NTNU
Kunnskap for en bedre verden

Department of Marine Technology

Master Thesis Marine Cybernetics

Simulation-based Validation of a Situational Awareness Simulator

Author:
Robin Stokke

Supervisor:
Øyvind Smogeli

Co-Supervisors:
Kjetil Vasstein
Erik Wilthil

June, 2022

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER OF TECHNOLOGY THESIS DEFINITION (30 SP)

Name of the candidate:	Robin Stokke
Field of study:	Marine Cybernetics
Thesis title (Norwegian):	Simulasjonsbasert Validering av en Situasjonsforståelsessimulator
Thesis title (English):	Simulation-based Validation of a Situational Awareness Simulator

Background

As essential component in the design and deployment of an autonomous vessel is the test system, validating the safety and performance of the system before it enters regular operation. Although physical testing is a necessary step in this process, it is too expensive to scale and too dangerous to test certain scenarios, motivating the need for simulation-based testing. Thus, complex closed-loop simulators are used to simulate the full autonomy stack as well as the vessel's operating environment, resulting in a computationally expensive simulation helping validate system behaviour. A challenge with these simulations is the specific graphical hardware requirements of the sensor simulation component, bottlenecking the speed and scalability of simulations. A proposed solution is to use an alternative, low-fidelity sensor simulation component without the same hardware requirements, that is compared and validated with respect to the original. This low-fidelity simulator would not replace its high-fidelity counterpart, but rather inform it of relevant scenarios to be analyzed in further detail. The viablility of this simulation-based simulator validation process is the research question this thesis intends to answer, where the sensor simulation components and situational awareness algorithms used at Zeabuz are the example case.

Scope of work

- Review relevant literature of verification and validation of autonomous marine and automotive systems.
- Discuss how the performance of a situational awareness simulator can be quantified and compared.
- Implement the necessary interfaces to complete the proposed low-fidelity simulator pipeline.
- Design and implement a result management tool capable of calculating performance benchmarks and visualizing system behaviour to facilitate in-depth analysis.
- Discuss the validity of the proposed simulation-based simulator validation process in the chosen use-case, and for interchangeable simulators in general.

Specifications

The student shall at startup provide a maximum 2-page week plan of work for the entire project period, with main activities and milestones. This should be updated on a monthly basis in agreement with supervisor.

Every weekend throughout the project period, the candidate shall send a status email to the supervisor and co-advisors, providing two brief bulleted lists: 1) work done recent week, and 2) work planned to be done next week.

The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, problem/research statement, design/method, analysis, and results. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 70 A4-pages, 100 B5-pages, from introduction to conclusion, unless otherwise agreed. It shall be written in English (preferably US) and contain the elements: Title page, project definition, preface (incl. description of help, resources, and internal and external factors that have affected the project process), acknowledgement, abstract, list of symbols and acronyms, table of contents, introduction (project background/motivation, objectives, scope and delimitations, and contributions), technical background and literature review, problem formulation or research question(s), method/design/development, results and analysis, conclusions with recommendations for further work, references, and optional appendices. Figures, tables, and equations shall be numerated. The contribution of the candidate shall be clearly and explicitly described, and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a

Harvard citation style (e.g. natbib Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct, which is taken very seriously by the university and will result in consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed.

The thesis shall be submitted with an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis definition shall be included after the title page. Computer code, pictures, videos, data, etc., shall be included electronically with the report.

Start date:	15 January, 2022	**Due date:**	11 June 2022

Supervisor:	Professor Øyvind Smogeli, NTNU/Zeabuz
Co-advisor(s):	Dr. Erik Wilthil, Zeabuz
	PhD. candidate Kjetil Vasstein, NTNU

Trondheim 11.06.2022

[illegible]

Professor Øyvind Smogeli

Abstract

Robust situational awareness is an essential component in an autonomous vessel, which is both expensive and time consuming to test and validate using field data alone. As has become commonplace in the self-driving car industry, game-engine based simulators are used to generate data for entire sensor suites, allowing for closed-loop simulations of the entire autonomy stack. A similar end-to-end simulator for autonomous vessels is being developed at Zeabuz, where the sensor simulation is conducted in the Unity-based tool Gemini. The high graphical processing power Gemini demands limits the rate at which testing can be scaled up, which motivates the proposal of using a simpler, faster, low-fidelity simulator replacing Gemini as the sensor simulation component. The performance of this simulator is validated with respect to the high-fidelity Gemini simulator using a dedicated software tool developed for comparing simulator benchmarks, which quantifies the differences between the simulators. This allows the low-fidelity simulator to be used as a replacement component, assuming an allowable margin of error, greatly accelerating test coverage of the autonomous vessel's autonomy stack.

Sammendrag

En robust situasjonsforståelsesalgoritme er en essensiell komponent i et autonomt fartøy, som er både dyrt og tidskrevende å test og validere ved bruk av kun feltdata. I selvkjørende bilindustrien har det blitt vanlig å bruke simulatorer bygd på spillmotorer for å generere sensordata, som tillater lukket-krets simuleringer av hele autonomisystemet. Et tilsvarende 'end-to-end' simulator for autonome fartøy utvikles av Zeabuz, hvor sensorsimuleringer gjennomføres av et Unity-basert verktøy ved navn Gemini. Den høye grafiske prosessorkraften krevd av Gemini begrenser oppskaleringen av tester, som motiverer forslaget om å anvende en enklere, raskere, lavpresisjonssimulator som erstatter Gemini som sensorsimuleringskomponent. Ytelsen av denne simulator blir validert i forhold til høypresisjonssimulatoren Gemini ved hjelp av et dedikert softwareverktøy utviklet for sammenligningen av simulatorytelse, som kvantifiserer forskjellene mellom simulatorene. Dette tillater at lavpresisjonssimulatoren kan erstatte Gemini, med en gitt tillatt feilmargin, og akselerere testingen av fartøyets autonomisystem.

Acknowledgements

This thesis would not have been possible without the extensive work done on the end-to-end simulator and all of its components, by a large number of people at Zeabuz. I'd like to thank my co-supervisor Erik Wilthil for the help with his situational awareness algorithm and simulator, as well as my other co-supervisor Kjetil Vasstein for his input on simulator performance metrics and general advice on what to focus on underway. I'd also like to acknowledge the work done by my fellow students Hanna Hjelmeland and Eivind Løvoll under the guidance of Vinicius de Oliveira, for laying the groundwork of the Result Manager tool which had only begun development a few months ago. Thanks to Brage Sæther as well, for showing genuine interest in my progress and discussing the implications of my results.

A thanks goes to my former colleagues in the student organisation Njord - The Autonomous Ship Challenge. The idea for this thesis, as well as the necessary experience with the software platforms, would not have been possible without this student initiative. Related to this, I greatly appreciate the continued support from Thomas Skarshaug, not only with this thesis, but starting over two years ago with technical support and an endless stream of questions regarding Gemini.

A final, special thanks goes to my supervisor Øyvind Smogeli, for believing in the project suggestion, always pointing me in the right direction, and being patient when things other than this thesis took priority. I am grateful to have received the opportunity to write this thesis in collaboration with Zeabuz.

Preface

This thesis has unsurprisingly been written in close collaboration with Zeabuz, as their core autonomy stack and test system is the basis for the conducted simulator validation. As such, there has over the course of the work period been a number of hurdles resulting from working on an evolving codebase that is yet to become properly robust. Thankfully, many of these hurdles were overcome with help from a number of different engineers at Zeabuz, however some proved too large to solve given the time schedule, and became evident in the results of this thesis. These factors, such as the sensor suite lacking a radar in the high-fidelity simulator, required some creative solutions to work around. Prior knowledge of a number of the system components proved to be valuable, having worked with Gemini and ROS in other projects, without which I likely would have been a larger burden for Zeabuz's engineers.

Table of Contents

List of Figures **viii**

List of Tables **ix**

1 Introduction **1**

2 Theory **3**

2.1 Simulation Fidelity 3

2.2 Simulator Verification and Validation 3

2.2.1 Regulatory requirements for MASS Simulators 4

2.3 Situational Awareness performance metrics 5

2.3.1 Batched Benchmarks 5

2.4 Coordinate frames and transforms 6

3 Method **7**

3.1 MilliAmpère 2 7

3.2 Software Architecture 8

3.2.1 ROS 10

3.2.2 Traffic Simulator 10

3.2.3 Situational Awareness Simulator 10

3.2.4 Gemini 11

3.2.5 Result Manager 12

4 Results **13**

4.1 Individual Scenarios 13

4.2 Simulator Comparisons 17

4.3 Batched Simulator Comparisons 20

4.4 Intermediate Results 21

5 Results with Radar enabled **22**

6 Discussion **24**

6.1 Simulator Performance Comparison 24

6.2 Result Manager Evaluation 25

6.3 Simulation Validation Viability 26

7 Conclusion **27**

7.1 Further Work 27

Bibliography **28**

Appendix **29**

A Exhaustive Results **29**

A Scenario 2 29

B Scenario 3 32

C Scenario 4 35

List of Figures

1 The software testing and verification hierarchy, where validation of the lower levels facilitate assurance of the higher levels. Source: Smogeli 2021 2

2 Relationship between reality, conceptual and simulation models, with related verification and validation activities. Based on (Schlesinger, 1979). Source: *Assurance of simulation models, DNV-RP-0513* 2021 4

3 The milliAmpère 2 ferry docked in Nyhavna, Trondheim. Source: Robin Stokke . . 7

4 The Shore Control Lab located in Trondheim Maritime Center. The aluminum profile structure is meant to allow for rapid prototyping of different control room configurations. Source: Robin Stokke 8

5 End-to-end simulation architecture used in Zeabuz. Source: Smogeli 2021 8

6 High-level overview of the system components and their interaction 9

7 A detailed overview of the system components, their language or platform of implementation, and the interactions between them 9

8 A screenshot of Gemini, with milliAmpère 2 docked at Ravnkloa 11

9 Overview of Scenario 1, Low-Fidelity Simulator 14

10 Overview of Scenario 1, High-Fidelity Simulator 14

11 SITAW algorithm position error in Scenario 1, Low-Fidelity Simulator 15

12 SITAW algorithm position error in Scenario 1, High-Fidelity Simulator 15

13 SITAW algorithm heading error in Scenario 1, Low-Fidelity Simulator 16

14 SITAW algorithm heading error in Scenario 1, High-Fidelity Simulator 16

15 Freeze-frame of scenario animation 17

16 Comparison of SITAW algorithm performance in Scenario 1 18

17 Comparison of SITAW algorithm performance in Scenario 2 18

18 Comparison of SITAW algorithm performance in Scenario 3 19

19 Comparison of SITAW algorithm performance in Scenario 4 19

20 The performance trend of the SITAW algorithm measured with respect to the time the target vessel enters sensor range 20

21 Intermediate Result Manager position error results from Scenario 1, Low-Fidelity Simulator 21

22 Intermediate Result Manager heading error results from Scenario 1, Low-Fidelity Simulator 21

23 Overview of Scenario with radar enabled, Low-Fidelity Simulator 22

24 SITAW algorithm position error in scenario with radar enabled, Low-Fidelity Simulator 22

25 SITAW algorithm heading error in scenario with radar enabled, Low-Fidelity Simulator 23

26 Overview of Scenario 2, Low-Fidelity Simulator 29

27 Overview of Scenario 1, High-Fidelity Simulator 29

28 SITAW algorithm position error in Scenario 2, Low-Fidelity Simulator 30

29 SITAW algorithm position error in Scenario 2, High-Fidelity Simulator 30

30 SITAW algorithm heading error in Scenario 2, Low-Fidelity Simulator 31

31 SITAW algorithm heading error in Scenario 2, High-Fidelity Simulator 31

32 Overview of Scenario 3, Low-Fidelity Simulator 32

33 Overview of Scenario 3, High-Fidelity Simulator 32

34 SITAW algorithm position error in Scenario 3, Low-Fidelity Simulator 33

35 SITAW algorithm position error in Scenario 3, High-Fidelity Simulator 33

36 SITAW algorithm heading error in Scenario 3, Low-Fidelity Simulator 34

37 SITAW algorithm heading error in Scenario 3, High-Fidelity Simulator 34

38 Overview of Scenario 4, Low-Fidelity Simulator 35

39 Overview of Scenario 4, High-Fidelity Simulator 35

40 SITAW algorithm position error in Scenario 4, Low-Fidelity Simulator 36

41 SITAW algorithm position error in Scenario 4, High-Fidelity Simulator 36

42 SITAW algorithm heading error in Scenario 4, Low-Fidelity Simulator 37

43 SITAW algorithm heading error in Scenario 4, High-Fidelity Simulator 37

List of Tables

1 Parameters used for transformation from LLA to NED 6

Glossary

COLAV Collision Avoidance. 10, 11

COLREGs Convention on the International Regulations for Preventing Collisions at Sea. 7

LLA Latitude, Longitude, Altitude. ix, 6

MPCS Motion Planning Control System. 10, 12, 25–27

NED North-East-Down. ix, 6

RMS Root Mean Square. 5, 12

ROS Robotics Operating System. vi, 6, 10, 12, 24

SITAW Situational Awareness. viii, ix, 5–7, 10–13, 15–20, 22–27, 30, 31, 33, 34, 36, 37

V&V Verification and Validation. 3

1 Introduction

Water-based modes of transport are the amongst the most energy efficient in widespread use today, largely due to their favorable economy of scale. Roughly 90% of global trade is transported by shipping vessels (OECD 2019) of varying sizes, with the majority on massive container ships and carriers. According to the International Maritime Organization, the shipping industry accounted for 2.89% of global CO_2 emissions as of 2020 (IMO 2020), with reports estimating that the industry's emissions could continue to grow between 50% and 250% within 2050 (EU Commission 2021) in a 'business-as-usual' scenario. With global trade showing no signs of shrinking, numerous measures must be taken to limit shipping's environmental impacts, as well as prevent accidents due to human error with potentially enormous consequences.

One promising measure that addresses these issues is the introduction of autonomous unmanned ships. The advantages are many, including reduced crew cost, possibility for more efficient low-speed transport, as well as improved scalability. They do, however, come with great complexity, which the autonomous automotive industry can attest to. Before long distance ocean crossings or minute-long urban waterway crossings can be carried out autonomously, a vessel's onboard autonomy must be capable of operating safely in every situation, be robust to unforeseen events, and the necessary regulations must be defined and met. In practice, there are many intermediate steps such as supervised autonomy, both onboard and onshore, where some argue that completely unsupervised autonomy is unlikely to be feasible in commercial applications.

The intermediate steps taken in the automotive industry include self-driving modes offered on regular consumer vehicles, where the driver remains responsible for the vehicle and is ready to intervene should the autonomy system lose confidence in its own decision making. A similar system is proposed for autonomous vessels, where a shore control center can monitor one or more vessels and intervene if necessary. To do this remotely, the vessel must be outfitted with a suite of sensors capable of understanding and communicating information about its surroundings.

An additional challenge highlighted by the autonomous automotive industry is the need for testing using simulations. The California-based self-driving car company Waymo claim to simulate nearly 1000 miles of driving for every mile driven by a real vehicle, generating data at an enormous scale (Hawkins 2021). The cost of constructing, outfitting and operating autonomous vessels for testing purposes is large enough to make simulation-based testing essential in building trust in the vessel's autonomy. As essential as it is, this testing infrastructure including simulated operational environments, sensors and decision-making algorithms is a vast, complex system in itself, with the following data analysis being yet another component. In Zeabuz, the autonomous passenger ferry start-up this thesis is written in collaboration with, the complexity of the simulated testing infrastructure and accompanying result analysis is estimated to be at least as large of a task to implement as the autonomy stack itself.

A significant challenge in simulation-based testing of complex systems is that the results are only as good as the precision of the simulation and its parameters, which motivates the need for verification and validation of the simulation systems. As these simulations are used to verify and validate the autonomy system, the testing architecture can be modeled as a pyramid where each level is tested and verified using the level below, as modeled in Figure 1. To further complicate these testing systems, there are a number of reasons why conventional testing, verification and validation methods are insufficient in this application. The non-deterministic nature of autonomous systems that are capable of learning and adapting continuously (Helle et al. 2016) may prevent them from achieving formal verification. This is also a likely result of the nearly infinite input space within which an autonomous vessel operates, where new edge-case scenarios, however unlikely, could always be designed to invalidate the system.

