function without such information.
- The proposed solution must not include software of which the (source) code is unaccusable to the RAS Cell.
    - As mentioned in section 1.3.3, there are multiple reasons as to why the RAS Cell must have access to the (source) code of any software that is running on the autonomous system.

Performance requirements

- The position accuracy of the proposed solution (higher is better).
    - With no usable GNSS signal, consistently high position accuracies cannot be guaranteed. Thus, the proposed solution's position accuracy is defined as a performance requirement, with higher accuracies being better.
- The computational requirements of the proposed solution (lower is better).
    - Due to the limited availability of computing power on autonomous platforms, lower computational requirements are beneficial. This not only reduces the power consumption of the robot, but it also allows other functionalities to operate simultaneously without a performance loss.
- The power consumption of the proposed solution (lower is better).
    - Autonomous platforms have a limited battery capacity, which introduces the need for energy-efficient hardware and software. As it is undesired to run out of power during the mission, a proposed solution with a lower power consumption is better.

Excitement requirements

- The proposed solution can create a 3D map of the area during navigation.
    - Available information of the deployment area is often not up-to-date, or it lacks sufficient detail. Hence, any collected environment data can be used for military purposes. Thus, it is desirable to have a system capable of scanning its environment in 3D, regardless of the size of the 3D map
- The proposed solution can be applied to other autonomous platform types.
    - As mentioned in section 1.1, the goal of the RAS Cell is to deploy different autonomous platform types in the future. Hence, it is desirable to have a proposed solution that can be efficiently adapted in the future.

Boundaries

Besides the aforementioned requirements, several boundaries about the deployment of the autonomous platform are defined. These describe the presumptions about the available data for the autonomous platform that are assumed to be valid for this project. The following boundaries are defined:

- The absolute starting location of the reconnaissance robot is always known and can be configured on the autonomous platform to provide a known starting point for the localization system.
- Map data for the deployment area is obtained from public sources, which provide an incomplete, global coverage [100]. These sources provide terrain classification data, albeit with a varying accuracy and age.
- Satellite imagery data is obtained from public sources, which provide a global coverage with a maximum resolution of 60 meters and an age of 10 days [101].
- Environmental 3D point cloud data is obtained from public sources, which do not provide a global coverage with an unknown resolution and maximum age.

5 Proposed solution

Based on the information that has been gathered up to this point, a proposed solution is designed that combines SLAM-based localization methods with position corrections and an adaptive sensor fusion framework. This framework is controlled by analyzing the data from active sensors. All aforementioned functionality is combined in a behaviour tree for modularity and expandability. Figure 14 shows an overview of the proposed solution's components and functionalities. The following sections will further discuss each component in more detail.

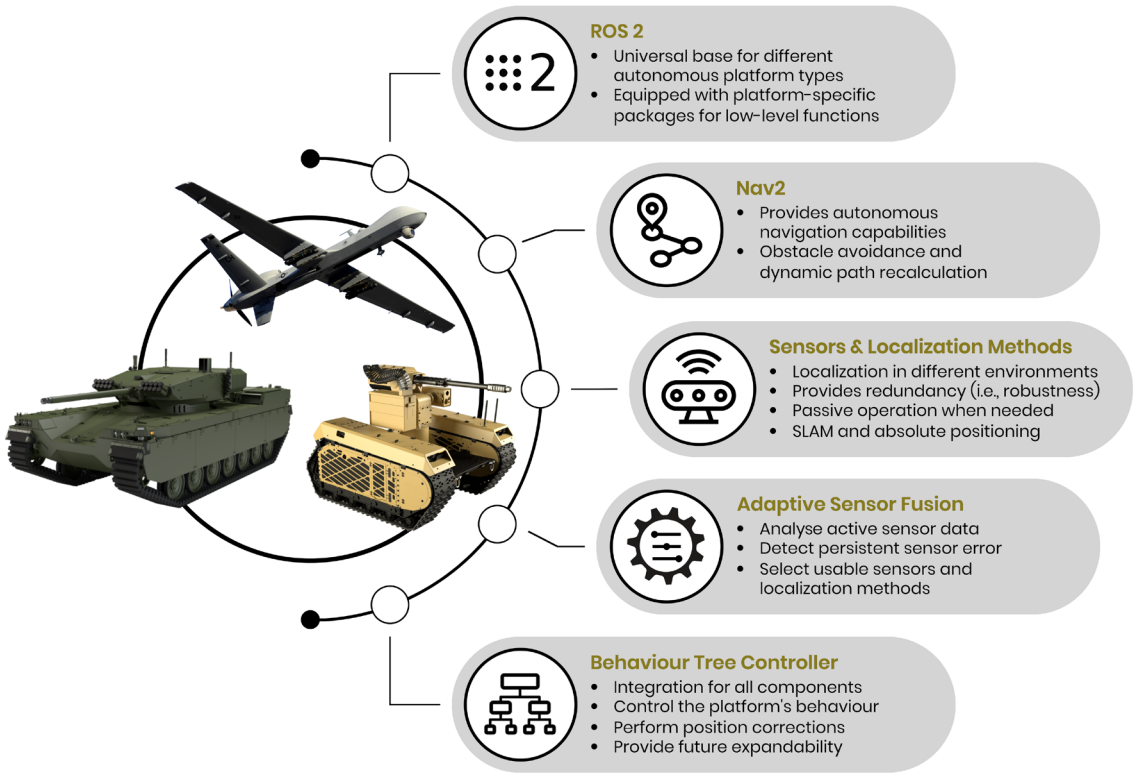


Figure 14: Overview of the proposed solution, its components, and their functionalities.

5.1 SLAM-based localization and navigation

As shown in section 2.1, there are multiple methods for GNSS-denied localization. However, many of these options are not robust enough for the military application. Therefore, a combination of SLAM-based localization methods and position corrections are proposed. This latter part will be further discussed in the next section.

There are multiple reasons for using SLAM-based localization methods as the main source for estimating the robot's position. First, SLAM algorithms allow for the detection of obstacles during navigation. Regardless of the used sensor type (e.g., vision or LiDAR), SLAM-based localization methods can perceive the environment and create a costmap based on the resulting data. Costmaps are used to define (non-)traversable parts of an environment. Other localization methods, such as SoOP, are cannot detect environmental obstacles, making them unsuitable as the primary localization method.

Another advantage of using SLAM-based localization methods is that they generate a 3D map of the environment while the robot navigates from waypoint to waypoint. Both LiDAR- and vision-based SLAM methods can create a point cloud of the environment, which can be converted to a mesh model or viewed as a raw point cloud. As mentioned in the previous chapter, this functionality is an *excitement requirement* for the RAS Cell, as the 3D maps can be used by the military to get a better understanding of the deployment area. This information can be used to make mission-related decisions and provide military personnel with up-to-date knowledge of the environment. Section 5.5 provides more information on how this point cloud data can be processed to reduce the storage and computational requirements.

Furthermore, the examples discussed in section 2.1 have showcased that SLAM-based localization methods are capable of achieving a consistently high position accuracy under the right conditions. This position accuracy reinforces the choice for SLAM-based localization methods, as it meets the proposed solution's *performance requirement*. To ensure that the position accuracy is maintained over longer distances and under suboptimal conditions, position corrections are applied. The implementation of these corrections will be discussed in the following section.

Finally, SLAM algorithms are independent localization methods, meaning that they do not require external data (e.g., magnetic field data, star maps, or satellite imagery) to operate. As a result, SLAM-based localization methods can always be used, assuming that the required sensors are usable. The use of independent localization methods is important, as it increases the robustness and self-reliance of the autonomous platform.

For this project, the SLAM implementation will be accomplished using RTAB-Map [102]. This is a ROS package (for both ROS 1 and ROS 2) that provides real-time appearance-based SLAM functionalities. One advantage of RTAB-Map is that it works with multiple sensor types, including: IMU, stereo camera, depth camera, 2D LiDAR, 3D LiDAR and wheel odometry. These different sensors can be used in different configurations, where RTAB-Map takes care of the sensor fusion process of the active/chosen sensors.

5.2 Position corrections

Despite the accuracy of SLAM-based localization methods, their position estimate is calculated using relative positioning. Consequently, it will drift over time, resulting in a lower position accuracy for longer trajectories. Additionally, SLAM's high position accuracy can only be achieved under the right conditions. Two examples of this are documented for the ORB-SLAM2 and S-PTAM SLAM methods, that achieved a relative position error of approximately 1.15% and 1.19% respectively in the KITTI Benchmark test sequences [103, 104]. However, when testing these algorithms under nonideal conditions in a forest environment, their relative position error increased to approximately 8.53% and 7.96% respectively [105]. Additionally, both algorithms failed to localize the robot within the forest environment when the sampling rate of the sensor data was reduced.

Position corrections can be implemented to compensate for the inconsistent position accuracy of SLAM-based localization methods. As discussed in section 2.1, there are multiple localization methods with which an absolute position estimate can be obtained. First, GPS or other GNSS signals can be used when they are available. While the assumption that usable GNSS signals are not present in the deployment area is based on valid concerns, and making this assumption improves the robustness of the proposed solution, the lack of usable GNSS signals in the deployment area is not a given. Therefore, GNSS signals should be considered as a feasible option to efficiently determine an absolute position estimate. Subsequently, it is advised that the autonomous platform is equipped with a GNSS receiver. To determine whether an available GNSS signal is usable, its position estimate can be compared to the relative position estimate. If the GNSS position estimate is not within an expected range of the relative position estimate, then the signal can be assumed to either be spoofed or too weak to be usable.

Other options to obtain an absolute position estimate include celestial localization methods, magnetic field-based solutions, signals of opportunity localization (when the transmitters' positions are known), or by matching ground images with satellite imagery. These absolute positioning methods are capable of achieving a position error between approximately 1.0 and 50.0 meters, which is similar to the position accuracy that can be achieved by GNSS receivers [106]. More importantly, these options provide a more accurate position estimate over longer distances (more than 1 km) than SLAM algorithms with an accuracy of 5% of the total distance travelled.

Continuous corrections

There are two options for the position corrections: continuous or periodic corrections. With continuous corrections, an absolute position estimate is continuously obtained in parallel with the SLAM algorithm's relative position estimate. These two position estimates are then combined to improve the accuracy. This task is commonly performed through an EKF [107, 108]. In navigation systems, an EKF is used to combine the output from multiple sensors with the goal of improving the overall position accuracy. To achieve this, the EKF applies a weight to each input. This weight is continuously corrected by calculating a prediction of the future estimated position, and comparing this estimate with the measured estimated position in the future state. The benefit of this approach is that the final position estimate is more accurate as a result of the continuous input weight adjustments. Note that this approach is similar to the concept of soft sensor fusion.

Continuous corrections require more computational power and thus result in a higher platform power consumption. This is due to the concurrent execution of two localization algorithms, and the subsequent combination of the two position estimates in an EKF. Moreover, the increased computational requirements can lead to a reduced localization performance for the SLAM algorithm as a result of lower execution frequencies. As shown in research, a drop of 50% in execution frequency can result in up to 160% lower position accuracies [105, 109]. While the EKF can partially compensate for the lower accuracy, the final position estimate will still be negatively impacted by the lower execution frequency.

Additionally, the implementation of continuous corrections is relatively complex due to the required compatibility of the EKF with multiple, changing localization methods. Figure 15 shows a flowchart of the continuous correction implementation.

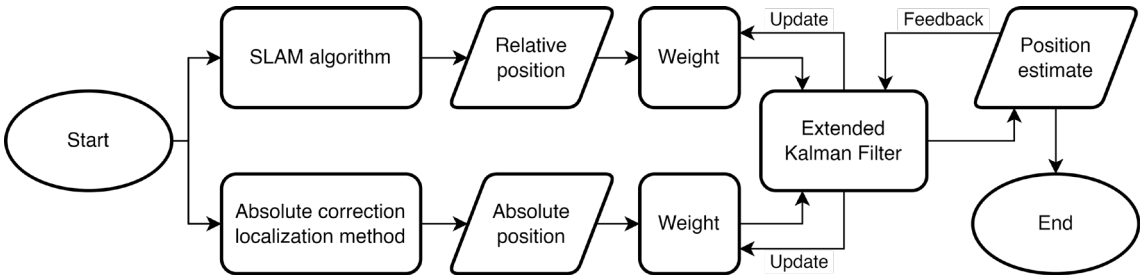


Figure 15: Flowchart of the continuous correction process using an EKF.

The continuous corrections can be implemented with the open-source robot_localization package for ROS 2. This package provides a configurable and extendable EKF that can be used for localization and navigation purposes [110].

Periodic corrections

Periodic corrections are based on the same absolute position estimate as the continuous variant, but the relative position estimate is only corrected at a specific rate or at specified locations. For example, the absolute corrections can be applied after every 30 minutes, or at every waypoint. The advantage of periodic corrections is that it requires less computational power, resulting in a lower power consumption for the autonomous platform. Additionally, it is less complex to implement and easier to extend with new localization methods. However, periodic corrections provide a less accurate position estimate since the relative and absolute position estimates are not continuously calculated and optimized through the EKF.

There are three possible scenarios that can occur with the estimated relative and absolute position: partial intersection, no intersection and full intersection. These three options can be seen in Figure 16 from left to right. In this image, point A and point B are position estimates (one relative, the other absolute), where the radius of each circle indicates the error of the position estimate.

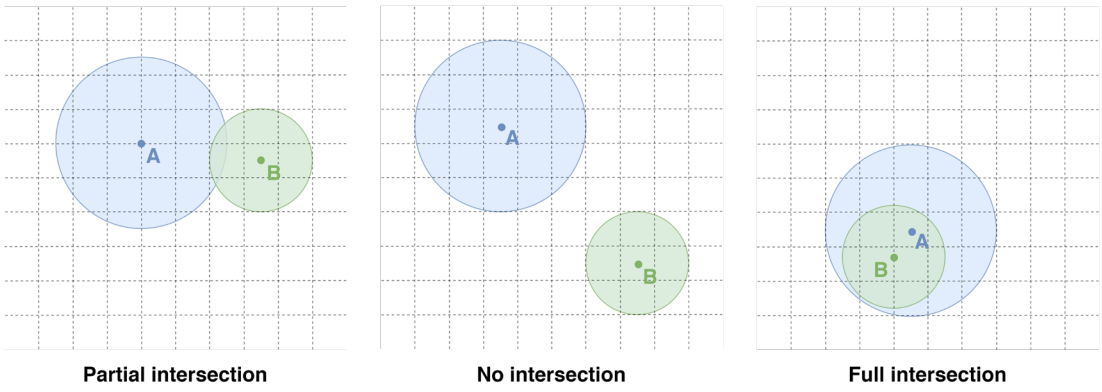


Figure 16: The different intersection scenarios for combining relative and absolute position estimates.

If there is a partial intersection between the two position estimates, as shown in Figure 16 (left), then the true position can be determined by calculating the intersection area's centre. The following equations describe the mathematics behind this approach:

Navigation statistics:

$$D_{travel} = 100m$$
$$Error_{rel} = 6\%$$
$$Error_{abs} = 10m$$

Relative position estimate:
$x_1 = 72m$
$y_1 = 36m$
$r_1 = 0.06 \times 100 = 6m$

Absolute position estimate:
$x_2 = 69m$
$y_2 = 41m$
$r_2 = 10m$

Distance between estimates:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \approx 5.83\ m$$

Parametric line equations:

$$x = (x_2 - x_1)t + x_1 = -3t + 72 \qquad y = (y_2 - y_1)t + y_1 = 5t + 36$$

Parametric intersection values:

$$t_1 = \frac{r_1}{D} = \frac{6}{5.83} \approx 1.03 \qquad t_2 = \frac{D - r_2}{D} = \frac{5.83 - 10}{5.83} \approx -0.72$$

Point A coordinates:
$x_A = (-3 \times 1.03) + 72 = 68.91$
$y_A = (5 \times 1.03) + 36 = 42.15$

Point B coordinates:
$x_B = (-3 \times -0.72) + 72 = 74.16$
$y_B = (5 \times -0.72) + 36 = 32.4$

True coordinates:

$$x_{true} = \frac{x_A + x_B}{2} = \frac{68.91 + 74.16}{2} = 71.5 \qquad y_{true} = \frac{y_A + y_B}{2} = \frac{42.15 + 32.4}{2} = 37.3$$

Final position estimate:

$$x_{true} = 71.5m$$
$$y_{true} = 37.3m$$

In the equations above, a position correction is applied after a trajectory length of 100 meters. This distance is an estimate that can be retrieved from the navigation algorithm. The relative and absolute error are estimates based on the results from simulations and experiments performed in theoretical research. These values represent the hypothetical relative and absolute localization algorithms used for this example. Note that these two error values can be adjusted if necessary. The following paragraph provides a textual explanation of the previously shown calculations.

To determine a final position estimate, the circle parameters for both position estimates are defined. These circles indicate the range of coordinates in which the true position lies (centred around the algorithm's reported position estimate). Then, a formula is created that represents a straight line between the two position estimates. This line intersects the edges of the intersection area at two points, creating a smaller line segment. The midpoint of this line segment is the final position estimate (Figure 17).

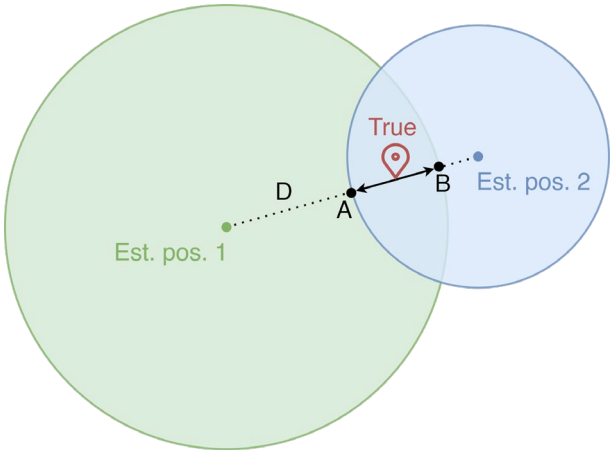


Figure 17: Visual representation of the combined position estimate calculation (image not to scale).

When there is no intersection between the two position estimates, then the most accurate position estimate can be used as the true position. As shown in Figure 16 (middle), this would result in the coordinates of point B being used as the true position. However, because there is no overlap in the two position estimates, another option would be to use a second absolute positioning algorithm to potentially discover an intersection between the position estimates. In the case of a full intersection, the most accurate position estimate can again be used for the true position. For Figure 16 (right), this means that the coordinates of point B are assumed to be the true position.

Due to the relatively simple mathematics behind the periodic correction method, it can be efficiently implemented by writing custom behaviour tree nodes (the explanation of the behaviour tree's functionality will be described in section 5.4). Figure 18 shows the flowchart of the periodic correction process.

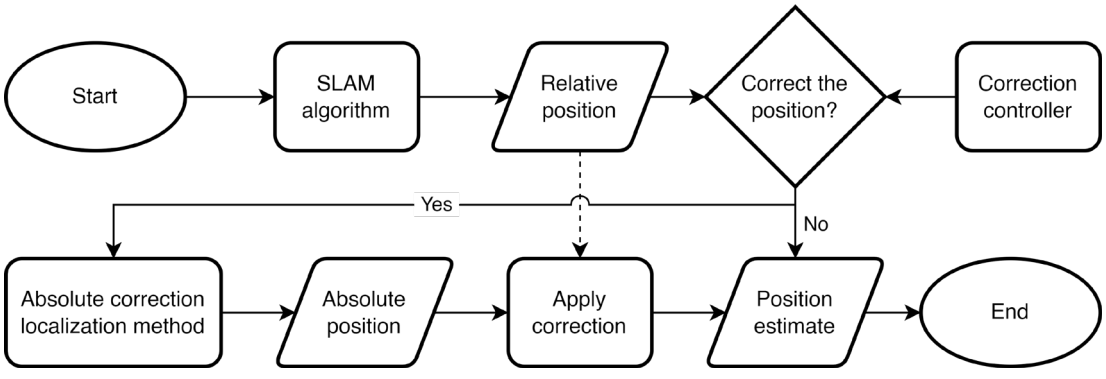


Figure 18: Flowchart of the periodic correction process using the previously described mathematics.

Note that the periodic correction method mimics the human approach of navigating in unknown environments when only given a map and a set of waypoints:

1. Determine or estimate the starting position on a map, based on the surroundings.
2. Determine the first waypoint's location on the map using its coordinates.
3. Plan a path to the aforementioned waypoint.
4. Start following the planned path.
5. Stop navigating when the waypoint is assumed to be reached.
6. Estimate the current position on the map using the surroundings.
7. Plan a path to the next waypoint using the previously determined position.
8. Repeat from step 4 onwards.

Selected correction method

Based on the results from the analysis of the two different correction methods, periodic corrections are selected as the most suitable option. Their efficient expendability and implementation make this solution preferable. This underpinning is strengthened by the complexity and computational requirements of the continuous corrections that can negatively impact the accuracy of the position estimates.

5.3 Adaptive sensor fusion framework

The adaptive sensor fusion framework is based on the researched adaptive sensor fusion frameworks, with an initial selection stage for the most suitable sensor(s) and localization method. This initial selection is combined with a continuous data analysis of the active sensor(s) to trigger a new selection during the mission if necessary (Figure 19). This framework runs in parallel with the navigation process of the autonomous platform.

