technische universität
dortmund

Master Thesis

Extended Sonar-based SLAM for GNSS-free Localization of Autonomous Vessels

Sushmitha Govindaraj
May 2025

Supervisor:
Univ.-Prof. Dr. rer. nat. Peter Buchholz
Dr.-Ing. Alexander Puzicha

Technical Unisersity Dortmund
Faculty for Computer Science
Chair for Modelling and Simulation (LS-4)
http://ls4-www.cs.tu-dortmund.de

Acknowledgement

I am grateful for all exceptional individuals who have played a significant role in my thesis journey. First and foremost, I would like to express my sincere gratitude to my supervisor Dr.-Ing. Alexander Puzicha, for his invaluable guidance, constructive feedback, and unwavering support throughout the course of this work. The expertise and encouragement not only enabled me to accomplish the objectives of this thesis but also motivated me to surpass them and pursue excellence. I would also like to express my gratitude to Univ.-Prof. Dr. rer. nat. Peter Buchholz, Chair for Modelling and Simulation (LS-4) at the Faculty for Computer Science, Technical University Dortmund, for providing a stimulating research environment and the resources that enabled me to complete this project. I am deeply grateful to my family and friends for their unwavering support, motivation, and understanding throughout this time. Their encouragement played a vital role in helping me stay focused and positive. I am also thankful to my colleagues and peers for their invaluable assistance, as well as to the larger research community, whose published work provided a solid foundation for my own research. Their comments provided valuable insights that helped shape and improve the foundation for this thesis. Thank you all for being a part of this adventure.

Abstract

Autonomous navigation in Global Navigation Satellite System (GNSS) denied environments poses major difficulties for maritime vessels, particularly in inland waterways where signal obstruction is frequent and common. This thesis proposes a ideal approach of fusing deep learning methods with traditional sensor fusion techniques to enhance navigation accuracy in such environments to improve localization and mapping performance.

Starting from a comprehensive problem analysis of maritime sensor, the research develops a multi-stage solution combining Doppler Velocity Log (DVL) and Inertial Measurement Unit (IMU) for estimating odometry, and equipping sonar technology for environmental perception.The principal challenge addressed by this research is the improvement of odometry estimation through the application of Long Short-Term Memory (LSTM) networks to enhance the raw measurements from Inertial Measurement Units (IMUs) and Doppler Velocity Logs (DVLs). These enhanced measurements are then passed into an Extended Kalman Filter (EKF) framework to create a robust odometry solution.

The system uses SOCA-CFAR algorithms to extract features from sonar data for environmental perception, while motion compensation utilizes the enhanced odometry to correct distortions caused during the scanning process. This corrected data goes into a pose-graph based SLAM implementation that utilizes the improved odometry for more accurate mapping and localization.

An extensive and comprehensive simulation environment was developed, using ROS 2 and Gazebo, to test the proposed method, to allow for the realistic modeling of the dynamics of marine vehicles and the behaviors of the sensors. The simulation allowed for collection of extensive training data for the LSTM models and facilitated rigorous evaluation of the complete navigation system.

Experimental results illustrates that the LSTM enhanced odometry significantly outperforms traditional filtering methods, with enhancements in position accuracy during GNSS outages. The integration with sonar-based SLAM further refines the performance, enabling reliable autonomous navigation in difficult maritime environments.

This work contributes to the field of autonomous vessel navigation by showing the efficiency of combining machine learning approaches with conventional navigation methods, offering a path towards more reliable operation of autonomous vessels in GNSS-denied environments.

Contents

Nomenclature

Vectors and Matrices

$\mathbf{a}_b$	Acceleration vector in body frame
$\mathbf{a}_g$	Acceleration vector in global frame
$\mathbf{b}_i$	Beam direction vector for beam i
$\mathbf{g}$	Gravity vector in global frame
$\mathbf{h}$	Hidden state in LSTM network
$\overrightarrow{h}$	Forward hidden sequence in BiLSTM
$\overleftarrow{h}$	Backward hidden sequence in BiLSTM
$\mathbf{x}$	System state vector
$\mathbf{x}_k$	System state vector at discrete time k
$\hat{\mathbf{x}}_k$	Estimated system state vector at discrete time k
$\mathbf{v}$	Velocity vector
$\mathbf{v}_b$	Vector of beam velocities
$\mathbf{v}_{\text{world}}$	Velocity vector in world frame
$\mathbf{y}$	Innovation vector (difference between measured and predicted values)
$\mathbf{z}$	Measurement vector
B	Matrix of beam direction vectors
$\mathbf{C}$	Covariance matrix
$\mathbf{F}$	State transition matrix
$\mathbf{H}$	Measurement matrix
$\mathbf{I}$	Identity matrix
$\mathbf{K}$	Kalman gain matrix
$\mathbf{P}$	State covariance matrix
$\mathbf{Q}$	Process noise covariance matrix
$\mathbf{R}$	Measurement noise covariance matrix
R_{bg}	Rotation matrix from body to global frame
$\mathbf{S}$	Innovation covariance matrix

Scalars and Variables

c	Constant (general)
d	Number of input dimensions
d'	Number of output dimensions
Δf	Frequency shift (Doppler shift)
f_0	Transmitted frequency
k	Discrete time point
n	Sequence length in time steps
$N_{samples}$	Total number of samples

$N_{sequences}$	Total number of sequences
$p_{est}(t)$	Estimated position at time t
$p_{gt}(t)$	Ground truth position at time t
t	Time
Δt	Time interval
v_i	Measured velocity along beam i
v_r	Relative velocity component along beam direction
v_x	Velocity component in x-direction
v_x^{GT}	Ground truth velocity in x-direction
v_y	Velocity component in y-direction
v_y^{GT}	Ground truth velocity in y-direction
v_z	Velocity component in z-direction
v_z^{GT}	Ground truth velocity in z-direction
x	x-coordinate
X	Input features for machine learning model
X_i	Input sequence for time step i
y_i	Target output for time step i
Y	Output targets for machine learning model

Greek Symbols

$\omega_x, \omega_y, \omega_z$	Angular velocities around x, y, and z axes
ω_x^{GT}	Ground truth angular velocity around x-axis
ω_y^{GT}	Ground truth angular velocity around y-axis
ω_z^{GT}	Ground truth angular velocity around z-axis
ϕ	Roll angle
ψ	Yaw angle
θ	Orientation (general)
θ	Pitch angle
θ_k	Orientation at discrete time k
$\dot{\theta}$	Angular velocity (general)
$\dot{\theta}_k$	Angular velocity at discrete time k

Navigation and Sensors

CFAR	**C**onstant **F**alse **A**larm **R**ate
DVL	**D**oppler **V**elocity **L**og
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
IMU	**I**nertial **M**easurement **U**nit
LIDAR	**LI**ght **D**etection **A**nd **R**anging
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SOCA-CFAR	**S**mallest-**O**f **C**ell-**A**veraging **C**onstant **F**alse **A**larm **R**ate
SONAR	**SO**und **N**avigation **A**nd **R**anging

Machine Learning and Neural Networks

Adam	**Ada**ptive **m**oment estimation optimizer
BiLSTM	**Bi**directional **L**ong **S**hort-**T**erm **M**emory
CNN	**C**onvolutional **N**eural **N**etwork

EKF	**E**xtended **K**alman **F**ilter
GRU	**G**ated **R**ecurrent **U**nit
LSTM	**L**ong **S**hort-**T**erm **M**emory
MSE	**M**ean **S**quared **E**rror
ReLU	**Re**ctified **L**inear **U**nit
RMSE	**R**oot **M**ean **S**quared **E**rror
RNN	**R**ecurrent **N**eural **N**etwork
R^2	Coefficient of determination
F1 score	Harmonic mean of precision and recall

SLAM and Navigation Metrics

APE	**A**bsolute **P**osition **E**rror
ATE	**A**bsolute **T**rajectory **E**rror
ISAM2	**I**ncremental **S**moothing **A**nd **M**apping 2
ODE	**O**pen **D**ynamics **E**ngine

Software and Tools

CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
Gazebo	Open source robotics simulator
PyTorch	Open source machine learning framework
REP-105	ROS Enhancement Proposal 105 – Coordinate Frames for Mobile Platforms
ROS	**R**obot **O**perating **S**ystem
ROS2	**R**obot **O**perating **S**ystem 2
$SE(3)$	**S**pecial **E**uclidean group in 3D space (representing rigid body transformations)
TF2	**T**rans**f**orm library 2

File Formats

XML	E**X**tensible **M**arkup **L**anguage

1 Introduction

Autonomous navigation technology has transformed numerous transportation sectors, but maritime environments especially inland waterways pose distinct challenges that require specialized solutions. Inland vessels navigate in diverse environments, including narrow canals, underneath bridges, along winding rivers, and through areas of varying water depth and changeable weather. In these situations, conventional navigation methods that rely heavily on signals from the Global Navigation Satellite System (GNSS) are often inadequate because the signals are obstructed, reflected, or otherwise degraded (Fukuda et al. 2021).A reliable alternative system that can provide positioning, navigation, and timing information in GNSS-denied environments is essential for the implementation and operation of autonomous surface vessels in inland waterways.

The autonomous navigation of inland vessels is a difficult problem that goes beyond simple positioning to encompass environmental perception, accurate motion estimation, and compute a reliable map of the surroundings. Previous approaches to this problem have made great advances, using various kinds of sensors and new algorithmic ideas. However, these attempts had their own environmental limitations, such as sensor noise characteristics, motion induced distortions in environmental perception data, and the difficulties of fusing information from multiple sensors (Paull et al. 2014). This thesis is built upon previous research and presents enhancements that leverages deep learning techniques to improve sensor data quality before it is passed into traditional navigation pipelines. Through thorough simulation and proper evaluation, the work explores the potential and setbacks of this method, analyzing how these systems might be optimised to bridge the differences between theoretical development and practical deployment in operational inland vessel.

1.1 Motivation

Autonomous vessels are a growing sector within the maritime transportation industry with the potential to transform the movement of goods, reduce operational costs, and enhance safety by ensuring fewer human error in the vessel operation (Gu and Wallace 2021). To ensure efficient navigation of the vessel, particularly in GNSS-denied environments, accurate odometry that estimates the vessel's position based on its past

movements becomes highly important. Unlike land based autonomous vehicles which can utilize wheel encoders for reliable odometry, vessels face the challenge of operating in dynamic fluid environments where currents, waves, and other external forces constantly influence movement.