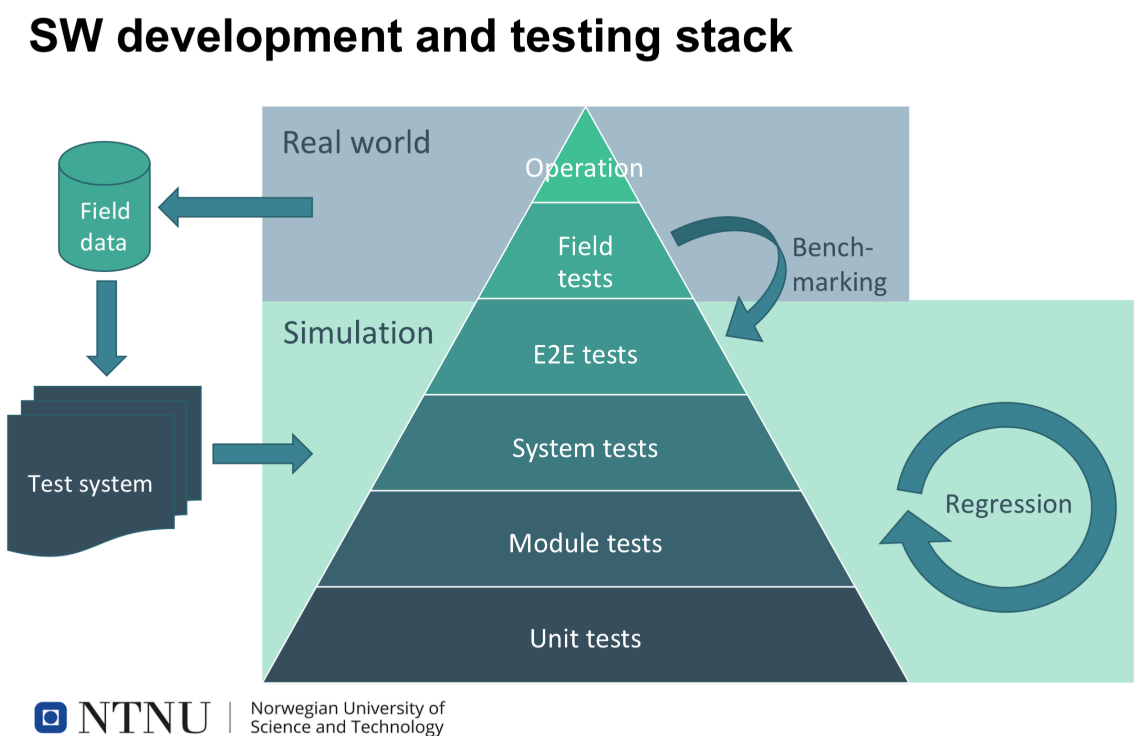

Figure 1: The software testing and verification hierarchy, where validation of the lower levels facilitate assurance of the higher levels. Source: Smogeli 2021

This thesis investigates whether the components of the aforementioned simulation pyramid can be inverted, specifically in the case of the situational awareness pipeline. Rather than using smaller, simpler building blocks to verify more complex, end-to-end simulations, this thesis proposes using compute-heavy simulations to validate the performance of simpler simulation models of the same system components. If the simulation results are similar enough, it is suggested that the simpler simulation could be used to verify portions of the input space at a much smaller compute cost, and reserving heavier simulations for edge-cases where the added uncertainty could influence whether the system reaches a failure state or not.

2 Theory

2.1 Simulation Fidelity

The term 'fidelity' can be understood differently in different contexts. In simulators intended for training of maritime personnel, simulator fidelity is defined as the degree to which a simulation imitates reality. It is classed as high or low, depending on how immersive or complex the simulations are perceived (Wahl 2019). This definition is similar to what is used in this thesis. The high-fidelity simulator indicates that the simulator is more complex and a better representation of reality. The low-fidelity simulator is in turn simpler, with a greater number of assumptions involved, but also much less complex.

This classification is somewhat of a simplification, as the current state of the Gemini simulator representing the high-fidelity option in this thesis is still very much a work-in-progress. Its performance is limited by the quality of the 3D models making up the rendered environment, and the validation of the simulator compared to reality is also an ongoing process. This is not considered to be problematic for the research question posed in this thesis as the focus is validation between two simulators, leaving the primary validation with reality as a high-priority suggestion for further work.

2.2 Simulator Verification and Validation

Verification and validation, often abbreviated as V&V, are a collection of procedures or tools used to determine whether a system or product fulfills all requirements and specifications. The terms are by some used interchangeably, but when it comes to V&V of simulators there is a well-defined distinction.

Verification is defined as 'a formal proof of correctness of the system for a (possibly infinite) set of parameters and inputs' (Kapinski et al. 2016). For a control system, for example, this implies that all specifications and regulatory requirements are fulfilled for all possible external parameters. For complex systems, formal verification can be very challenging or arguably impossible to achieve as the input space can have an infinite number of parameters which need to be verified. The same applies to the formal verification of an end-to-end autonomous ship simulator, where the simulator is verified if it can be shown that for every set of input parameters, the vessel's performance and behaviour matches the simulator's real-life counterpart exactly. Intuitively, this is an unrealistic goal, and defeats the purpose of simulations. For simulations to be useful, they must make certain assumptions simplifying the behaviour of the system they are modelling, and produce results that are within an acceptable margin of error. More complex simulators can be allowed to run slower and/or with greater computational requirements with the assumption that their results are more precise, and vice versa with simpler, faster simulators.

Although formal verification is at best impractical, some metric is required to measure the performance of the simulator to build trust in the results it produces. This is a non-trivial task for complex systems because there are a large number of factors influencing the results, including potential bugs in the autonomy stack that might be misunderstood as bugs in the simulator. Additionally, the question of which metrics should be used to measure the simulators performance is an ongoing field of research, and is dependant on the system components that are in focus. This process of quantifying simulator performance measured with respect to the real-life system is what is meant by simulator validation. This is shown in Figure 2, where model validation relates reality, in the physical domain, with the simulation model, in the digital domain. The figure is from the classification society DNV's Recommended Practice DNV-RP-0513, on Assurance of simulation models.

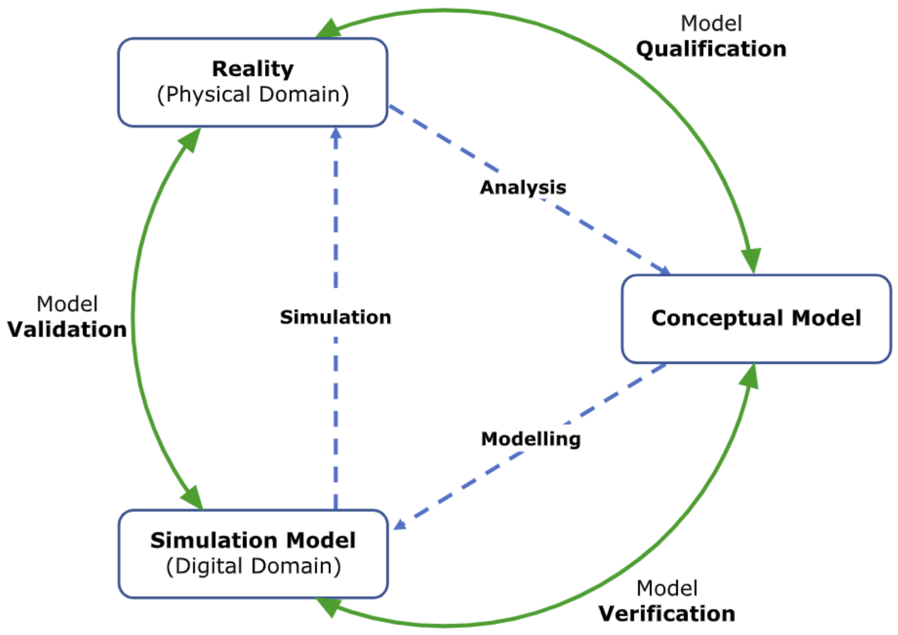

Figure 2: Relationship between reality, conceptual and simulation models, with related verification and validation activities. Based on (Schlesinger, 1979). Source: *Assurance of simulation models, DNV-RP-0513* 2021

For use in this thesis, the relationship shown in Figure 2 is iterated one step further down. The high-fidelity simulator takes the place of the 'Reality' component, and the low-fidelity simulator takes the role of 'Simulation Model'. As such, the high-fidelity simulator is regarded as the 'ground truth', and the performance of the low-fidelity simulator is measured with respect to the high-fidelity simulator rather than real data from the physical domain.

This tiered validation process facilitates the building of confidence in the low-fidelity simulator, as well as exposes cases where the assumptions of the simulation model break, necessitating the use of a higher-fidelity simulator.

2.2.1 Regulatory requirements for MASS Simulators

DNV has also published a class guideline on Autonomous and remotely operated ships in 2018 (*Autonomous and remotely operated ships, DNV-CG-0264* 2018). The document's objective, as it states in its introduction, is to provide guidance in the implementation of novel technologies related to remotely operated and autonomous ships. It provides a framework for the safe implementation for these technologies resulting in 'a safety level equivalent to- or better than conventional vessel operations'.

The guidelines do not go into great detail regarding the technical requirements of simulators meant to verify the performance of autonomy algorithms. It does, however, detail which requirements simulations can help fulfill, and underline the importance of a comprehensive verification and validation strategy, which a simulator can be a central part of. It also specifies that simulations are complimentary to standard regulatory requirements, which include physical testing in the vessel's operational environment.

Perhaps the most challenging requirement to meet regarding simulators is how they should 'render the real-life behaviour of the system'. The extent to which a simulator can fulfill this is difficult to determine, and requires a validation strategy of its own to quantify.

2.3 Situational Awareness performance metrics

The metric used to validate the low-fidelity simulator with respect to the high-fidelity simulator considers the situational awareness algorithm performance. Since this algorithm is configured identically in both simulators, one assumes that any discrepancies between them is thanks to limitations, differences or errors in the simulators. As a result, the absolute performance of the SITAW algorithm is of no direct interest in this thesis, but a well-functioning algorithm allows for greater confidence in the conclusions drawn from the results. Rather, the relative performance between the simulators can offer insights into the strengths and weaknesses of them, and even potentially help unearth issues in the SITAW algorithm itself.

A challenge in defining useful performance metrics for the SITAW algorithm is the large portions of scenarios lacking data. For example, a traffic vessel defined in the scenario configuration might operate outside of the ownships' maximum sensor range for the entire length of the scenario, in which case it will understandably never be detected. At a glance, this result might be misunderstood as a failure of the SITAW algorithm, and as a perfect correspondence between the two simulators; As they have identical results. As such, the performance metric must be biased towards 'populated' time periods, where the SITAW algorithm is publishing a state estimate, whilst also accounting for the possibility that the algorithm is not providing a state estimate when the target vessel is within range, and therefore detectable.

The Result Manager, described in Section 3.2.5, is the software component designed to quantify performance and visualize results from the simulator. An important parameter to take into account is to determine at what points the ownship could plausibly track the target vessel, as most often it will be out of range or obstructed from view. This is done by estimating a line-of-sight boolean state, using a simple geometric map of the ownship's immediate surroundings, and determining whether the sightline to a target vessel is obstructed by land.

2.3.1 Batched Benchmarks

Although single scenarios can provide useful insight into the performance of the SITAW algorithm, to properly test it a large number of simulations with different scenarios must be conducted. This raises the issue of how to effectively present the results of these simulations, without losing valuable information by reducing time series data to single values.

A proposed benchmark is to align data based on the time when the target vessel becomes detectable, as in, enters within maximum sensor range or becomes unobstructed from view. This is by no means an extensive benchmark, and its usefulness can be questioned when very long ranged sensors such as radars are used. Despite this, the results generated using this benchmark have proven useful in the overall simulator evaluation.

$$\begin{aligned} e_a(t) &= \Big(\frac{1}{n}\sum_{i=1}^{n}\sqrt{e_i(t-t_i)}\Big)^2 \quad , \quad t > t_0 \\ t_0 &= \Big(\frac{1}{n}\sum_{i=1}^{n}\sqrt{t_i}\Big)^2 \end{aligned} \tag{1}$$

Equation 1 shows how the 'batched average' position error trend is calculated. For i scenarios with a respective position error time series e_i, an inverse RMS value is calculated aligned on the time the target vessel entered range, t_i. The inverse RMS is well suited for this application as the goal is to isolate the data's trend, rather than highlight specific scenarios with large positional errors as the regular RMS would. The same procedure is done with the timestamp t_i, allowing for the calculation of the batched average detection delay time. In the software implementation of this benchmark, additional measures are taken to handle the case where one or more scenarios lack data at given timesteps.

2.4 Coordinate frames and transforms

To be able to correctly calculate performance metrics of the SITAW algorithm, a number of coordinate frames and transformations between them are used to relate data from different sources. ROS has built-in functionality handling euclidean transforms between frames, which simplifies this process significantly. For example, in the configuration file specific to milliAmpère 2, there is a tree of transforms relating the position and rotation of various sensors and other points of interest to the vessel center. ROS's transform tool can use these to lookup the current pose of a given vessel component in the local NED frame. The local NED frame used in milliAmpère's current operation is named 'piren', and has its origin on the north shore of the canal.

Some datasources, such as the geometric map used for line of sight estimation, are non-euclidean and therefore must be manually transformed into the piren NED frame. More specifically, the data was given in latitude (ϕ) and longitude (λ) in WGS 84. The NED frame coordinates can be calculates as shown in Equation 2 (Foster and Mullaney 2014), using parameters given in Table 1.

$$\begin{aligned}
R_N &= \frac{r_e}{\sqrt{1 - e^2 \sin(\phi_0)}} \\
R_M &= R_N \frac{1 - e^2}{\sqrt{1 - e^2 \sin(\phi_0)}} \\
x_{NED} &= \frac{\phi - \phi_0}{\operatorname{atan2}(1, R_M)} \\
y_{NED} &= \frac{\lambda - \lambda_0}{\operatorname{atan2}(1, R_N \cos(\phi_0))}
\end{aligned} \tag{2}$$

Parameter	Value	Name
e	0.0818	Eccentricity [-]
r_e	6378137	Equatorial Radius [m]
ϕ_0	63.43890291	Piren Frame Origin Latitude [deg]
λ_0	10.39908278	Piren Frame Origin Longitude [deg]

Table 1: Parameters used for transformation from LLA to NED

3 Method

3.1 MilliAmpère 2

The milliAmpère 2 passenger ferry, shown in Figure 3, is part of a research project lead by NTNU with the intention of facilitating innovation in marine autonomy. It is the second generation prototype ferry outfitted with an extensive sensor suite and DP-capable azimuth propulsion system, becoming a capable test platform for research in situational awareness, computer vision, collision avoidance, autonomous COLREGs compliance and perceived safety of autonomous systems, to name just a few. Building on this research, the intent is for milliAmpère 2 to enter regular operation transporting passengers across the canal from Ravnkloa to Vestre Kanalkai in Trondheim starting the summer of 2022. In the first phase, a ferry operator will remain aboard the vessel ready to override the autonomous control in case of malfunctions or emergencies, however the longer term goal is for this operator to be moved onshore in a remote control center.

As significant portions of the autonomy stack were developed as a part of the milliAmpère research project, milliAmpère 2 and its sensor suite was chosen as the ownship to simulate in this thesis. Being the default configuration of the end-to-end simulator, opting for an alternative vessel would have required extensive work not directly relevant to the research question. Of the many cameras and sensors making up MilliAmpère 2's sensor suite, the Navico Halo Radar and two Ouster Lidars are used by the SITAW algorithm. The computer vision algorithm using cameras is a separate system component also contributing to the vessel's situational awareness, but is not considered in the scope of work.

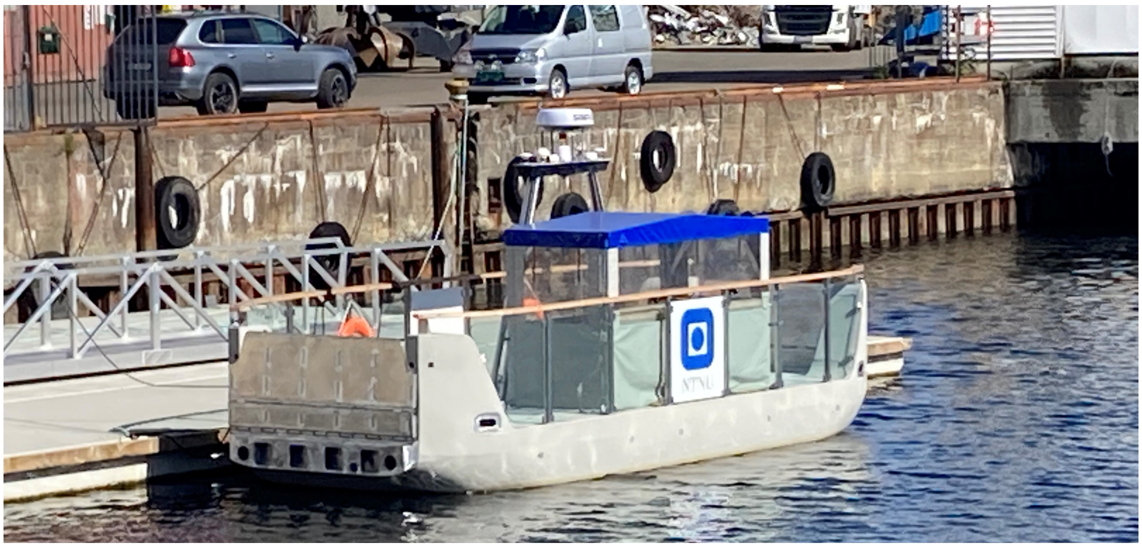

Figure 3: The milliAmpère 2 ferry docked in Nyhavna, Trondheim. Source: Robin Stokke

Vessel remote control and surveillance is a central component of the sister research project lead by NTNU's Department of Design, namely the Shore Control Lab, shown in Figure 4. Here, the open-ended questions regarding how best to remotely operate a vessel and how best to communicate the vessel's surroundings to an operator are being investigated. Taking it one step further, research is also being conducted on the remote operator his/herself, testing which system configurations aid their performance, and how stress might influence their decisions. Simulations play an essential role in recreating challenging situations that are unlikely to happen in real life, but could be dangerous if not properly handled.