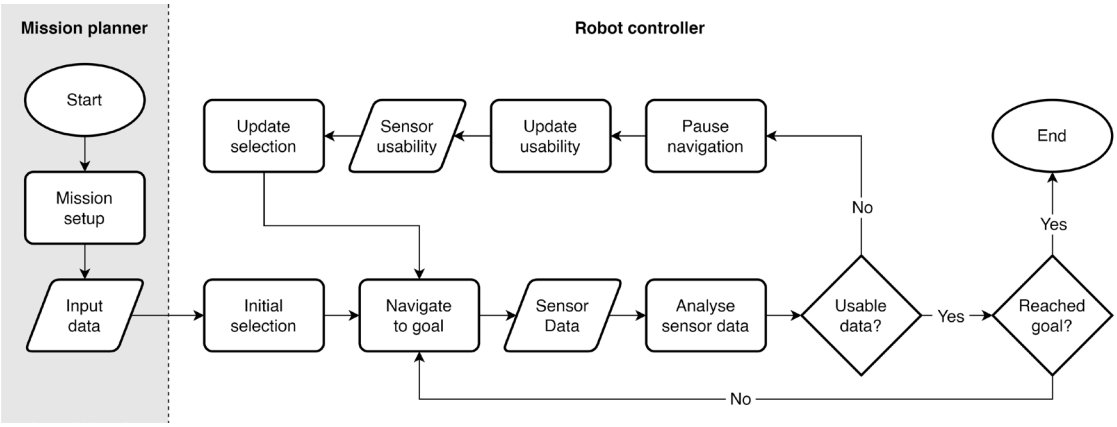


Figure 19: Flowchart of the adaptive sensor fusion framework.

The first step of the adaptive sensor fusion framework is the mission setup. This phase is performed in the mission planner, meaning that it is not located on the reconnaissance robot itself. During this step, the commander defines the mission parameters, and the environmental variables are measured/determined. The input data generated in this phase is sent to the robot, where it will be used for the robot controller setup. From this transition onward, all steps are performed on the autonomous platform itself.

Based on the input data from the mission setup, an initial selection is performed to determine the most suitable localization method. The selection model used in this step will be further described in section 5.3.1. Once the initial selection has been performed, the reconnaissance robot can run the selected localization method algorithm and start executing the reconnaissance mission. Note that the waypoint navigation is handled by the behaviour tree controller, not by the adaptive sensor fusion framework itself. More information about the behaviour tree will be provided in section 5.4.

Data from the active sensor(s) is continuously analysed during the mission. Section 5.3.2 describes the analysis methods for the different sensor types. Each sensor data analysis script contains multiple parameters that are used to determine whether the sensor data is unusable. These parameters can be modified by the user to adjust the adaptive sensor fusion framework's behaviour. Parameters can also be added or removed, if desired. If a persistent sensor data error is detected, a correction is triggered that results in a new localization method. This selection process uses the same algorithm as the initial selection, but this time the sensor list has been updated to reflect the unusable sensor data. The user can define per sensor type how many consecutive errors result in a persistent error. As long as this error limit is not reached, the robot will continuously check whether the final waypoint has been reached. While the final waypoint has not been reached, the sensor check and navigation process will run in a loop. Once the final waypoint has successfully been reached, the aforementioned processes are halted.

5.3.1 Localization method selection algorithm

The selection model of the adaptive sensor fusion framework determines the most suitable localization method based on the usable sensors and localization methods. For this process, a list of usable sensors is first created based on the mission parameters and environmental variables (i.e., the input data). Then, using this list and the input data, a list of usable localization methods is defined. Based on the characteristics and performance metrics of the usable localization methods, the most suitable localization method is determined. The current goal of the selection model is set to select the most accurate localization method. However, it is possible to add user-defined parameters to the final selection. This can be used to balance certain characteristics, such as accuracy and execution speed. The flowchart of the selection model can be seen in Figure 20.

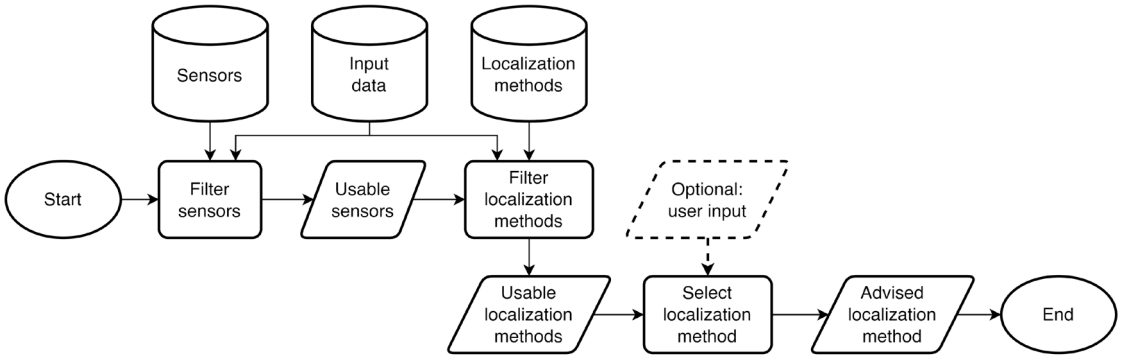


Figure 20: Flowchart of the localization method selection model process.

The sensor database contains a list of the available sensors with conditions under which the sensor cannot be used. These parameters are the characteristics of the sensors, which are used in the localization method selection. The input data database contains a list of mission parameters and environmental variables. These values are defined during the mission setup. Finally, the localization methods database contains a list of the available localization methods with their required sensors, the conditions under which they cannot be used, and their characteristics and performance metrics. These final two data types are used to determine the most suitable localization method from the filtered list of usable localization methods.

Due to the use of databases and a three-stage filtering approach, more sensors and localization methods can be added in the future without adding any code. They can simply be added to the respective databases, and are then automatically taken into account in the selection process. Additionally, the characteristics for both sensors and localization methods can be adjusted when necessary. Finally, the selection model can be used on multiple robot platforms equipped with different sensor configurations.

The pseudocode for the localization method selection script can be seen in Figure 21.

Algorithm 1: Selection of the most suitable localization method

Input: Input data, list of sensors and list of localization methods
Output: The most suitable localization method

```
1  def select_localization_method():
2          for sensor in available_sensors:
3                  if sensor == usable and sensor[charac] >= input_data[values]:
4                          add sensor to usable_sensors
5
6          if usable_sensors == empty:
7                  return error
8
9          for method in available_localization_methods:
10                 if method[charac] >= input_data[values]
11                 and method[charac] == usable_sensors:
12                         add method to usable_localization_methods
13
14         if usable_localization_methods == empty:
15                 return error
16
17         sort usable_localization_methods on method[accuracy]
18         advised_localization_method = usable_localization_method[0]
19
20         return advised_localization_method
```

Figure 21: Pseudocode for the localization method selection algorithm.

5.3.2 Sensor data analysis

Analyzing the data from actively used sensors requires a different approach for different sensor types. For this project, a sensor data analysis script has been written for vision sensors (i.e., cameras) and LiDAR sensors.

For both sensor types, the data is collected at a topic-level in ROS. This means that the raw sensor data has been processed by the sensor's driver. There are multiple reasons for using sensor data from this stage. First, measuring the raw signal coming from a sensor requires specific hardware and algorithms to analyse their usability. Additionally, due to the unprocessed nature of this signal, it is not possible to determine what the cause for an unusable sensor signal would be. Finally, different sensors of the same sensor type require a different analysis algorithm to obtain the same result.

Another option is to use feedback or error messages from the localization method's algorithm to determine whether a sensor can be used or not. This solution would be applicable for different sensors within the same sensor category. However, this method does not provide a cause behind the unusable sensor signal. Additionally, this approach requires a different message handling approach for the different localization method algorithms that the autonomous platform could use. Finally, not all localization methods will output a specific error when sensor data becomes unusable.

Analysing sensor data at a topic-level is universal across different sensors within the same sensor category, while also providing more detail on the cause of an unusable sensor signal. This information can be used to determine whether the unusable data is temporary or permanent. Additionally, since the sensor's signal has already been processed by its driver, no additional hardware is required for the analysis. Finally, this approach is applicable to other autonomous platforms, as the sensor drivers are inherent to the sensor itself, not the platform that it is used on.

Vision data analysis

The vision data collected by cameras is analysed using two different checks. First, the script responsible for the vision data analysis checks whether the camera data is actively being published to its corresponding ROS topic. In ROS, sensor data is published to topics, where it can be retrieved by other scripts, nodes, or software components. By checking if the camera topic is actively being published to, possible hardware or driver failures can be detected.

If the camera is correctly publishing the vision data to its topic, the vision data is further analysed on image brightness and the file size. Images that are either too dark or too bright can be unusable due to a lack of visual information in the image. Additionally, if the brightness of the image is not within a specified range, it could mean that there are obstructions blocking the camera, or that there are unfavourable lighting conditions. Similarly, if the file size of a still image capture is too small, this could indicate that there is too little detail in the image for it to be usable. Another cause could be a hardware failure, where the published image is a solid colour without any actual information.

Both checks are performed periodically due to the behaviour tree's mechanism (more about this in section 5.4.2). The camera topic is checked first, after which the image analysis function is called. Retrieving the vision data is done using Python's ROS subscriber functionality, which allows a specific function to be triggered when a message is published to a specified topic. For the analysis script, this means that the vision analysis will be triggered automatically when the sensor's driver publishes the data to its topic.

If one of the checks determines that the vision data is unusable, it will return an error through the socket communication interface. This error is received by the behaviour tree node, which will stop the navigation process if the defined consecutive error limit is reached. This logic and the consecutive error limit are programmed into the behaviour tree node, which will be discussed in section 5.4.2.

The pseudocode for the vision data analysis script can be seen in Figure 22.

Algorithm 2: Analysis of the camera's vision data

Input: Camera topic and vision data
Output: A usable or unusable designation for the camera sensor

```
1  def check_camera_topic():
2          active_topics = get_active_topics()
3          if active topics != camera_topic:
4                  return error
5
6          check_vision_data()
7
8  def check_vision_data(data):
9          img = data.convert_to_image()
10
11         if img.brightnes() < brt_limit_low or img.brightness() > brt_limit_high:
12                 return error
13
14         if img.file_size() < fs_limit:
15                 return error
16
17         return success
18
19 def main():
20         check_camera_topic()
```

Figure 22: Pseudocode for the vision data analysis algorithm.

LiDAR data analysis

LiDAR data is analysed using a similar approach as the vision data analysis. First, the script checks whether the topic for LiDAR data is active. Then, the following analyses are performed on the LiDAR data itself: checking if the LiDAR outputs the expected number of scan points; checking if the scan point distances are not all 0 or infinite; and checking if there are enough valid scan points compared to the total number of scan points.

Similar to the vision data, these analyses are performed periodically, following the behaviour tree's execution. The result of the LiDAR data analysis will be parsed to the corresponding node in the behaviour tree. Based on the consecutive error limit defined by the user, this node will trigger a new selection for the localization method if necessary.

The pseudocode for the LiDAR data analysis script can be seen in Figure 23.

Algorithm 3: Analysis of the LiDAR scan points

Input: LiDAR topic and scan points
Output: A usable or unusable designation for the LiDAR sensor

```
1  def check_lidar_topic():
2          active_topics = get_active_topics()
3          if active topics != lidar_topic:
4                  return error
5
6          check_lidar_data()
7
8  def check_lidar_data(data):
9          valid_data_pct = data[valid] / data[raw] * 100
10         summed_dist = sum(data[valid])
11
12         if valid_data_pct < valid_data_pct_limit:
13                 return error
14
15         if summed_dist == 0:
16                 return error
17
19         if data[raw] != expected_scan_points:
19                 return error
20
21         return success
22
23 def main():
24         check_lidar_topic()
```

Figure 23: Pseudocode for the LiDAR data analysis algorithm.

5.4 Behaviour tree controller

A behaviour tree is used to control the behaviour of the autonomous platform, which contains all the platform's functionalities. Behaviour trees are a concept that allow tasks to be dynamically organized in a tree-like structure, using nodes and subtrees [111]. One advantage of using a behaviour tree is that tasks can easily be reused in different stages of the robot's behaviour by copying existing nodes or subtrees to a different location in the behaviour tree. Additionally, due to the structure of behaviour trees, the robot's behaviour can be efficiently modified between missions. This modular and dynamic approach is important for the military application, as it allows the execution plan of missions to be adjusted. Another advantage is that the robot's behaviour capabilities can be extended in the future by adding additional nodes that contain/perform new tasks. Finally, by adapting the behaviour tree of an existing robot, other robotic platforms (e.g., drones) can be configured and deployed more efficiently.

The RAS Cell is already using behaviour trees in Project Sentinel for the autonomous platform's behaviour control mechanism. From their perspective, the full control and expandability that behaviour trees provide are important aspects, as it allows them to efficiently modify the robot's behaviour and implement new capabilities in the future. Another reason is that behaviour trees are easily integrated with robotic platforms running on ROS. This topic will also be discussed in the next section.

The main drawback of behaviour trees is the relatively complex implementation. Due to the modular approach, there are multiple behaviour tree structures to achieve the same functionality. Additionally, it can take some time to fully understand the behaviour tree's logic, how it executes certain nodes, and how it responds to different node outcomes (i.e., Success, Running, Failure). As a result, the initial implementation of behaviour trees can be relatively complicated compared to alternatives.

An alternative to controlling the robot's behaviour would be to use finite state machines (FSMs). This is a concept in which robotic behaviour can be defined using states and transitions [111]. In general, FSMs are easier to setup for smaller, less complex behaviours. The main disadvantage of an FSM is that each state and transaction must be manually coded, meaning that each task is configured to be executed after a specified transition has occurred. As a result, FSMs can be modified or extended less efficiently, and are more difficult to reuse on other robotic platforms. Another disadvantage is that an FSM can only have a single active state at any time, whereas behaviour trees can have multiple nodes running simultaneously. This capability is crucial, as the autonomous platform's envisioned functionality requires multiple tasks to be performed at the same time. Because of these disadvantages, behaviour trees are the more suitable option that match the requirements for the autonomous platform.

5.4.1 Implementation

The behaviour tree is designed and visualized using Groot, which is a visual editor for behaviour trees created with the BehaviorTree.CPP library [112, 113]. These two software packages are used as they are compatible with ROS 2 and the Nav2 navigation stack [114]. Another advantage of using Groot is that it allows behaviour trees to be modified efficiently without the need for coding (knowledge). This allows end users to modify a robot's behaviour for different (reconnaissance) missions. Finally, Groot visually displays the execution status of an active behaviour tree, visualizing what the robot is doing. This is advantageous as it provides commanders and legal personnel with an explanation of what the autonomous platform is doing and why, without requiring in-depth knowledge of the code that runs in the background of the behaviour tree. Figure 24 shows the BehaviorTree.CPP (left) and Groot interface (right) for a running behaviour tree.

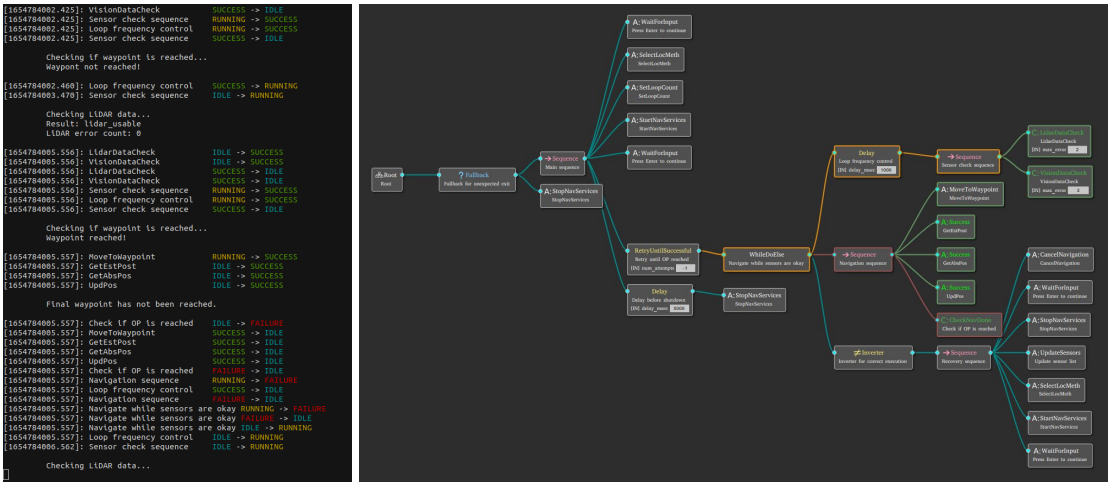


Figure 24: BehaviorTree.CPP interface (left) and Groot interface (right).

Custom nodes are written in C++, after which they can be placed in the behaviour tree using Groot. This scripting language is used as the BehaviorTree.CPP library is also written in C++. Some custom nodes communicate with Python scripts using ZeroMQ socket connections, which is an efficient implementation for inter-process communication [115]. Python is used to control certain parts of the robot's behaviour, since not all of the ROS 2 functionality is included with the C++ API. Other nodes that control part of the behaviour tree's execution behaviour can be inserted from Groot's built-in node library. Examples of such nodes are the sequence node, delay node and fallback node.

Alternatives to BehaviorTree.CPP and Groot are PyTrees [116] and the PyTrees ROS Viewer [117] respectively. These are the Python equivalent for BehaviorTree.CPP and Groot, allowing ROS behaviour trees to be programmed using the Python language. One disadvantage of this alternative is that the PyTrees ROS Viewer can only be used for visualization purposes, not for editing an existing behaviour tree. Additionally, in contrast to BehaviorTree.CPP, PyTrees is not integrated with the Nav2 navigation stack. These disadvantages make PyTrees and the PyTrees ROS Viewer an inferior option, as they lack integration and a user-friendly visual editing functionality.

5.4.2 Behaviour tree design

Figure 25 shows the behaviour tree that contains the relevant nodes for the proposed solution. This is a simplified diagram of the complete behaviour tree that is designed in Groot, which can be found in Appendix F. This full behaviour tree from the appendix contains additional nodes to correctly execute the behaviour tree's logic and to perform debugging tasks.

Figure 25: Simplified diagram of the proposed solution's behaviour tree.

The behaviour tree in Figure 25 is executed from left to right, from top to bottom. After the root node, the main sequence is started. First, an initial selection is performed for the most suitable localization method. If that succeeds, the total number of waypoints is set. This number is determined using the JSON file containing all waypoint coordinates. Then, the navigation services related to the selected localization method are started. Once these services are successfully started, the navigation loop is executed.

This loop consists of a while-statement that first checks the sensor data. However, the sensor check sequence is preceded by a loop frequency control node. This node pauses the behaviour tree execution for a specified amount of time. Users can modify the frequency of the sensor check loop by adjusting the delay value in this node. When more sensor checks nodes are added in the loop, the value of this frequency control node can be reduced, and vice versa. After the delay is finished, each sensor check is performed consecutively in a separate node. These nodes have an input port where users can define the maximum number of consecutive errors per sensor. If a sensor is not used for the active localization method, then the corresponding node will skip the sensor check automatically. This logic is programmed into the node itself. If all applicable sensor data checks are executed, the behaviour tree will start the navigation sequence. Additional nodes can be added to the sensor data check sequence for newly implemented sensors.

The first node in the navigation sequence sends the autonomous platform to the first upcoming waypoint, and checks whether it has been reached. If the waypoint has not been reached, this node will return a "Running" status. As a result, the behaviour tree will return to the sensor data check sequence. This parallel behaviour is known as a coroutine node, which allows the waypoint navigation to remain active while checking the sensor data. Due to this node type, the behaviour tree is capable of consecutively running multiple sensor data checks while navigating to a waypoint and subsequently checking whether it has been reached.

Once a waypoint has been reached, the waypoint navigation node will return a "Success" status, triggering the execution of the next node. This is where the position correction is applied, using three separate nodes for the different process steps. This process has been described in section 5.2. After the correction, the final navigation sequence node checks whether the final waypoint (i.e., the OP) has been reached. If this is the case, then the entire behaviour tree stops. If the final waypoint has not been reached, then the navigation loop is repeated for the next waypoint from the JSON file.

If a sensor data check returns a failure during the navigation sequence, then the recovery sequence is triggered. When this sequence is triggered, the behaviour tree's logic automatically halts the execution of the waypoint navigation. The first node from the recovery sequence pauses the autonomous platform. Then, the navigation services belonging to the active localization method are shut down. Based on the results from the sensor data checks, the sensor list is updated to classify the faulty sensor as unusable. Next, using this updated sensor list, a new localization method is selected. Subsequently, the corresponding nodes and navigation services are started. Once these services are successfully running, the navigation loop continues with the waypoint navigation. This navigation loop exits once the final waypoint has been reached. This will also trigger the shutdown of the active localization method algorithm and navigation services, which is executed after a customizable delay. This delay is integrated to ensure a clean shutdown. Once the localization method and navigation services are off, the behaviour tree itself will exit with a "Success" status.

5.5 Point cloud data compression

A disadvantage of SLAM-based localization methods is the relatively large amount of data that is used in the process. Both vision and LiDAR-based SLAM methods use point clouds to update the position estimate of a platform and to perform loop closure detection. The size of the point clouds not only affects the storage requirements of the autonomous platform, but it also increases the computational requirements.

One possible solution would be to use semantic SLAM methods, where the raw point cloud data is converted to a semantic segmentation map. As described in section 2.1, these localization methods are less accurate than regular SLAM methods. Additionally, semantic segmentation maps of the environment contain less information than raw 3D point clouds, which makes them less informative and more difficult to analyse for human operators. Due to these disadvantages, and the fact that a 3D map of the environment is an *excitement requirement*, semantic SLAM is not used for point cloud data compression.