While Pazarci (2024) presented promising results with his pose-graph based SLAM approach using mechanical scanning sonar and IMU-based motion estimation, there are still opportunities to enhace the odometry for reliable performance.Building upon this foundation, this thesis aims to improve vessel odometry by integrating Doppler Velocity Log (DVL) measurements with IMU data. By incorporating machine learning techniques to identify and compensate for complex error patterns in these sensors, the research focuses in producing more accurate motion estimates compared to the estimates of traditional filtering methods. This enhanced odometry serves as the base for more reliable navigation, as it directly improves the ability of SLAM algorithms to create accurate maps and determine vessel position.

1.2 State of the Art

Research in the area of autonomous navigation in environments where Global Navigation Satellite Systems (GNSS) are unreliable, like in inland waterways or underwater, is becoming quite important. To obtain accurate localization under such conditions, recent navigation systems depend on dead reckoning using data from Inertial Measurement Units (IMUs) and Doppler Velocity Logs (DVLs). Fukuda et al. (2021) highlights that IMU and DVL sensors are important alternatives during GNSS free conditions, enabling vessels to sustain basic navigation. Although, they provide valuable insights, these sensors suffer from drift and noise, causing cumulative errors in position estimates over time if not externally corrected.

The sensor fusion algorithms utilizes traditional methods such as the Extended Kalman Filter (EKF) to mitigate these errors. As discussed by Paull et al. (2014), the EKF remains a promising solution in marine robotics, by combining IMU and DVL data with dynamic models to estimate a vessel's state. However, the performance of EKF is highly dependent on the quality and the properties of the input data. In real world marine environments, sensor readings are often distorted by complex non-linearities due to various factors like currents, wave motion, and mechanical vibrations. These factors violate the assumptions of gaussian noise and linear motion models that the EKF relies upon, resulting in degraded accuracy and reliability.

Recent investigations have begun to blend learning-based techniques into the model-driven approaches to address their limitations. Inertial Navigation Meets Deep Learning, by Cohen and Klein (2023), applied Long Short-Term Memory (LSTM) networks to learn temporal patterns in sequences of sensor measurements. They regressed position displacements directly using the LSTM and integrated these outputs into the traditional filtering pipeline to enhance localization. Thus, their hybrid method could

harness the predictive power of deep learning and the structure of traditional filtering. Other efforts, such as Ye et al. (2023), have employed learning mechanisms to predict filter residuals or dynamically adjust noise parameters, thereby providing enhanced adaptability to environmental changes and sensor behavior.

Although these advancements are promising, most previous research has concentrated on underwater vehicles, with their associated navigation difficulties. For inland surface vessels, especially those relying on sonar for simultaneous localization and mapping (SLAM) navigation continues to pose truly unique challenges. Pazarci (2024) addressed these problems by developing a SLAM approach for inland waterway navigation, however this approach had limited performance due to the lack of reliable odometry based localization. The experimental use of laser for velocity estimation was effective in controlled conditions but posed challenges when applied to diverse range of operational conditions.

Using this foundation, the proposed strategy adopts an alternative focus that emphasizes a refinement of IMU and DVL sensor information based on learned representations.In this method, an LSTM network, trained on temporally structured IMU and DVL data, serves to extract more coherent and informative representations of the motion dynamics of the vessel ensuring more robust localization. These refined signals are then fed to a conventional EKF, which benefits from both enhanced input quality and the robustness of model-based filtering. This balanced approach leverages the adaptability of deep learning and the interpretability of classical estimation to improve localization reliability in sonar-based navigation systems for autonomous inland vessels operating in GNSS-denied environments.

1.3 Objectives

The primary objectives of this research are:

- To develop a comprehensive simulation environment that realistically models inland vessel dynamics and sensor behaviors.
- To create a machine learning approach using LSTM networks to enhance the quality of IMU and DVL measurements.
- To integrate the enhanced odometry with sonar-based feature detection and SLAM techniques.
- To quantitatively evaluate the performance achieved through the proposed approach.

1.4 Thesis Structure

The structure of this thesis is organized as follows:

- Chapter 2 examines the primary sensors used in the navigation system: sonar for environmental perception, and DVL and IMU for odometry.
- Chapter 3 presents the overall system architecture and approach.
- Chapter 4 details the development of the simulation environment using ROS 2 and Gazebo.
- Chapter 5 describes the data collection and preprocessing methodology.
- Chapter 6 presents the LSTM model architecture and training process.
- Chapter 7 discusses the sensor enhancement and robot localization.
- Chapter 8 discusses the integration with the SLAM pipeline and presents mapping results.
- Chapter 9 evaluates the performance of the enhanced odometry system.
- Chapter 10 concludes the thesis with a summary of findings and suggestions for future work.

2

Sensors for Navigation

Autonomous vehicles face a significant challenge when it comes to navigating in environments where the Global Navigation Satellite System (GNSS) is unavailable. To achieve reliable positioning and mapping (Paull et al. 2014), these vehicles must rely on alternative sensing modalities and sophisticated fusion techniques. This chapter looks at three main types of sensors to be used in a GNSS-free navigation system: for understanding the environment, sonar; and for establishing and maintaining a reliable odometry, the Doppler Velocity Log (DVL) and Inertial Measurement Unit (IMU). Grasping the fundamental principles of operation, their capacities, and constraints of these sensors is fundamental to the development of efficacious navigation methods in situations where satellite positioning is unattainable.

2.1 Environmental Perception

For obstacle detection and mapping the vehicle's environment, sonar technology that allows to analyze acoustic reflections from a variety of sources in the water environment is utilized.

2.1.1 Sonar Systems for Environmental Perception

Sonar (Sound Navigation and Ranging) is the main source perception sensor in marine environments where optical sensors cannot function as they are limited by light absorption and turbidity (Christ and Sr 2007). Unlike electromagnetic waves that rapidly weaken and do not penetrate the water, acoustic signals travel far and effectively. In marine applications, the acoustic signals of sonar sensors travel substantial distances, making it preferable in long range sensing modalities.

Basic Principles and Operation

Sonar devices work by sending out sound waves and interpreting the echoed sounds that bounce back, with the time delay of the return indicating how far away the reflecting object is, and the strength of the returning sound wave indicating how much

the object reflects sound. The basic equation that relates the time of an emitted sound wave to the distance it has traveled is:

$$d = \frac{c \cdot t}{2} \tag{2.1.1}$$

where d represents the distance, c is the speed of sound in water (approximately 1500 m/s), and t is the time taken for the signal to travel to the target and return. The factor of 2 accounts for the round trip travel of the signal, as the signal travels to the target and then reflects back to the receiver.

The work focuses particularly on mechanically scanning imaging sonars. These operate by rotating a transducer to insonify different sectors of the environment sequentially. The echo intensity received at each bearing and range bin creates a polar image of the surroundings.

These sonars have a unique set of characteristics that clearly distinguish them from environmental sensors. They obtain data sequentially, and a full 360° scan takes several seconds to complete. The sonar beam typically has a very narrow horizontal aperture (1-3°) and a much wider vertical aperture (10-40°), creating a fan-shaped sensing pattern. Each ping returns intensity information across a range of distances, producing a radial slice of the environment. The resolution of these sonars varies with respect to range, providing better detail at shorter distances. These characteristics require varied processing approaches, particularly for extracting the features and compensating the motion, which will discuss in subsequent chapters.

2.2 Odometry

The solution to the odometry problem takes advantage of the complementary skills of the Doppler Velocity Log and Inertial Measurement Unit. The DVL provides direct velocity measurements relative to the seafloor, while the IMU supplies angular velocity data for orientation tracking.

2.2.1 Doppler Velocity Log (DVL)

The Doppler Velocity Log (DVL) gives direct measurements of a velocity of the vehicle relative to the seafloor or water column, making it valuable for marine navigation where position drift must be reduced.

Operational Principles

DVLs use the Doppler effect to measure velocity. The Doppler effect states that the frequency shift of an acoustic wave reflected from a moving target is proportional to the relative velocity between the wave source and the target (Yuh, Marani, and Blidberg 2011). This relationship is given by:

$$\Delta f = -\frac{2f_0}{c} v_r \tag{2.2.1}$$

where **f** is the frequency shift, **f0** is the transmitted frequency, $\mathbf{v}_r$ is the relative velocity component along the beam direction, and c is the speed of sound in water. The factor of 2 accounts for the signal's round-trip travel, as the velocity component affects both the outgoing and incoming signals.

A conventional DVL utilizes a Janus configuration with four acoustic beams. These beams form two perpendicular pairs and are oriented downward approximately 30 degrees from vertical. Each beam measures the velocity component in its direction, by combining these components, the full 3D velocity vector can be calculated as shown in Figure 2.1

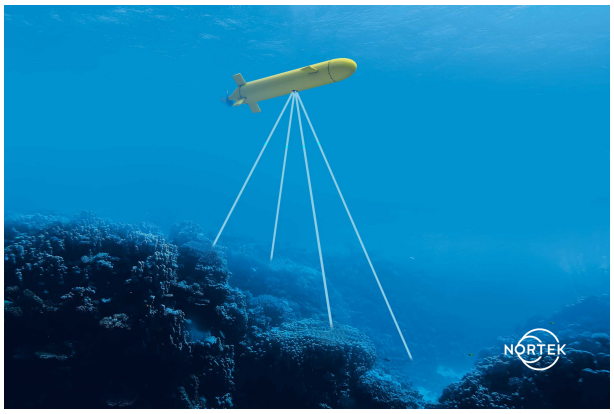


Figure 2.1: DVL Beam Geometry
(*New to subsea navigation? — Nortek* 2025)

A DVL with several beams measures Doppler shift from each beam providing a projection of the velocity vector from each beam. Given a beam direction vector $\mathbf{b}_i$ and a vehicle velocity vector **v**, the measured velocity along beam i is:

$$v_i = \mathbf{b}_i \cdot \mathbf{v} \tag{2.2.2}$$

With at least three non-coplanar beams, the full 3D velocity vector can be reconstructed by solving the system of equations:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \mathbf{b}_3^T \\ \mathbf{b}_4^T \end{bmatrix} \mathbf{v} \tag{2.2.3}$$

This can be solved using least squares methods when more than three beams are available:

$$\mathbf{v} = (B^T B)^{-1} B^T \mathbf{v}_b \tag{2.2.4}$$

where B is the matrix of beam direction vectors and $\mathbf{v}_b$ is the vector of beam velocities.