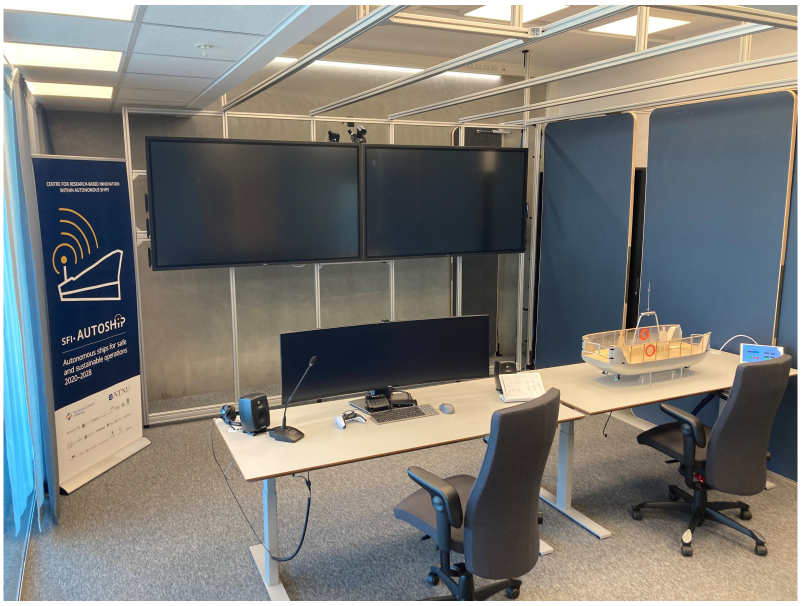

Figure 4: The Shore Control Lab located in Trondheim Maritime Center. The aluminum profile structure is meant to allow for rapid prototyping of different control room configurations. Source: Robin Stokke

3.2 Software Architecture

Due to the nature of the complex system architecture, no single diagram could intuitively portray the intricacies of communications between the many system components. Therefore, a number of diagrams are shown to explain different aspects of the system.

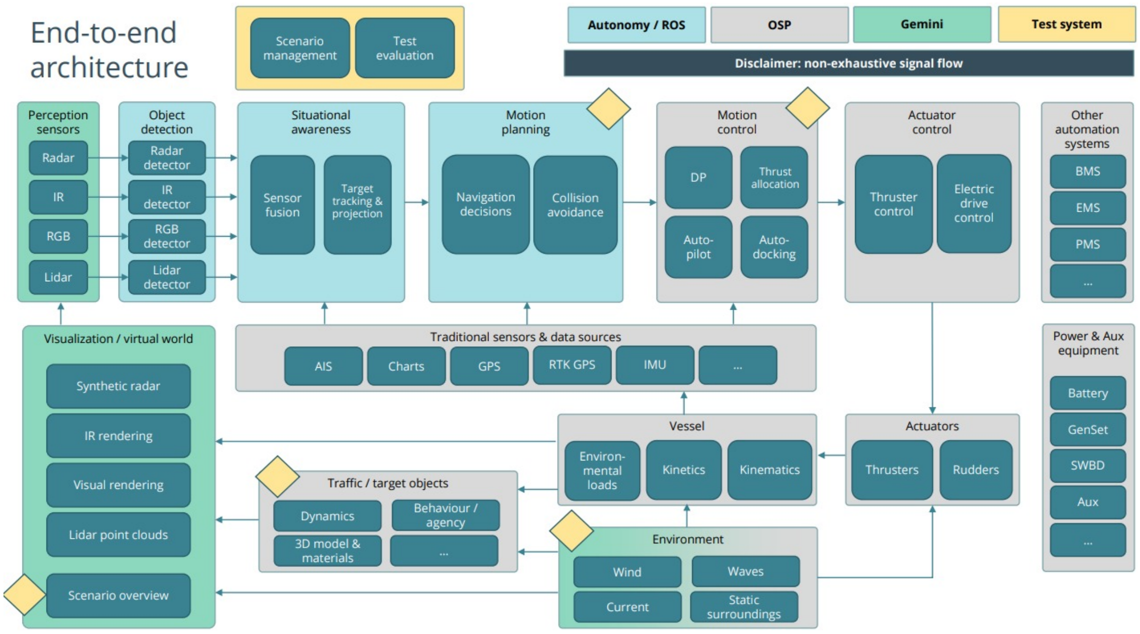

Figure 5: End-to-end simulation architecture used in Zeabuz. Source: Smogeli 2021

Figure 5 shows the end-to-end simulator used for the assurance of the autonomy stack intended for Zeabuz' passenger ferries. Interfaces between components mimic the real interfaces as far as practically possible, and the components responsible for simulating and interacting with the vessel's operating environment are designed to be so realistic that the autonomy stack operates identically as on a real vessel. The operating environment has so many variables that achieving identical performance in an end-to-end simulation and in real life is likely impossible, however quantifying these similarities and differences is the reason why simulator validation is essential.

The entire end-to-end simulator is outside of the scope of this thesis, as only the left-hand side of Figure 5 concerning sensor simulation and situational awareness is considered. Of all of the system components, the 'Visualization/Virtual World' and 'Perception Sensors' components are the most computationally expensive, mostly due to their reliance on GPU compute. To run the entire end-to-end simulator in real time, a relatively expensive computer with a ray-tracing enabled graphics card is necessary, making scaling of the simulations more challenging and more expensive. Therefore, the mentioned rendering components are selected as the focus of this thesis, where the potential performance increase balances the lower simulation precision.

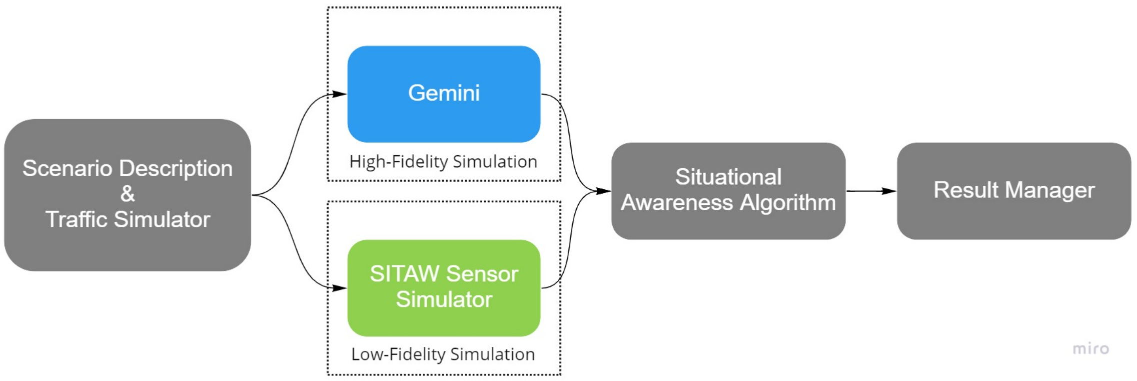

Figure 6: High-level overview of the system components and their interaction

Figure 6 demonstrates the scope and information flow of the software components used in this thesis. The green, 'low-fidelity' simulation described in Section 3.2.3 is proposed as a low compute cost alternative to Gemini, in blue. As the diagram makes clear, both components have identical inputs and outputs, allowing them to be used interchangeably in the pipeline.

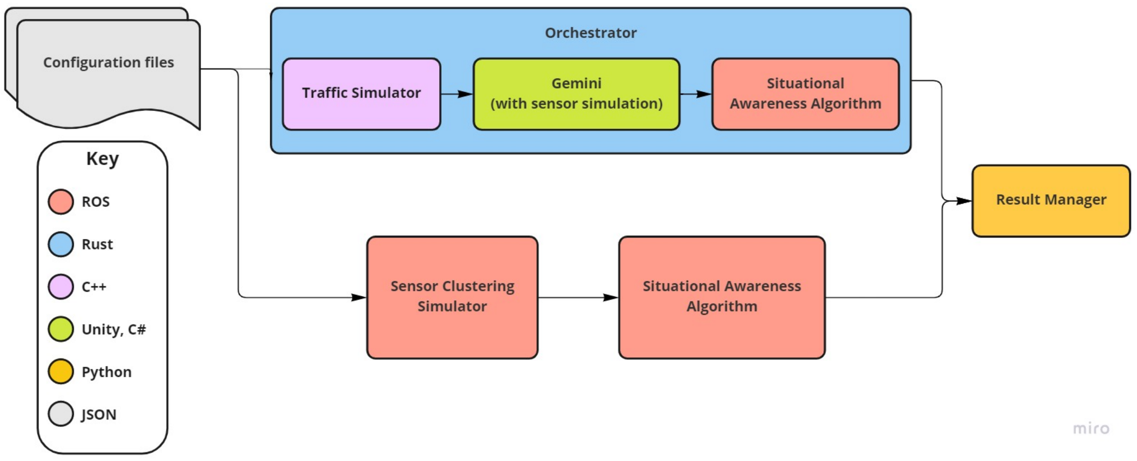
Configuration files
Orchestrator
Traffic Simulator
Gemini (with sensor simulation)
Situational Awareness Algorithm
Result Manager
Sensor Clustering Simulator
Situational Awareness Algorithm
Key
ROS
Rust
C++
Unity, C#
Python
JSON
miro

Figure 7: A detailed overview of the system components, their language or platform of implementation, and the interactions between them

Figure 7 shows in greater detail the information flow as implemented specifically for this thesis. The simulation orchestrator shown in blue is a framework that starts up and initializes all necessary software components to run a full end-to-end simulation. The orchestrator can be configured to spin up only specific components or portions of the end-to-end simulator, however there was no practical need to do this in the thesis work. As such, the orchestrator in practice initialized many more components than is shown in Figure 7, such as the MPCS and vessel dynamics.

The orchestrator outputs a rosbag with logs of all relevant system states, such as the ownship position and heading, as well as the SITAW estimates. This logfile is read by the Result Manager to generate plots, animations, and to determine whether the system performance is acceptable with respect to a number of predefined error tolerances.

To ensure the low fidelity simulation shown in the bottom pipeline in Figure 7 is run with the same parameters, the ownship data from the orchestrator's rosbag is replayed and used as reference. In a proposed end-to-end simulation using the low-fidelity component, the ownship would be controlled by the MPCS, in the same was as in the high-fidelity case.

Each of the components shown in Figure 7 are described in further detail in the following subsections.

3.2.1 ROS

The Robotics Operating System[1], generally referred to as ROS, is a widely-adopted communication framework for robotics applications. By standardizing interfaces between all software and hardware components in a realtime system, ROS simplifies integration for the end user significantly. From the manufacturer's perspective, offering a ready-to-use ROS interface for their product can save many of their customers the work of writing custom drivers for their system, giving them a competitive edge.

ROS is used for a significant portion of the end-to-end simulation at Zeabuz. The traffic simulation, SITAW and COLAV algorithms run in ROS, simplifying the complex realtime interaction between the components. ROS also has dedicated logging tools, where all communications between components are saved in a rosbag, which can later be replayed or post-processed to validate system performance.

3.2.2 Traffic Simulator

The Traffic Simulator is a C++ library that deterministically simulates the movement of vessels in the ownship's surrounding area. It has a number of different path finding and control algorithms which a vessel can be configured to use, such as A-star and a Line-of-sight waypoint controller. Its deterministic behaviour is important, as it allows a given scenario to be recreated identically with a slightly changed parameter elsewhere in the system. Generating pseudo-random traffic to test the autonomy system in many situations is a necessary step in the assurance process, but this is instead done by randomizing the input parameters in the Traffic Simulator's configuration file.

In this thesis, the waypoint controller is used to imitate regular vessel behaviour in the canal across which milliAmpère 2 operates. This includes abiding by the speed limit of 5 knots, and generally remaining closer to the starboard side of the canal.

3.2.3 Situational Awareness Simulator

The situational awareness simulator, adapted from the work of PhD. Erik Wilthil (Wilthil 2019), simulates the output of the SITAW algorithm without visually rendering the vessel's operational environment, making it much less computationally intensive to run. The simulator consists of two parts, the first of which generates emulated clustered pointcloud scans, and the second of which evaluates the scans to output tracks of estimated vessel positions. Using the sensor's position in relation to these traffic vessels, the scan generator generates corresponding ellipses accounting for the distance between them and whether line of sight is broken by major geographic features.

The simulator is by nature limited in use, as it does not account for a number of factors including

[1] https://www.ros.org/

weather, pitch and roll of the ownship or minor obstacles completely or partially obstructing the sensor's line of sight. It importantly only generates data based on actual target vessels, making false-positives impossible, and without the inclusion of sensor noise the SITAW algorithm is given an easier task. Simultaneously, it provides a more realistic simulation than bypassing the SITAW algorithm altogether and relaying ground truth positions of traffic vessels to the COLAV algorithm, which is a solution still used in practice for faster simulations run without graphical rendering.

Additionally, the simulator is well-suited for machine learning applications where the scan generator could be trained to mimic the output of a real sensor or a simulated sensor in Gemini, for example. This could allow for the generation of multiple sensor profiles applicable in different conditions, such as varying weather, or in open or closed waters.

3.2.4 Gemini

Gemini[2] is an open-source project originating as a student project at NTNU in the course 'Experts in Teamwork'. The software is best described as a toolbox for electromagnetic sensor emulation built on the video game engine Unity. Taking advantage of modern 3D rendering techniques, such as hardware accelerated raytracing, Gemini simulates the output of optical and infrared camera, lidar and radar, and broadcasts the feeds and point clouds to other software systems over gRPC. As such, Gemini is a necessary component for full closed-loop software-in-the-loop simulations, including vessel dynamics, SITAW and COLAV algorithms.

As with any simulation, the simulation results are only as trustworthy as the assumptions and limitations of the simulator itself. Gemini is no exception, where the simulated sensor outputs are directly reliant on the fidelity of the 3D models making up the environment around them. This is a significant challenge, as it is currently an open research question how the performance of computer vision models differ with real and generated data. PhD candidate Kjeil Vasstein, one of Gemini's main contributors, discussed this in his masters thesis (Vasstein 2020), proposing a performance metric quantifying this difference.

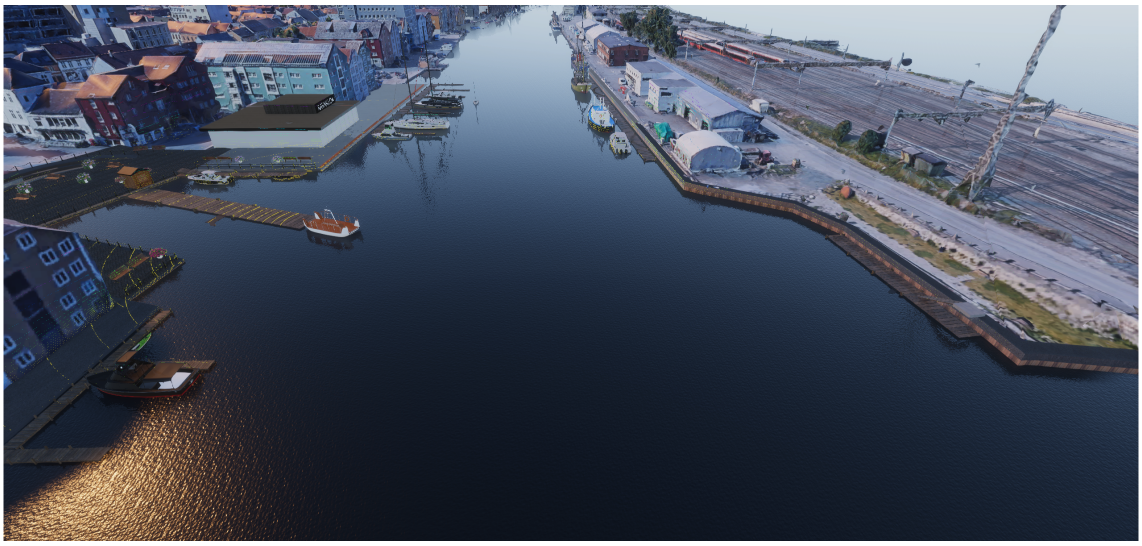

Figure 8: A screenshot of Gemini, with milliAmpère 2 docked at Ravnkloa

Figure 8 shows milliAmpère 2 in its simulated operational environment. On some of the features in the environment, small, colored dots can be seen, which are a visualization of the generated lidar point cloud.