Instead of using semantic segmentation, it is advised to use point cloud compression. Research indicates a positive viability of point cloud compression with SLAM-based localization methods. Cui et al. [118] proposed a compression method that compares consecutive point clouds and only saves identical points with the highest occurrence count. Using this approach, the researchers were able to reduce the number of points in the point cloud with ~95% while maintaining a position error of less than 0.04 m. Similarly, Tu et al. [119] demonstrate a compression method that compresses the raw point cloud data by periodically saving certain point cloud scans as reference scans, that are then used to estimate the point cloud for other scans. This point cloud estimation is performed using the pose information (position and rotation) generated by the SLAM algorithm. With this approach, the bitrate can be reduced from ~8,500 kb/s to ~4,700 kb/s, while maintaining a sub-meter position accuracy in multiple urban environments.

Implementing point cloud compression within project Sentinel could be done using Google's Draco library [120]. This is an open-source library designed to compress and decompress 3D point cloud data or geometric meshes with the goal to reduce storage and computational requirements. Paplhám and Petříček [121] implemented this library in an open-source ROS node that compresses and transports 3D point cloud data using ROS's built-in message type sensor_msgs/PointCloud2. Similarly, Wiemann et al. [122] tested the Draco library in a ROS test setup where a Velodyne Puck VLP-16 was used to generate a 3D point cloud. Using compression, the point cloud data was reduced with 12% while retaining a position error less than 1 mm.

Another option would be to use the point cloud compression ROS node from Dybedal and colleagues [123], which uses voxels for compression to reduce a point cloud's size. With a voxel resolution of 4 cm, a point cloud compression ratio of 40.5 was achieved. Increasing the voxel resolution to 2 cm resulted in a 22.5 compression ratio. Despite the fact that this solution is not open-source, it is based on the pcl_octree library from the open-source Point Cloud Library (PCL) [124].

6 Simulation testing

Simulations have been performed to test the functionality of the proposed solution. Based on the results from this testing, recommendations are provided on how the current state of the proposed solution can be improved.

6.1 Simulation setup

All simulations are performed on a Lenovo P51 laptop with an Intel Core i7-7700HQ processor (four physical processor cores running at 2.8 GHz), 16 GB of 2400 MHz RAM and an NVIDIA Quadro M1200 dedicated graphics card.

The adaptive sensor fusion framework and behaviour tree are tested in ROS 2 on Ubuntu 20.04 LTS. Gazebo is used to simulate an autonomous platform in a testing environment, while BehaviorTree.CPP and Groot are used for the behaviour tree execution and visualization respectively. The TurtleBot3 Waffle is used as the simulated robot, which is equipped with a 2D LiDAR and stereo camera. This platform and sensor configuration are chosen for their clear documentation, relatively simple setup, and the availability of working examples with multiple localization algorithms. Note that the platform itself does not impact the goal of the simulation tests, as these are aimed at the functionality of the adaptive sensor fusion framework and behaviour tree.

The testing environment consists of a single-story house, where the second half of the building is covered with a roof (Figure 26). Due to this setup, the environment's brightness will permanently decrease in the second half of the navigation trajectory. A home environment is used for the testing environment, since the 2D LiDAR is not designed to operate in natural environments. 3D LiDAR sensors are more suitable for such environments, which are less structured and more open.

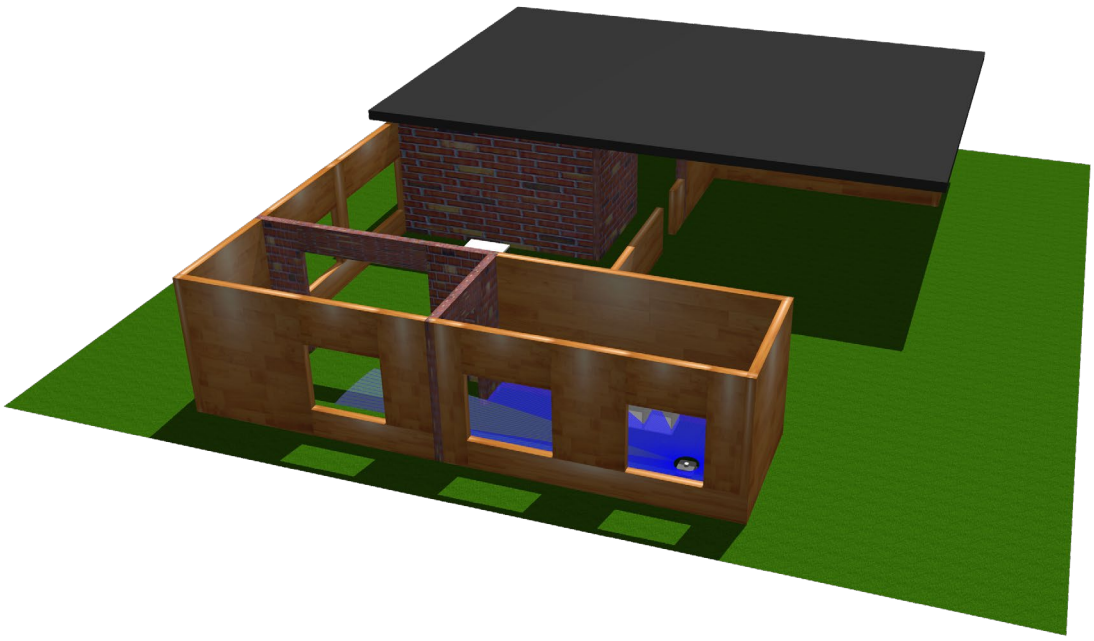


Figure 26: Gazebo environment in which the simulation runs are conducted.

A list with 10 waypoints is used for the navigation process. These waypoints navigate the robot from one side of the house to the other, with multiple obstacles along the way. Since the robot has no prior knowledge of these obstacles (nor of the environment itself), the obstacle avoidance capabilities are also indirectly tested.

There are two localization methods that can be used for navigation purposes: RTAB-Map with the 2D LiDAR and stereo camera, and RTAB-Map with only the 2D LiDAR. Data from these two localization algorithms are communicated with the Nav2 navigation stack, which provides the navigation and obstacle avoidance functionalities.

The goal for the simulations is to test the detection of unusable sensor data, followed by a successful switch to a new, usable localization method. During the switch, it is expected that the robot will pause the navigation process. Once the new localization method has been initialized, the robot should continue the navigation process to the upcoming waypoint(s), until it successfully reaches the final waypoint (i.e., the OP).

6.2 Test results

Three configurations are tested in the simulations, in which the maximum consecutive error value for the vision data is set to 1, 2 and 3. Each configuration is tested three times to test for consistency. This results in a total of nine simulation runs. For every run, the ground truth (Gazebo position), estimated position (RTAB-Map position) and switchover coordinates are logged. The switchover coordinate indicates the location at which the behaviour tree pauses the navigation process to switch between localization methods.

The results from the first configuration (maximum consecutive error = 1) can be seen in Figure 27. The graphs indicate that the adaptive sensor fusion framework consistently detected unusable camera data at an average location of X = -5.82 and Y = 2.38. While not expected, the environment brightness is too low around this location for the camera data to be usable. Due to the “strict” maximum consecutive error value of 1, the robot switches from using both the LiDAR and camera (ld_vs_slam) to using only the LiDAR (ld_slam). Once the switch has been concluded, the robot successfully continues its navigation process to the final waypoint. In all three runs, the final waypoint was reached.

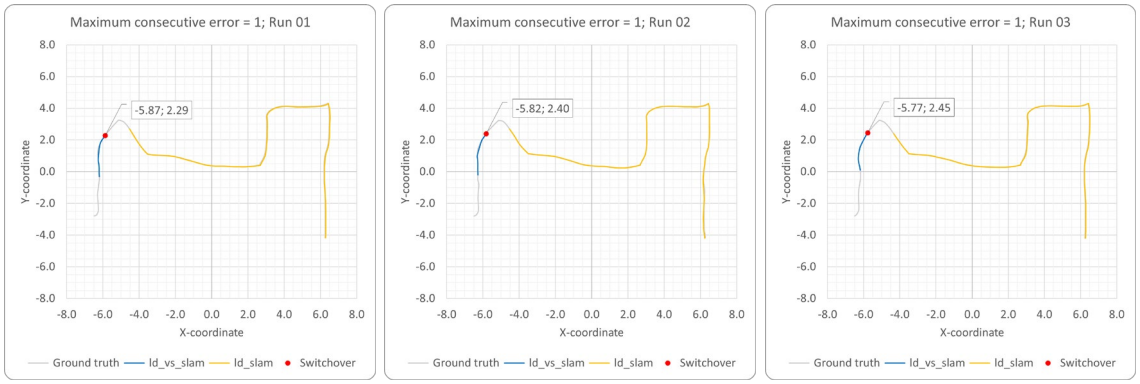


Figure 27: Results from the three runs testing the first configuration setup.

Figure 28 shows the results from the second configuration, which uses a maximum consecutive error limit of 2. Using this configuration, the robot again detected unusable sensor data in a consistent manner. With an average switchover coordinate of X = 5.21 and Y = 3.17, the adaptive sensor fusion framework detects unusable camera data in a similar location as the previous configuration. The distance between the two average switchover positions is determined by the frequency at which the sensor data is checked. This distance can be altered by modifying the loop frequency value in the behaviour tree.

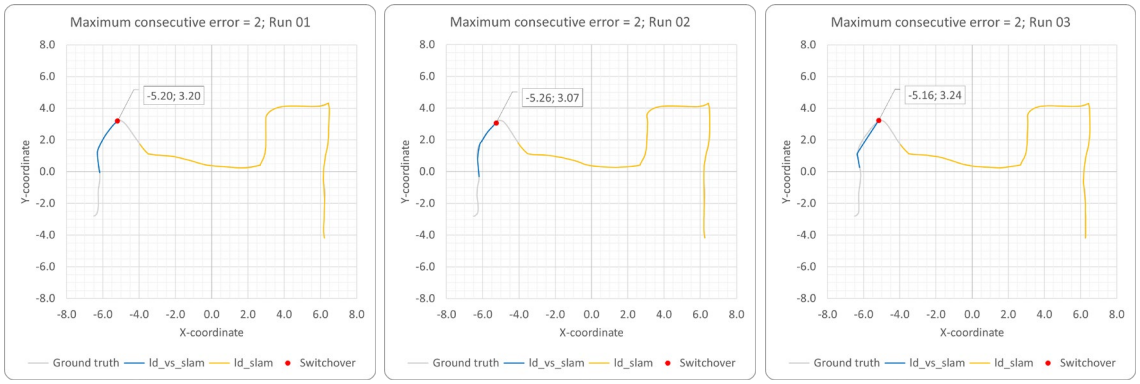


Figure 28: Results from the three runs testing the second configuration setup.

This configuration exhibited inconsistent behaviour during a previous, unrecorded set of simulation runs. During one of these runs, the TurtleBot3 drove past the temporarily dark environment and triggered a localization method switch at the starting point of the roof. This inconsistent behaviour is most likely the result of the behaviour tree's timing, as it was observed that the simulation speed fluctuated when running other programs during the simulations. As a result, the robot navigated past the darker location before the vision data check could be performed twice. Due to the various tasks the autonomous platform is expected to perform simultaneously, such timing inconsistencies are a valid possibility.

The final configuration results can be seen in Figure 29, where a maximum consecutive error of 3 was tested. Similar to the previous two set of tests, the system consistently detected unusable camera data at an average switchover coordinate of X = -0.38 and Y = 0.43. As expected, there is a low variance between the switchover coordinates, as this location provides a consistent darkness level. Like all other runs, the robot successfully navigated to the final waypoint after the localization algorithm switch.

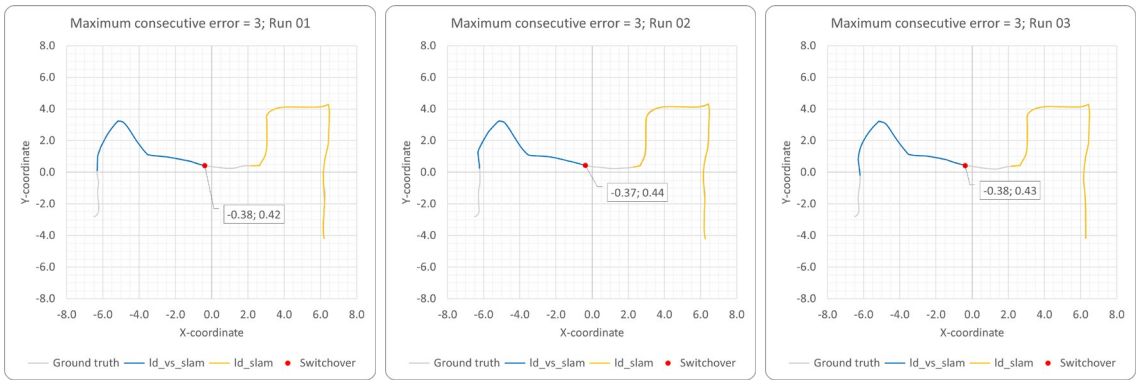


Figure 29: Results from the three runs testing the third configuration setup.

Note that the ld_vs_slam and ld_slam paths do not start at the location where they are selected. This is the result of an initialization phase in which the algorithms collect data about the environment. During this phase, the localization algorithm only uses odometry for the localization process. Once enough data points are collected, the algorithm can start determining an estimated pose based on the generated environment map.

The results from all nine runs are shown in Table 2, which lists the length of the ld_vs_slam and ld_slam trajectories, the switchover coordinate and the maximum position error for each run. This error is measured as an absolute position error (APE) and as a relative position error (RPE). Combining the data from all runs results in an average maximum absolute position error of 0.37 m and an average maximum relative position error of approximately 1.20 %.

	ld_vs_slam path (m)	*ld_slam path (m)*	*Switchover coordinate (m)*	*APE (m)*	*RPE (%)*
Configuration 1					
Run 01	5.34	25.58	-5.87; 2.29	0.42	1.36
Run 02	5.43	25.50	-5.82; 2.40	0.39	1.26
Run 03	5.44	25.37	-5.77; 2.45	0.39	1.27
Configuration 2					
Run 01	6.40	24.48	-5.20; 3.20	0.37	1.20
Run 02	6.31	24.67	-5.26; 3.07	0.32	1.03
Run 03	6.50	24.47	-5.16; 3.24	0.34	1.10
Configuration 3					
Run 01	12.46	18.49	-0.38; 0.42	0.41	1.33
Run 02	12.53	18.50	-0.37; 0.44	0.31	1.00
Run 03	12.42	18.43	-0.38; 0.43	0.38	1.23

Table 2: Simulation test results for the different runs and test configurations.

6.3 Test conclusion

Based on the simulation results (including the observation made during the simulation runs with incorrect data sources), it can be concluded that the proposed solution is successfully capable of consistently detecting unusable sensor data and switching to a new, usable localization method. However, this does require a correct configuration of the maximum consecutive error setting. The simulation results indicate that a maximum consecutive error value of 3 would be optimal, as this number provides the most stable and consistent detection of long-term unusable data.

Note that this optimal value holds true for vision data only. It is possible that more robust sensors, such as LiDAR and RADAR sensors, could operate consistently with a lower value. The same holds true for standalone sensors that are not linked to environmental changes, such as IMUs. Therefore, further tests should be performed per sensor type to determine the optimal value for the maximum consecutive error value.

Another important note is that the optimal value for the maximum consecutive error is linked to the sensor data check loop frequency. If the sensor data is checked once every two seconds (i.e., at a rate of 0.5 Hz), it will most likely have a different optimal value compared to a loop frequency of 10 Hz. Consequently, the maximum consecutive error value should be defined after the loop frequency is set. A higher loop frequency ensures that unusable sensor data can be detected quickly, but it does increase the computational power, and vice versa.

6.4 Future improvements

Despite the successful simulation runs, multiple improvements can be made to improve the proposed solution's usability and performance. These improvements are applicable regardless of whether they are implemented by the RAS Cell themselves, or by an external company. The following paragraphs will elaborate these possible improvements.

Behaviour tree improvements

The first improvement is to divide the functionality of complex nodes over multiple separate nodes where possible. For example, the "navigate to waypoint" node currently performs multiple tasks: it moves the autonomous platform to a waypoint (which includes obstacle avoidance functionalities); it checks whether the waypoint has been reached; and it increases the waypoint counter to properly navigate through the list of waypoints. Ideally, these functionalities should be appointed to individual nodes to increase the transparency of the behaviour tree. Having nodes with only one function or task also makes it easier to adapt a behaviour tree to different platforms, or to reuse existing nodes in other behaviour trees.

Secondly, multiple nodes use Python scripts to execute their functionality. These nodes should be rewritten in C++ code where possible. This makes the codebase more uniform, reduces latency and allows for the segregation of node functionalities. Note that not all Python-dependent nodes can be adapted to C++, since certain ROS 2 functionalities are only available through the Python API. Forcefully converting these Python nodes to C++ nodes might require cumbersome workarounds that could nullify the benefits from the scripting language conversion.

Envisioned functionality implementations

The position corrections and point cloud compression are not implemented in the proposed solution's prototype. However, these functionalities should be implemented to improve the system's position accuracy and to reduce the storage requirements of the SLAM-based localization methods. Note that the tested behaviour tree does include dummy nodes for the position corrections. While not functional, the position of these nodes within the behaviour tree is correct, as the corrections should be applied after each time a waypoint is reached. This is a logical place to perform the corrections, as the robot is stationary for a short period at each waypoint. Correcting the location can lead to position jumps, which could disturb the navigation algorithm if performed during the active navigation process.

A selection model can also be used when multiple localization methods for absolute positioning are implemented on the system. Similar to the current selection model, this process can filter unusable absolute positioning methods and determine the most accurate option. This localization method can then be called when the behaviour tree reaches the node that retrieves the absolute position. The implementation of the point cloud compression varies depending on the used localization methods, since every algorithm uses its own topics and message types to distribute information.

Finally, as mentioned in section 6.3, optimal values for the maximum consecutive error must be analysed for different sensor types. This step can be performed when the autonomous platform's hardware and sensor configuration are determined. Once the sensor configuration is known, the optimal maximum consecutive error value can be determined using the physical setup or a simulated version of the sensor configuration.

Externally dependent improvements

Finally, there are two possible improvements that depend on external parties. First, with a tighter integration between the autonomous platform and the behaviour tree, low-level behaviour such as battery charging or lighting controls can be managed and modified in real-time. This improvement can only be made with the cooperation of the robot's hardware manufacturer.

Another point of improvement is the implementation of real-time interventions for meaningful human control. At this time, behaviour trees created with BehaviorTree.CPP cannot be paused during execution. This means that the behaviour tree must include all tasks the autonomous platform is expected to perform during a mission, before the start of the mission. If the real-time intervention feature is added to the library in the future, end users can pause an active mission to add, remove or modify upcoming tasks. Example use cases for this functionality are to change mission types or modify waypoints and/or observation points during an active mission.

7 Future expandability

Up to now, this thesis has focused on the design and implementation of the proposed solution for Project Sentinel's current state: a single UGV executing a reconnaissance mission in one type of environment. However, as mentioned in section 1.1, the end goal is to expand the capabilities of the autonomous platform in (any combination of) three different directions. This chapter describes how the proposed solution can be adapted to fit these future use cases, and how the future deployment scenarios can be exploited to improve the proposed solution's capabilities and performance.

7.1 Number and type of autonomous platforms

Future versions of the proposed solution must also be applicable to different types of autonomous platforms (e.g., UAVs). To realize this extension of the current state prototype, the following adaptations must be made (regardless of the platform type):

- Extending the list of sensors;
- Extending the list of localization methods;
- Implement new sensor check nodes corresponding to the new sensors;
- Process the new localization methods in the nodes that start and stop the navigation services and localization method algorithm(s).

This list assumes that there is a common base platform that is equipped with ROS 2, the Nav2 navigation stack, the BehaviorTree.CPP library and the Groot package. These software requirements are mandatory to ensure the universal deployment of the proposed solution to other autonomous platforms. When these requirements are met, the behaviour tree from the proposed solution can be integrated as a subtree into the overarching behaviour tree of the autonomous platform, which consists of multiple subtrees that cover the entire robot's functionality management. Because this common software platform is needed to efficiently apply the proposed solution to other platform types, it is recommended for the RAS Cell to define this as a requirement for external companies or organizations that develop the physical platforms.

While the proposed solution should be applicable for different platform types, this thesis has focused on the application for ground vehicles. Drones or other airborne vehicles require a different approach for the proposed solution, as their moving mechanics are significantly different. Additionally, there are major differences between rotor-based drones (i.e., single-rotor or multi-rotor drones) and fixed-wing drones. When UAVs are used for reconnaissance purposes, and they maintain a relatively high cruising altitude, obstacle avoidance is not required. Furthermore, airborne vehicles commonly have smoother travel trajectories due to the lack of ground surfaces. Additionally, the cruising speed of airborne platforms is often higher compared to ground vehicles. Consequently, different localization methods should be utilized. For example, drones can use ground-to-satellite image matching as a continuous, absolute localization method. To provide redundancy, this can be combined with IMU data, visual odometry, and LiDAR or RADAR odometry (depending on the cruising altitude).