The Doppler Velocity Log (DVL) has a significant advantage for inland vessel navigation because it directly measures velocity, which provides drift-free speed without error accumulation seen in acceleration based systems. DVLs are highly suitable for long-duration operations, because they have bounded errors that do increase over time. Modern DVLs achieve new heights in performance and accuracy, with velocity measurements within 0.2-0.3 % of the distance traveled, outperforming dead reckoning from accelerometers. With these advantages, DVL data complements IMU measurements by addressing their disadvantage of velocity and position drift.

2.2.2 Inertial Measurement Units (IMUs)

Inertial Measurement Unit gives high frequency measurements of a vehicle's motion where external references are unavailable making them essential for navigation in GNSS-denied environments.

Basic Principles

An IMU has three gyroscopes and three accelerometers mounted orthogonally, giving the system six degrees of freedom. The gyroscopes measure angular velocity around each axis. The accelerometers measure specific force, which includes the acceleration due to gravity and the linear acceleration of the vehicle (Woodman 2007).
Inertial navigation is based on the fundamental principle of integrating acceleration measurements mathematically to obtain velocity and position. Likewise, angular velocity measurements are integrated to get orientation. For a rigid body in three-dimensional space, the linear acceleration in the global reference frame can be expressed in terms of the body-frame acceleration measured by the accelerometer through:

$$\mathbf{a}_g = R_{bg}\mathbf{a}_b + \mathbf{g} \tag{2.2.5}$$

where, $\mathbf{a}_g$ is the acceleration vector in the global frame,$\mathbf{a}_b$ is the acceleration vector measured in the body frame,R_{bg} is the rotation matrix that transforms from body to global frame, $\mathbf{g}$ is the gravity vector in the global frame
The rotation matrix R_{bg} can be obtained from the angular velocity measurements by integrating the following differential equation:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \tag{2.2.6}$$

where ϕ, θ, and ψ are the roll, pitch, and yaw angles, and ω_x, ω_y, and ω_z are the angular velocities measured by the gyroscopes.

In our navigation system, the IMU's angular velocity measurements are favored over accelerometer data for a number of reasons. One is that the DVL provides a direct, drift-free velocity measurement that does not require the integration of an accelerometer

signal. Another reason is that angular velocity determined from a gyroscope generally has a better signal-to-noise ratio than does an accelerometer signal. Orientation tracking, essential for transforming DVL body frame velocities to the global reference frame, relies on accurate angular velocity. The IMU provides the Yaw (heading), crucial for navigation, which cannot be reliably determined from other sensors in marine environments. This approach combines the strengths of the IMU and DVL through complementary sensor fusion, addressing the limitations of both sensors.

2.2.3 Complementary Nature of IMU and DVL

The complementary data from IMU and DVL, if correclty fused results in higher navigation performance compared to either sensor alone can perform. This advantage comes in from their different characteristics as shown in Table 2.1:

Table 2.1: Comparison of IMU and DVL characteristics

Characteristic	**IMU**	**DVL**
Update Rate	High (100-1000 Hz)	Moderate (1-10 Hz)
Measurement Type	Acceleration & Angular Velocity	Direct Velocity
Error Growth	Unbounded (drift)	Bounded
Short-Term Accuracy	Excellent	Good
Long-Term Stability	Poor	Excellent

The IMU gives high frequency updates with excellent short term accuracy but has long term drift. The DVL gives direct velocity measurements with bounded errors but at a lower rate . With the combination of IMU and DVL,we can obtain the best of both worlds by having both the high update rate of the IMU and the long-term stability of the DVL.

Advantages Over Previous Approaches

Pazarci (2024) develped a SLAM system for autonomous inland vessels using an IMU and a laser scanner for velocity estimation. Although the system was effective in environments where laser scanners performed well, this approach faces limitations in varied dynamic environments. The proposed DVL based approach offers several advantages like providing direct velocity measurements and ensuring reliable velocity estimation compared to laser scanners, which infer velocity information from sequential scans. The laser scanners are ineffective due to light attenuation, whereas DVLs are especially designed for underwater use. Additionally, DVLs do not rely on environmental features and can operate in various environments, like flat, muddy seafloor. Furthermore, DVLs offer full 3D velocity measurement, overcoming the corridor effect observed in laser scanners (Caballero et al. 2009). These advantages makes the sensor fusion approach ideal for navigation , providing reliable odometry that serves as the base for the sonar-based SLAM system.

This chapter has covered the three main sensors found in the GNSS free navigation system. The mechanical scanning sonar furnishes environmental perception capability, yielding data that is used for feature extraction and mapping. The DVL serves as the system's direct, accurate velocity measurement device, with errors that are bounded, while the IMU provides high-frequency angular velocity data for tracking the system's orientation. By fusing the complementary DVL and IMU data, we create a robust odometry system that serves as the foundation for our navigation approach.

In the following chapters, we will explore how these sensor inputs are processed, enhanced, and combined to create a complete navigation solution for GNSS-denied environments.

3

System Architecture and Overall Approach

This chapter provides a comprehensive overview of the GNSS-free navigation system developed in this thesis. The essential aim is to develop a strong localization and mapping answer for inland vessels that can function in places where satellite navigation is impossible. The solution must rely on enhanced odometry derived from inertial and velocity measurements, complemented by sonar as the perception sensor.

This work's main breakthrough is using machine learning to enhance the quality of the sensor data before it combines with data from other sensors and fed into traditional SLAM like algorithms. Figure 3.1 shows the overall system architecture, the high-level view of complete system starting from flow of data from raw sensors into various processing steps to the final results to estimate vehicle position and create a map of the environment.

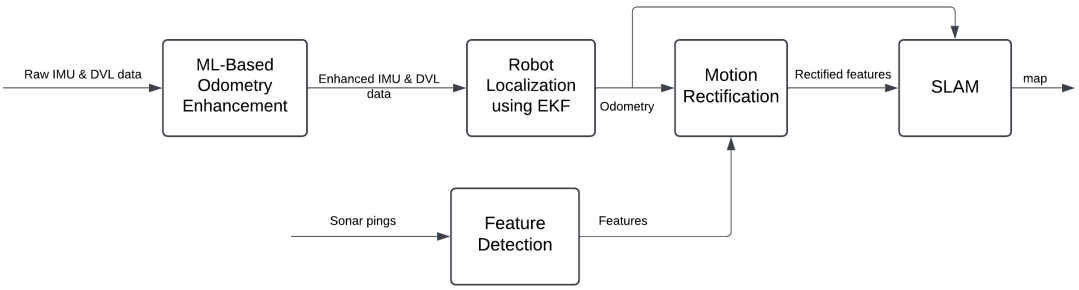


Figure 3.1: System Architecture

The pipeline consists of several interconnected components: ML-Based Odometry Enhancement, Robot Localization using EKF, Feature Detection, Motion Rectification, and SLAM. While the complete navigation system integrates multiple components, the main contribution of this thesis is the ML-Based Odometry Enhancement and its integration with the sensor fusion and SLAM framework to create a robust navigation solution.

This thesis develops a machine learning strategy to improve odometry data from raw IMU and DVL readings. Both types of sensors are required for use in inland water

conditions, where external references for navigation are limited (Snyder 2010). However, traditional state estimation approaches including the use of Kalman filters work poorly with these sensors because they are not designed to account for the significant, environmentally induced errors that DVLs and IMUs experience.

LSTM networks were chosen for odometry enhancement,as they effectively model temporal sequences and capture long-term dependencies in the sensor data (Hochreiter and Schmidhuber 1997). The architecture takes normalized IMU and DVL inputs and processes them through LSTM layers to yield accurate predictions of the angular and linear velocities rather than direct position estimates. This aligns with the physical properties of the sensors and avoids the difficult and complex task of directly learning to estimate position.

Different models tackle the unique error patterns of each type of sensor. The IMU model hones in on angular velocity corrections and the DVL model deals with terrain and current effects. These were trained on a range of simulated trajectories paired with diverse kinds of ground truth data, making the system robust across a variety of operational scenarios. The trained LSTMs are integrated into the ROS 2 framework, where they process raw data from the sensors and turn it into better measurements, greatly improving the odometry.

The sensor fusion based on the Enhanced Kalman Filter (EKF) utilizes the data from the machine learning-enriched IMU and DVL to produce a more accurate and reliable stream of odometry.The EKF implementation by Moore and Stouch (2013) was reimagined to better accommodate the characteristics of the enhanced sensor data. Tuned to reflect the reliability of the ML-processed sensor data, the EKF produces a consistent and high quality estimate of the motion that serves as the foundation for the processes of mapping and localization that follow.

The Feature Detection component takes the raw sonar ping data and processes it to extract useful environmental features. Distortions in the sonar data caused by vehicle motion during scanning are corrected for by motion rectification using the enhanced odometry and extracted sonar features.
These components have been adapted and optimized for the simulation environment used in this thesis.

The SLAM module concludes the pipeline. It takes the sonar features, which have been corrected for motion, along with the enhanced odometry, and fuses their information to create a consistent map. At the same time, it is refining the vehicle's position estimate. By utilizing the improved odometry from previous stages,the SLAM could create accurate and relaible mapping. This approach builds upon the work of Ribas et al. (2014), who demonstrated the importance of accurate motion estimation for effective SLAM.

An essential element for this research was the establishment of a complete simulation environment within ROS 2 and Gazebo. The simulation environment provided access to ground truth data that was necessary for training the LSTM models and judging the overall system performance. Thorough tests were conducted using the simulation to assess the effectiveness of the odometry enhancement approach.

Testing in simulated conditions avoids the expense and time commitment of trials in real-world conditions.The total system integration is realized with an architecture based on ROS 2, which connects all components through topic interfaces that are well defined. This architecture allows the real time processing of data from the sensors and makes possible both test development for the systems and performance evaluation of the completed systems.