[2]https://github.com/Gemini-team/Gemini

3.2.5 Result Manager

The Result Manager is a standalone software module written in Python meant to evaluate the autonomy system's performance based on a number of metrics. The module currently contains metrics for the MPCS and SITAW systems, of which the latter are of relevance to this thesis. The SITAW metrics are calculated by comparing the ground truth positions and headings of nearby vessels with the estimates delivered by the SITAW algorithm, all of which are recorded in a ROS bag file. The calculated metrics include the RMS position and heading error, initial detection delay and tracking time delay. In addition to publishing the system performance, acceptance thresholds for the various metrics can be defined to determine whether the system meets given criteria.

Over the course of the thesis work this software module has grown significantly to include post-processing, scenario batching, plotting and animation tools, playing an important part in the generation of results.

4 Results

Unfortunately, the current state of the high-fidelity simulator is limited to using only lidar sensors. This means that the milliAmpère 2's sensor suite is reduced from a radar, a fore- and an aft lidar to only the lidars. This limits the performance of the SITAW algorithm, and to maintain the greatest possible similarity between the simulators, the radar was disabled in the low-fidelity simulator as well. Simulation results with the radar enabled are included mainly for demonstration purposes in Section 5.

4.1 Individual Scenarios

The following figures display results from a single test scenario, run in both the low-fidelity and high-fidelity simulators. A total of four scenarios were used, with figures from the remaining three included in Appendix A.

Please note that in all of the scenarios the ownship remains completely stationary. This is due to an issue in the end-to-end simulator, which was unfortunately not addressed in time for this thesis. This does not prevent properly comparing the performance of the simulators, although it will be an incomplete comparison not accounting for potential discrepancies introduced by movement and rotation of the ownship and its sensors. An advantage of the ownship not moving is it offers a more intuitive understanding of the scenarios and when target vessels enter and leave maximum sensor range.

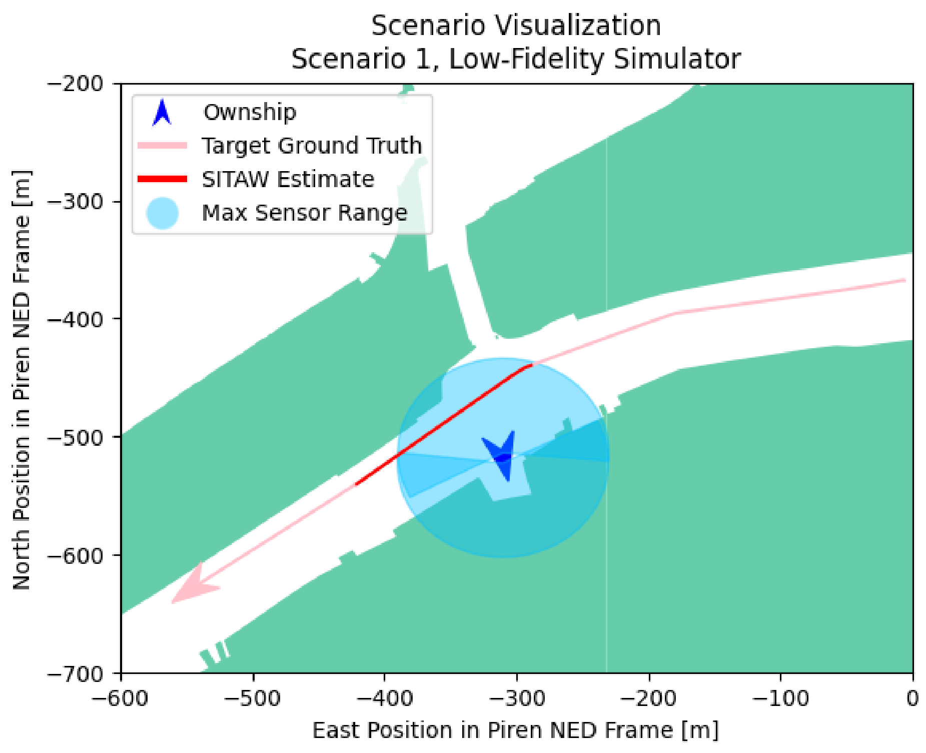

Figure 9: Overview of Scenario 1, Low-Fidelity Simulator

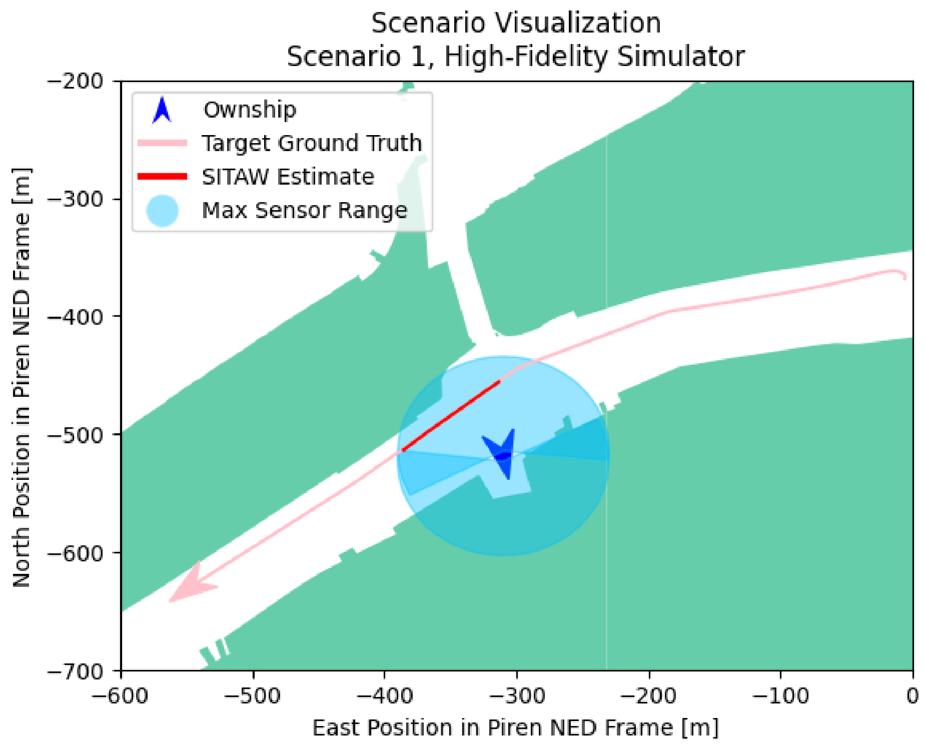

Figure 10: Overview of Scenario 1, High-Fidelity Simulator

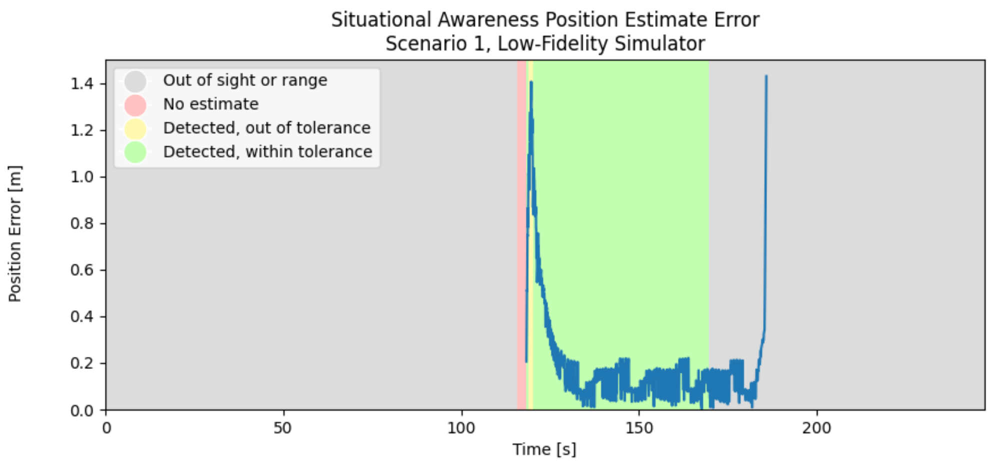

Figure 11: SITAW algorithm position error in Scenario 1, Low-Fidelity Simulator

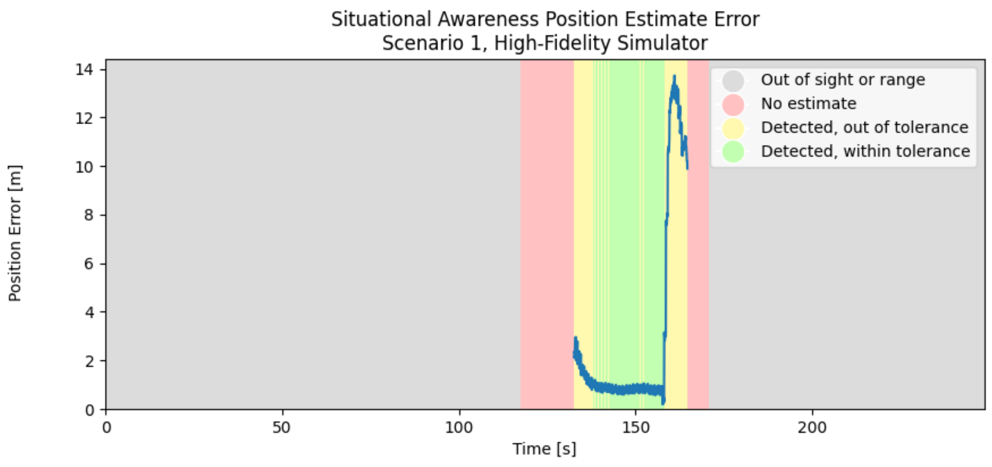

Figure 12: SITAW algorithm position error in Scenario 1, High-Fidelity Simulator

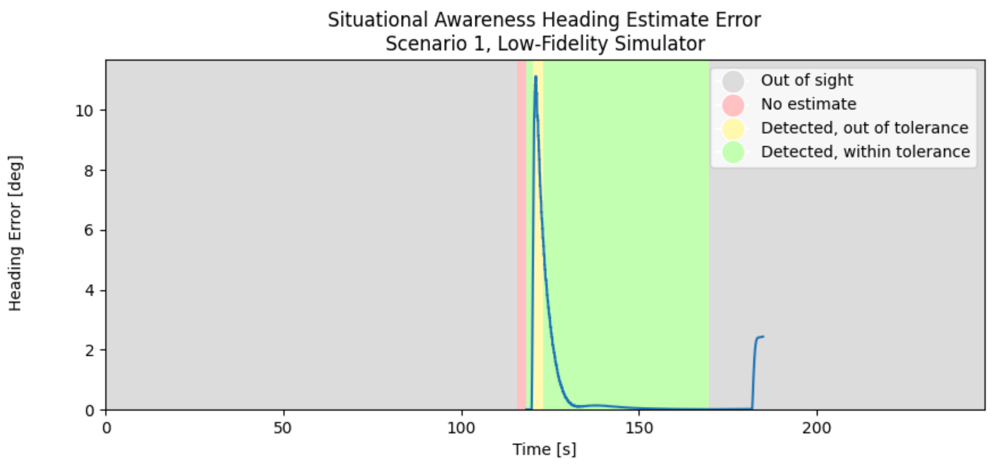

Figure 13: SITAW algorithm heading error in Scenario 1, Low-Fidelity Simulator

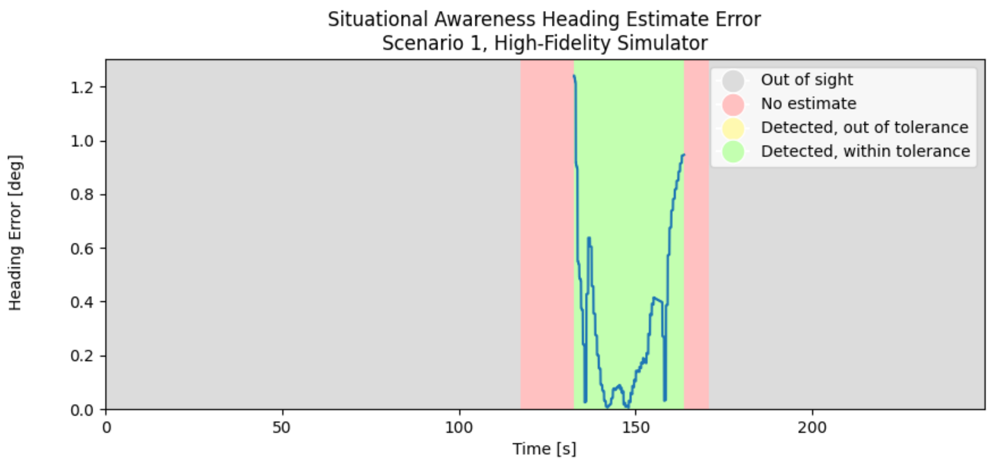

Figure 14: SITAW algorithm heading error in Scenario 1, High-Fidelity Simulator

In addition to the scenario visualizations, animated GIFs have been generated to more intuitively describe the scenarios. These are included in a folder accompanying the document. Figure 15 shows an example of what these animations look like.

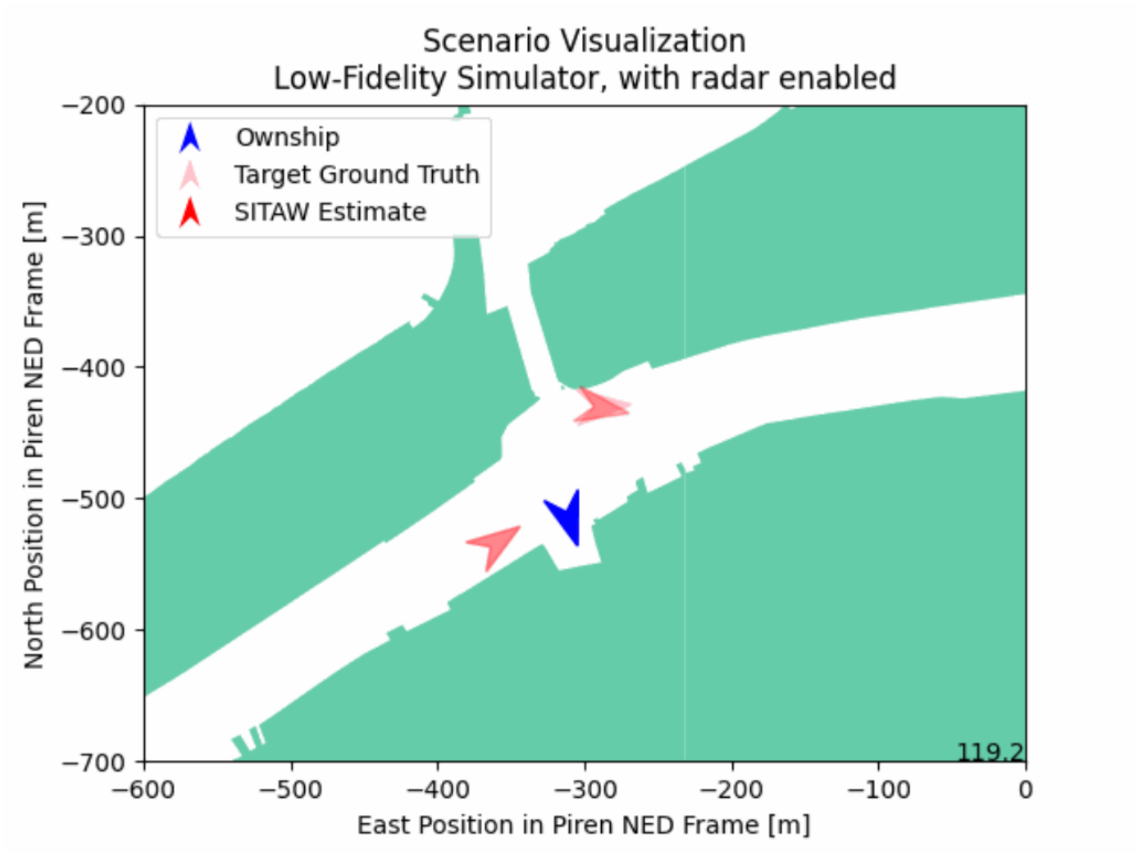

Figure 15: Freeze-frame of scenario animation

4.2 Simulator Comparisons

As discussed in Section 2.3, the SITAW algorithm performance is in itself of no direct interest to this thesis. The comparison between the high- and low-fidelity simulators is the focus, shown in the following figures. Note that the time axis is zoomed in to the periods where the target vessel is in close vicinity to the ownship.