As mentioned in the previous paragraph, fixed-wing drones introduce additional challenges. While rotor-based drones can hover at specified waypoints, similar to how a UGV operates, fixed-wing drones can only circle or loiter around a specific location. Subsequently, they cannot pause the active navigation process to switch between localization methods. Instead, a recovery cruise mode could be activated during which the drone only flies in a straight line and reduces its cruising speed. A simplified example of a behaviour tree for fixed-wing drones can be seen in Figure 30.

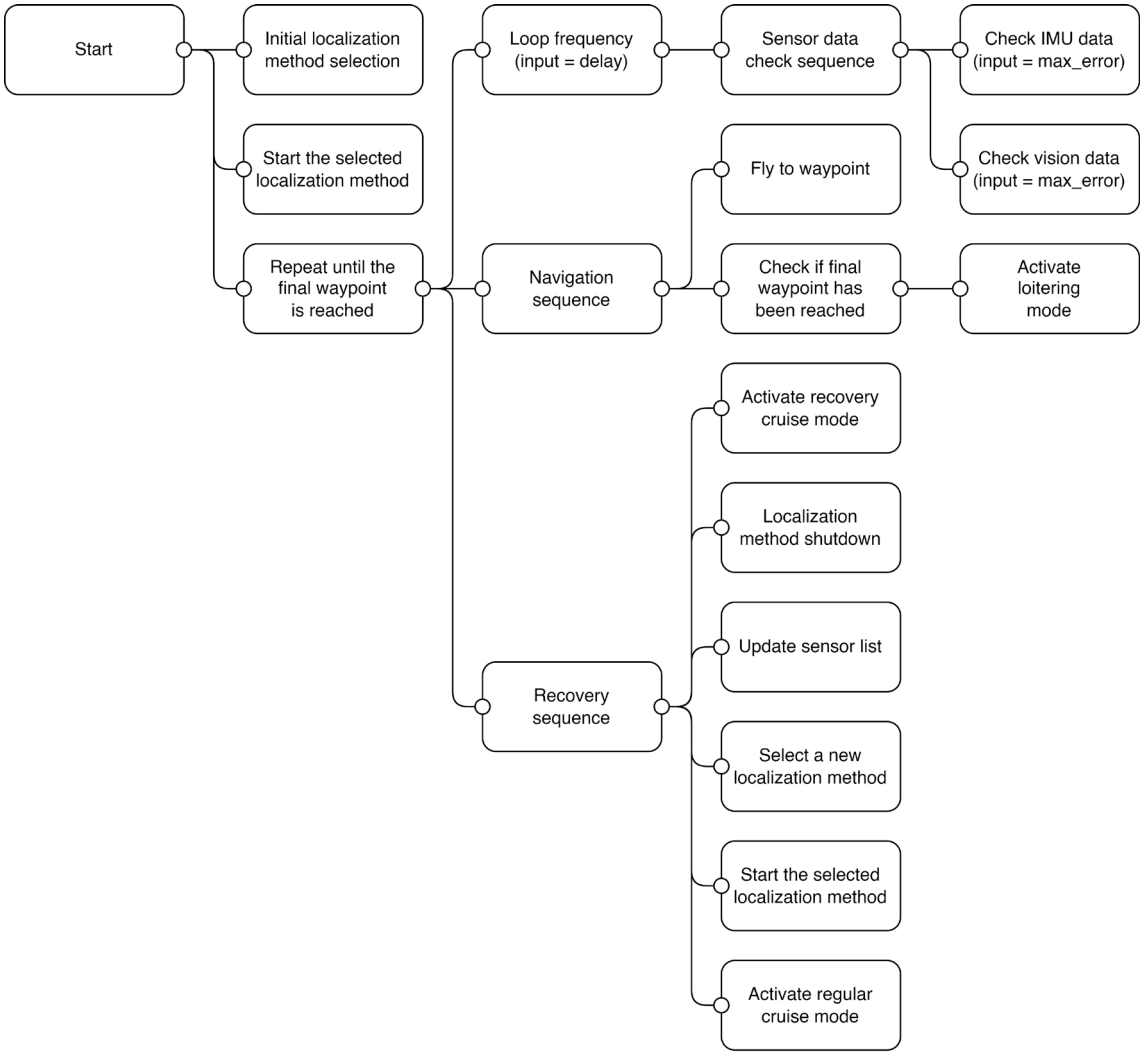


Figure 30: Behaviour tree configuration for fixed-wing drones.

The use of different autonomous platforms provides new possibilities that can improve the proposed solution's performance. For example, UAVs can be used to capture current aerial imagery of an AoI that can then be used by a UGV to perform the position corrections using ground-to-aerial image matching. Similarly, UAVs could generate an environment's magnetic field map, allowing other unmanned vehicles (UXVs) to perform passive navigation and/or position corrections in GNSS-denied environments. This collaborative approach is a valuable aspect that strengthens the proposed solution.

Applying the proposed solution to a group of UXVs requires no modifications to the behaviour tree, assuming that new sensors and localization methods have been implemented as described above. However, any collaboration or communication between the different UXVs will have to be implemented in a separate subtree of the overarching behaviour tree. While the proposed solution's behaviour tree performs a significant part of the autonomous platform's functionalities, it is not complete. Other tasks, such as target observation and status reporting functionalities, are part of a larger tree that encompass all of the platform's functionalities.

Simultaneously deploying multiple autonomous platforms also provides new possibilities that can improve the proposed solution's performance. As described in section 2.1.6, multiple vehicles in a GNSS-denied environment can be used to improve the localization performance of each platform. By creating an ad hoc network of mobile nodes that communicate with each other, and performing timing measurements on these communication signals, the relative position of each platform can be determined. This estimate can be combined with the results from the SLAM algorithms running on each autonomous platform to improve the position estimate of all autonomous robots.

7.2 Military capabilities

Project Sentinel's current state goal is to create an autonomous UGV that performs a reconnaissance mission. Future versions should also be capable of performing other military functions, such as engaging specified targets. While this is a significant extension of the platform's capabilities, it does not require any modifications to the proposed solution. Both the behaviour tree and the adaptive sensor fusion network are focused on the localization and navigation functionalities. However, extending the autonomous platform with more military capabilities will require modifications to the overarching behaviour tree. New subtrees will have to be added that control these new military capabilities. An example of how this structure with behaviour trees and subtrees functions, can be seen in Figure 31. Here, each subtree node is an entire behaviour tree in itself. For example, the "Subtree 03 - Waypoint navigation" node is the entire subtree as previously shown in Figure 25 in section 5.4.2.

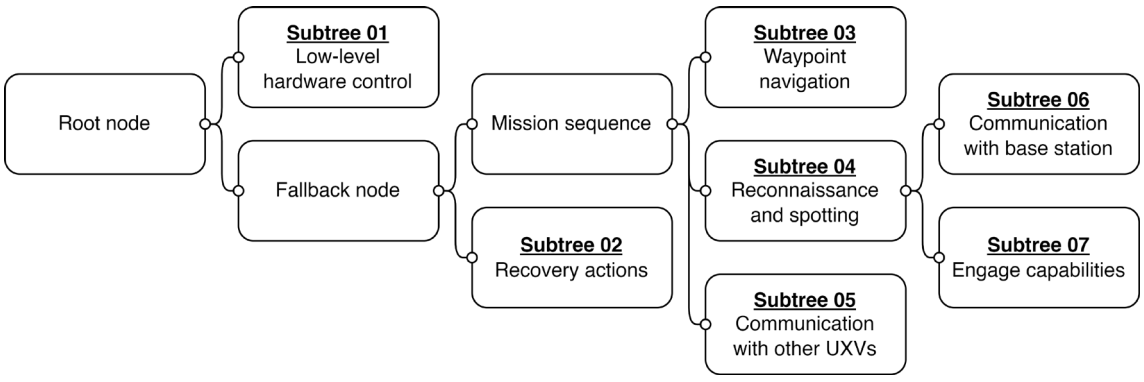


Figure 31: Overarching behaviour tree with subtrees as nodes for specific functionalities.

7.3 Deployment environments

Increasing the types of environments in which the autonomous platform is deployed, does not directly require modifications to the proposed solution. However, it is likely that challenging environments for GNSS-denied navigation (e.g., desert or polar areas) benefit from specialized sensors and/or localization methods. Implementing these can be done using the same list of steps as described in the beginning of section 7.1.

The proposed solution's performance can increase through the addition of more specialized localization methods for specific deployment environments. For example, localization within (sub)urban environments can be performed more accurately and efficiently compared to traditional SLAM algorithms with the method presented by Wang and colleagues [33]. This approach uses a 3D LiDAR to detect and map buildings in an environment, which are then compared to OpenStreetMap data, resulting in sub-meter position errors while maintaining real-time performance.

8 Conclusion

The aim of this thesis is to solve the challenges inherent to autonomous, military vehicles with regards to localization and navigation in GNSS-denied environments. Knowing how these obstacles can be overcome is important for Project Sentinel, as it allows the RAS Cell to define a set of underpinned requirements for the outsourced development of (parts of) the autonomous platform. Deploying autonomous military vehicles is a relevant goal for the RNLA, as they allow for the effective deployment of scarce human resources, increased personnel safety and they aid in achieving battlespace supremacy.

A proposed solution is defined to overcome the challenges and provide value for Project Sentinel. The solution consists of an adaptive sensor fusion framework that selects usable sensors and localization methods based on mission parameters and the deployment environment. This framework combines the researched concepts of hard sensor fusion and predetermined sensor fusion. Furthermore, a behaviour tree controller is used to integrate this framework with the autonomous platform's navigation capabilities. A behaviour tree is chosen for its flexibility, which allows the proposed solution to be extended and applied to other autonomous platforms in the future. The proposed solution is capable of autonomous waypoint navigation while continuously checking the active sensors for unusable data. The navigation process is paused if the adaptive sensor fusion framework detects a persistent sensor error, after which it switches to a different localization method. After a successful switch, the waypoint navigation process is continued until the autonomous platform reaches the final waypoint.

The simulation results indicate that the adaptive sensor fusion framework is capable of consistently detecting unusable sensor data. Moreover, the behaviour tree demonstrated successful control of the waypoint navigation process, while being tightly integrated with the adaptive sensor fusion framework. Another important result from the simulation tests is the importance of the sensor check frequency and maximum consecutive error values. These two user-adjustable parameters define the behaviour of the adaptive sensor fusion framework. Consequently, defining suboptimal values can result in undesired behaviour and/or inconsistent detection of unusable sensor data. In addition to this valuable lesson, the simulations validate that the proposed solution can successfully switch between localization methods during the waypoint navigation process, after which the autonomous platform consistently reached the final waypoint.

To conclude, the proposed solution - consisting of an adaptive sensor fusion framework and behaviour tree controller - provides a solid foundation for the autonomous platform. The flexible setup of the behaviour tree allows the design to grow with Project Sentinel as it progresses and aims for a more complete autonomous platform that can be integrated with multiple, different autonomous platform types. Furthermore, the adaptive sensor fusion framework provides a robust and extendable base that can be improved through the addition of new sensor types and specialized localization methods.

9 Discussion and recommendations

This chapter first discusses multiple aspects of the thesis and proposed solution. Then, the second part of this chapter presents a set of recommendations for future research.

9.1 Discussion

The first part of this section reflects upon the proposed solution by comparing it with the results from the theoretical research. Then, the assumptions made throughout the report are discussed. Next, other applications for the proposed solution are reviewed.

9.1.1 Theoretical research compared to the proposed solution

The first part of the proposed solution consists of an adaptive sensor fusion framework. The theoretical research indicated that there is no single sensor-localization method combination that is capable of robust localization and navigation. Hence, the autonomous platform must be equipped with multiple sensors and localization methods to fill the functionality gaps and provide a level of redundancy.

Choosing the most suitable localization method is initially done using the predetermined sensor fusion concept, as this provides an efficient selection process at the start of the mission. The proposed solution first generates a list of usable sensors, which is done using the mission parameters and environment variables. Next, a list of suitable localization methods is created using the data from the previous steps. Finally, the most accurate localization method is selected from this list.

While the autonomous platform is navigating through waypoints, the adaptive sensor fusion framework continuously analyses the active sensors' data. Depending on the parameters defined in the corresponding scripts, a sensor's data can be labelled as unusable when it goes outside the defined values. If this error is persistent according to the user-defined limit, then the sensor will be disabled. This behaviour is consistent with the concept of hard sensor fusion, which provides an effective implementation of adaptive sensor fusion during the mission execution.

While the soft sensor fusion concept is not directly implemented, there are localization methods that are based on this approach. As mentioned before, the concept of soft sensor fusion is similar to the use of an EKF for the fusion of position estimates. Therefore, by implementing EKF-based localization methods, the soft sensor fusion approach can still be integrated into the proposed solution.

The research on behaviour trees indicated that they provide a more flexible approach to the control of robotic behaviour than alternatives. During the design of the proposed solution's behaviour tree, this advantage was indeed experienced. The node-based structure allowed for rapid prototyping and testing of various behaviour tree setups, while minimizing the use of scripts and code. Using Groot's graphical interface made it easy to analyse, debug and improve the robot's behaviour in a short-cycle process. This experience validates the theory and strengthens the argument for using behaviour trees to control the autonomous platform's (future) behaviour.

9.1.2 Assumption analysis

The following paragraphs will elaborate on the assumptions that have been made throughout this project. First, the proposed solution is designed with the assumption that there is no usable GNSS signal in the deployment environment. This assumption is based on valid concerns regarding the availability of GNSS signals. However, it is not a given that the autonomous platform will always operate in GNSS-denied environments. Since usable GNSS signals can provide efficient and accurate absolute position estimates, it is still advised to equip the autonomous platform with a GNSS receiver. This allows the robot to use GNSS signals for position corrections when possible.

Other assumptions are made about the detectability of LiDAR, RADAR and structured light-based sensors. Here, the detection risk of active sensors is analysed using basic testing equipment and theoretical research, due to the unavailability of proper testing equipment and environments. Furthermore, the assumptions regarding the detection risk err on the side of caution, as the mission integrity depends on remaining undetected.

The simulations on precipitation resistance of vision-based localization methods also include two assumptions. Here, it is assumed that visual feature extraction-based algorithms are negatively impacted by precipitation, both in terms of position accuracy and computational speed. On the other hand, semantic segmentation-based solutions are assumed to only suffer from increased computation times. These assumptions are made based on theoretical research and simulation runs in MATLAB and an Unreal Engine environment. While these two sources provided similar outcomes, real-world tests could be performed to provide a more definitive conclusion on the precipitation resistance of vision-based localization methods.

Finally, an assumption was made on the impact of computational loads on the behaviour tree's timing. During an unlogged simulation run, it was observed that the simulation speed, and thus the timing of the behaviour tree, reduced when multitasking during the simulations. This led to unexpected behaviour that did not occur in other test runs. Because this inconsistent behaviour was only observed while multitasking, the assumption was made that multitasking was the most likely cause. Since the autonomous platform will perform more functionalities besides localization and navigation, it is still important to take into account that multitasking could result in inconsistent behaviour tree timing, resulting in unexpected and possibly undesired behaviour.

9.1.3 Other use cases for the proposed solution

While the proposed solution is designed for autonomous, military robots operating in GNSS-denied environments, its functionality can be applied in other use cases. Examples include: autonomous cars, autonomous agricultural machinery, and autonomous mining trucks. While the aforementioned autonomous platforms often operate in environments with a usable GNSS signal, they still require a system that is capable of navigating from A to B, without hitting any obstacles. Additionally, while not required, these platforms can still benefit from a robust system capable of switching between sensors and localization methods when needed, as this can reduce downtime or accidents.

9.2 Recommendations

There are multiple research topics related to the proposed solution that did not fit within the scope of this assignment. The following sections elaborate on these research topics and describe the possible benefits that could be achieved.

9.2.1 Machine learning applications

There are two possible applications for machine learning (ML) algorithms in the proposed solution. First, ML can be used to identify the cause of unusable sensor data. Currently, the sensor data from active sensors is analysed using a script that checks multiple data characteristics. While this method works to detect unusable sensor data, ML algorithms could be used to determine if unusable data is generated due to a hardware failure, software error, or a change in the environment. This is valuable information, as it can be used to either permanently or temporarily disable a sensor. For example, if a ML algorithm determines that a sensor is defective, it can be disabled for the entire mission. This means that the sensor does not have to be periodically checked throughout the mission. However, if unusable data is the result of a changing environment, then the sensor should be tagged as being temporarily unusable. By periodically checking this sensor's data and/or the environment, it can be re-enabled when possible. Using ML for this use case can increase the position accuracy while reducing power usage.

Secondly, ML algorithms can be used to adjust the characteristics of sensors and localization methods. Currently, these characteristics are estimated based on research. By adjusting the characteristics of each sensor and localization method, and testing these different configurations in simulations, ML algorithms can perfect the values for each characteristic. These values could even be optimized for specified goals, such as position accuracy or mission execution speed. Utilizing ML algorithms for this application could lead to a more efficient localization method selection and improved position accuracy.

9.2.2 Environment-dependent localization methods

The proposed solution uses RTAB-Map as the SLAM algorithm. While this algorithm works in all environments with the correct sensor configuration, section 2.1 indicates that there are other SLAM algorithms that are specifically optimized for certain environments, such as urban areas. Implementing multiple SLAM algorithms that are optimized for specific environments can yield a more efficient and accurate localization and navigation process. Benefits of this approach could include: lower computational requirements, improved position accuracy and lower energy consumption. For example: deploying the OpenStreetMap-based localization method from Wang et al. [33] in urban environments, combined with a basic obstacle avoidance algorithm, can provide sub-meter accuracy while only using two sensors (3D LiDAR with an IMU).

In theory, this solution could result in a more efficient mission execution. However, the implementation will require a more complex selection algorithm, and several tests on the usability of different SLAM algorithms in multiple environments. Therefore, research should be performed to analyse whether this strategy can result in a net benefit.

9.2.3 Low-light localization methods

Currently, the proposed solution is capable of autonomous localization, navigation and obstacle avoidance under low-light conditions using a LiDAR sensor. Since regular cameras are not usable in the dark, LiDAR sensors are the only option for such conditions. However, if the LiDAR sensor is also unusable in low-light situations, then the robot will not be able to autonomously perform the aforementioned tasks. This situation can occur when the LiDAR sensor has a failure, or when active sensors cannot be used due to hostile presence. To overcome this issue, the RAS Cell should research visual SLAM methods that are capable of operating under low-light conditions.

Wang and colleagues [125] presented a visual SLAM algorithm that combines thermal infrared images with regular camera images for visual odometry and point cloud generation. While this method allowed the researchers to successfully complete four low-light test sequences, the achieved position accuracy does not meet that of regular SLAM algorithms. Another method is presented by Benbihi et al. [126], who propose a SLAM algorithm for low-light situations that uses semantic segmentation and object detection for navigation. Similar to the first solution, this method outperforms regular SLAM algorithms in low-light conditions, since the former often fail to complete the test sequence under such conditions, but the position accuracy is less than regular SLAM.

Aladem and colleagues [127] presented the use of image processing to improve the recognizability of features in low-light image captures made with regular cameras. While this method does improve the position accuracy of visual SLAM algorithms, it comes at the cost of increased computational requirements due to the required image processing. Adding the most accurate image processing algorithm resulted in an additional ~120 ms of processing time for each frame.

9.2.4 Human interaction and live updates

In its current form, the proposed solution does not allow for human interactions with the autonomous platform during the execution of a mission. However, in a real deployment scenario there could be multiple reasons for which human interaction is required or preferable. For example, if the selection algorithm cannot determine a usable localization method due to a change in the environment, then a human operator should be able to directly interact with the robot to provide further instructions. Additionally, with live interaction capabilities enabled, a commander would be able to change the mission parameters and/or goal(s) while a robot is already executing the mission.

To incorporate such a system, possible interactions between a human operator and the autonomous platform must be defined. This could be done through brainstorming sessions with end users and members of the RAS Cell. Then, based on the list of possible interactions, a technical implementation concept can be defined, created, tested, and improved where necessary. An important factor in this interaction system will be the communication, as the transmission of data could alert hostiles of the robot's presence (as mentioned in section 1.3.3).

References

[1] "RAS eenheid - Op zoek naar de meerwaarde van robotic warfare", WerkenbijDefensie.nl [Online]. Available: https://werkenbijdefensie.nl/onze-technici/ras-eenheid.

[2] The Hague Centre for Strategic Studies, "The Military Applicability of Robotic and Autonomous Systems", HCSS Security, The Hague, 2021.

[3] D. Amoroso and G. Tamburrini, "Autonomous Weapons Systems and Meaningful Human Control: Ethical and Legal Issues", Current Robotics Reports, vol. 1, no. 4, pp. 187-194, 2020. Available: 10.1007/s43154-020-00024-3.

[4] P. Dutta, T. Halder, S. Banerjee, A. Basak, S. Nanda and D. Chakravarty, "Analysis of jamming and anti jamming techniques for Galileo GNSS", Materials Today: Proceedings, 2022. Available: 10.1016/j.matpr.2022.03.009.

[5] E. Ochin and Ł. Lemieszewski, "Security of GNSS", GPS and GNSS Technology in Geosciences, pp. 51-74, 2021. Available: 10.1016/b978-0-12-818617-6.00015-9.

[6] A. Grant, P. Williams, G. Shaw, M. De Voy and N. Ward, "Understanding GNSS availability and how it impacts maritime safety", in International Technical Meeting of the Institute of Navigation, San Diego, 2011.