This chapter has presented the architecture and overall approach of the navigation system without global navigation satellite system (GNSS) access, which has been developed in this thesis. The primary innovation lies in the ML-Based Odometry Enhancement component. This component uses long short-term memory (LSTM) networks to enhance the quality of inertial measurement unit (IMU) and Doppler velocity log (DVL) data before it enters the traditional sensor fusion and simultaneous localization and mapping (SLAM) pipeline. This approach to odometry addresses the challenges in marine robotics, enabling accurate navigation in GNSS free enviroments.

4

Simulation Environment Development

4.1 Overview of the Simulation Framework

A complete simulation environment using ROS 2 (Robot Operating System) and Gazebo was constructed for the development and testing of our GNSS-free navigation system. This combination offers a well-suited framework for developing any robotics applications, offering necessary tools for physics-based simulation, sensor modeling, and a standardized communication architecture. The simulation environment was particularly designed to recreate conditions where GNSS signals are unavailable, forcing the navigation system to rely entirely on inertial, velocity, and sonar measurements.

4.1.1 Robot Operating System 2 (ROS 2)

Robot Operating System 2 (ROS 2) is a flexible, modular framework designed to support the development of robotic systems. Building on the experience from the original ROS, ROS 2 handles the needs of modern robotics by providing improved communication reliability, real time capabilities, enhanced security, and cross-platform support (Macenski et al. 2022). With its focus on modularity through a node-based architecture, ROS 2 provides a solid foundation for building, testing, and deploying complex, autonomous robotic applications in both research and industrial settings.

In our implementation, we especially harness the node-based architecture of ROS 2. In it, each component of the system operates as an independent node. As shown in Figure 4.1, the nodes communicate with each other through a publish-subscribe messaging pattern (Quigley et al. 2009). This sophisticated approach enables to develop and test individual components separately before integrating them into the complete system.

The tf2 (transform library) is equally crucial for our simulation, maintaining the spatial relationships among the various coordinate frames associated with the vessel, the sensors, and the environment. This framework automatically propagates transforms through the frame tree, ensuring consistent spatial reasoning throughout the system.

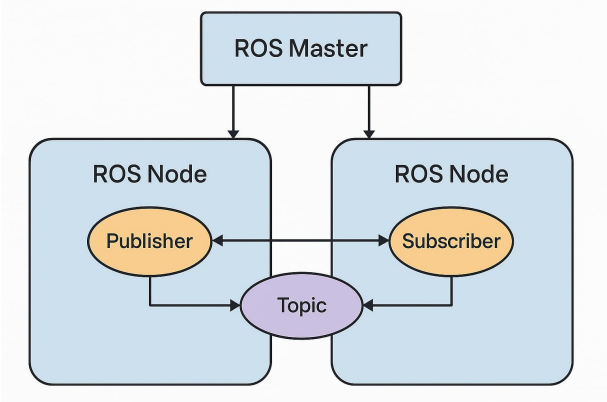


Figure 4.1: Node Architecture

4.1.2 Gazebo Simulation

Gazebo is a powerful open-source simulator that integrates well with ROS 2 to provide realistic physics, sensor modeling, and environment interaction (Koenig and Howard 2004). It supports dynamic simulation of rigid body systems using engines like ODE and offers a plugin-based architecture for extending functionality. Gazebo's capabilities to simulate complex environments and its compatibility with ROS 2 make it an perfect choice for developing and testing robotic systems in realistic virtual environments.

In our work, Gazebo is used to simulate the environment for a vessel that is denied Global Navigation Satellite System (GNSS) signals. The simulation was customized with plugins to model the physics of an environment, including buoyancy, drag forces, and current-induced resistance. Standard sensor models were extended with realistic noise and limitations, and a custom plugin was added to simulate the performance of Doppler Velocity Log (DVL). In order to simulate sonar-based perception, LiDAR sensor with adjusted range and operational parameters was adapted that closely mimicked how a sonar sensor works to replicate acoustic behavior in water. Through gazebo_ros_pkgs,the simulation with ROS 2 was interfaced, enabling synchronized access to all sensor data and control inputs, which is used for developing and validating our SLAM pipeline.

4.2 Simulation World Design

The environment seen in the simulation as shown in Figure 4.2 was created to deliver realistic testing conditions for the navigation system. The environment is built inside a large water tank, with walls on all sides and two floor levels, one at the surface (z=0) and a second floor positioned 5 meters beneath the surface for DVL bottom tracking. In order to achieve realistic physics, we incorporated several specialized plugins. The plugins used include a hydrodynamics plugin which simulates water properties with appropriate fluid density and drag coefficients. The buoyancy plugin creates appropriate buoyant forces to the vessel. Visual water effects are added by

creating transparent surfaces and particle emitters to enhance the visual fidelity of the simulation.

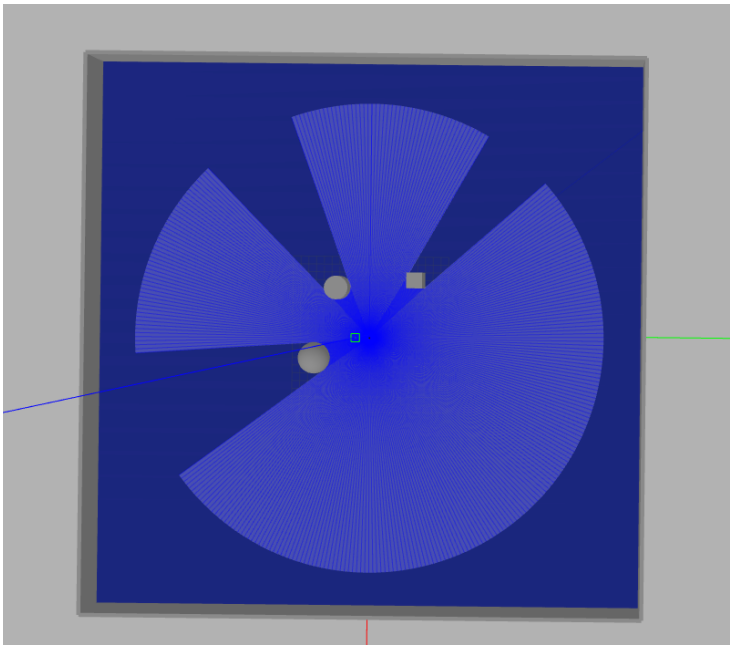


Figure 4.2: Simulated Environment

Several static obstacles were placed in the environment to develop a complex testing scenario for the navigation system. These consist of cylindrical pillars, spherical objects, and rectangular blocks positioned at various locations within the tank. These static obstacles provide features for the sonar-based mapping system to detect and present navigation challenges that require path planning.

4.3 Vessel Model Implementation

This section details the design and implementation of the simulated vessel model used within the Gazebo environment.

4.3.1 Physical Structure

The simulated vessel was designed as a simplified inland vehicle with hydrodynamic properties appropriate for the intended use. The main body is a rectangular hull with overall dimensions of 0.6 * 0.5 * 0.2 meters. To ensure realistic behavior, the center of mass was placed below the geometric center of the vehicle.
The vessel is attached with a vertical base link that acts as the mounting point for sensors as shown Figure 4.3. This design creates a realistic representation of a small autonomous inland vehicle, with appropriate masses, inertia tensors, and points of attachment for sensors. Controlled movement is enabled in the horizontal plane by utilizing the planar movement plugin (Sendobry et al. 2012).

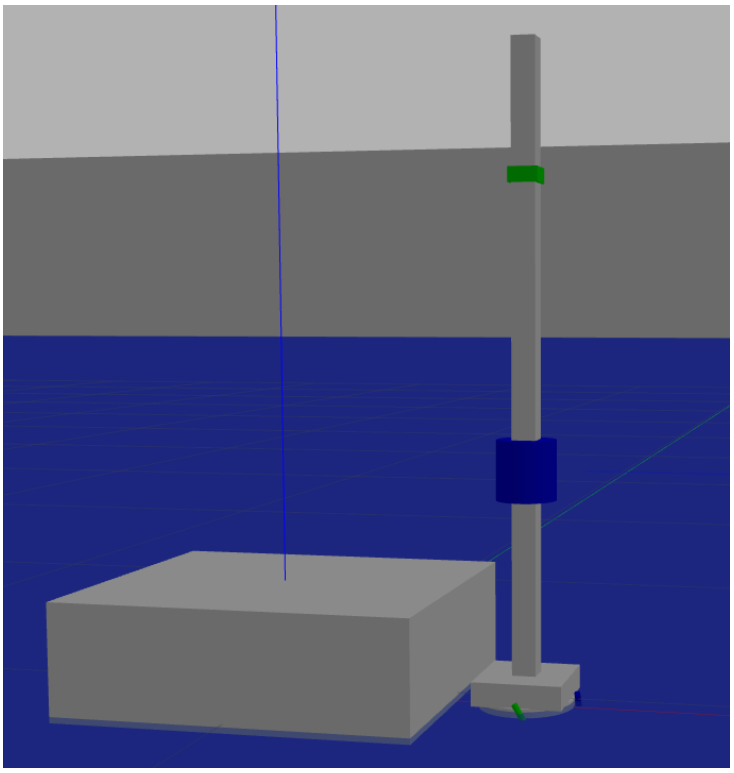


Figure 4.3: Simulated Vessel Setup

4.3.2 Motion Control

To achieve controlled movement of the vessel, the gazebo_ros_planar_move plugin was enabled, which accepts velocity commands from ROS 2 and applies appropriate forces and movement to the simulated vessel. This plugin offers a basic model of propulsion, which takes linear and angular velocity commands on the cmd_vel topic.

Although this does not completely account for the complicated hydrodynamics of marine vehicles, it offers enough accuracy for the examination of navigation algorithms. This is because our emphasis is on aspects related to the processing and fusion of sensor data.

4.4 Simulated Sensor Implementation

Accurate simulation of sensor behavior is important for developing and evaluating a navigation system designed for GNSS-denied environments. The proposed navigation system relies on data from an Inertial Measurement Unit (IMU), a Doppler Velocity Log (DVL), and a sonar. Therefore, realistic models for these sensors, including appropriate noise characteristics and limitations, were implemented within the Gazebo simulation framework. These simulations were designed to recreate the data outputs expected from physical hardware, enabling the development and testing of downstream navigation algorithms.

4.4.1 Inertial Measurement Unit (IMU)

The IMU was configured using built-in IMU sensor plugin present in Gazebo. This sensor provides linear acceleration and angular velocity measurements in three dimensions, mimicking the real world IMU sensor behaviour.