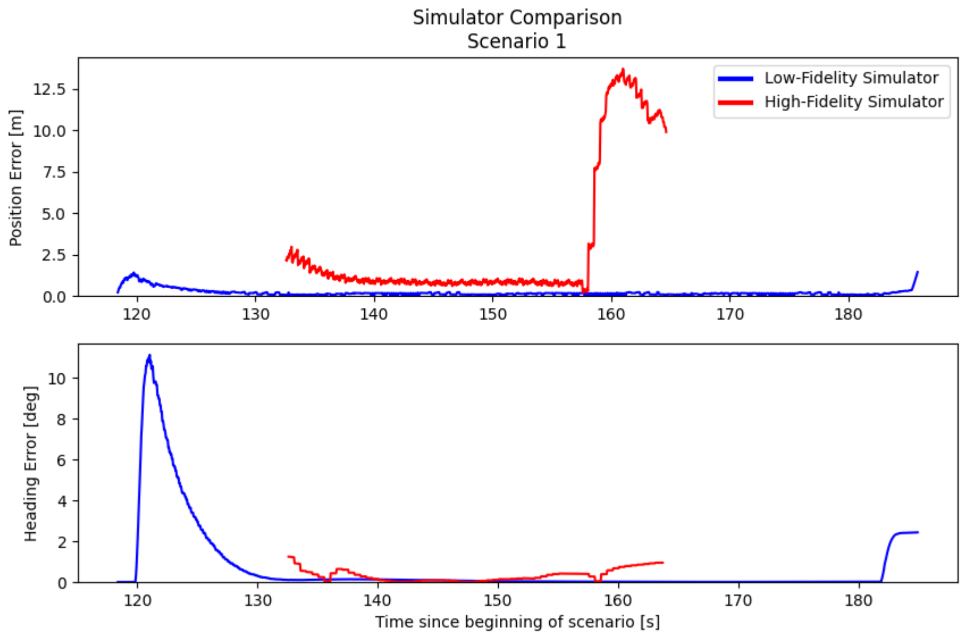

Figure 16: Comparison of SITAW algorithm performance in Scenario 1

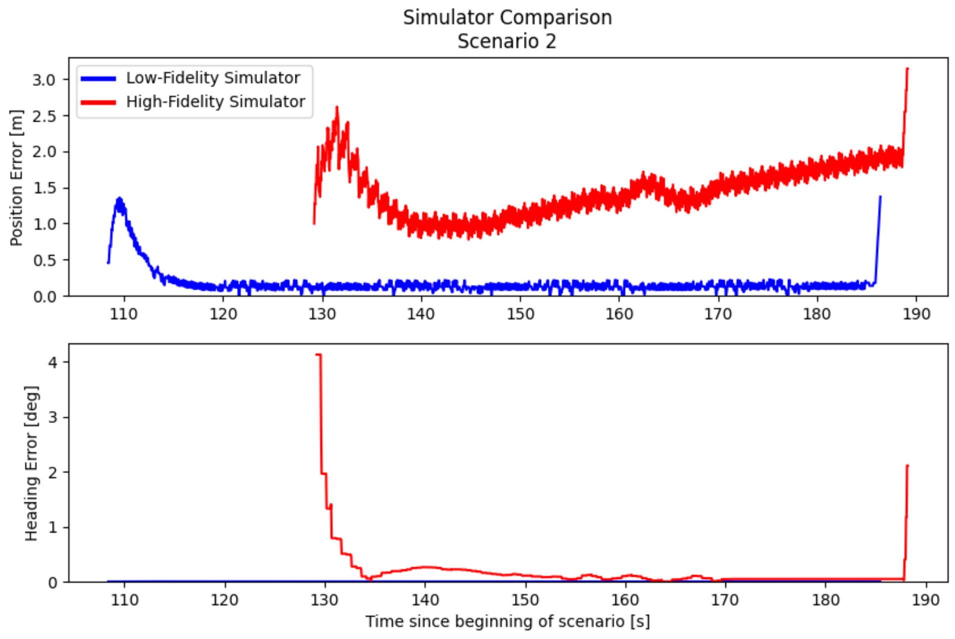

Figure 17: Comparison of SITAW algorithm performance in Scenario 2

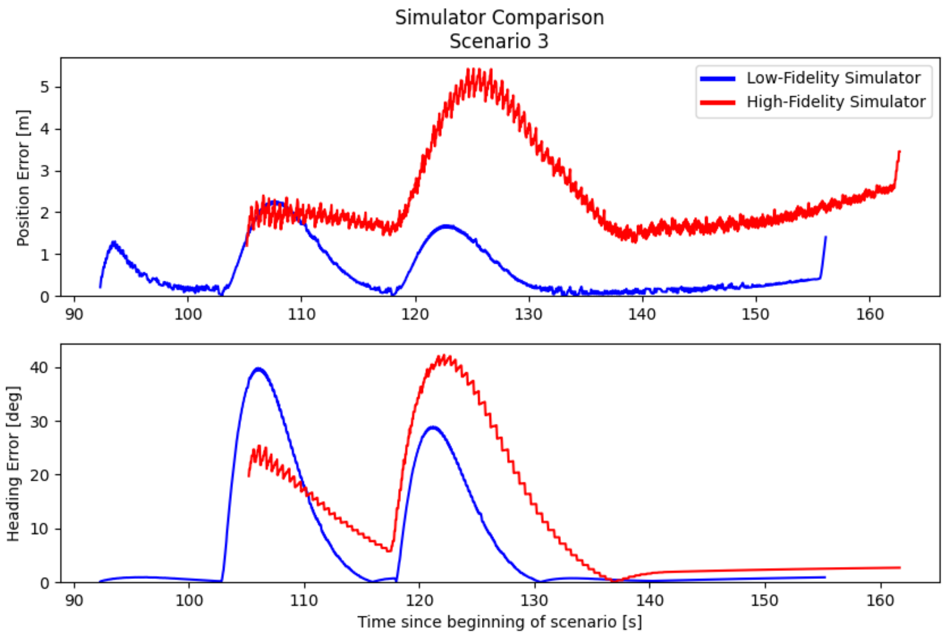

Figure 18: Comparison of SITAW algorithm performance in Scenario 3

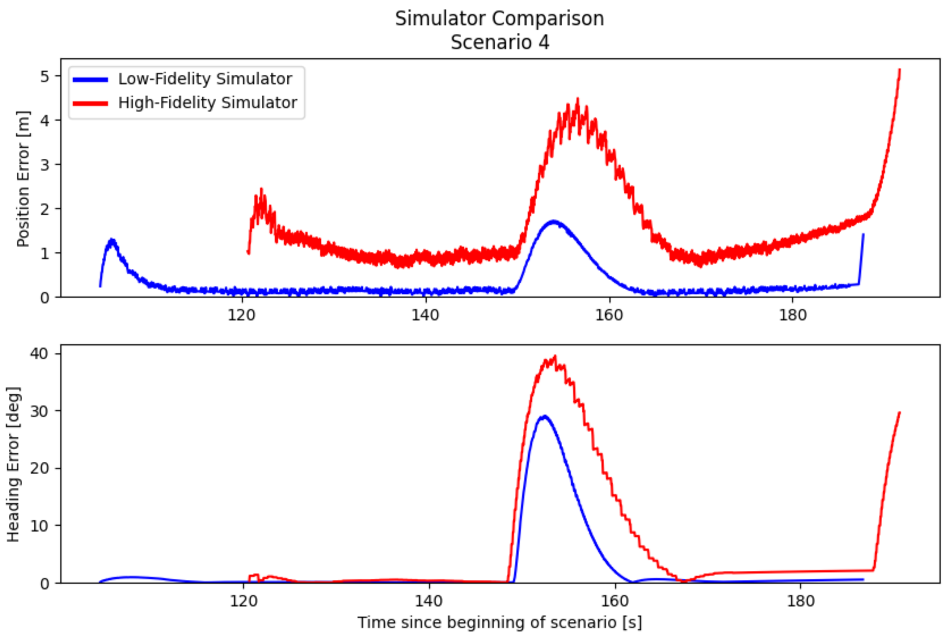

Figure 19: Comparison of SITAW algorithm performance in Scenario 4

4.3 Batched Simulator Comparisons

Figure 20 is the culmination of the four scenarios run on each of the simulators. The scenario result batching method proposed in Section 2.3 is used to plot the 'Batch Average' values. The average detection times T_{Lo} and T_{Hi} are displayed with dotted lines and their values written explicitly.

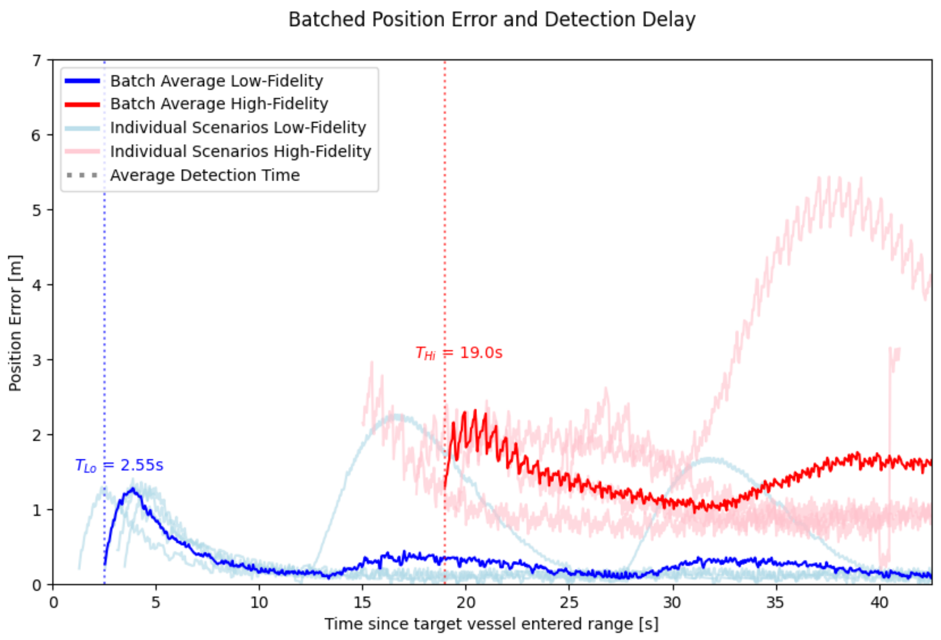

Figure 20: The performance trend of the SITAW algorithm measured with respect to the time the target vessel enters sensor range

4.4 Intermediate Results

During the development of the Result Manager application used to generate the figures in this section, seemingly erroneous results were occasionally discovered, an example of which is shown in Figure 21 and 22.

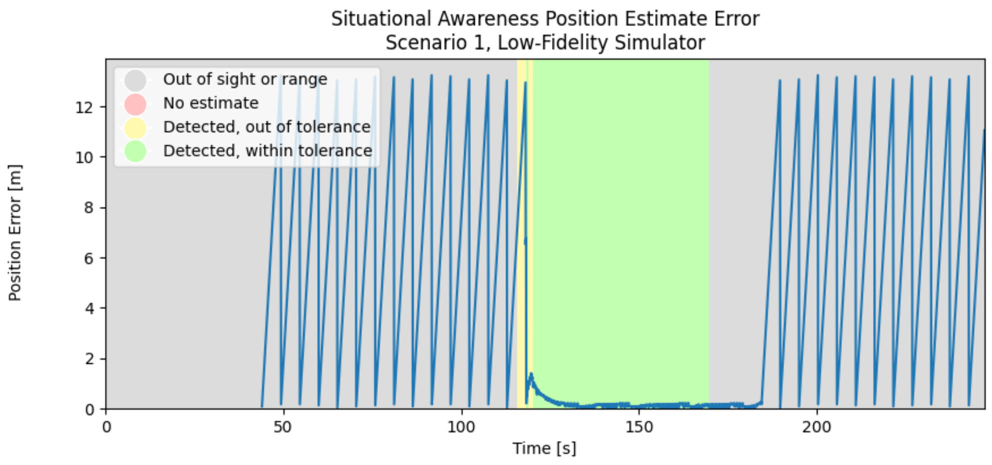

Figure 21: Intermediate Result Manager position error results from Scenario 1, Low-Fidelity Simulator

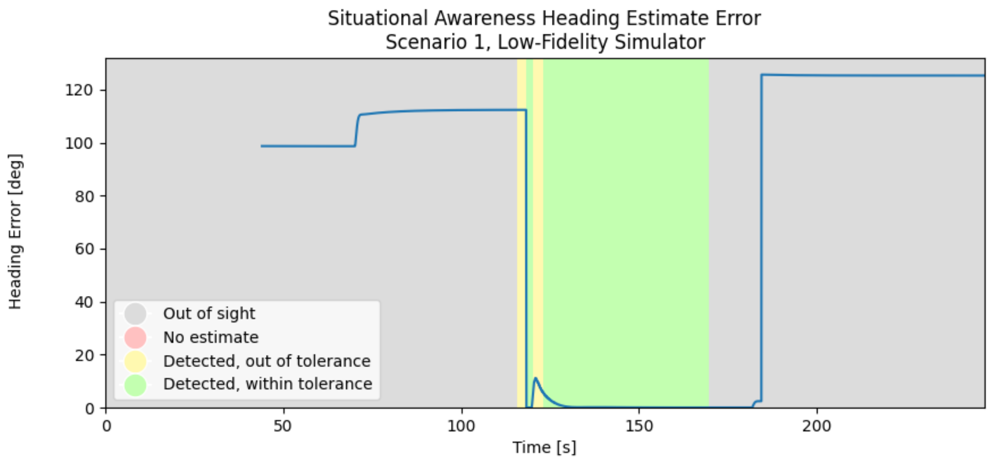

Figure 22: Intermediate Result Manager heading error results from Scenario 1, Low-Fidelity Simulator

5 Results with Radar enabled

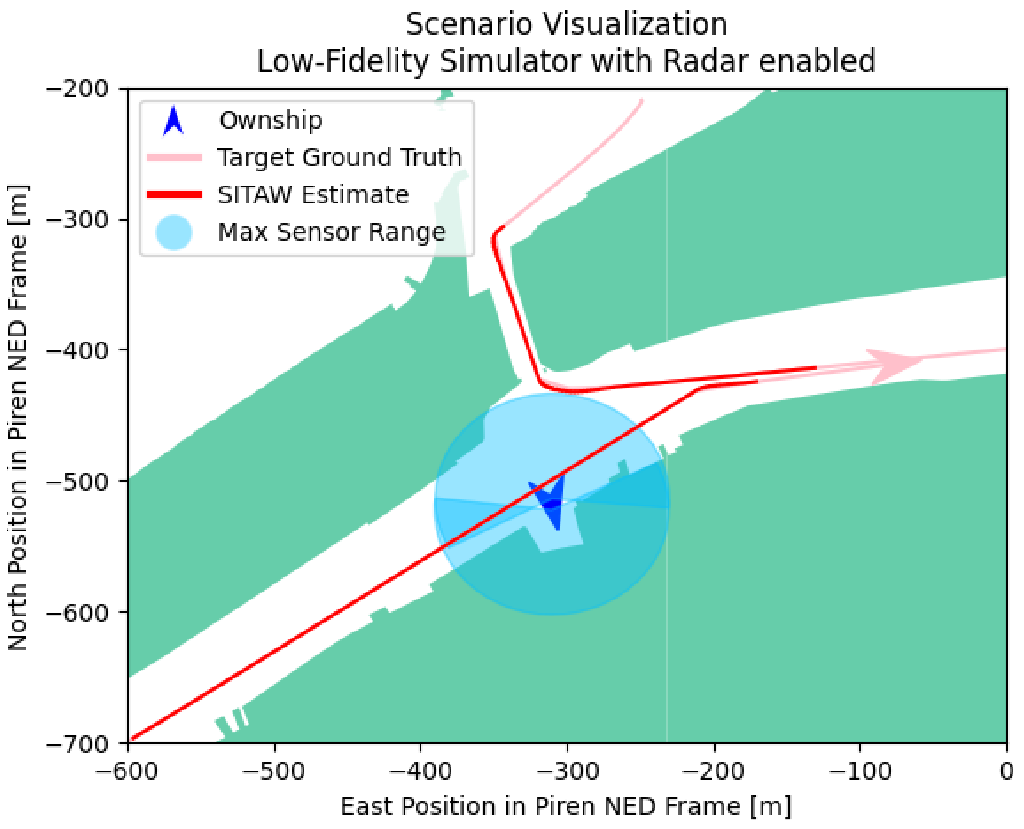

Figure 23: Overview of Scenario with radar enabled, Low-Fidelity Simulator

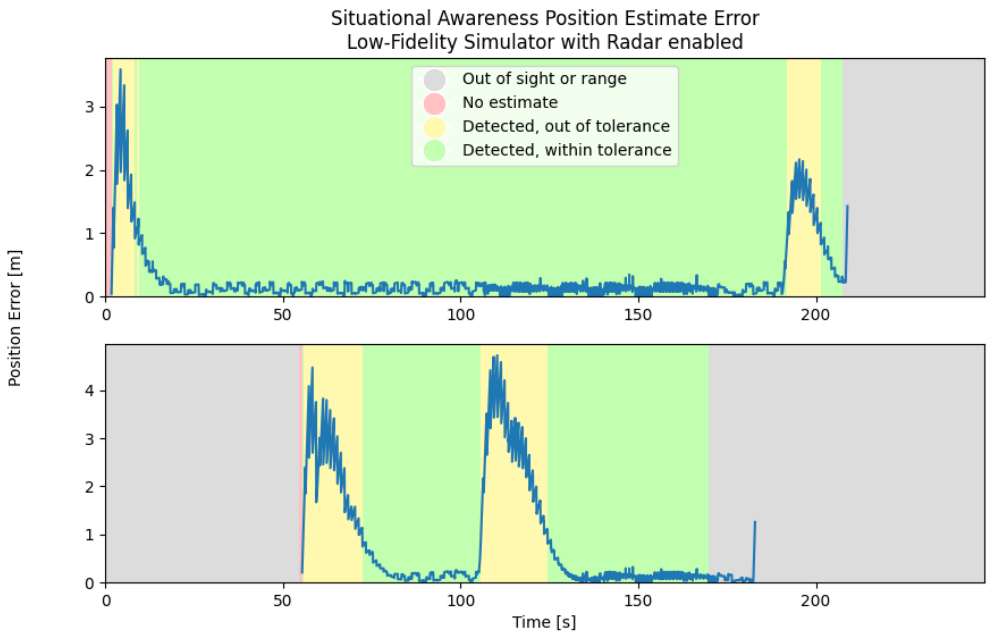

Figure 24: SITAW algorithm position error in scenario with radar enabled, Low-Fidelity Simulator