[7] A. Kumar, S. Kumar, P. Lal, P. Saikia, P. Srivastava and G. Petropoulos, "Introduction to GPS/GNSS technology", GPS and GNSS Technology in Geosciences, pp. 3-20, 2021. DOI: 10.1016/b978-0-12-818617-6.00001-9.

[8] Technology for the United States Navy and Marine Corps 2000-2035, 2nd ed. Washington, D.C.: National Academy of Sciences, 1997.

[9] A. Benaskeur and F. Rhéaume, "Adaptive data fusion and sensor management for military applications", Aerospace Science and Technology, vol. 11, no. 4, pp. 327-338, 2007. Available: 10.1016/j.ast.2007.01.005.

[10] "X-300 Integrator onbemand verkenningssysteem", Defensie.nl, 2020. [Online]. Available: https://www.defensie.nl/onderwerpen/materieel/vliegtuigen-en-helikopters/x-300-integrator-onbemand-verkenningssysteem.

[11] "Integrator Specifications", Insitu Products, 2020. [Online]. Available: https://www.insitu.com/products/integrator.

[12] Ministerie van Defensie, Joint Doctrine Publicatie 5 Commandovoering. Ministerie van Defensie, 2012, pp. 36-37.

[13] P. Whissell, "Reveal The Invisible", SBQuantum [Online]. Available: https://sbquantum.com/.

[14] A. Canciani, "Magnetic navigation", Redondo Beach, 2018.

[15] “Alternative Navigation Systems - Magnetic Anomaly Aided Navigation", Honeywell Aerospace [Online]. Available: https://aerospace.honeywell.com/us/en/learn/products/sensors/alternative-navigation-systems.

[16] J. Raquet, J. Shockley and K. Fisher, "Determining Absolute Position Using 3-Axis Magnetometers and the Need for Self-Building World Models", in Navigation Sensors and Systems in Global Navigation Satellite Systems (GNSS) Degraded and Denied Environments, 2013.

[17] "Alternative Navigation Systems - Celestial Navigation", Honeywell Aerospace [Online]. Available: https://aerospace.honeywell.com/us/en/learn/products/sensors/alternative-navigation-systems.

[18] T. Stephens, R. Merritt, B. Schipper, P. Samanant and H. International, "Honeywell Celestial-Aided Navigation Performance in GPS-Denied Ground Testing at PNTAX 2020", in Joint Navigation Conference, San Diego, 2021.

[19] X. Wei, C. Cui, G. Wang and X. Wan, "Autonomous positioning utilizing star sensor and inclinometer", Measurement, vol. 131, pp. 132-142, 2019. DOI: 10.1016/j.measurement.2018.08.061.

[20] X. Ning and J. Fang, "A new autonomous celestial navigation method for the lunar rover", Robotics and Autonomous Systems, vol. 57, no. 1, pp. 48-54, 2009. Available: 10.1016/j.robot.2008.02.006.

[21] Polaris SkyPASS. Huntsville: Polaris Sensor Technologies, Inc., 2020.

[22] A. Lompado, T. Aycock and B. Wheeler, "Using Atmospheric Polarization Patterns for Azimuth Sensing", Naval Surface Warface Center, Dahlgren, 2014.

[23] J. Zhang, J. Yang, S. Wang, X. Liu, Y. Wang and X. Yu, "A self-contained interactive iteration positioning and orientation coupled navigation method based on skylight polarization", Control Engineering Practice, vol. 111, p. 104810, 2021. Available: 10.1016/j.conengprac.2021.104810.

[24] T. Aycock, A. Lompado, T. Wolz and D. Chenault, "Passive optical sensing of atmospheric polarization for GPS denied operations", SPIE Proceedings, 2016. Available: 10.1117/12.2227140.

[25] K. Song et al., "Navigation Control Design of a Mobile Robot by Integrating Obstacle Avoidance and LiDAR SLAM," 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018, pp. 1833-1838, DOI: 10.1109/SMC.2018.00317.
[26] Y. Zhuang, S. Yang, X. Li and W. Wang, "3D-laser-based visual odometry for autonomous mobile robot in outdoor environments," 2011 3rd International Conference on Awareness Science and Technology (iCAST), 2011, pp. 133-138, DOI: 10.1109/ICAwST.2011.6163127.
[27] "Lidar Mapping Technology | Exyn Technologies", Exyn.com. [Online]. Available: https://www.exyn.com/technology/lidar-mapping-technology.
[28] "Autonomous Drone Navigation System Ends Reliance on GPS | NASA Spinoff", Spinoff.nasa.gov, 2020. [Online]. Available: https://spinoff.nasa.gov/Spinoff2020/ps_5.html.
[29] System Specifications. Philadelphia: Exyn Technologies.
[30] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley and C. Stachniss, "SuMa++: Efficient LiDAR-based Semantic SLAM," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 4530-4537, DOI: 10.1109/IROS40897.2019.8967704.
[31] F. Yan, J. Wang, G. He, H. Chang and Y. Zhuang, "Sparse semantic map building and relocalization for UGV using 3D point clouds in outdoor environments", Neurocomputing, vol. 400, pp. 333-342, 2020. Available: 10.1016/j.neucom.2020.02.103.
[32] B. Suger and W. Burgard, "Global outer-urban navigation with OpenStreetMap," 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 1417-1422. Available: 10.1109/ICRA.2017.7989169.
[33] P. Wang, L. Mihaylova, P. Bonnifait, P. Xu and J. Jiang, "Feature-refined box particle filtering for autonomous vehicle localisation with OpenStreetMap", Engineering Applications of Artificial Intelligence, vol. 105, p. 104445, 2021. Available: 10.1016/j.engappai.2021.104445.
[34] "Honeywell Radar Velocity System", Honeywell Navigation & Sensors. [Online]. Available: https://aerospace.honeywell.com/us/en/learn/products/sensors/radar-velocity-system.
[35] E. Quist, P. Niedfeldt and R. Beard, "Radar odometry with recursive-RANSAC", IEEE Transactions on Aerospace and Electronic Systems, vol. 52, pp. 1618-1630, 2016. DOI: 10.1109/taes.2016.140829.
[36] A. Scannapieco, A. Renga, G. Fasano and A. Moccia, "Experimental Analysis of Radar Odometry by Commercial Ultralight Radar Sensor for Miniaturized UAS", Journal of Intelligent & Robotic Systems, vol. 90, no. 3-4, pp. 485-503, 2017. Available: 10.1007/s10846-017-0688-1.
[37] "Alternative Navigation Systems - Vision Navigation", Navigation & Sensors. [Online]. Available: https://aerospace.honeywell.com/us/en/learn/products/sensors/alternative-navigation-systems.
[38] "P3DR | Precision 3D Registration", Maxar.com. [Online]. Available: https://www.maxar.com/products/precision-3d-registration.
[39] "5 Small Unmanned Products to Watch This Year", Lockheed Martin, 2021. [Online]. Available: https://www.lockheedmartin.com/en-us/news/features/2021/5-small-unmanned-products-to-watch-this-year.html.
[40] A. Viswanathan, B. R. Pires and D. Huber, "Vision based robot localization by ground to satellite matching in GPS-denied situations," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 192-198, DOI: 10.1109/IROS.2014.6942560.
[41] B. Patel, "Visual Localization for UAVs in Outdoor GPS-denied Environments", Master of Applied Science, University of Toronto Institute for Aerospace Studies, 2019.
[42] G. Floros, B. van der Zander and B. Leibe, "OpenStreetSLAM: Global vehicle localization using OpenStreetMaps," 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 1054-1059, DOI: 10.1109/ICRA.2013.6630703.
[43] "Real-Time High-Precision Navigation and Mapping", SRI International, 2020. [Online]. Available: https://www.sri.com/case-studies/real-time-high-precision-navigation-and-mapping/.
[44] H. Chiu, M. Sizintsev, X. Zhou, P. Miller, S. Samarasekera and R. Kumar, "Sub-meter vehicle navigation using efficient pre-mapped visual landmarks", 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016. Available: 10.1109/itsc.2016.7795602.
[45] Y. Fan, Q. Zhang, Y. Tang, S. Liu and H. Han, "Blitz-SLAM: A semantic SLAM in dynamic environments", Pattern Recognition, vol. 121, p. 108225, 2022. DOI: 10.1016/j.patcog.2021.108225.
[46] L. Xiao, J. Wang, X. Qiu, Z. Rong and X. Zou, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment", Robotics and Autonomous Systems, vol. 117, pp. 1-16, 2019. Available: 10.1016/j.robot.2019.03.012.
[47] A. Couturier and M. Akhloufi, "A review on absolute visual localization for UAV", Robotics and Autonomous Systems, vol. 135, p. 103666, 2021. Available: 10.1016/j.robot.2020.103666.

[48] E. Yao, H. Zhang, H. Xu, H. Song and G. Zhang, "Robust RGB-D visual odometry based on edges and points", Robotics and Autonomous Systems, vol. 107, pp. 209-220, 2018. Available: 10.1016/j.robot.2018.06.009.

[49] A. Badshah et al., "Vehicle navigation in GPS denied environment for smart cities using vision sensors", Computers, Environment and Urban Systems, vol. 77, p. 101281, 2019. Available: 10.1016/j.compenvurbsys.2018.09.001.

[50] N. Haigh, "Navigation via Signals of Opportunity (NAVSOP)", BAE Systems. [Online]. Available: https://www.baesystems.com/en/product/navigation-via-signals-of-opportunity-navsop.

[51] N. Souli, P. Kolios and G. Ellinas, "Relative Positioning of Autonomous Systems using Signals of Opportunity", 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), 2020. Available: 10.1109/vtc2020-spring48590.2020.9128912.

[52] K. Ismail, R. Liu, J. Zheng, C. Yuen, Y. Guan and U. Tan, "Mobile Robot Localization Based On Low-Cost LTE And Odometry In GPS-Denied Outdoor Environment", in Proceeding of the IEEE International Conference on Robotics and Biomimetics, Dali, 2019, pp. 2338-2343.

[53] S. Singh and P. Sujit, "Landmarks based path planning for UAVs in GPS-denied areas.", IFAC-PapersOnLine, vol. 49, no. 1, pp. 396-400, 2016. Available: 10.1016/j.ifacol.2016.03.086.

[54] M. Page and T. L. Wickramarathne, "Enhanced Situational Awareness with Signals of Opportunity: RSS-based Localization and Tracking," 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019, pp. 3833-3838, DOI: 10.1109/ITSC.2019.8917116.

[55] M. Miller, J. Raquet and M. Uijt de Haag, "Navigating in Difficult Environments: Alternatives to GPS - 2", 2009.

[56] S. Nightingale, "No GPS, no problem: Next-generation navigation", University of California, 2016. [Online]. Available: https://www.universityofcalifornia.edu/news/no-gps-no-problem-next-generation-navigation.

[57] "WarLoc™ - Robotic Research", Robotic Research. [Online]. Available: https://www.roboticresearch.com/WarLoc/.

[58] S. Freedberg Jr., "WarLoc: A GPS Alternative On Your Boot - Breaking Defense", Breaking Defense, 2020. Available: https://breakingdefense.com/2020/01/warloc-a-gps-alternative-on-your-boot/.

[59] "NEON Personnel Tracker Military", TRX Systems. [Online]. Available: https://www.trxsystems.com/pt-mil.html.

[60] TRX NEON Personnel Tracker – Military Dismount. TRX Systems, Inc., 2021.

[61] J. Russell, M. Ye, B. Anderson, H. Hmam and P. Sarunic, "Cooperative Localisation of a GPS-Denied UAV in 3-Dimensional Space Using Direction of Arrival Measurements.", IFAC-PapersOnLine, vol. 50, no. 1, pp. 8019-8024, 2017. Available: 10.1016/j.ifacol.2017.08.1226.

[62] H. Zhang, S. Chen, C. Seow and S. Tan, "Localization in GPS Denied Environment", in Advances in Networks, Security and Communications: Reviews, Vol. 2, 2nd ed., S. Yurish, Ed. International Frequency Sensor Association (IFSA) Publishing, 2020, pp. 209-224.

[63] X. Ma, S. Djouadi, P. Crilly, S. Sahyoun and S. Smith, "Improved localization in GPS-denied environments using an autoregressive model and a generalized linear model", 2011 - MILCOM 2011 Military Communications Conference, 2011. Available: 10.1109/milcom.2011.6127768.

[64] A. Coluccia, F. Ricciato and G. Ricci, "Positioning Based on Signals of Opportunity", IEEE Communications Letters, vol. 18, pp. 356-359, 2014. DOI: 10.1109/lcomm.2013.123013.132297.

[65] "Autonomous Shuttle", Unmanned.tamu.edu. [Online]. Available: https://unmanned.tamu.edu/projects/autonomous-shuttle/.

[66] J. Lääne, "How Starship Delivery Robots know where they are going", STARSHIP, 2020.

[67] "Oshkosh Autonomous Technology - Oshkosh Defense", Oshkosh Defense. [Online]. Available: https://oshkoshdefense.com/technology/oshkosh-autonomous-technology/.

[68] "Oshkosh TerraMax Unmanned Ground Vehicle Technology", Army Technology, 2013. Available: https://www.army-technology.com/projects/oshkosh-terramax-unmanned-ground-vehicle/.

[69] F. Wang, J. Cui, B. Chen and T. Lee, "A Comprehensive UAV Indoor Navigation System Based on Vision Optical Flow and Laser FastSLAM", Acta Automatica Sinica, vol. 39, no. 11, pp. 1889-1899, 2013. Available: 10.1016/s1874-1029(13)60080-4.

[70] Y. Shi et al., "Fusion of a panoramic camera and 2D laser scanner data for constrained bundle adjustment in GPS-denied environments", Image and Vision Computing, vol. 40, pp. 28-37, 2015. Available: 10.1016/j.imavis.2015.06.002.
[71] A. Viswanathan, B. R. Pires and D. Huber, "Vision-based robot localization across seasons and in remote locations," 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 4815-4821, DOI: 10.1109/ICRA.2016.7487685.
[72] D. Maturana, P. Chou, M. Uenoyama and S. Scherer, "Real-Time Semantic Mapping for Autonomous Off-Road Navigation", Field and Service Robotics, pp. 335-350, 2017. Available: 10.1007/978-3-319-67361-5_22.
[73] M. Mostafa, S. Zahran, A. Moussa, N. El-Sheimy and A. Sesay, "Radar and Visual Odometry Integrated System Aided Navigation for UAVS in GNSS Denied Environment", Sensors, vol. 18, no. 9, p. 2776, 2018. Available: 10.3390/s18092776.
[74] Defence Research and Development Canada, "Sensor management for tactical surveillance operations", Defence R&D Canada – Valcartier, Saint-Gabriel-de-Valcartier, 2007.
[75] "Puck Lidar Sensor, High-Value Surround Lidar | Velodyne Lidar", Velodyne Lidar. [Online]. Available: https://velodynelidar.com/products/puck/.
[76] "Police laser jamming Laser Defusers | K40 Electronics", K40.com. [Online]. Available: https://k40.com/product/laser-defusers/.
[77] Israel Military Industries Systems Ltd., "Iron Fist Light Configuration", 2017.
[78] Elbit Systems, "Iron Fist Series of Active Protection Systems", 2019.
[79] Thales Optronics, Thales Group UK, "Laser Warning System", 2012.
[80] Ford Project Nightonomy. Arizona: Ford, 2016.
[81] Camacho, J., 2000. Federal Radar Spectrum Requirements. [ebook] U.S. Department of Commerce. Available at: https://www.ntia.doc.gov/files/ntia/publications/ntia00-40.pdf
[82] Lacomme, P., Marchais, J., Hardange, J. and Normant, E., 2007. Air and Spaceborne Radar Systems: An Introduction. Burlington: Elsevier.
[83] 2019. IEEE Standard Letter Designations for Radar-Frequency Bands. [online] p.10. Available: https://ieeexplore.ieee.org/document/8999849.
[84] "Radenso Pro M Radar Detector with 1 year Radar Ticket Free Guarantee", Radenso.com. [Online]. Available: https://radenso.com/products/radenso-pro-m-radar-detector.
[85] Xbox 360 Kinect on the infrared spectrum. NickInTimeFilms, 2011.
[86] E. Reagan, Xbox Kinect and Infrared Camera. Photography Bay, 2012.
[87] M. Newman, B. Gatersleben, K. Wyles and E. Ratcliffe, "The use of virtual reality in environment experiences and the importance of realism", Journal of Environmental Psychology, vol. 79, p. 101733, 2022. Available: 10.1016/j.jenvp.2021.101733.
[88] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, DOI: 10.1109/TRO.2015.2463671.
[89] T. Curcic, "Quantum-Assisted Sensing and Readout (QuASAR) (Archived)", DARPA. [Online]. Available: https://www.darpa.mil/program/quantum-assisted-sensing-and-readout.
[90] DARPA, "Emerging Microsystem Technologies for Autonomous Positioning, Navigation, and Timing (PNT)".
[91] F. Kobayashi, F. Arai, T. Fukuda, M. Onoda and Y. Hotta, "Sensor Selection by Reliability Based on Possibility Measure and Its Application to Grinding Process", Transactions of the Society of Instrument and Control Engineers, 2000. [Online]. Available: https://www.jstage.jst.go.jp/article/sicetr1965/36/3/36_3_290/_article.
[92] O. Cohen and Y. Edan, "A Sensor Fusion Framework for On-Line Sensor and Algorithm Selection," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 3155-3161, Available: 10.1109/ROBOT.2005.1570596.
[93] Y. Jia et al., "Lvio-Fusion: A Self-adaptive Multi-sensor Fusion SLAM Framework Using Actor-critic Method," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 286-293, DOI: 10.1109/IROS51168.2021.9635905.

[94] G. Lee, S. Jung and D. Han, "An Adaptive Sensor Fusion Framework for Pedestrian Indoor Navigation in Dynamic Environments", IEEE Transactions on Mobile Computing, vol. 20, no. 2, pp. 320-336, 2021. Available: 10.1109/tmc.2019.2946809.

[95] H. Hamadi, B. Lussier, I. Fantoni and C. Francis, "Data fusion fault tolerant strategy for a quadrotor UAV under sensors and software faults", ISA Transactions, 2022. DOI: 10.1016/j.isatra.2022.01.007.

[96] "EKF3 Affinity and Lane Switching — Copter documentation", Ardupilot.org, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-ek3-affinity-lane-switching.html.

[97] H. Sankhla, "GSoC 20: EKF3 Affinity and Lane Switching (Merged!)", ArduPilot, 2020. [Online]. Available: https://discuss.ardupilot.org/t/gsoc-20-ekf3-affinity-and-lane-switching-merged/61320.

[98] C. Silva, P. Borges and J. Castanho, "Environment-Aware Sensor Fusion using Deep Learning", in 16th International Conference on Informatics in Control, Prague, 2019.

[99] N. Kano, N. Seraku, F. Takahashi and S. Tsuji, "Attractive Quality and Must-Be Quality", Journal of The Japanese Society for Quality Control, vol. 14, no. 2, pp. 147-156, 1984. Available: https://doi.org/10.20684/quality.14.2_147.

[100] C. Barrington-Leigh and A. Millard-Ball, "The world's user-generated road map is more than 80% complete", PLOS ONE, vol. 12, no. 8, p. e0180698, 2017. Available: 10.1371/journal.pone.0180698.

[101] European Space Agency, "Sentinel-2 - Missions - Sentinel Online - Sentinel Online", Sentinel.esa.int [Online]. Available: https://sentinel.esa.int/web/sentinel/missions/sentinel-2.

[102] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation", Journal of Field Robotics, vol. 36, no. 2, pp. 416-446, 2018. Available: 10.1002/rob.21831.

[103] R. Mur-Artal and J. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras", IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, 2017. Available: 10.1109/tro.2017.2705103.

[104] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera and J. Jacobo Berlles, "S-PTAM: Stereo Parallel Tracking and Mapping", Robotics and Autonomous Systems, vol. 93, pp. 27-42, 2017. Available: 10.1016/j.robot.2017.03.019.

[105] I. Ali et al., "FinnForest dataset: A forest landscape for visual SLAM", Robotics and Autonomous Systems, vol. 132, p. 103610, 2020. Available: 10.1016/j.robot.2020.103610.

[106] M. Jiménez-Martínez, M. Farjas-Abadia and N. Quesada-Olmo, "An Approach to Improving GNSS Positioning Accuracy Using Several GNSS Devices", Remote Sensing, vol. 13, no. 6, p. 1149, 2021. Available: 10.3390/rs13061149.

[107] I. Ullah, S. Qian, Z. Deng and J. Lee, "Extended Kalman Filter-based localization algorithm by edge computing in Wireless Sensor Networks", Digital Communications and Networks, vol. 7, no. 2, pp. 187-195, 2021. Available: 10.1016/j.dcan.2020.08.002.

[108] S. Tzafestas, "Mobile Robot Localization and Mapping", Introduction to Mobile Robot Control, pp. 479-531, 2014. Available: 10.1016/b978-0-12-417049-0.00012-2.

[109] S. Li, "Testing and Evaluation of Collaborative SLAM", Graduate, MSc., Ohio State University, 2017.

[110] T. Moore, "robot_localization wiki", docs.ros.org, 2016. [Online]. Available: https://docs.ros.org/en/noetic/api/robot_localization/html/index.html.