The IMU sensor was set with a high update rate of 200 Hz to capture rapid dynamics, with angular velocity measurements being of particular concern for our navigation system. Although the IMU can also provide linear acceleration data, this system primarily uses the angular velocity measurements for orientation tracking, as the translational velocity information can be achieved from DVL.

The IMU simulation creates a sophisticated angular velocity noise model based on the XSENS MTI-100 (*Xsens MTi-100 IMU — movella.com* 2025) to simulate difficult conditions for robustness testing. The model utilizes various error sources to bring in realistic degradation of gyroscope measurements. The baseline white noise has a standard deviation of 0.0004 rad/s (0.023°/s). Bias instability is implemented through a random walk process with a coefficient of 0.00005 rad/s/sqrt(s), producing slowly evolving offset errors with a maximum drift of ±0.005 rad/s (±0.286°/s), compared to the actual sensor's ±0.0045 rad/s (±0.26°/s) specification.

The model also creates a temperature sensitivity of 0.00025 rad/s/°C with simulated temperature fluctuations having a standard deviation of 0.5°C, creating bias shifts with relative to time. Scale factor errors of ±0.5% introduce multiplicative inaccuracies that increase with angular rate, while cross-axis sensitivity of 0.25% simulates mechanical coupling between sensing axes. Signal processing artifacts are implemented through quantization effects with 0.00035 rad/s resolution, making step patterns in the output data. Occasional measurement outliers will occur with magnitudes of ±0.05 to ±0.15 rad/s (±3° to ±8.6°/s).

The reported values in the output messages has a standard deviation of 0.002 rad/s (0.11°/s) in each axis, providing navigation algorithms with a realistic estimate of measurement uncertainty. This moderately enhanced noise profile brings in a realistic simulation of gyroscope performance in challenging environments with moderate vibration or electromagnetic interference.

4.4.2 Doppler Velocity Log (DVL)

The DVL presented unique challenges for simulation since Gazebo does not include a built-in DVL sensor. A custom approach was implemented which combines a visually representative model with functional behavior (Manhães et al. 2016).

The DVL simulation incorporates four acoustic beams that are directed at 30-degree angles from vertical. These are arranged in a Janus configuration. The plugin interacts

with the physics engine to gain access to the vessel's real velocity and transforms that into the DVL reference frame, and then adds realistic noise patterns to simulate the behavior of a physical sensor.

For visual representation,beam indicators using thin cylinders with different colors for each beam was created as seen in Figure 4.4, allowing intuitive visualization of the DVL's field of view and operation.

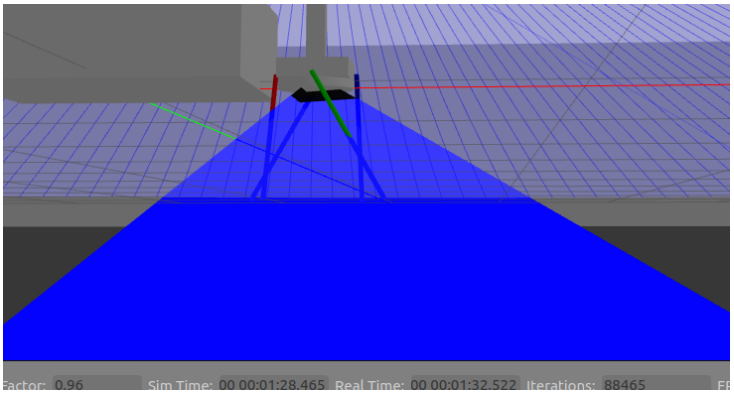


Figure 4.4: DVL Beam Arrangement

To closely simulate real world sensor behavior, we implemented a comprehensive multi-modal noise model that provides various error sources. The DVL noise model mimics the complex error patterns observed in commercial underwater velocity sensors. The implementation had a multi-component noise structure with fixed bias components (0.01 m/s horizontal, 0.008 m/s vertical) and random noise with 0.03 m/s standard deviation, matching the real world sensor specified precision. The model incorporated bias instability through a slow random walk (0.0001 m/s rate), creating realistic long-term drift characteristics. Non-Gaussian error components include impulse noise (±0.08 m/s magnitude), complete outliers (up to ±0.5 m/s), and signal loss events. Environmental factors degrade measurements through simulated water stratification (0.008 m/s std. dev.) and turbidity effects (0.015 m/s std. dev.). Range-dependent errors (0.0008 coefficient) create the characteristic degradation in accuracy with increasing distance from the bottom. Temporal error correlation (0.15 coefficient) provides realistic persistence in measurement errors across consecutive readings, which prevents the unrealistic white noise assumption present in many simplified simulations.

4.4.3 Sonar Implementation

The approach to sonar simulation tackled the Gazebo's lack of native sonar support through by adapting existing sensor plugins. Since Gazebo does not support sonar simulation, we adjusted the ray sensor plugin to produce a working sonar analog. This approach consists of repurposing a LiDAR sensor with suitable parameters to behave like a sonar system.

A scanning mechanism based on a continuous joint was implemented that allows the sonar link to rotate, simulating a mechanically scanning sonar. This joint is controlled via ROS 2 control, allowing programmatic positioning to make the joint act like a sector scanning sonar. The joint does a full rotation in 35 seconds, which matches the specifications of real mechanical scanning sonars.

A key element of our sonar implementation is the ray-to-sonar converter node. This node takes the ray sensor data and converts it into realistic sonar echoes. For each angular position, it takes the incoming rays and transforms them into intensity values along each beam, creating a sonar echo message with appropriate intensity distributions that simulate acoustic properties rather than optical ones.

The sonar simulation replicates the noise traits of the BlueRobotics Ping360 Mechanical Scanning Imaging Sonar (*Ping360 Sonar — bluerobotics.com* 2025), with a detailed noise model merged into the echo conversion process. The noise model contains ambient background noise across all range bins, setting up a baseline signal-to-noise ratio that changes with range. Each detected object creates intensity patterns with fluctuations of ±10-20 units on a 0-255 scale, simulating the random variations in acoustic reflections like in real environments. Range precision is approximately ±0.5% of the measured range, in comparison with the manufacturer's specification of ±1%. Angular precision is ±0.3°, closely matching the Ping360's mechanical accuracy of ±0.25°.

While this adaptation cannot capture all the acoustic complexities of sonar propagation, it provides a functional approximation that is sufficient for testing feature detection and mapping algorithms. The resulting data closely matches the output of a real mechanical scanning sonar.

4.5 Ground Truth Data Generation

A vital component of the simulation is the ability to obtain ground truth data, which works as a reference for training machine learning models and for evaluating the performance of navigation algorithm. The simulation provides ground truth information through odometry messages obtained from the planar move plugin, containing the true position, orientation, and velocity of the vessel. This data serves multiple purposes, validating the accuracy of the navigation algorithms, providing training data for machine learning models, initializing filter states, and enabling comparative benchmarking over different approaches.

4.6 ROS 2 Integration

The entire simulation is closely connected with ROS 2, allowing all components to communicate using the standard types of messages and communication patterns. Sensors publish data using standard ROS 2 formats: IMU data as sensor_msgs/Imu, DVL velocity as a custom message derived from geometry_msgs/TwistWithCovarianceStamped,

and sonar data as sensor_msgs/LaserScan for the raw range data and a custom message type for processed intensity-based sonar returns.
Sensors publish to appropriately namespaced topics like /imu/data, /dvl/data, and /sonar/scan_echo. The simulation implements a complete transform tree that follows the REP-105 standard, with global map frame as the starting frame, through odom frame, to base link, and on to individual sensor frames (Foote 2013). This integral part of the ROS 2 structure makes it possible for all the simulation components to be accessed through the same interfaces as those used with the actual physical hardware, which smooths the way for a potential real-world deployment of the simulation components.Figure 4.5 shows the transformation tree of the simulation.

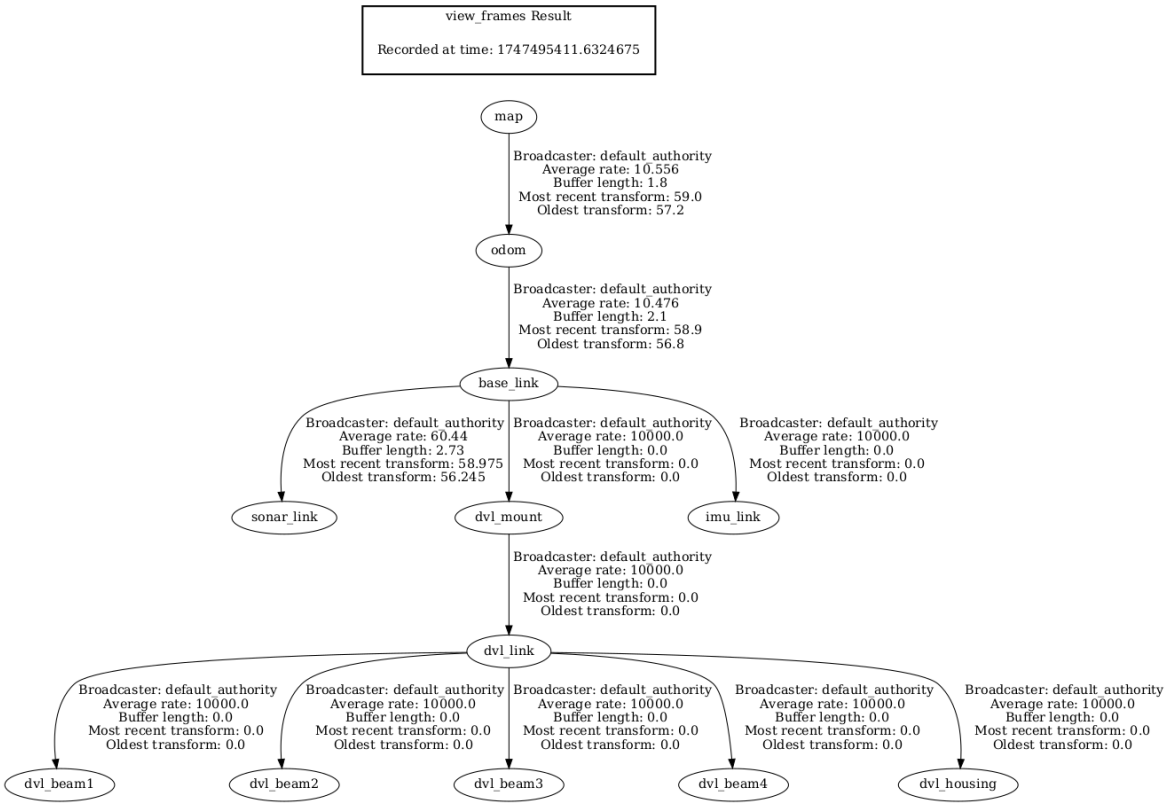


Figure 4.5: TF tree generated

This ROS 2 integration ensures that all components of a simulation can be accessed via the same interfaces as would be used with physical hardware making the transition from simulation to potential real-world deployment much smoother. The comprehensive testbed for our GNSS-free navigation approach is the simulation environment. Using ROS 2 and Gazebo, along with custom plugins and models, a convincing scenario with realistic physical properties and sensor behaviors was created. The implemented noise models assure that the navigation algorithms are tested under challenging conditions, while the generation of ground truth data enables rigorous evaluation of performance.