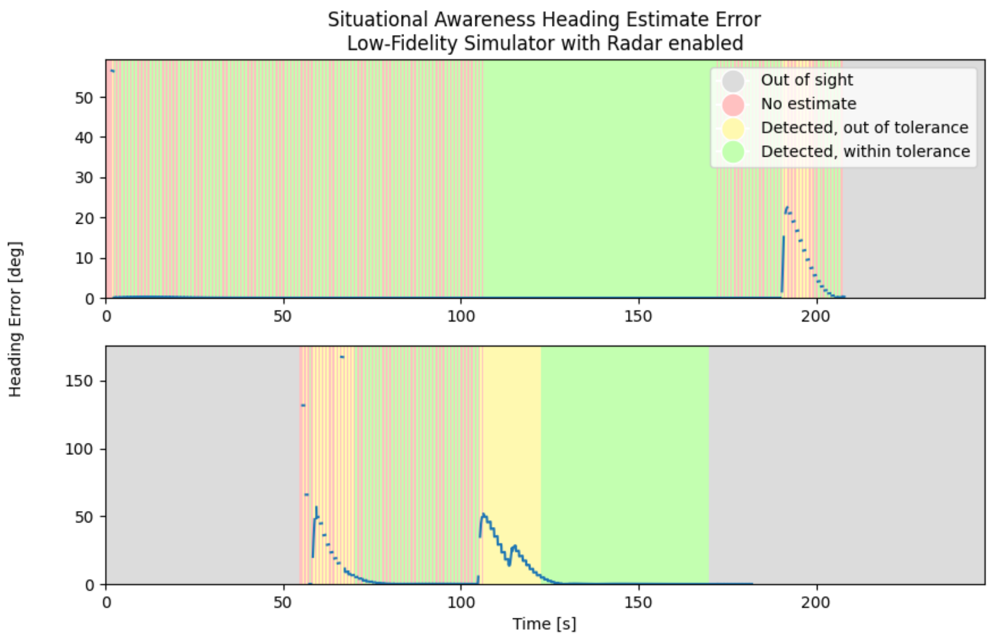

Figure 25: SITAW algorithm heading error in scenario with radar enabled, Low-Fidelity Simulator

6 Discussion

6.1 Simulator Performance Comparison

To preface this discussion, it is reiterated that the focus of this thesis is on the comparison between the high- and low-fidelity simulators, and not on the performance of these simulators or the SITAW algorithm. Despite this, some improvements were introduced in the interest of obtaining more useful results. First and foremost substantial work was required to complete the low-fidelity simulator pipeline, as there was no existing interface between the Traffic Simulator and the SITAW clustering simulator. Furthermore, the SITAW simulator did not account for any geographic features, which required the implementation of a line-of-sight condition based on a geometric map of the vessel's operational environment.

Comparing Figure 9 and 10, showing an overview of the first scenario, shows that the scenarios run in the high- and low-fidelity scenarios match almost perfectly. This is an essential assumption all further results are based on. The figures do not portray the time component of the scenario, but glancing at Figure 11 and 12 confirms that the target vessel enters and exits maximum sensor range at the same time. The animations included with this document also confirm this. The only discrepancy between the target trajectory in each of the simulators can be seen at the beginning of their paths, on the right-hand side of the overview figures. The high-fidelity simulator has a small turn, which is not present in the low-fidelity simulator. This difference is a result of how the two logging tools operate. The logging tool in the high-fidelity simulator begins logging roughly 0.08 seconds after the simulation is initiated, whereas its low-fidelity counterpart begins roughly 0.85 seconds after initiation. This split-second difference means some data is lost in the beginning of low-fidelity simulation, however this does influence the rest of the scenario due to the deterministic design of the Traffic Simulator.

The transparent, blue wedges emanating from the ownship representing the max range and angle of the two lidars onboard milliAmpère 2 are useful as a 'sanity check'. In both Figure 9 and 10 the SITAW estimate is mostly limited to within the sensor range, and importantly is not established before the target vessel has entered the sensor range. The low-fidelity simulator continues to have a confident position estimate roughly 25 meters after the target vessel has left this range, which is plausible as the algorithm has established a course and velocity estimate which can be used to extrapolate position for a certain time period before the uncertainty grows too large. The visualization does not account for where the sensor's view is obstructed by geographical features, which is a potential for improvement.

The color coding on Figure 11 and 12 offers a quick, but descriptive, understanding of the SITAW algorithm performance. Without the grey areas, for example, it would be more difficult to judge whether the algorithm performance is satisfactory, as there are large periods of time where the target vessel is not tracked. Furthermore, the user is able to define an error tolerance which results in either a yellow or green status, which is useful as the y-axis varies in scale based on the largest observed position error. This applies to Figure 13 and 14 as well.

Moving on to the comparison figures in Section 4.2, there are two main points that all four scenarios have in common. Firstly, the low-fidelity simulations provide an initial state estimate significantly earlier than the high-fidelity simulations. This can be explained by the low-fidelity simulator's sensor cluster generator not having any noise modelling, and as such the SITAW algorithm quickly achieves a high confidence in the simulated sensor data. As the target vessel leaves the maximum sensor range, there is not a clear trend, where the high-fidelity simulator maintains an estimate longer in Scenario 3, the opposite in Scenario 1, and in Scenarios 2 and 4 the difference is small. Secondly, the high-fidelity simulations generally have a significantly higher position and heading error than the low-fidelity simulations. This is also likely a result of the lacking noise contribution in the low-fidelity simulator, but could also be a result of lower latency in the low-fidelity simulator. Since the entirety of the low-fidelity simulator is run in ROS, lower latency is expected than in the high-fidelity simulator's case where data must first be exported from Unity over gRPC. This latency of each case is however not measured, so the importance of this contribution is uncertain. As for similarities between the simulators, Figure 18 and 19 from Scenarios 3 and 4 show that the response from the target vessel changing course is relatively similar. The high-fidelity simulator's state error increases a bit more, and takes longer for the error to lower and level out, but the general shape is alike. The width and height of these bell-like curves could be a useful tool in quantifying the performance of the SITAW algorithm, as it is in many ways a more important performance indicator than the steady-state error when the target vessel has an unchanging heading.

The batched simulator comparison shown in Figure 20 clearly show the aforementioned trends in the two simulators. The target vessel detection delays T_{Lo} and T_{Hi} are not far from an order of magnitude apart, and the same goes for the steady state position errors of the simulators. The resulting batched average performance trends would benefit from a greater sample size, as only using four scenarios makes the results susceptible to large variations due to edge-cases in each of the scenarios. This was unfortunately impractical, as scenario generation is at the moment a manual process, where a scheduling system to queue simulations would be useful. Despite this, there is no indication that a greater number of scenarios would significantly influence the conclusions drawn from the figure.

6.2 Result Manager Evaluation

The Result Manager tool used to generate the results laying the foundation for the evaluations in the previous subsection is where the majority of work in this thesis was focused. As a tool to evaluate the performance of a situational awareness algorithm, and in extension compare the performance of the simulators generating sensor data, it offers a number of insights which are not immediately evident from the raw data.

For example, during the development of the Result Manager, it was discovered that the tool was logging both preliminary and confirmed tracks from the SITAW algorithm. A preliminary track is created in the intermediate phase whilst the algorithm is building confidence in its position estimate. This can be a useful feature, but for most use cases only the confirmed tracks are of interest, as only the confirmed tracks are sent to the MPCS component for motion planning and collision avoidance. Figure 21 and 22 show an example from the low-fidelity simulator where both preliminary and confirmed tracks are displayed. The repetitive, sawtooth behaviour of the position error are from preliminary tracks, and helped uncover an issue in the SITAW algorithm's track generation logic. Rather than first checking the distance to the input cluster to determine whether the data is likely noise or not, the algorithm initialized a track first, and immediately after disregarded it as the distance was well outside of the max range of sensor. This resulted in periodic preliminary tracks being broadcast.

The Result Manager is still primarily built for evaluating individual scenarios, and as such does not provide much in terms of long-term, general performance metrics. These metrics are necessary to validate a simulator with a higher-fidelity simulator, or with real data. The scenario batching suggested in Section 2.3 and shown in Figure 20 is a step in the right direction, where a trend begins to emerge from the small sample size. It can also be argued that the detection delay benchmark used in Figure 20 is a poor parameter to compare simulators with, as there are a large number of factors influencing this delay, but there are few proposed alternatives to offsetting time series data to be able to align them like is done here. Many benchmarking parameters can be boiled down to a single, time independent value, however much of data's intricacies are lost in doing so. As such, a combination of single-value (1-dimensional) and time dependent (2-dimensional) benchmarks is suggested.

To properly facilitate for the necessary data processing for this style of scenario batching, it is proposed that the Result Manager tool should be split into two distinct parts. The first should parse the simulation logs, and generate all necessary additional data to store the simulation results. This includes all of the existing metric calculations, such as position and heading error, line of sight, distance between ownship and target vessel, as well as additional parameters of interest. This component could also include simple plotting functions similar to those that generated the figures in Section 4.1. All of these metrics, in addition to the original log data, should be saved in a database, which interfaces with the second Result Manager component. This component is responsible for querying the database, combining data based on the scenario configurations, the type of simulation (or real data), and the results themselves. Using this, the user can request plots displaying various performance metrics given a number of parameters, and even comparing current algorithm performance to previous iterations. A tool such as the one described here is a necessary step in the assurance of the autonomy stack in an autonomous vessel, as both simulations and field tests are nearly worthless without the ability to properly evaluate the results from them.

The figures in Section 5, where the Navico Halo Radar onboard milliAmpère 2 is enabled, suggests an additional use for the Result Manager tool. The scenario includes 2 target vessel approaching from two different directions, and it is immediately apparent that the SITAW algorithm begins tracking them much earlier than only with lidars, as the radar has a much greater maximum range. It should be noted that the radar detection range is not shown with a blue indicator in Figure 23. Referring to Figure 24, there is an observable change in the data once the target vessel (particularly the first) enters the lidar's range. This is even more apparent in Figure 25, where the radar data alone is not sufficient to maintain a continuous heading signal, resulting in the colored stripes in the background. As such, the Result Manager can be used to quickly test the performance of different sensor suite configurations, either with entirely different sensors, or sensors positioned or configured differently.

6.3 Simulation Validation Viability

To address the main research question, the presented results do not support the validation of the low-fidelity simulator with respect to the high-fidelity simulator. The differences between them, especially in the detection delay, are great enough to significantly influence the MPCS system and the scenario's outcome. As such, the low-fidelity simulator would not operate well as a replacement for the high-fidelity simulator for portions of the input space at this point in time. However, the results do not indicate that this proposal wouldn't work on a general basis, and with targeted improvements to the low-fidelity simulator using the Result Manager as an aid there is reason to believe a greater similarity could be achieved without greatly increasing compute cost.

The nature of the research question makes it difficult to conclude, as the validity of the hypothesis is heavily dependant on the two simulators included in the validation process. In this specific case, the low-fidelity simulator proved to have too many assumptions and simplifications to properly model all the system dynamics at play. As such, the hypothesis can only be confirmed, not falsified, as there are always unmodeled systems and behaviours which can help reduce the differences between the simulators within the defined acceptable error.

The defined acceptable error is another important aspect to consider. Depending on the motivation behind running a specific simulation, various magnitudes of error can be accepted, and this heavily influences the validation process. For example, if the main goal is to extensively test the collision avoidance algorithm in an end-to-end simulator, using the current low-fidelity simulator could make sense. Using the high-fidelity simulator would take too long and require too much computing power, whereas the final alternative would be to bypass the SITAW algorithm altogether and directly feed target vessel ground truth positions to the collision avoidance algorithm. For this application, it could be argued that the error introduced by low-fidelity simulator is acceptable and a significant improvement on the bypassing solution. Since this error can be quantified by the Result Manager, certain edge-case scenarios where the system is within this margin of error to failure (collision) could be re-run using the high-fidelity simulator, where the time and compute cost can be accepted due to the lower quantity of scenarios.

7 Conclusion

As briefly mentioned in Section 6.3, the nature of the posed research question prevents a conclusive answer applicable to any two simulators modelling the behaviour of the same physical system. At best, one can conclude that for two specific simulators, and for a given margin of error, the low-fidelity simulator can be validated using results from the high-fidelity simulator. The low-fidelity simulator can then provide results within said margin of error in either the entire input space, or at least in the area of the input space covered in the validation process.

In the specific case of the two simulators used in this thesis, such a conclusion cannot be drawn, as the differences between them surpass any reasonable margin of error. An exception may be for use in end-to-end simulations where the SITAW algorithm is not the primary focus, however even these results should be used with caution. Using the estimated detection delays from Figure 20, the $2.5ms^{-1}$ forward velocity of the target vessels, and a sensor range of 80 meters, the low-fidelity simulator would inform the MPCS system of vessels on a potential collision course at a distance of roughly 75 meters, and the high-fidelity simulator at only roughly 40 meters. This would considerably influence the behaviour of the MPCS system. Despite this conclusion, the work conducted in this thesis has quantified the differences between the simulators, and helped suggest a number of changes that would reduce these differences to a level which would make the low-fidelity simulator useful in many more situations.

7.1 Further Work

The Result Manager tool has the potential to play an important role in simulator validation, but at the time being it lacks a wider range of performance benchmarks on which the validation can be based. The detection delay after a target vessel enters sensor range, and the following position error trend, is a good performance indicator but does not capture other potential differences. For example, the SITAW algorithm response when a target vessel changes course could be another benchmark. Measuring the increase in position and heading error, as well as the time before the error returns to steady state, would lay the groundwork of a useful benchmark. This suggestion, as well as others, require extensive post-processing capabilities with access to the smallest details from all system components; Such as the exact moment a target vessel initiates a course change, or the point at which a target vessel enters the detection range of each sensor onboard. This requirement can be met with the database querying component proposed in Section 6.2, where all of these parameters are saved in a database such that a high-level evaluation tool can quickly query and compile data from relevant scenarios and present them to the user. The same database could be queried by automated testing tools, that can compare performance of the most recent changes to historical data to ensure the changes are maintaining or improving performance.

Further work on the low-fidelity simulator itself would first and foremost include modelling sensor noise, which should impair the SITAW algorithm's performance to a level closer to the high-fidelity simulator. The generated sensor data from the high-fidelity simulator could in fact be very helpful in informing the noise modelling. Machine learning could be applied to train a model imitating the output of the high-fidelity simulator. Achieving improved correspondence between the simulators would facilitate further research into the use of dual simulators, such as using adaptive stress-testing on the low-fidelity simulator to design high-risk scenario configurations which can be examined in closer detail with the high-fidelity simulator.

An additional suggestion for further work relevant to the research question would be to test the hypothesis on two completely different simulators, modelling a different physical system. Situational awareness is a well-suited application for dual simulators, as the difference in compute cost between a low-fidelity and high-fidelity simulator is large, but there are many other systems where high-precision simulations are expensive and time consuming, such as computational fluid dynamics (CFD) analysis. In such a case, the tools developed in this thesis would be of little help as completely different performance metrics are used, but the concept remains the same.

Bibliography

[1] *Assurance of simulation models, DNV-RP-0513*. 2021. URL: https://rules.dnv.com/docs/pdf/DNV/RP/2021-06/DNV-RP-0513.pdf.

[2] *Autonomous and remotely operated ships, DNV-CG-0264*. 2018. URL: https://rules.dnv.com/docs/pdf/DNV/CG/2021-09/DNV-CG-0264.pdf.

[3] EU Commission. *Reducing emissions from the shipping sector*. 2021. URL: https://ec.europa.eu/clima/eu-action/transport-emissions/reducing-emissions-shipping-sector_en.

[4] Roger Foster and Dan Mullaney. 'Basic Geodesy Article 018: Conversions and Transformations'. In: (2014).