[111] Colledanchise M, Ögren P. "Behavior trees in robotics and AI: An introduction". CRC Press; 2018 Jul 20. Available: https://doi.org/10.48550/arXiv.1709.00084.

[112] D. Faconti, "GitHub - BehaviorTree/Groot: Graphical Editor to create BehaviorTrees. Compliant with BehaviorTree.CPP", GitHub, 2018. [Online]. Available: https://github.com/BehaviorTree/Groot.

[113] D. Faconti, "BehaviorTree.CPP", Behaviortree.github.io, 2018. [Online]. Available: https://behaviortree.github.io/BehaviorTree.CPP/.

[114] "Nav2 Behavior Trees — Navigation 2 1.0.0 documentation", Navigation.ros.org, 2020. [Online]. Available: https://navigation.ros.org/behavior_trees/index.html.

[115] P. Hintjens, "ZeroMQ", Zeromq.org, 2007. [Online]. Available: https://zeromq.org/.

[116] D. Stonier, "GitHub - PyTrees: Python implementation of behaviour trees.", GitHub, 2020. [Online]. Available: https://github.com/splintered-reality/py_trees.

[117] D. Stonier, "PyTrees ROS Viewer: A Qt-Js hybrid application for visualisation of executing or log-replayed behaviour trees in a ROS2 ecosystem.", GitHub, 2020. [Online]. Available: https://github.com/splintered-reality/py_trees_ros_viewer.

[118] J. Cui et al., "PocoNet: SLAM-oriented 3D LiDAR Point Cloud Online Compression Network", _2021 IEEE International Conference on Robotics and Automation (ICRA)_, 2021. Available: 10.1109/icra48506.2021.9561309.
[119] C. Tu, E. Takeuchi, C. Miyajima and K. Takeda, "Continuous point cloud data compression using SLAM based prediction", _2017 IEEE Intelligent Vehicles Symposium (IV)_, 2017. Available: 10.1109/ivs.2017.7995959.
[120] "Draco 3D Data Compression", Draco 3D, 2017. [Online]. Available: https://google.github.io/draco.
[121] J. Paplhám and T. Petříček, "point_cloud_transport", GitHub, 2019. [Online]. Available: https://github.com/paplhjak/point_cloud_transport.
[122] T. Wiemann et al., "Compressing ROS Sensor and Geometry Messages with Draco", 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2019. Available: 10.1109/ssrr.2019.8848965.
[123] J. Dybedal, A. Aalerud and G. Hovland, "Embedded Processing and Compression of 3D Sensor Data for Large Scale Industrial Environments", _Sensors_, vol. 19, no. 3, p. 636, 2019. Available: 10.3390/s19030636.
[124] R. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)", 2011 IEEE International Conference on Robotics and Automation, 2011. Available: 10.1109/icra.2011.5980567.
[125] R. Wang et al., "DVT-SLAM: Deep-Learning Based Visible and Thermal Fusion SLAM", Lecture Notes in Electrical Engineering, pp. 394-403, 2021. Available: 10.1007/978-981-16-3142-9_37.
[126] A. Benbihi, C. Pradalier and O. Chum, "Object-Guided Day-Night Visual Localization in Urban Scenes", 2022. Available: https://doi.org/10.48550/arXiv.2202.04445.
[127] M. Aladem, S. Baek and S. Rawashdeh, "Evaluation of Image Enhancement Techniques for Vision-Based Navigation under Low Illumination", Journal of Robotics, vol. 2019, pp. 1-15, 2019. Available: 10.1155/2019/5015741.
[128] E. Kim and D. Choi, "A 3D Ad Hoc Localization System Using Aerial Sensor Nodes," in IEEE Sensors Journal, vol. 15, no. 7, pp. 3716-3723, July 2015, DOI: 10.1109/JSEN.2015.2420598.
[129] X. Song, X. Li, W. Tang and W. Zhang, "A fusion strategy for reliable vehicle positioning utilizing RFID and in-vehicle sensors", Information Fusion, vol. 31, pp. 76-86, 2016. Available: 10.1016/j.inffus.2016.01.003.
[130] H. Erturk, G. Tuna, T. Mumcu and K. Gulez, "A Performance Analysis of Omnidirectional Vision Based Simultaneous Localization and Mapping", Lecture Notes in Computer Science, pp. 407-414, 2012. Available: 10.1007/978-3-642-31588-6_53.
[131] S. Goh, O. Abdelkhalik and S. Zekavat, "A Weighted Measurement Fusion Kalman Filter implementation for UAV navigation", Aerospace Science and Technology, vol. 28, no. 1, pp. 315-323, 2013. Available: 10.1016/j.ast.2012.11.012.
[132] X. Liu, X. Liu, Y. Yang and W. Zhang, "An INS/Lidar Integrated Navigation Algorithm Based on Robust Kalman Filter", Lecture Notes in Electrical Engineering, pp. 1027-1037, 2021. Available: 10.1007/978-981-15-8155-7_86.
[133] L. Xu, C. Feng, V. Kamat and C. Menassa, "An Occupancy Grid Mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments", Automation in Construction, vol. 104, pp. 230-245, 2019. Available: 10.1016/j.autcon.2019.04.011.
[134] K. C. Saranya, V. P. S. Naidu, V. Singhal and B. M. Tanuja, "Application of vision based techniques for UAV position estimation," 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), 2016, pp. 1-5, DOI: 10.1109/RAINS.2016.7764392.
[135] M. Nahangi, A. Heins, B. McCabe and A. Schoellig, "Automated Localization of UAVs in GPS-Denied Indoor Construction Environments Using Fiducial Markers", Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC), 2018. Available: 10.22260/isarc2018/0012.
[136] A. Wu, E. Johnson, M. Kaess, F. Dellaert and G. Chowdhary, "Autonomous Flight in GPS-Denied Environments Using Monocular Vision and Inertial Sensors", Journal of Aerospace Information Systems, vol. 10, no. 4, pp. 172-186, 2013. Available: 10.2514/1.i010023.
[137] F. Cazaurang, K. Cohen and M. Kumar, Multi-rotor platform-based UAV systems, 1st ed. Elsevier Ltd., 2007, pp. 153-175.

[138] T. Qin, T. Chen, Y. Chen and Q. Su, "AVP-SLAM: Semantic Visual Mapping and Localization for Autonomous Vehicles in the Parking Lot," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 5939-5945, Available: 10.1109/IROS45743.2020.9340939.

[139] A. Mayalu, "Beyond LiDAR for Unmanned Aerial Event-Based Localization in GPS Denied Environments", Doctor of Philosophy, Virginia Polytechnic Institute and State University, 2021.

[140] D. Cucci, M. Rehak and J. Skaloud, "Bundle adjustment with raw inertial observations in UAV applications", ISPRS Journal of Photogrammetry and Remote Sensing, vol. 130, pp. 1-12, 2017. Available: 10.1016/j.isprsjprs.2017.05.008.

[141] S. Tomažič, D. Dovžan and I. Škrjanc, "Confidence-Interval-Fuzzy-Model-Based Indoor Localization," in IEEE Transactions on Industrial Electronics, vol. 66, no. 3, pp. 2015-2024, March 2019, Available: 10.1109/TIE.2018.2840525.

[142] H. P. Chiu, X. S. Zhou, L. Carlone, F. Dellaert, S. Samarasekera and R. Kumar, "Constrained optimal selection for multi-sensor robot navigation using plug-and-play factor graphs," 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 663-670, Available: 10.1109/ICRA.2014.6906925.

[143] X. Bin, Y. Sen and Z. Xu, "Control of a quadrotor helicopter using the COMPASS (BeiDou) system and on-board vision system", Optik, vol. 127, no. 17, pp. 6829-6838, 2016. Available: 10.1016/j.ijleo.2016.05.022.

[144] P. Castillo-García, L. Muñoz Hernandez and P. García Gil, "Data Fusion for UAV Localization", Indoor Navigation Strategies for Aerial Autonomous Systems, pp. 109-129, 2017. Available: 10.1016/b978-0-12-805189-4.00007-x.

[145] J. Kläß, J. Stückler and S. Behnke, "Efficient Mobile Robot Navigation using 3D Surfel Grid Maps", Proceedings of the German Conference on Robotics (ROBOTIK), 2012. Available: http://www.ais.uni-bonn.de/papers/robotik2012_klaess_3dnav.pdf.

[146] C. Cheng, "Exploring the use of signals of opportunity for practical localization", 2015. Available: 10.32657/10356/62929.

[147] E. Javanmardi, M. Javanmardi, Y. Gu and S. Kamijo, "Factors to Evaluate Capability of Map for Vehicle Localization", IEEE Access, pp. 49850-49867, 2018. DOI: 10.1109/access.2018.2868244.

[148] A. Hata and D. Wolf, "Feature Detection for Vehicle Localization in Urban Environments Using a Multilayer LIDAR", IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 2, pp. 420-429, 2016. Available: 10.1109/tits.2015.2477817.

[149] N. Krombach, D. Droeschel, S. Houben and S. Behnke, "Feature-based visual odometry prior for real-time semi-dense stereo SLAM", Robotics and Autonomous Systems, vol. 109, pp. 38-58, 2018. Available: 10.1016/j.robot.2018.08.002.

[150] H. Deng, U. Arif, K. Yang, Z. Xi, Q. Quan and K. Cai, "Global optical flow-based estimation of velocity for multicopters using monocular vision in GPS-denied environments", Optik, vol. 219, p. 164923, 2020. Available: 10.1016/j.ijleo.2020.164923.

[151] M. Shan, F. Wang, F. Lin, Z. Gao, Y. Z. Tang and B. M. Chen, "Google map aided visual navigation for UAVs in GPS-denied environment," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, pp. 114-119, DOI: 10.1109/ROBIO.2015.7418753.

[152] A. Gupta, H. Chang and A. Yilmaz, "GPS-DENIED GEO-LOCALISATION USING VISUAL ODOMETRY", ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. -3, pp. 263-270, 2016. Available: 10.5194/isprs-annals-iii-3-263-2016.

[153] G. Chowdhary, E. Johnson, D. Magree, A. Wu and A. Shein, "GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft", Journal of Field Robotics, vol. 30, no. 3, pp. 415-438, 2013. Available: 10.1002/rob.21454.

[154] G. Vasiljević, D. Miklić, I. Draganjac, Z. Kovačić and P. Lista, "High-accuracy vehicle localization for autonomous warehousing", Robotics and Computer-Integrated Manufacturing, vol. 42, pp. 1-16, 2016. Available: 10.1016/j.rcim.2016.05.001.

[155] Z. Kassas, J. Khalife, A. Abdallah and C. Lee, "I am Not Afraid of the Jammer: Navigating with Signals of Opportunity in GPS-Denied Environments", Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020), 2020. Available: 10.33012/2020.17737.

[156] X. Wan, J. Liu, H. Yan and G. Morgan, "Illumination-invariant image matching for autonomous UAV localisation based on optical sensing", ISPRS Journal of Photogrammetry and Remote Sensing, vol. 119, pp. 198-213, 2016. Available: 10.1016/j.isprsjprs.2016.05.016.

[157] M. Ibrahim and O. Moselhi, "Inertial measurement unit based indoor localization for construction applications", Automation in Construction, vol. 71, 2016. Available: 10.1016/j.autcon.2016.05.006.

[158] A. Costley and R. Christensen, "Landmark Aided GPS-Denied Navigation for Orchards and Vineyards," 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), 2020, pp. 987-995, Available: 10.1109/PLANS46316.2020.9110130.

[159] K. Konolige, M. Agrawal and J. Solà, "Large-Scale Visual Odometry for Rough Terrain", Springer Tracts in Advanced Robotics, pp. 201-212, 2010. Available: 10.1007/978-3-642-14743-2_18.

[160] S. Du, Y. Li, X. Li and M. Wu, "LiDAR Odometry and Mapping Based on Semantic Information for Outdoor Environment", Remote Sensing, vol. 13, no. 15, p. 2864, 2021. DOI: 10.3390/rs13152864.

[161] F. Massa, L. Bonamini, A. Settimi, L. Pallottino and D. Caporale, "LiDAR-Based GNSS Denied Localization for Autonomous Racing Cars", Sensors, vol. 20, no. 14, p. 3992, 2020. Available: 10.3390/s20143992.

[162] S. James, R. A. Verrinder, D. Sabatta and A. Shahdi, "Localisation and mapping in GPS-denied environments using RFID tags," 2012 5th Robotics and Mechatronics Conference of South Africa, 2012, pp. 1-4, Available: 10.1109/ROBOMECH.2012.6558464.

[163] G. Hemann, S. Singh and M. Kaess, "Long-range GPS-denied aerial inertial navigation with LIDAR localization," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 1659-1666, Available: 10.1109/IROS.2016.7759267.

[164] C. Troiani, A. Martinelli, C. Laugier and D. Scaramuzza, "Low computational-complexity algorithms for vision-aided inertial navigation of micro aerial vehicles", Robotics and Autonomous Systems, vol. 69, pp. 80-97, 2015. Available: 10.1016/j.robot.2014.08.006.

[165] B. Wagstaff and J. Kelly, "LSTM-Based Zero-Velocity Detection for Robust Inertial Navigation," 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2018, pp. 1-8, Available: 10.1109/IPIN.2018.8533770.

[166] E. Royer, M. Lhuillier, M. Dhome and J. Lavest, "Monocular Vision for Mobile Robot Localization and Autonomous Navigation", International Journal of Computer Vision, vol. 74, no. 3, pp. 237-260, 2007. Available: 10.1007/s11263-006-0023-y.

[167] S. Weiss, D. Scaramuzza and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments", Journal of Field Robotics, vol. 28, no. 6, pp. 854-874, 2011. Available: 10.1002/rob.20412.

[168] M. Miller, M. Uijt de Haag, A. Soloviev and M. Veth, "Navigating in Difficult Environments: Alternatives to GPS - 1", 2009.

[169] H. Simkovits, A. Weiss and A. Amar, "Navigation by inertial device and signals of opportunity", Signal Processing, vol. 131, pp. 280-287, 2017. Available: 10.1016/j.sigpro.2016.08.022.

[170] G. Duckworth and E. Baranoski, Navigation in GNSS-Denied Environments: Signals of Opportunity and Beacons. NATO Science and Technology Organization, 2007.

[171] M. Miller et al., Navigation in GPS Denied Environments: Feature-Aided Inertial Systems. Eglin AFB: Air Force Research Laboratory, 2010.

[172] Raquet, J. F. (2013). Navigation using pseudolites beacons and signals of opportunity. NATO STO lecture series SET-197, navigation sensors and systems in GNSS degraded and denied environments, 1-18.

[173] S. Samarasekera and C. Bindra, "Next Generation Vehicle Positioning in GPS-Degraded Environments for Vehicle Safety and Automation Systems", 2009.

[174] F. Lazzari, A. Buffi, P. Nepa and S. Lazzari, "Numerical Investigation of an UWB Localization Technique for Unmanned Aerial Vehicles in Outdoor Scenarios", IEEE Sensors Journal, vol. 17, no. 9, pp. 2896-2903, 2017. Available: 10.1109/jsen.2017.2684817.

[175] Jian Lin, Jierui Peng, Zhichao Hu, Xiaofeng Xie and Rui Peng. ORB-SLAM, IMU and Wheel Odometry Fusion for Indoor Mobile Robot Localization and Navigation. Academic Journal of Computing & Information Science (2020), Vol. 3, Issue 1: 131-141. Available: https://doi.org/10.25236/AJCIS.2020.030114.

[176] C. Campos, R. Elvira, J. Rodriguez, J. Montiel and J. Tardos, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM", IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1874-1890, 2021. Available: 10.1109/tro.2021.3075644.

[177] A. Niewola and L. Podsędkowski, "PSD – probabilistic algorithm for mobile robot 6D localization without natural and artificial landmarks based on 2.5D map and a new type of laser scanner in GPS-denied scenarios", Mechatronics, vol. 65, 2020. DOI: 10.1016/j.mechatronics.2019.102308.

[178] A. Bachrach, A. de Winter, R. He, G. Hemann, S. Prentice and N. Roy, "RANGE - robust autonomous navigation in GPS-denied environments," 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 1096-1097, Available: 10.1109/ROBOT.2010.5509990.

[179] I. Belkin, A. Abramenko and D. Yudin, "Real-Time Lidar-based Localization of Mobile Ground Robot", Procedia Computer Science, vol. 186, pp. 440-448, 2021. DOI: 10.1016/j.procs.2021.04.164.

[180] J. Niedzwiedzki et al., "Real-Time Parallel-Serial LiDAR-Based Localization Algorithm with Centimeter Accuracy for GPS-Denied Environments", Sensors, vol. 20, no. 24, p. 7123, 2020. Available: 10.3390/s20247123.

[181] R. Leishman, T. McLain and R. Beard, "Relative Navigation Approach for Vision-Based Aerial GPS-Denied Navigation", Journal of Intelligent & Robotic Systems, vol. 74, no. 1-2, pp. 97-111, 2013. Available: 10.1007/s10846-013-9914-7.

[182] M. Caspers, "ROBOTIC NAVIGATION AND MAPPING IN GPS-DENIED ENVIRONMENTS WITH 3D LIDAR AND INERTIAL NAVIGATION UTILIZING A SENSOR FUSION ALGORITHM", Master of Science in Electrical Engineering, Naval Postgraduate School, 2021.

[183] M. Aldibaja, N. Suganuma and K. Yoneda, "Robust Intensity-Based Localization Method for Autonomous Driving on Snow–Wet Road Surface", IEEE Transactions on Industrial Informatics, vol. 13, no. 5, pp. 2369-2378, 2017. Available: 10.1109/tii.2017.2713836.

[184] X. Zhao, L. Liu, R. Zheng, W. Ye and Y. Liu, "A Robust Stereo Feature-aided Semi-direct SLAM System", Robotics and Autonomous Systems, vol. 132, p. 103597, 2020. Available: 10.1016/j.robot.2020.103597.

[185] H. Chiu, S. Williams, F. Dellaert, S. Samarasekera and R. Kumar, "Robust vision-aided navigation using Sliding-Window Factor graphs," 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 46-53, Available: 10.1109/ICRA.2013.6630555.

[186] M. Balcılar, S. Yavuz, M. Amasyalı, E. Uslu and F. Çakmak, "R-SLAM: Resilient localization and mapping in challenging environments", Robotics and Autonomous Systems, vol. 87, pp. 66-80, 2017. Available: 10.1016/j.robot.2016.09.013.

[187] R. Vivacqua, M. Bertozzi, P. Cerri, F. Martins and R. Vassallo, "Self-Localization Based on Visual Lane Marking Maps: An Accurate Low-Cost Approach for Autonomous Driving", IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 2, pp. 582-597, 2018. Available: 10.1109/tits.2017.2752461.

[188] J. Kim and S. Sukkarieh, "SLAM aided GPS/INS Navigation in GPS Denied and Unknown Environments", The 2004 International Symposium on GNSS/GPS, 2004. Available: http://users.cecs.anu.edu.au/~Jonghyuk.Kim/pdf/GNSS2004-SLAMaidedGPSINS-No37-Kim.pdf.

[189] I. Cvišić, J. Ćesić, I. Marković and I. Petrović, "SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles", Journal of Field Robotics, vol. 35, no. 4, pp. 578-595, 2017. Available: 10.1002/rob.21762.

[190] S. Prabu and G. Hu, "Stereo Vision based Localization of a Robot using Partial Depth Estimation and Particle Filter", IFAC Proceedings Volumes, vol. 47, no. 3, pp. 7272-7277, 2014. Available: 10.3182/20140824-6-za-1003.01599.

[191] K. Wang, G. Xia, Q. Zhu, Y. Yu, Y. Wu and Y. Wang, "The SLAM algorithm of mobile robot with omnidirectional vision based on EKF," 2012 IEEE International Conference on Information and Automation, 2012, pp. 13-18, Available: 10.1109/ICInfA.2012.6246774.

[192] M. Warren, M. Greeff, B. Patel, J. Collier, A. P. Schoellig and T. D. Barfoot, "There's No Place Like Home: Visual Teach and Repeat for Emergency Return of Multirotor UAVs During GPS Failure," in IEEE Robotics and Automation Letters, vol. 4, no. 1, pp. 161-168, Jan. 2019, Available: 10.1109/LRA.2018.2883408.

[193] Soloviev, A.; Bates, D.; van Graas, F. (2007) Tight Coupling of Laser Scanner and Inertial Measurements for a Fully Autonomous Relative Navigation Solution. In Military Capabilities Enabled by Advances in Navigation Sensors (pp. 4-1 – 4-30). Meeting Proceedings RTO-MP-SET-104, Paper 4. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int.

[194] Y. Song, B. Xian, Y. Zhang, X. Jiang and X. Zhang, "Towards autonomous control of quadrotor unmanned aerial vehicles in a GPS-denied urban area via laser ranger finder", Optik, vol. 126, no. 23, pp. 3877-3882, 2015. Available: 10.1016/j.ijleo.2015.07.058.

[195] C. Yang and T. Nguyen, "Tracking and Relative Positioning with Mixed Signals of Opportunity", Navigation, vol. 62, no. 4, pp. 291-311, 2015. Available: 10.1002/navi.122.

[196] V. Murali, H. -P. Chiu, S. Samarasekera and R. T. Kumar, "Utilizing semantic visual landmarks for precise vehicle navigation," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1-8, Available: 10.1109/ITSC.2017.8317859.

[197] T. Qin, P. Li and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004-1020, Aug. 2018, Available: 10.1109/TRO.2018.2853729.