5

Data Collection and Preprocessing

This chapter explains the methods that were used to collect and preprocess sensor data for the development of a learning based odometry enhancement system for autonomous vessels.

5.1 Data Collection

The foundation of learning-based odometry enhancement lies in the quality and diversity of its training data. This work collected systematic training data in the form of raw sensor measurements from an Inertial Measurement Unit (IMU) and a Doppler Velocity Log (DVL), along with precise ground truth velocities from the Gazebo simulation environment. The objective was to collect a representative dataset to train a Long Short-Term Memory (LSTM) model to estimate states of the vessel accurately.

5.1.1 Motion Scenarios

All data collection was carried out in the Gazebo simulation environment,where three main streams of information was recorded during each session.
Motion scenarios as seen in Figure 5.1 were manually generated by passing velocity commands to the `/cmd_vel` topic to control the execution of different movement patterns. The dataset was collected under a wide variety of vehicle behaviors and motion scenarios to allow the model to generalize well across the different types of vehicle behavior.

Straight-line motion at constant speed was performed. This provided simple, steady-state data with minimal rotational effects and allowed the model to observe basic translation behaviors without the complexity of combined motions.

Circular trajectories were followed at a constant radius and a constant rotational speed. This scenario experienced continuous angular motion along with linear translation, challenging the model to deal with combined dynamics accurately.

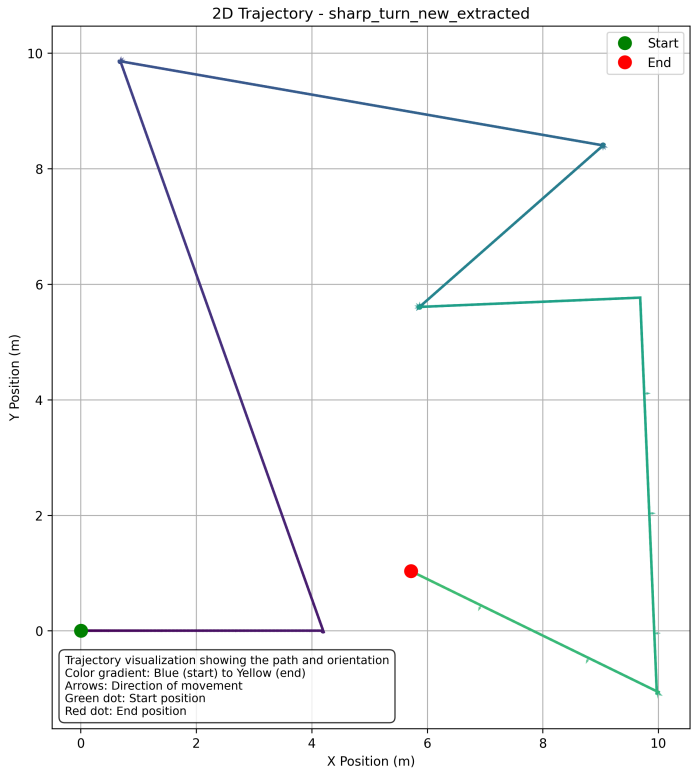


Figure 5.1: Example Trajectory from Collected Scenario

Sharp turns were included, in which the vehicle made rapid, high-curvature maneuvers. These events caused sudden changes in both linear and angular velocities, testing the model to the type of sudden, discontinuous motion patterns that are often encountered in real world settings.

In-place rotations were another vital type of motion, where the vehicle turned about its vertical axis with little translation. This isolated rotational dynamics, helped to model separate rotation from linear drift in noisy sensor readings.

Varying speed motions were performed, where the vehicle accelerated and decelerated during both straight and curved paths. By exposing the model to non-uniform motion, it was trained to expose the model to the dynamic noise characteristics associated with velocity changes.

Randomized movement sequences were executed, using sequences that combined translations, rotations, varying speeds, and unpredictable direction changes. These scenarios mimicked real-world navigation, where a vehicle's path is seldom perfectly structured. Randomness was especially important for ensuring that the LSTM model would not latch onto one or two simple motion patterns and overfit to them.

Each scenario was carried for a duration of two to five minutes, depending on the complexity of the motion. Longer sessions were used for more complex or random movements to guarantee a dataset that was both large and varied. After recording,

the datasets were organized and labeled according to the motion scenario performed, creating a structured dataset ready for preprocessing and model training (Fossen 2011).

5.2 Data Preprocessing

The data preprocessing pipeline transforms raw sensor data from ROS2 bags into a structured format that is appropriate for machine learning models, particularly LSTM networks. As highlighted by Wu et al. (2018), the proper preprocessing has a significant impact on the performance of deep learning models in time-series applications. Our preprocessing workflow handles multiple sensor streams, synchronizes them temporally, and generates sequential data that is suitable for time-series prediction tasks.

5.2.1 Data Extraction

Our preprocessing pipeline extracts and processes data from multiple complementary sensing modalities (Paull et al. 2014): **Ground Truth Odometry** (`/odom`) provides the reference velocity data that serves as the model training target; **IMU Data** (`/imu/data`) contains angular velocity measurements that capture and emphasize the vessel's motion dynamics; **DVL Measurements** (`/dvl/data`) supply linear velocity information, which directly complements the IMU data.

The first preprocessing step involves extracting raw sensor data from the ROS2 bag files using the `rosbag2_py` library. For each supported topic, relevant messages are deserialized and parsed into a structured format, making them ready for further processing. This includes normalizing timestamps to seconds, enabling temporal alignment across the different sensor streams.

5.2.2 Temporal Synchronization

One of the main problems in sensor fusion is the handling of the asynchronous nature of the different streams of sensors. The preprocessing pipeline puts into practice a temporal synchronization strategy that uses the IMU timestamps as reference points. The temporal synchronization employs a nearest-neighbor matching algorithm under a maximum time difference constraint, which can be mathematically expressed as:
For each IMU timestamp t, a measurement from another sensor is included if:

$$|t_{sensor} - t| < \text{threshold} \tag{5.2.1}$$

the threshold can be set to different levels (default: 0.5 seconds). This temporal proximity metric assures that all measurements in a synchronized record represent the system state at approximately the same moment. As noted by Schmidhuber (2015), consistent temporal alignment is essential to maintain the statistical validity of state estimation problems.

5.2.3 Feature Engineering

Subsequent to synchronization, the records are structured into datasets pertaining to features and targets, in accordance with the conventions of standard machine learning. The feature engineering process ensures that the input-output pairs are meaningful and capture the essence of the prediction task.
Input features gathers the full range of sensor measurements from IMU and DVL, which together provide a comprehensive representation of the vessel's dynamic state. These can be represented as:

$$X = \{x_{IMU}, x_{DVL}\} \tag{5.2.2}$$

Where x_{IMU} contains angular velocities, and x_{DVL} contains velocity measurements:

$$x_{IMU} = \{\omega_x, \omega_y, \omega_z\} \tag{5.2.3}$$

$$x_{DVL} = \{v_x, v_y, v_z\} \tag{5.2.4}$$

Output targets consist of ground truth velocity values extracted from the reference odometry data:

$$Y = \{v_x^{GT}, v_y^{GT}, v_z^{GT}, \omega_x^{GT}, \omega_y^{GT}, \omega_z^{GT}\} \tag{5.2.5}$$

This clear separation between inputs and outputs makes the training of models to predict the ground truth state based on raw sensor inputs. The relationship can be formulated as:

$$f : X \to Y \tag{5.2.6}$$

Where f represents the learning algorithm discovering the complex relationships between sensor measurements and the actual vessel state.

5.2.4 Sequence Generation

For learning from time series, especially when using LSTM networks, the synchronized data is taken and arranged into a sequential format, using a sliding window approach. This method creates a representation of the data that is structured allowing the system to discern the underlying temporal patterns in the motion of the vessels.
Using a default window length of 20 time steps, overlapping sequences are created from our dataset that act as input-output pairs for training. The formulation for this sequence generation process is:

$$X_i = [x_i, x_{i+1}, ..., x_{i+n-1}] \tag{5.2.7}$$

$$y_i = x_{i+n} \tag{5.2.8}$$

Where $X_i \in \mathbb{R}^{n \times d}$ denotes the input sequence with n time steps and d dimensions, and $y_i \in \mathbb{R}^{d'}$ represents the target values. The sequence length n is configurable, with a default value of 20 derived from empirical testing on marine navigation data (Längkvist, Karlsson, and Loutfi 2014).
The window slides forward with stride $s = 1$, as shown in Figure 5.2, creates a new sequence for each time step:

$$X_{i+1} = [x_{i+1}, x_{i+2}, ..., x_{i+n}] \tag{5.2.9}$$

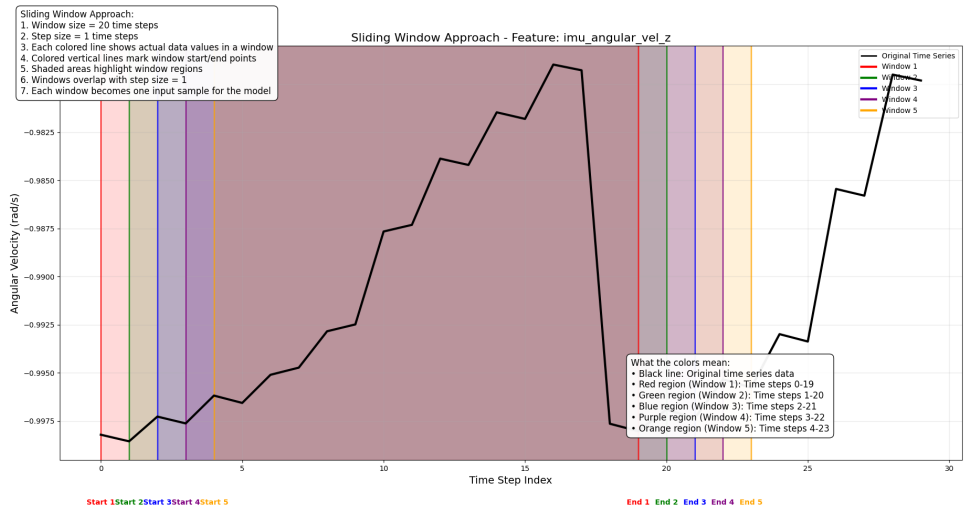


Figure 5.2: Sliding Window Approach

$$y_{i+1} = x_{i+n+1} \tag{5.2.10}$$

This creates significant overlap between sequences, which boosts the training data and helps the model learn patterns at various positions in the time series. The total number of sequences generated can be calculated as:

$$N_{sequences} = N_{samples} - n \tag{5.2.11}$$

Where $N_{samples}$ represents the total number of synchronized measurements in our dataset.
We store these sequence-target pairs as NumPy arrays, enabling efficient memory usage with optimal training performance. The resulting data structure follows the format:

$$X_{sequences} \in \mathbb{R}^{N_{sequences} \times n \times d} \tag{5.2.12}$$

$$y_{targets} \in \mathbb{R}^{N_{sequences} \times d'} \tag{5.2.13}$$

The system enables efficient batch processing during model training, all while retaining the original temporal structure of the data.

5.2.5 Output Formats

The preprocessing pipeline creates various output files that serve different purposes in the machine learning workflow. For each sensor topic, it generates a raw CSV file that preserves the original measurements. At the same time, it also generates a dataset that contains aligned measurements from all sensors, which provides a comprehensive view of the system state at each timestamp. Separate feature and target CSV files were generated that could be used for traditional ML methods that does not require sequential data. NumPy sequence arrays optimized for LSTM training was created enabling efficient model training. Finally, dataset information files document preprocessing parameters and dataset statistics, enabling to reproduce the results in a robust way.

5.2.6 Implementation Details

The execution uses the pandas library to manipulate the data and NumPy to handle the array operations efficiently, following the modern data science best practices (McKinney 2017). The preprocessing comprises mathematical operations like temporal alignment and sequence generation, which are optimized for numerical stability and computational efficiency.

The implementation also includes sturdy error-handling systems that enforce the mathematical constraints of the sensor data, such as making sure that the velocity vector representations are kept within the bounds of physical plausibility. These mathematical safeguards are of utmost importance for ensuring the quality and reliability of the resulting datasets. The subsequent chapter will expand upon this prepared dataset, laying out the formation and teaching of the LSTM model that is meant to eliminate the errors in marine vehicle odometry.

6

Deep Learning Based Sensor Enhancement

This chapter examines a deep-learning framework aimed at enhancing the accuracy of the inertial measurement unit (IMU) and Doppler velocity log (DVL) sensor outputs in autonomous maritime navigation systems. The proposed approach clearly focuses on learning a direct mapping from raw sensor data to ground truth values, which in turn mitigates the negative effects of sensor noise, drift, and non-linear behaviors. Conventional enhancement techniques, like Kalman filter variations, rely heavily on predefined motion models and statistical assumptions, which limit their applicability in complex operational environments and require significant parameter tuning. In contrast, the integration of deep learning with the Extended Kalman Filter (EKF) allows for the provision of enhanced sensor measurements that accounts the systematic errors and unmodeled dynamics. According to empirical findings, estimates of DVL linear velocity and IMU angular velocity both show notable gains in accuracy under various circumstances. This enhancement results in increased resilience and precision in autonomous navigation, particularly in GNSS-denied or sensor-challenging scenarios.

6.1 Selection of LSTM over Alternative Deep Learning Techniques

Long Short-Term Memory networks were selected for sensor enhancement after careful analysis of maritime sensor data characteristics and evaluation of various deep learning approaches. Maritime sensor data are difficult to work with as it presents unique challenges like strong temporal correlations between states, non-linear relationships between raw readings and actual motion, variable length operational sequences, and multiple noise types including systematic biases and environment-specific disturbances.

Before selecting LSTM networks for our maritime sensor enhancement system, we considered several alternative deep learning approaches.

RNNs are basic neural network architectures. A basic type of RNN can handle sequential data, but it suffers from the vanishing gradient problem when processing long sequences. This makes them unsuitable for applications like the one discussed here, where dependencies between sensor readings may span significant time intervals, especially with vessels moving at varying speeds (Pascanu, Mikolov, and Bengio 2013). Their inability to maintain long-term dependencies makes them ineffective for complex maritime motion patterns.

Gated Recurrent Units (GRUs) provide a simpler gating mechanism than LSTMs, allowing them to be more efficient in computation. While effective for many sequence tasks, they lack LSTM's dedicated cell state and output gate which provide advantages when processing noisy sensor data. For maritime applications where accuracy is critical, LSTM's superior handling of complex noise patterns justifies its slightly higher computational cost.The architectural comparisons of RNN,GRU and LSTM is depicted in Figure 6.1

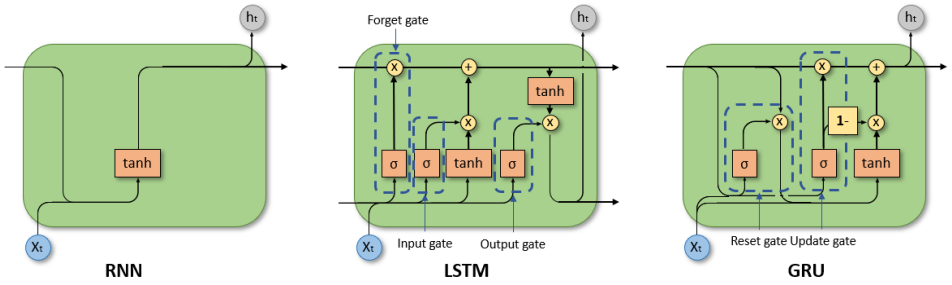


Figure 6.1: Comparison of RNN, GRU, and LSTM Cell Architectures
(*Unlocking the Power of LSTM, GRU, and the Struggles of RNN in Managing Extended Sequences — Murad Atcorvit* 2025)

Convolutional Neural Networks (CNNs) performs at their best at spatial feature extraction but lack inherent temporal modeling capabilities. They typically require transforming time-series data into image-like representations, bringing in more complexity and potential information loss. CNNs struggle to capture the intricate temporal dynamics which is essential for accurate vessel motion prediction, particularly with varying sampling rates.The architectural design of CNN is shown in Figure 6.2

Transformer Models yields great results for sequence modeling but their self-attention mechanisms are computationally expensive for real-time applications on embedded systems (Vaswani et al. 2017). They also need larger datasets than typically available for specialized maritime applications. Despite their theoretical advantages, their computational requirements make them impractical for onboard vessel systems.

Limitations in these systems arise from their typically simplistic architectures and computational models. LSTM networks address these limitations through their specialized architecture with its gating mechanisms that selectively retain relevant information while eliminating the noise. This architecture balances temporal modeling capability,

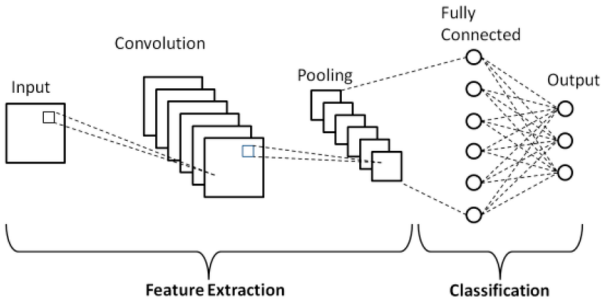


Figure 6.2: Basic CNN Architecture
(*Deep Learning Basics Part 8: Convolutional Neural Network — Sasirekha Rameshkumar* 2025)

noise resilience, and computational efficiency thereby making it optimal for the sensor enhancement tasks in navigation system. In addition, the bidirectional implementation provides each prediction with complete temporal context, further improving accuracy during complex maneuvers.

6.2 Fundamentals of LSTM Networks

The temporal dimension of sensor data is inherent, as current readings are contextually related to both previous and subsequent measurements. These temporal dependencies are especially well-suited for recurrent neural networks (RNNs), specifically Long Short-Term Memory (LSTM) networks. LSTMs use a number of mechanisms that enable them to selectively retain information over long sequences, in contrast to standard RNNs that have trouble with long-range dependencies because of vanishing gradients.

At its core, an LSTM network is made up of memory cells,each of which has three primary gating mechanisms: the forget gate determines which information from the previous cell state should be discarded; the input gate controls what new information should be stored in the cell state; and the output gate regulates what information from the cell state should be output.The architecture of LSTM is described in Figure 6.3

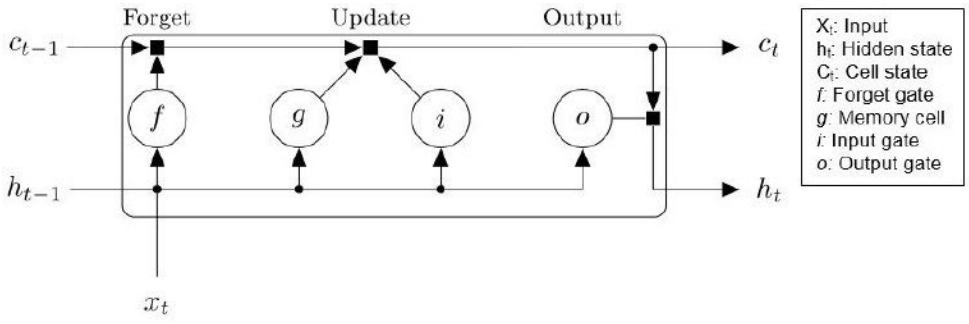


Figure 6.3: LSTM Architecture
(*What is LSTM (Long Short-Term Memory)? — MathWorks* 2025)

These gates perform as learnable filters, effectively forming an adaptive memory system that can selectively keep relevant information and toss out noise. This is a way to sharpen the sensor's capabilities while distinguishing between the motion of a real vessel and the noise from a sensor.