[5] Andrew J. Hawkins. *Welcome to Simulation City, The Virtual World where Waymo tests its Autonomous Vehicles*. 2021. URL: https://www.theverge.com/2021/7/6/22565448/waymo-simulation-city-autonomous-vehicle-testing-virtual.

[6] Philipp Helle, Wladimir Schamai and Carsten Strobel. 'Testing of autonomous systems–Challenges and current state-of-the-art'. In: *INCOSE international symposium*. Vol. 26. 1. Wiley Online Library. 2016, pp. 571–584.

[7] IMO. *Fourth IMO GHG study 2020*. 2020. URL: https://www.imo.org/en/OurWork/Environment/Pages/Fourth-IMO-Greenhouse-Gas-Study-2020.aspx.

[8] James Kapinski et al. 'Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and Verification Techniques'. In: *IEEE Control Systems* 36.6 (2016), pp. 45–64. DOI: 10.1109/mcs.2016.2602089.

[9] OECD. *Ocean Shipping and shipbuilding*. 2019. URL: https://www.oecd.org/ocean/topics/ocean-shipping/.

[10] Øyvind Smogeli. *Simulation-based testing and verification, TMR06*. 2021.

[11] Kjetil Vasstein. 'Autoferry Gemini: A real-time simulation platform for electromagnetic radiation sensors on autonomous ships'. In: *IOP Conference Series: Materials Science and Engineering* 929 (2020), p. 012032. DOI: 10.1088/1757-899x/929/1/012032.

[12] Aud M. Wahl. 'Expanding the concept of simulator fidelity: The use of Technology and collaborative activities in training maritime officers'. In: *Cognition, Technology amp; Work* 22.1 (2019), pp. 209–222. DOI: 10.1007/s10111-019-00549-4.

[13] Erik F. Wilthil. 'Maritime Target Tracking with Varying Sensor Performance'. In: (2019).