[198] S. Tennyson, "Vision-Aided Navigation for GPS-Denied Environments Using Landmark Feature Identification", MSc. in Aerospace Engineering, Embry-Riddle Aeronautical University, 2014.

[199] J. Courbon, Y. Mezouar, N. Guénard and P. Martinet, "Vision-based navigation of unmanned aerial vehicles", Control Engineering Practice, vol. 18, no. 7, pp. 789-799, 2010. Available: 10.1016/j.conengprac.2010.03.004.

[200] H. Lategahn and C. Stiller, "Vision-Only Localization," in IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 3, pp. 1246-1257, June 2014, DOI: 10.1109/TITS.2014.2298492.

[201] W. Low, R. Nagarajan and S. Yaacob, "Visual based SLAM using modified PSO", 2010 6th International Colloquium on Signal Processing & its Applications, 2010. Available: 10.1109/cspa.2010.5545267.

[202] A. McConville et al., "Visual Odometry Using Pixel Processor Arrays for Unmanned Aerial Systems in GPS Denied Environments", Frontiers in Robotics and AI, vol. 7, 2020. Available: 10.3389/frobt.2020.00126.

[203] Y. Ai et al., "Visual SLAM in dynamic environments based on object detection", Defence Technology, vol. 17, no. 5, pp. 1712-1721, 2021. Available: 10.1016/j.dt.2020.09.012.

[204] C. Gómez, M. Mattamala, T. Resink and J. Ruiz-del-Solar, "Visual SLAM-Based Localization and Navigation for Service Robots: The Pepper Case", RoboCup 2018: Robot World Cup XXII, pp. 32-44, 2019. Available: 10.1007/978-3-030-27544-0_3.

[205] H. Deng, U. Arif, Q. Fu, Z. Xi, Q. Quan and K. Cai, "Visual–inertial estimation of velocity for multicopters based on vision motion constraint", Robotics and Autonomous Systems, vol. 107, pp. 262-279, 2018. Available: 10.1016/j.robot.2018.06.010.

[206] V. Usenko, N. Demmel, D. Schubert, J. Stückler and D. Cremers, "Visual-Inertial Mapping With Non-Linear Factor Recovery," in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 422-429, April 2020, Available: 10.1109/LRA.2019.2961227.

Appendix A: 3D LiDAR sensor detectability

Back

The following images indicate the daylight visibility of IR light emitted by the Velodyne Puck LiDAR sensor. Each image is taken from a different angle while looking directly at the Velodyne LiDAR. The emitted infrared light can be seen as a bright, white spot in the images, on one of the corners of the vehicle.

Appendix B: Stereo vision sensor detectability

Back

The following images indicate the visibility of infrared light emitted by the Intel RealSense D435i depth camera. Each image is taken at a different distance while looking directly at the IR emitter from the RealSense camera. The capture distance is listed for each image.

Image capture distance: approximately 3 meters.

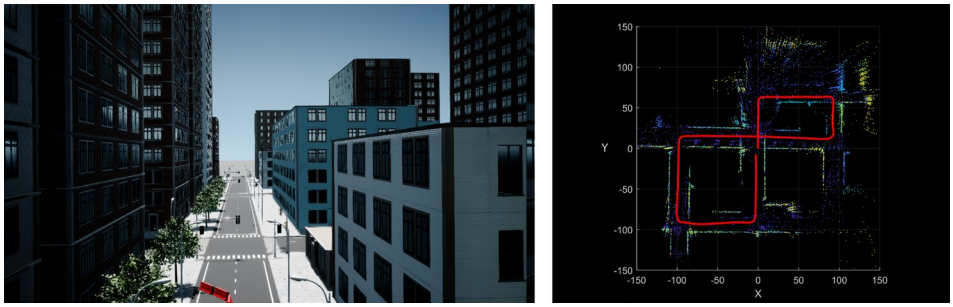
Image capture distance: approximately 6 meters.

Image capture distance: approximately 12 meters.

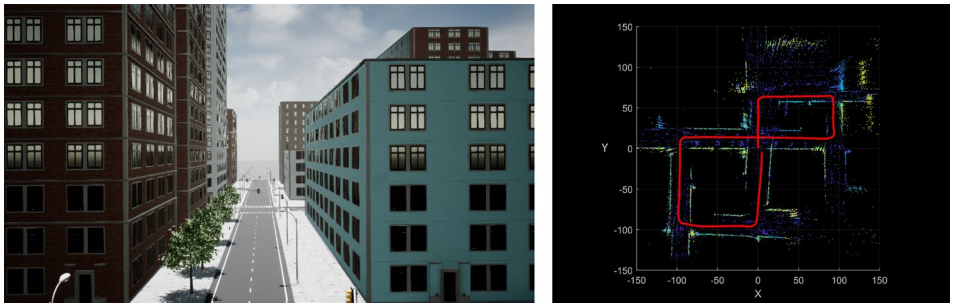
Appendix C: Feature extraction simulations

Back

The following images show the performance of a feature extraction SLAM algorithm under different weather conditions. The simulations are performed using MATLAB, which allows for consistent test runs with customizable weather conditions. On the left side of each image is a picture of the environment, showcasing the weather conditions. The right side contains a graph of the position estimate and the extracted features.

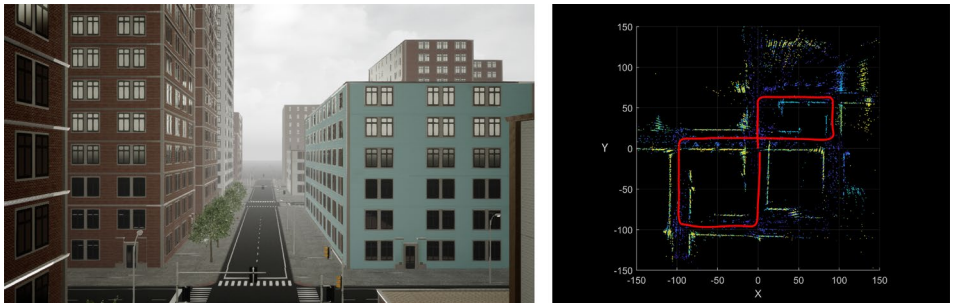


Clouds: 0% — Fog: 0% — Rain: 0%



Clouds: 25% — Fog: 20% — Rain: 0%

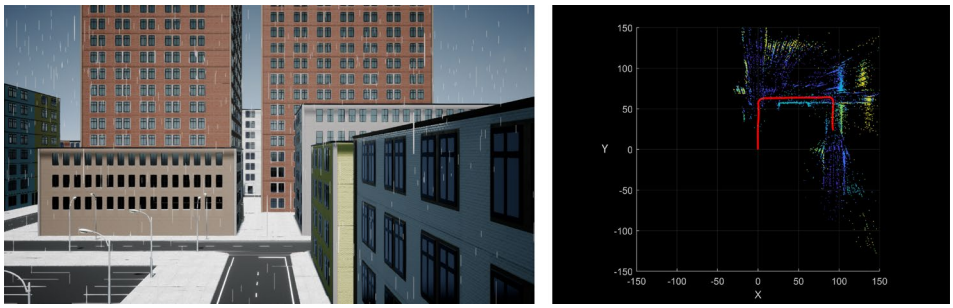
Clouds: 35% — Fog: 35% — Rain: 0%



Clouds: 60% — Fog: 60% — Rain: 0%

Clouds: 70% — Fog: 90% — Rain: 0%

Clouds: 100% — Fog: 100% — Rain: 0%



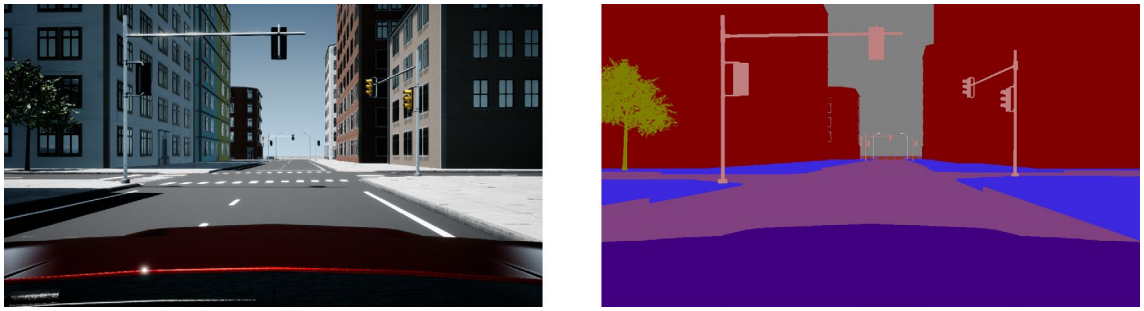
Clouds: 10% — Fog: 10% — Rain: 10%

Clouds: 40% — Fog: 20% — Rain: 30%

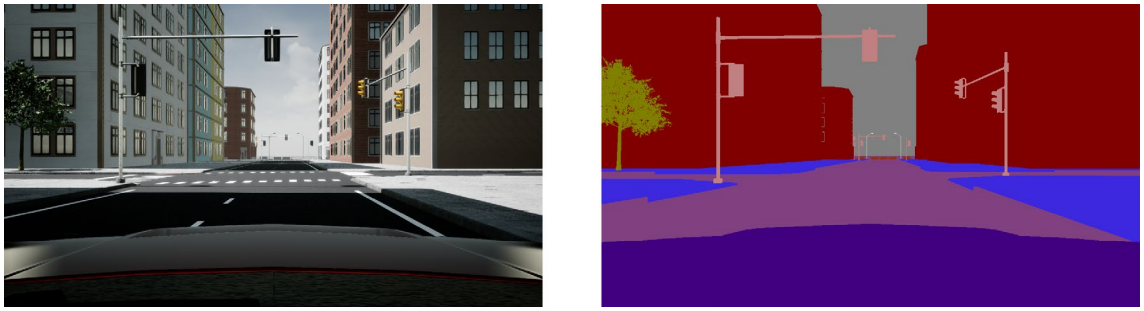
Appendix D: Semantic segmentation simulations

Back

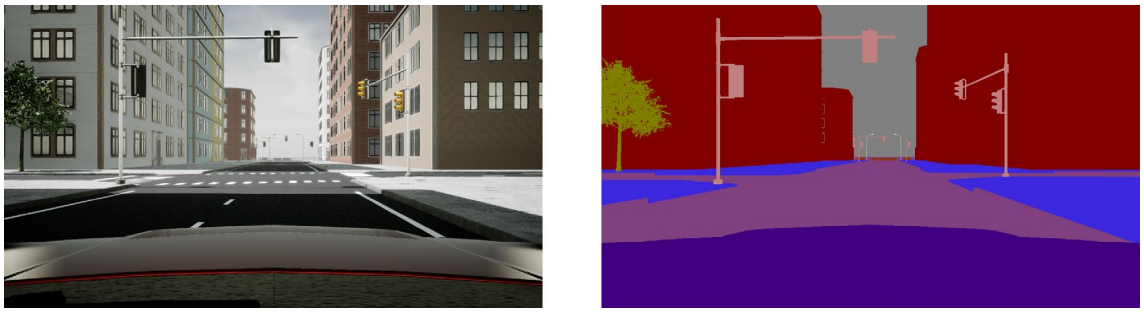
The following images show the performance of a semantic segmentation algorithm under different weather conditions. The simulations are performed using MATLAB, which allows for consistent test runs with customizable weather conditions. On the left side of each image is a picture of the environment, showcasing the weather conditions. The right side contains an image that shows the result of the semantic segmentation algorithm.



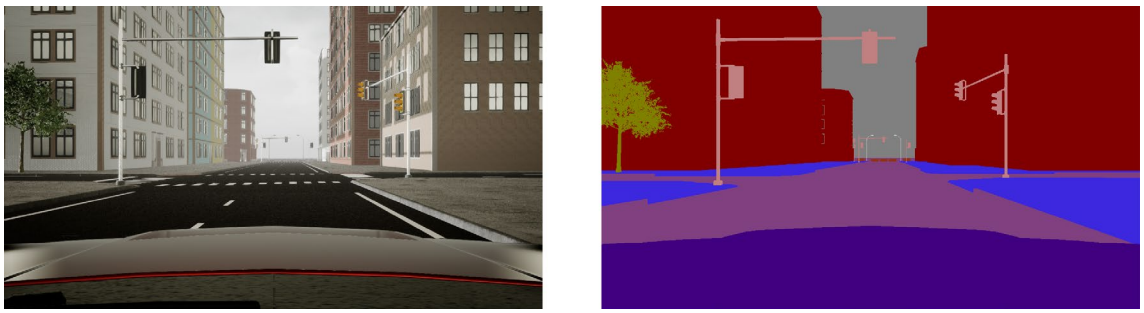
Clouds: 0% — Fog: 0% — Rain: 0%



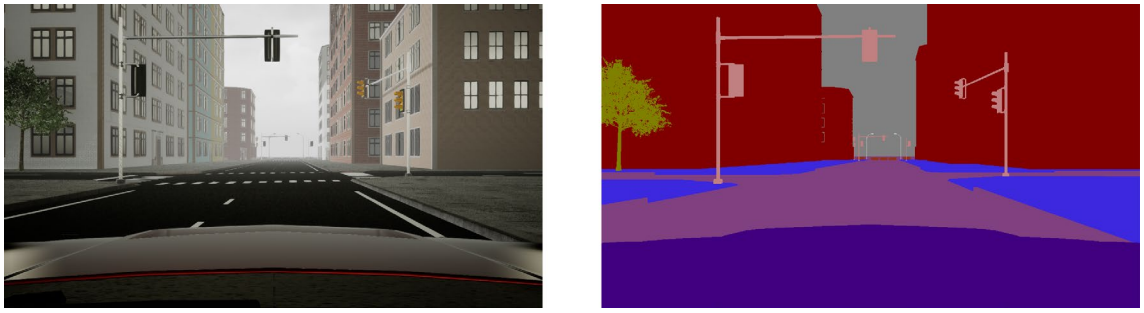
Clouds: 25% — Fog: 20% — Rain: 0%



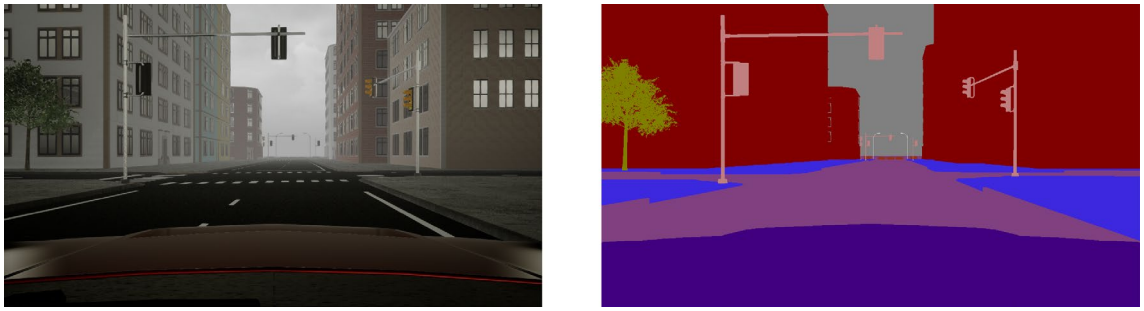
Clouds: 35% — Fog: 35% — Rain: 0%



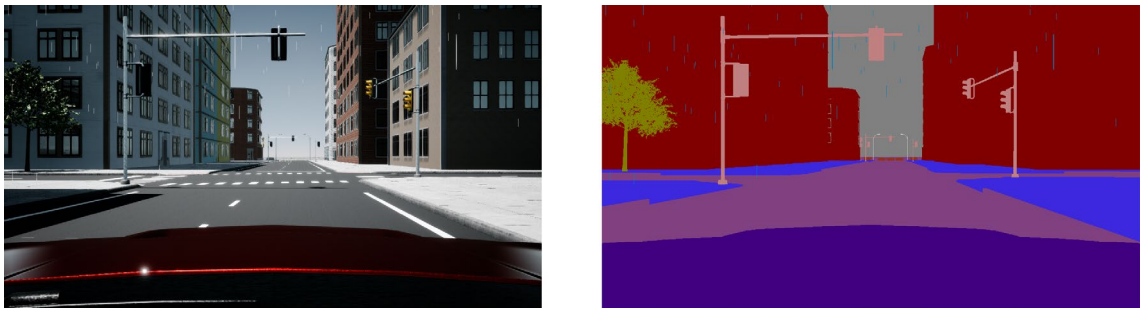
Clouds: 60% — Fog: 60% — Rain: 0%



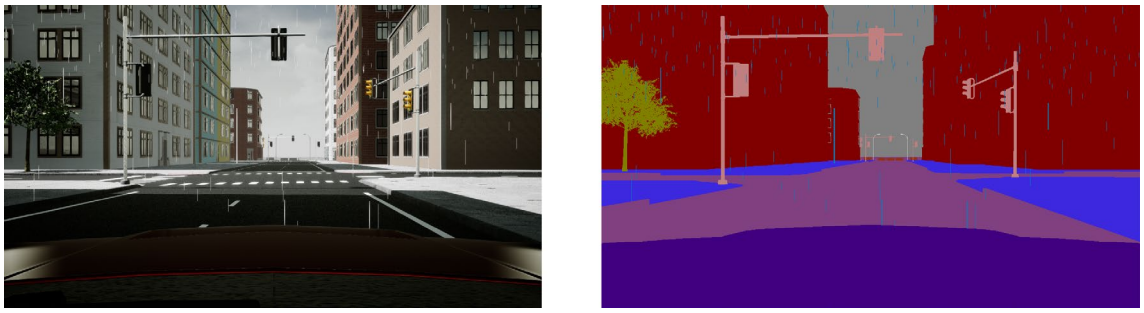
Clouds: 70% — Fog: 90% — Rain: 0%



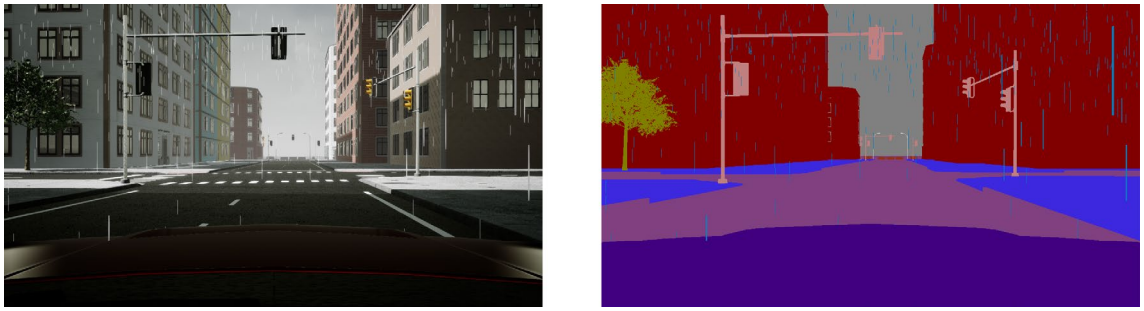
Clouds: 100% — Fog: 100% — Rain: 0%



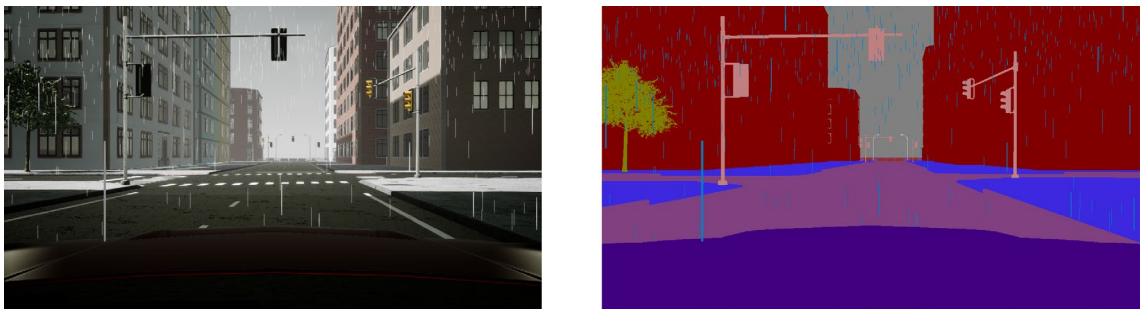
Clouds: 10% — Fog: 10% — Rain: 10%



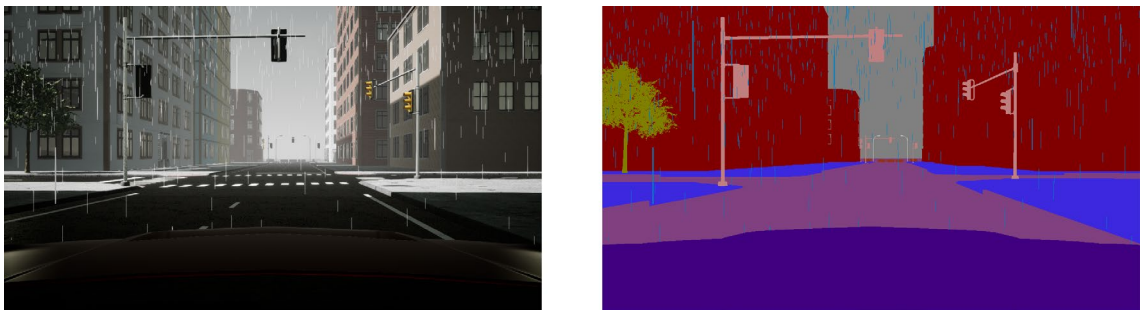
Clouds: 40% — Fog: 20% — Rain: 30%



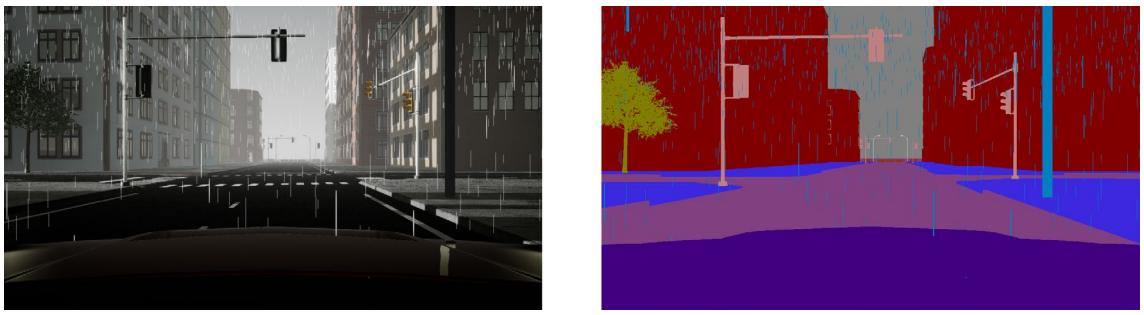
Clouds: 50% — Fog: 50% — Rain: 50%



Clouds: 65% — Fog: 65% — Rain: 65%



Clouds: 85% — Fog: 85% — Rain: 85%



Clouds: 100% — Fog: 100% — Rain: 100%

Appendix E: State-of-the-art summarizing table

Back

The following table contains an overview of all localization methods reviewed for the theoretical research that also published their accuracy data. Additional information, such as operational or computational requirements, are listed under the Notes column.