Appendix

A Exhaustive Results

A Scenario 2

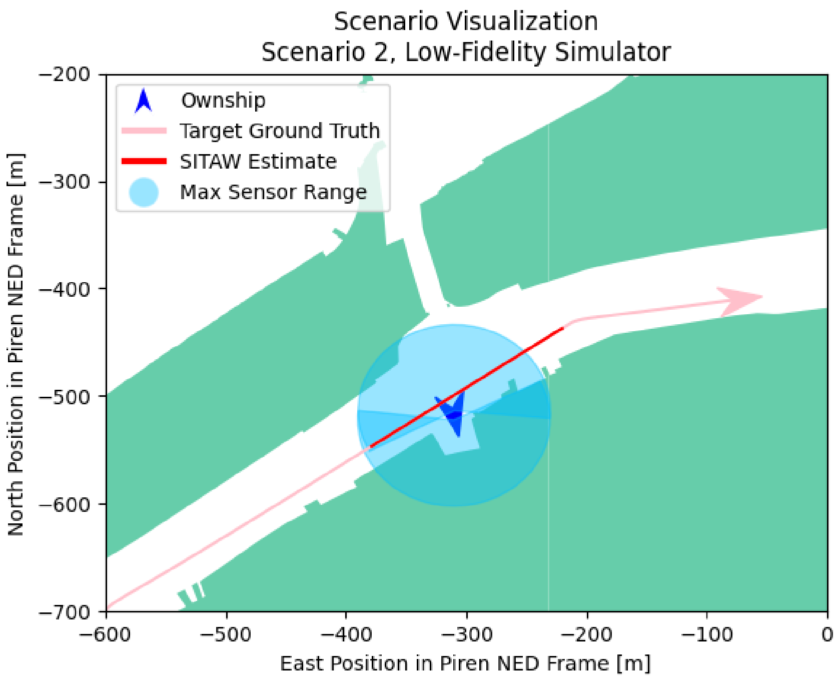

Figure 26: Overview of Scenario 2, Low-Fidelity Simulator

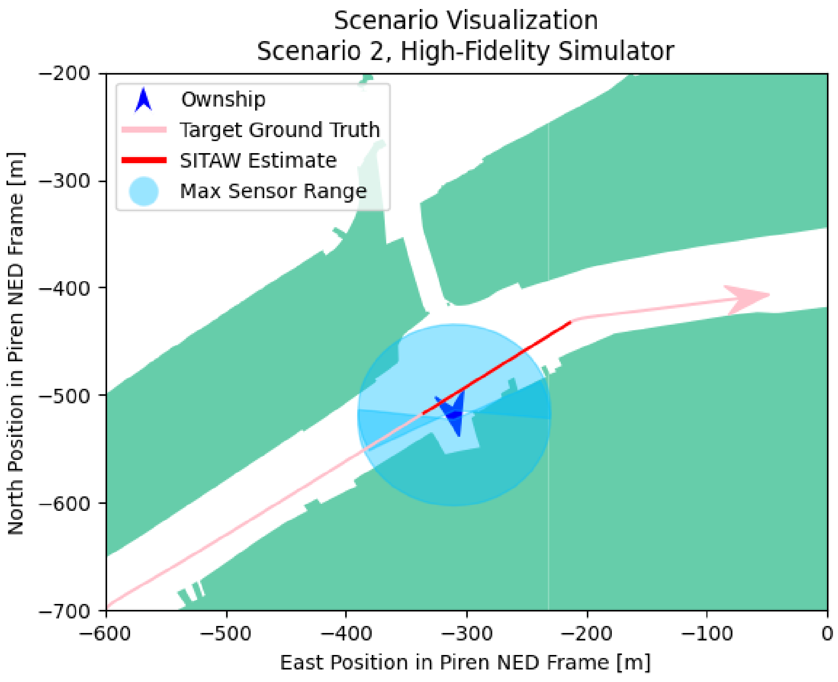

Figure 27: Overview of Scenario 1, High-Fidelity Simulator

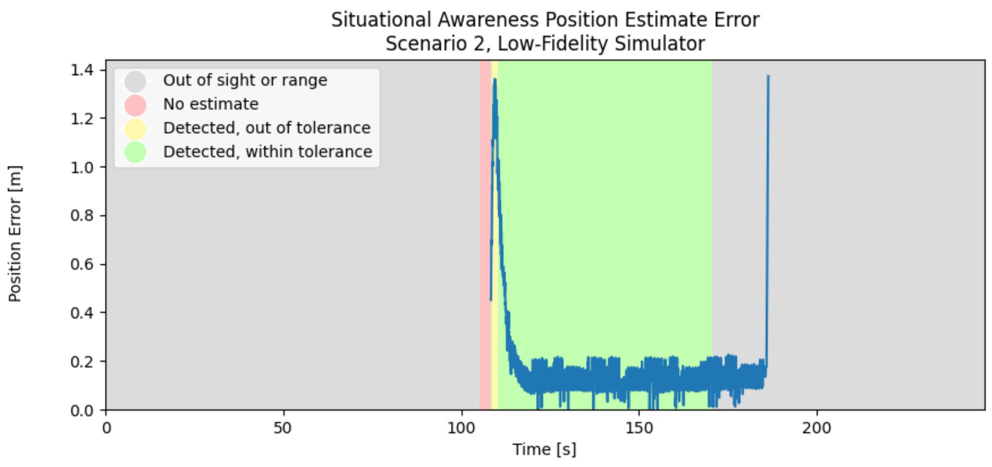

Figure 28: SITAW algorithm position error in Scenario 2, Low-Fidelity Simulator

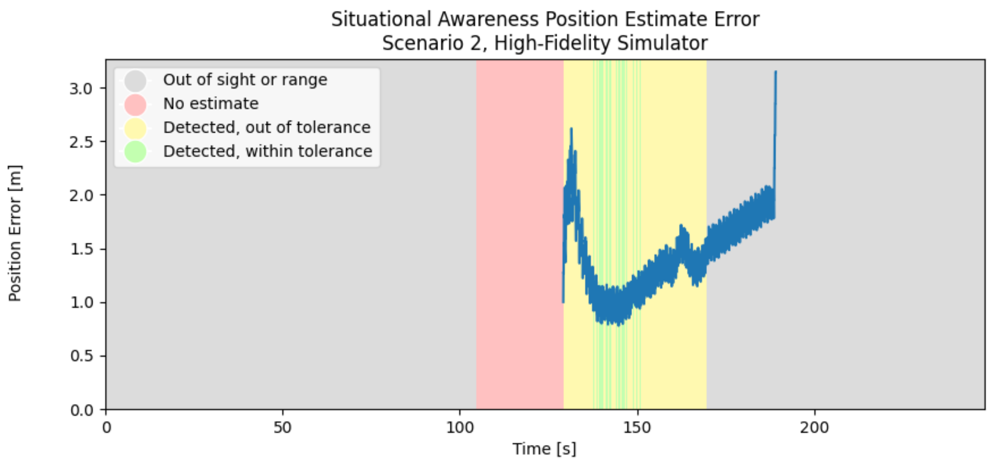

Figure 29: SITAW algorithm position error in Scenario 2, High-Fidelity Simulator

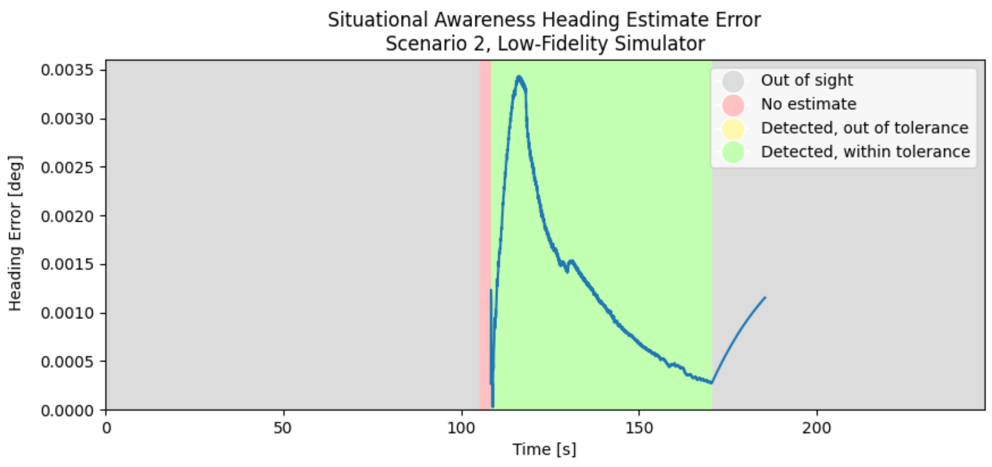

Figure 30: SITAW algorithm heading error in Scenario 2, Low-Fidelity Simulator

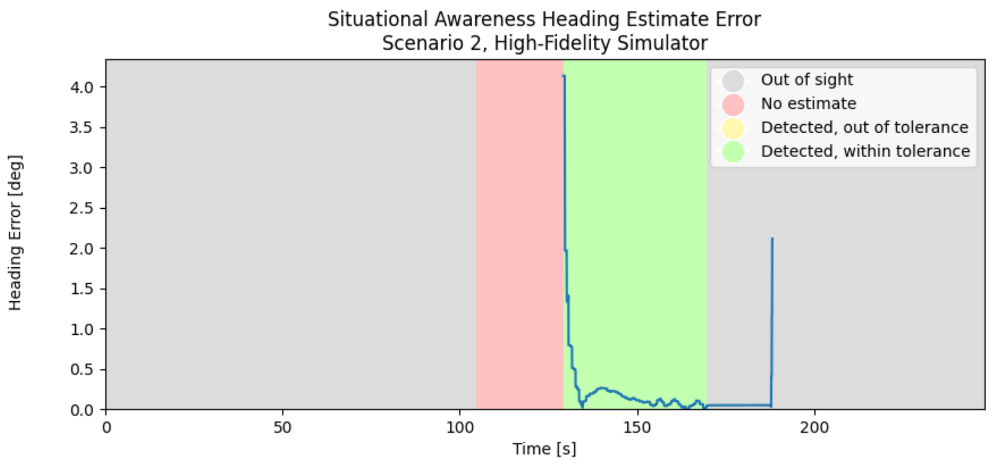

Figure 31: SITAW algorithm heading error in Scenario 2, High-Fidelity Simulator

B Scenario 3

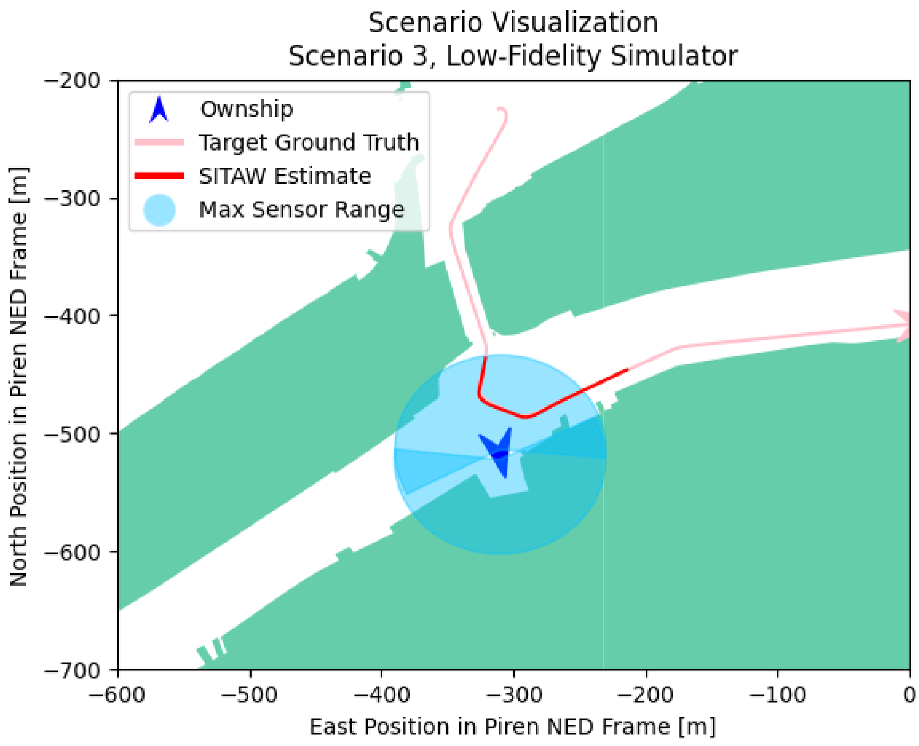

Figure 32: Overview of Scenario 3, Low-Fidelity Simulator

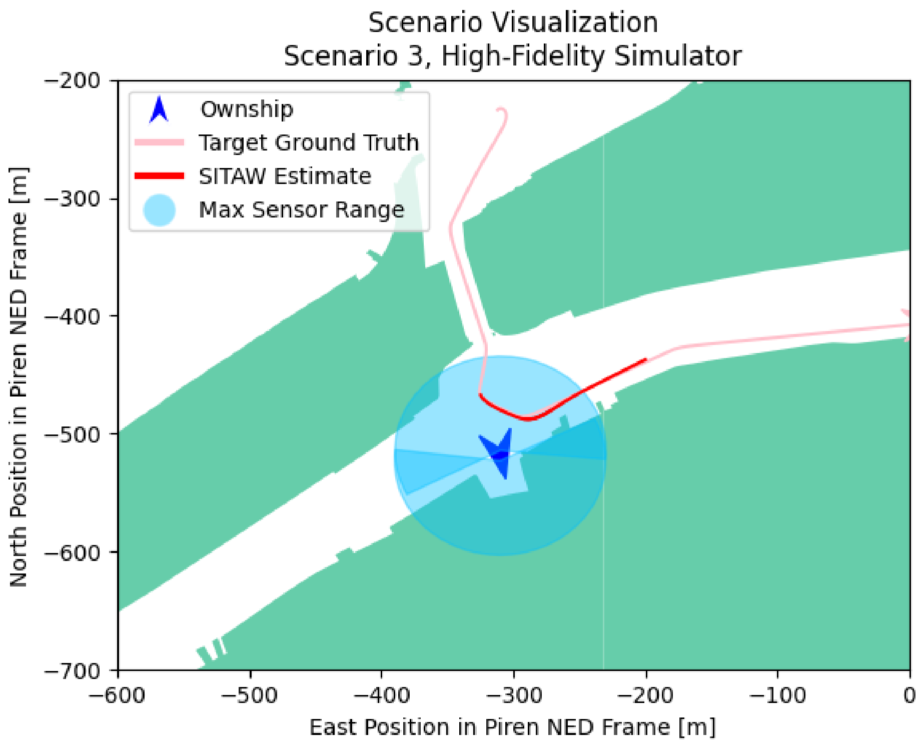

Figure 33: Overview of Scenario 3, High-Fidelity Simulator

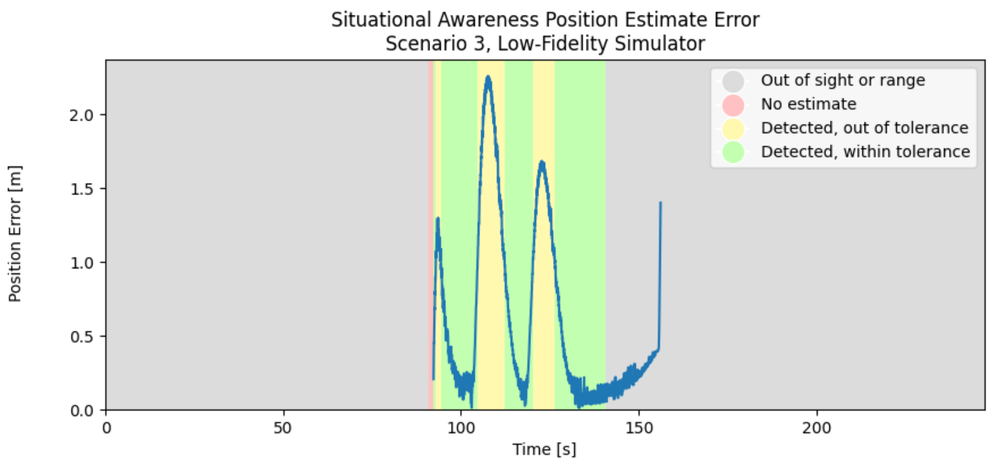

Figure 34: SITAW algorithm position error in Scenario 3, Low-Fidelity Simulator

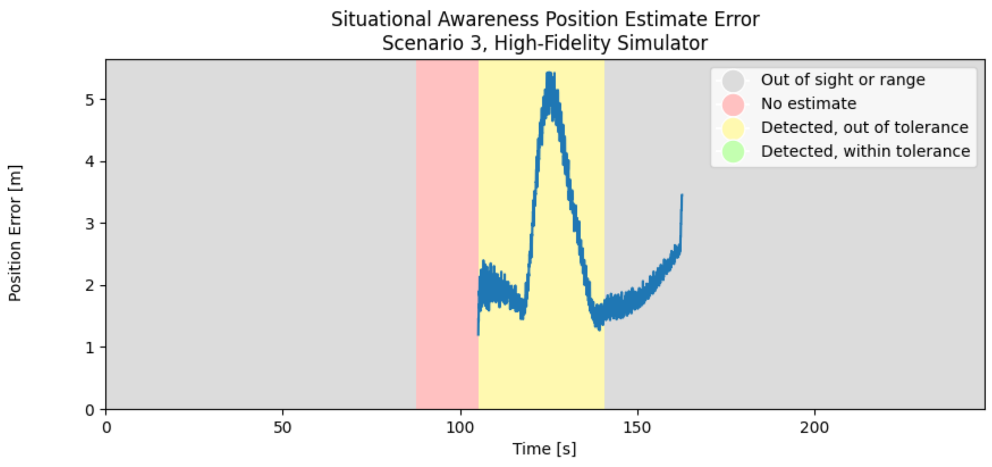

Figure 35: SITAW algorithm position error in Scenario 3, High-Fidelity Simulator

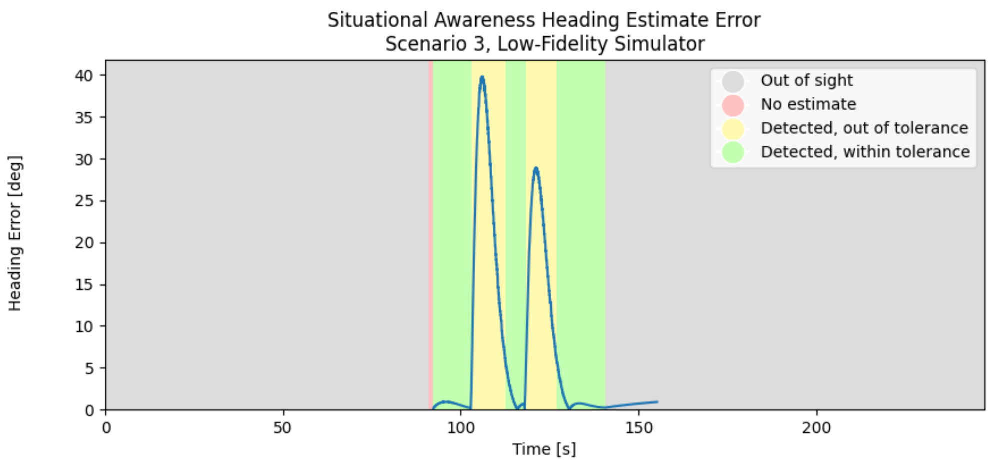

Figure 36: SITAW algorithm heading error in Scenario 3, Low-Fidelity Simulator

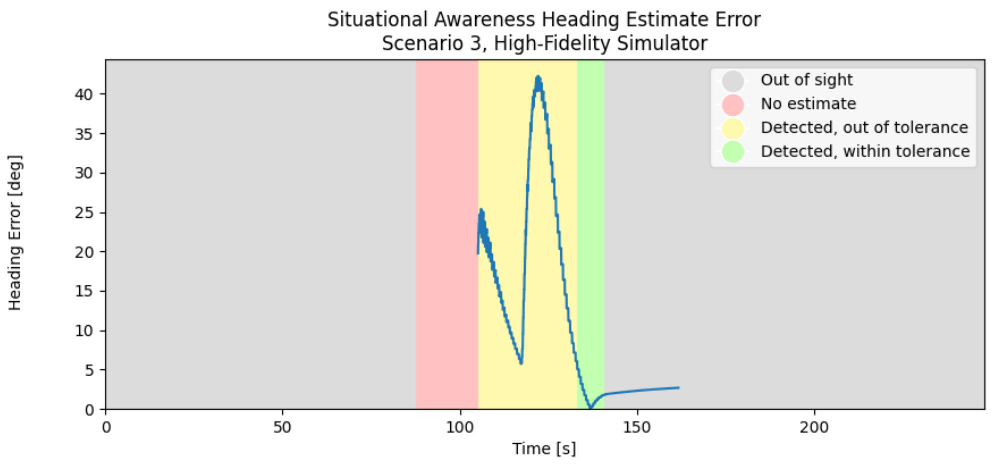

Figure 37: SITAW algorithm heading error in Scenario 3, High-Fidelity Simulator

C Scenario 4

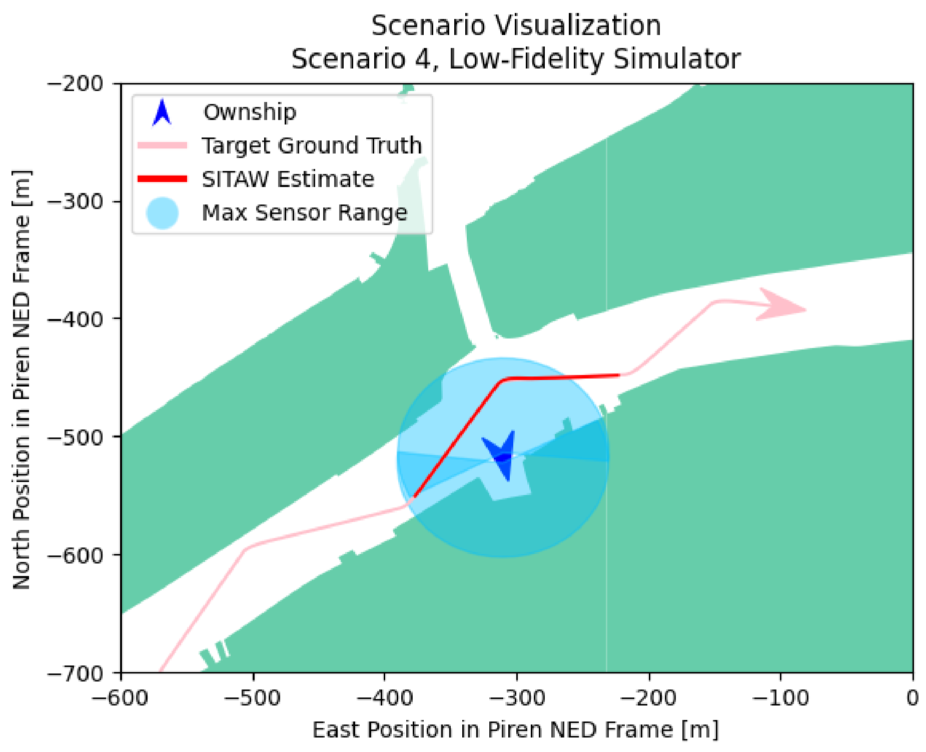

Figure 38: Overview of Scenario 4, Low-Fidelity Simulator

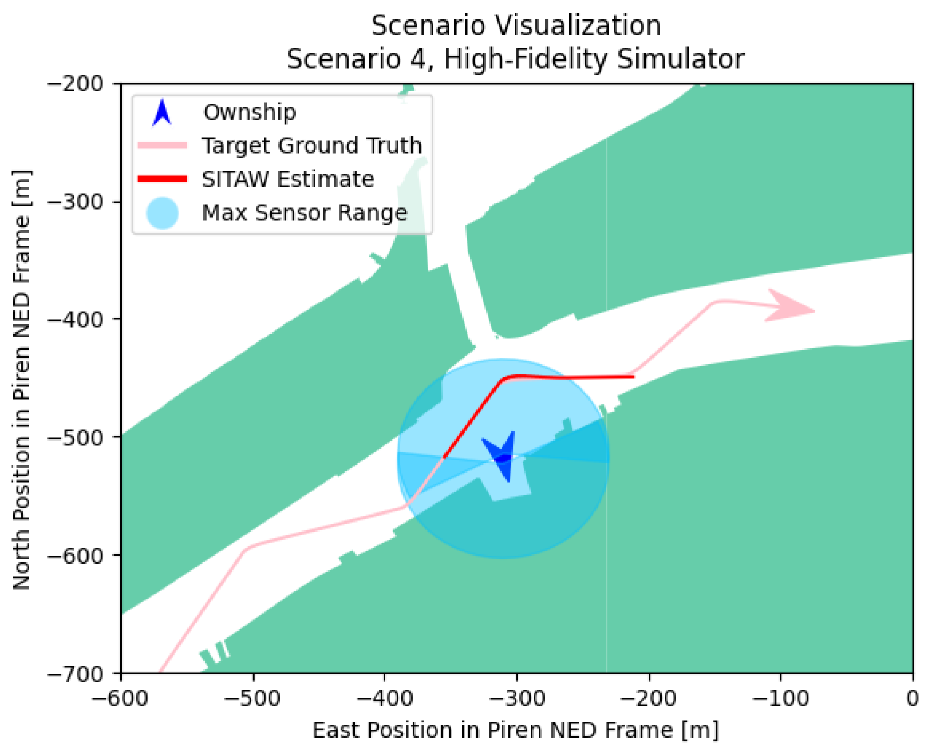

Figure 39: Overview of Scenario 4, High-Fidelity Simulator

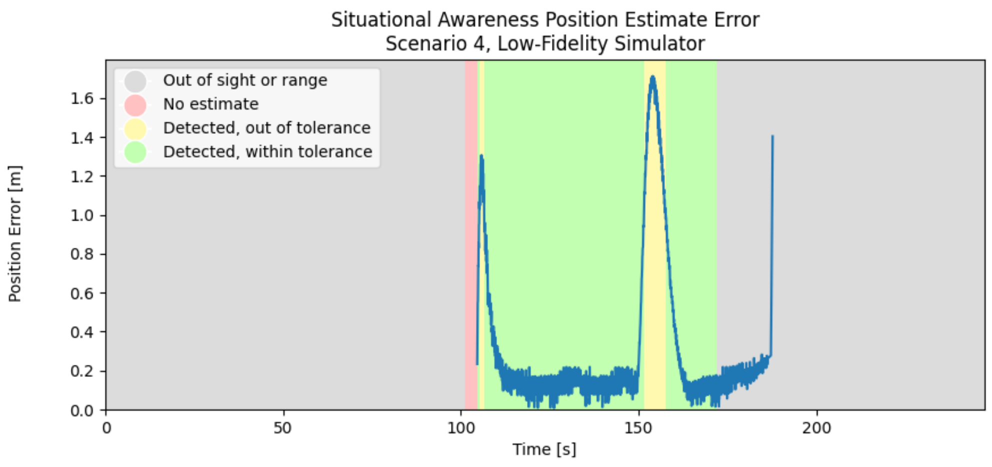

Figure 40: SITAW algorithm position error in Scenario 4, Low-Fidelity Simulator

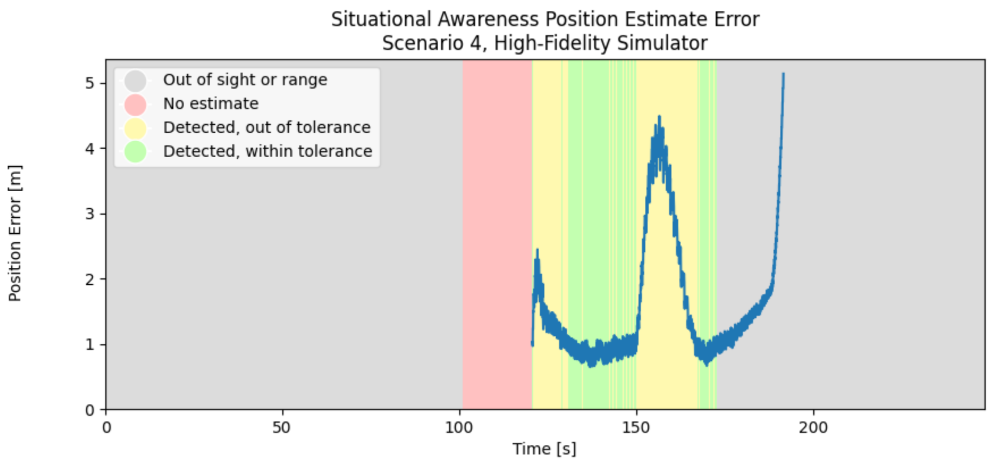

Figure 41: SITAW algorithm position error in Scenario 4, High-Fidelity Simulator

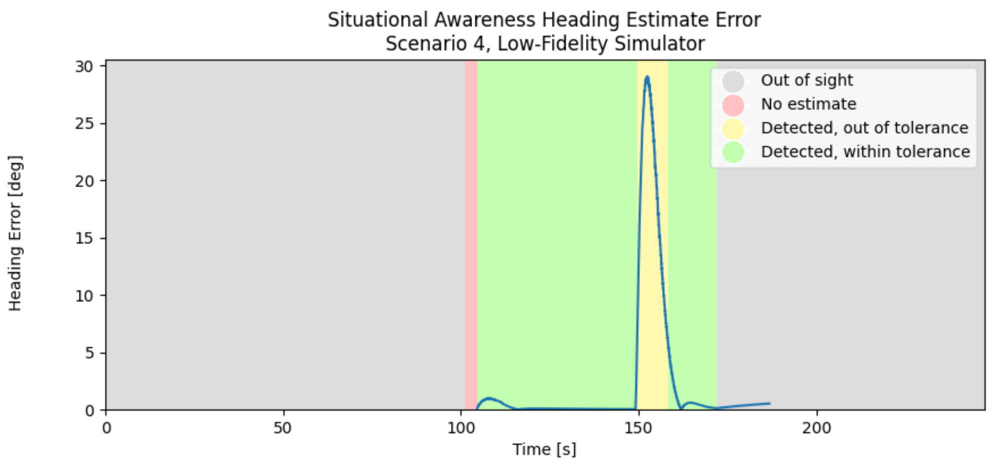

Figure 42: SITAW algorithm heading error in Scenario 4, Low-Fidelity Simulator

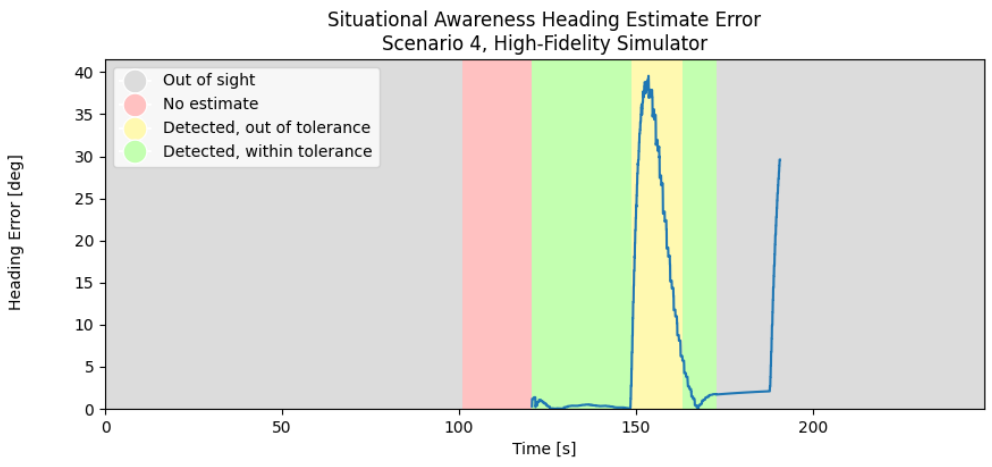

Figure 43: SITAW algorithm heading error in Scenario 4, High-Fidelity Simulator