Localization method	Used sensors	Average position error (m)	Notes	Reference
Visual odometry algorithm using 3D range data.	3D LiDAR	0.45	Not yet suitable for large-scale outdoor environments.	[26]
Ad hoc localization system with aerial and ground UWB sensor nodes.	UWB ranging sensor	0.07	The location of the UWB sensor nodes should be known.	[128]
Ad hoc localization system combined with IMU and wheel odometry.	RFID ranging sensor, IMU and wheel speed sensors	5.00	The described use case is for tunnels, where there is no GPS signal.	[129]
Tracking stars and measuring their altitudes to determine a position.	Star sensor	60.00	Requires clear visibility of the sky/stars to operate.	[20]
SLAM with omnidirectional camera and an Extended Kalman Filter (EKF).	Monocular omnidirectional camera	0.08	Only suitable for indoor environments due to the impact of environmental conditions.	[130]
Using skylight polarization data in a coupled iteration mechanism.	Polarization sensor and accelerometer	50,000.00	Sky obstructions can reduce the positioning accuracy.	[23]
Ad hoc localization using a Weighted Measurement Fusion Kalman Filter (WMFKF).	Wireless Local Positioning System (WLPS) sensors	50.00	Requires a WLPS detection range of more than 30 kilometres for accurate localization.	[131]
Combining INS and LiDAR with a Robust Kalman Filter (RKF).	IMU and 2D LiDAR	0.02	The average position error is achieved over a trajectory of approximately 20-30 meters in an indoor environment.	[132]
Visual SLAM combined with fiducial markers (visual landmarks).	RGB-D camera	0.25	Requires fiducial markers to be placed in the environment.	[133]
Using geo-referenced satellite imagery and a feature detection algorithm.	Monocular camera	0.25	Requires enough environmental features, and thus a sufficiently large image view.	[134]
Using vision data to detect fiducial markers with known 3D locations.	Monocular camera	0.60	Requires fiducial markers in the environment, with known 3D locations.	[135]
Combining a monocular camera with IMU data in an EKF.	Monocular camera, IMU and GPS	1.20	A GPS signal is required to create an initial environment map with feature points.	[136]
Combining LiDAR SLAM and camera images for target identification.	2D LiDAR, monocular camera and IMU	0.25	No computational requirements listed.	[137]
Starlight imaging combined with inclinometer.	Star sensor, a dual-axis inclinometer and a precision clock	70.00	Temperature, refraction of the atmosphere and earth oblateness influence the accuracy.	[19]

Using visual SLAM based on robust features commonly found in parking lots.	Four fisheye cameras, wheel encoders and an IMU	0.05	The proposed solution only works with robust features, such as parking lines or speed bumps.	[138]
Planar extraction from 3D point cloud.	3D LiDAR	5.00	Requires a 3D point cloud of the environment beforehand.	[139]
Semantic SLAM with noise block removal.	RGB-D camera	0.05	The described method is designed for indoor environments with moving objects.	[45]
Sensor fusion of camera and IMU data.	Monocular camera and IMU	10.00	The position error will increase over time.	[140]
Ad hoc localization using Bluetooth beacons.	Bluetooth radio sensor	0.48	Requires pre-installed Bluetooth beacons in the environment with known locations.	[141]
Sensor fusion using factor graphs.	Camera, 2D LiDAR, compass and a magneto-meter	3.62	Experiments are conducted using a single core of an Intel Core i7 CPU running at 2.70 GHz.	[142]
Sensor fusion of COMPASS, IMU and optical flow data.	COMPASS receiver, monocular camera and INS	5.00	Still requires COMPASS (similar to GPS) for accurate localization.	[143]
Cooperative localization using a GPS-equipped drone and Direction of Arrival (DOA) measurements.	GPS receiver and communi-cation sensor	500.00	Still requires GPS for one drone (or another vehicle) to determine another vehicle's position.	[61]
Sensor fusion using optical flow and an IMU in an EKF.	Monocular camera and IMU	0.50	The position errors are achieved in an indoor environment.	[144]
Measuring local magnetic field variations to compare it to a global magnetic field map.	3-axis magnetometers	200.00	Requires a global magnetic field map to determine the position.	[16]
Semantic SLAM using deep learning for object detection.	Monocular camera	30.00	The experiments are conducted using an Nvidia Jetson TX2 development board.	[46]
Using a 3D voxel map for Monte Carlo localization.	2D LiDAR or 3D LiDAR	0.50	Requires a 3D voxel map (OctoMap) beforehand.	[145]
Ad hoc localization using the Received Signal Strength (RSS) of 4G LTE beacons.	4G LTE receiver	50.00	The experiments deploy 4G LTE transmitters spaced 500 meters from each other.	[54]
Two-dimensional odometry using radar.	Radar	3.00	The experiments are conducted on an Intel Core i7 CPU running at 2.30 GHz and 6 GB of RAM.	[36]
Using Time Difference of Arrival (TDOA) from Signals of Opportunity (SoOP).	Universal Software Radio Peripherals (USRP)	10.00	No computational requirements listed.	[146]
Sensor fusion combined with a 3D map of the environment.	3D LiDAR, IMU, GPS, odometer and stereo camera	0.52	Still requires GPS for accuracy. Additionally, no computational requirements are listed.	[147]
Feature detection in a 3D environment map that is compared with a reference map.	3D LiDAR, GPS and INS	0.30	A geo-referenced 3D map of the environment is required.	[148]

Visual odometry combined with visual SLAM.	Stereo camera and 3D LiDAR	10.00	Experiments are conducted using an Intel Core i7 running at 2.2 GHz and 8 GB of RAM	[149]
Matching line features extracted from 3D LiDAR data with OpenStreetMap data.	3D LiDAR	0.60	Experiments are conducted using an Intel Core i7 running at 3.5 GHz and 16 GB of RAM.	[33]
Visual SLAM using both ORB-SLAM2 and S-PTAM algorithms	Stereo camera	115.00	The simulations are performed using an Intel Core i7 running at 1.90 GHz with 32 GB of RAM.	[105]
Sensor fusion of vision and LiDAR data for scale and loop closure constraints.	Panoramic camera and 2D LiDAR	0.20	Without loop closures, the position error will increase over time.	[70]
Optical flow navigation using a downward looking monocular camera and IMU.	Monocular camera and IMU	0.51	Processing times are around 25-33 ms, which is sufficient or real-time operation.	[150]
Comparing 3D LiDAR terrain classification with OpenStreetMap data.	3D LiDAR, gyroscopes, IMU and odometry	1.50	The position accuracy depends on the accuracy and correctness of the terrain classification.	[32]
Optical flow combined with feature detection on Google Maps and local imagery.	Monocular camera and IMU	6.70	The proposed solution provides drift-free localization without the need for loop closures.	[151]
Combining visual odometry with street and building data from public sources.	Monocular camera and IMU	15.00	The described solution is designed for urban use cases.	[152]
Sensor fusion of vision and IMU data using a corner detection algorithm and an EKF.	Monocular camera and IMU	5.00	The proposed solution can operate on low-end hardware in real-time.	[153]
Monte Carlo localization with Iterative Closest Point (ICP) optimization from 2D LiDAR data.	2D LiDAR	0.01	Requires an initial map of the environment beforehand.	[154]
Signals of Opportunity (SoOP) using different radio signals with known transmitter locations.	Radio receiver	0.62	The listed position accuracy was achieved with 9 CD-cellular CDMA towers with known locations.	[155]
Comparing on-board camera images with satellite imagery.	Monocular camera	1.31	This method only works with enough features in the on-board camera images.	[156]
Ad hoc localization based on the Theatre Positioning System (TPS).	TPS receiver	20.00	The listed position accuracy was achieved with 3 TPS transmitters, placed 150 kilometres apart.	[63]
Sensor fusion for IMU and barometric pressure sensor with a jerk integration algorithm.	IMU and barometric pressure sensor	121.89	The position accuracy will decrease over time, due to the types of used sensors.	[157]
Combining environmental knowledge with 2D LiDAR and IMU data in a neural network.	2D LiDAR and IMU	3.00	This method assumes that there are periodically placed landmarks in the environment with a similar shape.	[158]
Ad hoc localization combined with dead reckoning for areas without a signal.	Radio receiver	50.00	Requires cellular towers with known locations for accurate localization.	[53]
Combining visual odometry with IMU data to reduce drift rates.	Stereo camera and an IMU	20.48	The proposed method can operate in real-time on low-end hardware.	[159]
LiDAR Odometry and Mapping (LOAM) using semantic information.	3D LiDAR	80.00	The experiments are performed on an Intel Core i7 CPU running at 3.60 GHz.	[160]

Using LiDAR data with an EKF and Monte Carlo Localization.	2D LiDAR	4.92	Requires an occupancy grid of the environment beforehand.	[161]
Using RFID tags distributed throughout the environment for localization.	RFID ranging sensor, IMU and wheel speed sensors	0.35	The listed position accuracy was achieved with 800 RFID tags distributed in an area of 10 m^2.	[162]
Ad hoc localization using cooperative, geo-referenced vehicles.	Radio sensor	2.00	This method requires a cooperative vehicle with known location to achieve the listed position accuracy.	[62]
Measuring landscape elevation with LiDAR data and compare this with a known Digital Elevation Model (DEM) of the environment.	3D LiDAR and IMU	27.00	Requires a DEM of the deployment environment beforehand.	[163]
Sensor fusion of monocular camera and IMU with EKF and a feature detection algorithm.	Monocular camera and an IMU	0.02	The proposed solution can operate on low-end hardware in real-time.	[164]
Using a neural network to improve the accuracy of INS data for localization.	INS	1.10	No computational requirements listed.	[165]
Combining wheel odometry with LTE cellular network signals.	LTE radio sensor and a wheel speed sensor	13.07	Requires LTE cellular towers with known locations to achieve the listed position accuracy.	[52]
Natural landmark detection in a known 3D map of the environment.	Monocular camera and GPS	0.02	A GPS is still required to create a geo-referenced 3D map of the environment.	[166]
Visual SLAM using a single monocular camera.	Monocular camera	0.10	Requires a map of the environment beforehand to achieve the listed position accuracy.	[167]
Image-aided INS using a feature-tracking algorithm.	Monocular camera and an INS	0.10	The position error increases over time.	[168]
LiDAR-aided INS using a feature-detection algorithm.	2D LiDAR and an INS	1.59	The listed position accuracy was achieved in a 250-meter outdoor track.	[168]
Ad hoc localization using Time of Arrival (TOA) measurements combined with INS data.	Radio sensor and INS	175.00	The transmitter locations should be known to achieve the listed accuracy.	[169]
LiDAR-based SLAM and obstacle avoidance.	2D LiDAR	0.06	The listed position accuracy was achieved in an indoor environment.	[25]
Signals of Opportunity localization using TV signals.	Radio receiver	42.00	The location of the TV signal transmitters should be known.	[170]
Using a feature detection algorithm with vision data, combined with INS data.	Monocular camera and an INS	1.59	No computational requirements listed.	[171]
Using different radio frequency transmitters for Signals of Opportunity localization.	RF receiver	1.50	The listed accuracy requires multiple RF beacons with known locations.	[172]
Combining visual odometry with visual landmark detection and matching.	Stereo camera	0.54	No computational requirements listed.	[173]

Ad hoc localization using UWB nodes and Time of Flight (ToF) data.	UWB receiver	0.30	The listed position accuracy can only be achieved when the distance between the UWB nodes and receiver is < 80 m.	[174]
Combining VO with a chamfer matching algorithm for OpenStreetMap data.	Stereo camera	5.19	The proposed method works best in urban environments, due to the chamfer matching algorithm.	[42]
Sensor fusion using ORB-SLAM, IMU and wheel odometry.	Monocular camera, IMU and wheel speed sensors	0.50	The listed accuracy is achieved with a predefined map of the environment.	[175]
Visual SLAM with map reuse, loop closing and localization capabilities.	Stereo camera	0.60	The position accuracy decreases with high speeds and low frame-rates.	[103]
Visual-inertial SLAM algorithm.	Stereo camera and an IMU	4.50	The best position accuracy is achieved when loop closures are present.	[176]
Using a Point-to-Surfel Distance algorithm for 3D LiDAR data.	3D LiDAR	12.60	The proposed solution can be executed in real-time using one core of an Intel Core i7 running at 2.60 GHz.	[177]
Combining radar odometry, optical flow and IMU data with an EKF for robust navigation.	Radar, monocular camera and an IMU	5.00	The use of radar odometry allows the proposed method to be used in challenging environments and weather conditions.	[73]
Combining a recursive-RANSAC algorithm for radar odometry with IMU data.	Radar and an IMU	92.00	The localization error can be improved with a more accurate IMU.	[35]
Sensor fusion using 2D LiDAR and IMU data with an EKF.	2D LiDAR and an IMU	0.30	The proposed method is designed for indoor and urban environments.	[178]
Combining the LOAM method with corner and surface point extraction.	3D LiDAR	4.20	Real-time performance can be achieved on an Nvidia Jetson AGX Xavier.	[179]
Sensor fusion of 3D LiDAR data and an IMU with an EKF for each LiDAR scan point.	3D LiDAR and an IMU	0.13	The proposed method requires a 3D map of the environment beforehand.	[180]
Sensor fusion with vision and IMU data using visual graph-SLAM and an EKF.	RGB-D camera and an IMU	2.00	The proposed solution can achieve real-time performance on an Intel Core i7 running at 2.10 GHz.	[181]
Signals of Opportunity of multiple frequency band signals.	Software defined radio	150.00	The proposed solution does not require the position of the radio transmitters to be known.	[51]
Using a strap-down navigation algorithm and an iterative closest point (ICP) registration algorithm for LiDAR and IMU sensor fusion.	3D LiDAR and an IMU	1.23	The experiments are conducted using h an Intel Core i3 running at 2.00 GHz and 16 GB of RAM.	[182]
Sensor fusion of LiDAR and vision data with a principal component analysis (PCA).	3D LiDAR and a monocular camera	0.20	The experiments are conducted using an Intel Core i7 running at 3.40 GHz and 8 GB of RAM	[183]
Visual odometry for dynamic environments with RGB-D camera data.	RGB-D camera	0.04	The proposed method is designed for dynamic environments with moving objects.	[48]
Sensor fusion with 3D LiDAR, IMU and stereo camera based on the SLAM method.	3D LiDAR, IMU and a stereo camera	1.02	The proposed solution can achieve real-time performance on an Intel Core i7 running at 3.60 GHz.	[184]

Sensor fusion using Sliding-Window Factor Graphs (SWFG).	Monocular camera, IMU, odometer and a barometer	19.19	The proposed solution can achieve real-time performance on a single core of an Intel Core i7 running at 2.40 GHz.	[185]
Particle filter-based SLAM methodology for large, complex environments.	RGB-D camera, 2D LiDAR and an IMU	0.31	The proposed solution can operate with only a 2D LiDAR.	[186]
Comparing visually detected lane markers with a map of the environment, combined with dead reckoning.	Monocular camera	0.04	The described method only works on roads where there are lane markers.	[187]
Using a Kalman Filter (KF) for sensor fusion of vision and IMU data.	Monocular camera and an IMU	100.00	The proposed solution requires powerful hardware for real-time execution.	[188]
Visual SLAM for stereo camera with SOFT feature tracking for pose estimation, large loop-closing and global consistently.	Visual-inertial sensor (stereo camera with an onboard IMU)	0.72	The proposed solution can operate in real-time with an Intel Core i7 running at 3.40 GHz and 16 GB of RAM.	[189]
Using a stereo camera for 3D feature point detection, followed by partial depth estimation and a particle filter.	Stereo camera	2.80	No computational requirements listed.	[190]
Detecting and comparing visual landmarks with an existing map of visual landmarks.	Monocular camera and an IMU	0.94	Requires a map of the visual landmarks within the environment beforehand.	[44]
Surfel-based semantic mapping of 3D LiDAR data.	3D LiDAR	8.08	Real-time performance can be achieved with an Intel Xeon running at 3.60 GHz, 16 GB of RAM and an Nvidia Quadro P4000.	[30]
Using an omnidirectional camera for the SLAM method, combined with an EKF.	Omnidirectional camera	0.10	The proposed solution can be executed in real-time.	[191]
Combining feature point recognition for vision data and dead reckoning.	Stereo camera and an IMU	3.60	The proposed solution requires a visual map of the environment beforehand.	[192]
Sensor fusion using feature detection for 2D LiDAR data combined with IMU data.	2D LiDAR and an IMU	1.60	No computational requirements listed.	[193]
Sensor fusion using a Point to Linemetric Iterative Closest Point algorithm for 2D LiDAR data, combined with an IMU.	2D LiDAR and an IMU	0.20	The proposed solution can be executed in real-time on low-end hardware.	[194]
Using DTV and cell phone signals for SoOP localization.	Radio sensor	120.00	The achievable position accuracy depends on the quality of the radio signals.	[195]
Semantic mapping for the ORB-SLAM2 methodology using stereo vision.	Stereo camera	23.00	The proposed method does not require a map of the environment beforehand.	[196]
Using a modified normalized phase correlation algorithm for visual odometry.	Monocular camera	9.00	The proposed solution works with less textured images and can be executed in real-time.	[49]
Sensor fusion using a monocular camera and an IMU, combined with a loop detection module.	Monocular camera and an IMU	0.25	The proposed solution can be executed in real-time using three threads of an Intel Core i7 running at 3.60 GHz.	[197]

Matching vision data from a UGV with satellite imagery using feature detection.	Panoramic camera	10.00	No computational requirements listed.	[40]
Comparing visual landmarks from vision data to predefined visual landmarks for localization.	Monocular camera	0.30	The described approach requires pre-mapped visual landmarks.	[198]
Comparing natural landmarks with previously detected natural landmarks for localization.	Fisheye camera	2.00	The described approach requires pre-mapped natural landmarks.	[199]
Matching ground images and semantic environment data with satellite imagery.	Panoramic camera, 2D LiDAR and an IMU	5.40	No computational requirements listed.	[71]
Comparing monocular camera data with stereo vision data from the environment.	Monocular camera and a stereo camera	0.12	The described method requires the environment to be pre-mapped with a stereo camera.	[200]
Visual SLAM using a Particle Swarm Optimization (PSO) algorithm.	Stereo camera	0.10	The position error will increase over time due to the used sensor and localization method.	[201]
Comparing vision data with geo-referenced images from 3D Google Earth.	Stereo camera and an IMU	4.00	The proposed solution can be performed in real-time with an Nvidia Tegra TX2 computer.	[41]
Comparing vision data from Pixel Processor Arrays with geo-referenced satellite images.	Pixel Processor Array	4.00	The proposed solution uses components that are energy efficient	[202]
Applying deep learning object detection for visual SLAM in dynamic environments.	RGB-D camera	0.70	Due to the deep learning algorithm, powerful hardware is required for operation.	[203]
Sensor fusion of visual SLAM with wheel odometry.	Monocular camera and wheel speed sensor	0.85	The listed position accuracy is achieved in an indoor environment.	[204]
Sensor fusion based on a monocular camera, an IMU and an ultrasonic range finder.	Monocular camera, an IMU and an ultrasonic range finder	0.30	Environments with less texture will decrease the position accuracy.	[205]
Sensor fusion of visual odometry and inertial odometry with non-linear factors.	Monocular camera and an IMU	0.13	The proposed solution can run in real-time with an Intel Xeon running at 3.60 GHz.	[206]

Appendix F: Groot behaviour tree design

Back

The following image shows a screenshot from Groot with the behaviour tree used for the simulation testing of the prototype. The input value for all behaviour tree nodes with an input field can be modified by the user before the behaviour tree is executed (except the RetryUntilSuccessful node).