6.2.1 Bidirectional LSTM Architecture

We extend the standard LSTM concept in our implementation by using a bidirectional architecture (BiLSTM) as seen in Figure 6.4. In a BiLSTM, the input sequence is processed in both the forward and the backward directions simultaneously, and the outputs are combined to form a unified representation (Schuster and Paliwal 1997). This bidirectional processing gives the network a full temporal context by allowing each enhanced output to benefit from both the past and future sensor readings.

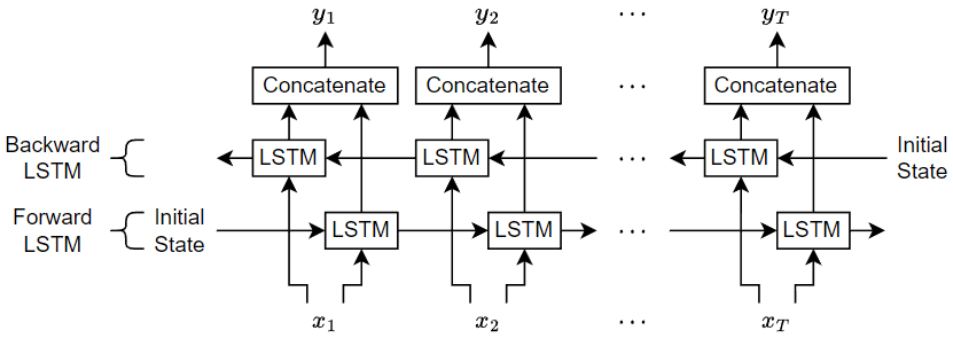


Figure 6.4: Bidirectional LSTM Architecture
(*What is LSTM (Long Short-Term Memory)? — MathWorks* 2025)

Mathematically, for a given input sequence $x = (x_1, x_2, ..., x_T)$, a BiLSTM computes:

- A forward hidden sequence $h = (h_1, h_2, ..., h_T)$
- A backward hidden sequence $h = (h_1, h_2, ..., h_T)$
- The combined output sequence $y = (y_1, y_2, ..., y_T)$ where each y_t is a function of both h_t and h_t

As it enables the model to take into account the entire trajectory context when enhancing each individual measurement point, this bidirectional processing is particularly useful for sensor enhancement.

6.3 Implemented Network Architecture

A carefully designed neural network architecture with precisely defined layers is used for the sensor enhancement system. In order to improve both IMU angular velocities and DVL linear velocities, this design approach is applied to specialised models. This allows for the targeted improvement of these crucial motion parameters while preserving computational efficiency appropriate for real-time deployment in autonomous

vessel systems.

The entire architecture starts with an input layer that receives sequences of a single sensor reading. Each sequence is configured to process 20 timesteps, which allows the model to gain sufficient temporal context from the input data. The main body of the architecture is made up of two layers of Bidirectional LSTMs. The first layer has 32 hidden units in each direction (forward and backward), so there are a total of 64 units in this layer and returns the full sequence output. The second BiLSTM layer also uses 32 hidden units per direction but only returns the output from the final timestep, creating a 64-dimensional feature vector.

After the stack of BiLSTMs, a Layer Normalization operation is performed to maintain stable training by normalizing activations across the feature dimension while using learnable scaling and shift parameters. Following this, a fully connected layer reduces dimensionality from 64 to 32 neurons with a ReLU activation (Nair and Hinton 2010). To stave off any overfitting, a dropout layer with a rate of 0.3 has been tossed in that randomly deactivates 30% of the neurons during training (Gal and Ghahramani 2016). Finally, the output layer consists of a single neuron with linear activation that maps the 32-dimensional feature vector to a single enhanced sensor reading.

This design enabled focused improvement of these important motion parameters while preserving computational efficiency fit for real-time operation in autonomous vessel systems. The models can more successfully correct systematic errors and noise by concentrating on the particular error patterns and features of every measuring method without compromising general system performance.

6.4 Data Processing Pipeline and Training Methodology

This section explains the data processing pipeline and training methodology used to prepare raw sensor data and train the LSTM models for sensor enhancement.

6.4.1 Dataset Construction

The first step in the training process is gathering a lot of data on vessel operations. We collect operational metadata that allows for context-aware evaluation, raw sensor readings at their native sampling frequency, and related ground truth values from high-precision reference systems.

After then, it undergoes a number of refinement steps to get ready for model training. In order to guarantee that every aspect of the input have comparable scales and to promote numerical stability during training, normalization is carried out using a fitted StandardScaler. Following that, the dataset is split into training and validation sections (80% and 20%, respectively), with extra effort taken to ensure that the distributions of vessel maneuvers and the surrounding conditions in each subset are representative.

To ensure consistency between training and operational use, the model deployment package includes all normalization settings.

6.4.2 Training Process

The training procedure combines different techniques to ensure that the model performs robustly. The data is passed through the system in mini-batches of 64 samples, achieving a fair balance between computational efficiency and the stochastic gradient benefits. We deploy the Adam optimizer, with an initial learning rate of 0.001 and a set of parameters for which Adam performs efficiently and effectively across many different types of networks.

The objective function is Mean Squared Error, and it is concerned with minimizing the squared difference between the improved readings and the actual, or ground-truth, values. To avoid overfitting, we automatically terminate training early if there is no improvement in validation loss after 15 epochs. Throughout training, we hold on to the best-performing model, as indicated by the validation metrics.

An essential consideration in our approach is ensuring batch sizes stays large to maintain the effectiveness of the LayerNorm (Ba, Kiros, and Hinton 2016). By removing extremely small batches from training, we avoid normalization instabilities and get more consistent learning dynamics.

6.4.3 Implementation Platform

PyTorch deep learning framework (version 1.9.0) is utilized for the implementation. It is chosen for its flexibility in defining custom architectures and better handling of sequential data. The training was conducted on NVIDIA CUDA-enabled hardware to accelerate the training process, with an approximate training time of 2-3 hours per sensor model depending on dataset size.

6.5 Ablation Studies and Architecture Validation

Our architectural selections were validated via comprehensive ablation studies. Testing BiLSTM layer counts showed that a single layer reduced performance , while three layers yielded minimal improvements despite more parameters, confirming that two layers strikes an optimal balance between efficiency and performance. For number of hidden units, we found that 16 units cause performance reductions due to underfitting, while 64 units offer relatively negligible benefits despite higher computational costs, confirming that 32 units per direction is a very reasonable tradeoff between performance, cost, and convergence speed.

Experiments with the dropout rate showed that a rate of 0.1 is obviously too low in allowing the model to overfit (at least by epoch 30), while a rate of 0.5 is obviously too

high and is causing excessive regularization and very poor convergence behavior, supporting the choice of 0.3 as a much safer rate. Of the various normalization approaches, BatchNorm is the most problematic, producing unstable results with variable batch sizes. Models without normalization converge slowly and are more sensitive to the learning rate.

In contrast, LayerNorm seems to have no such weaknesses. It is not only more stable than BatchNorm, but also more robust in yielding good results under a wide variety of conditions. These systematic evaluations show that every architectural element meaningfully contributes to the performance of our sensor enhancement models.

6.6 Experimental Results and Performance Analysis

This section presents the experimental results obtained from training and evaluating the Long Short-Term Memory (LSTM) models designed for enhancing Inertial Measurement Unit (IMU) and Doppler Velocity Log (DVL) sensor data

6.6.1 Model Performance Overview

All targeted sensor components showed notable increases in measurement accuracy when using the BiLSTM-based sensor enhancement models. Models successfully learn to map raw sensor readings to their ground truth counterparts after being trained on large datasets.A summary of each model's performance metrics can be found in Table6.1.

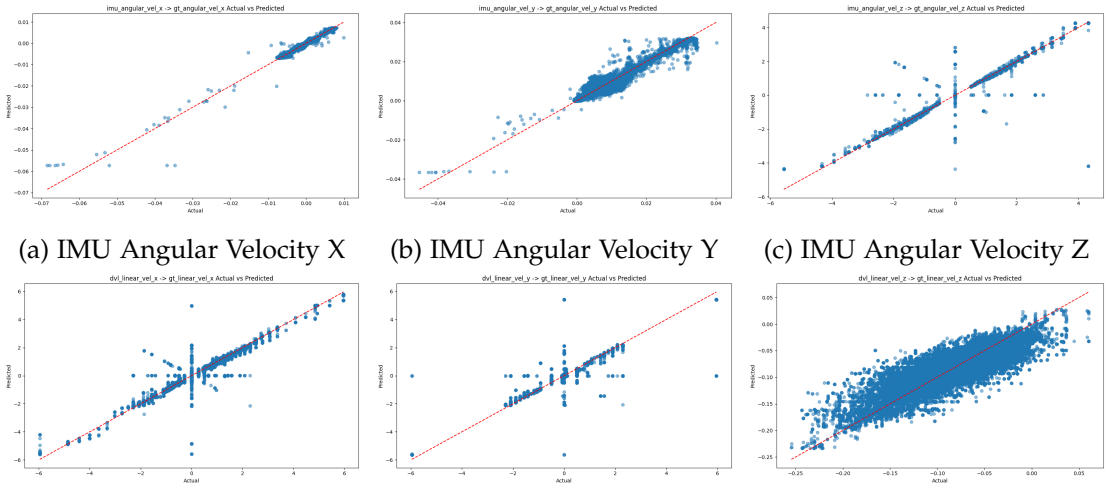
Table 6.1: Sensor-wise R^2 and RMSE Metrics

Sensor Type	**Component**	**R^2**	**RMSE**
IMU Angular Velocity	X	0.9768	0.0003
	Y	0.9666	0.0010
	Z	0.9948	0.1033
DVL Linear Velocity	X	0.9730	0.1461
	Y	0.9390	0.1319
	Z	0.6680	0.0210

The percentage of variance in the ground truth measurements that can be predicted from the model outputs is shown by the R^2 values. Five of the six components have values above 0.93, indicating that the models have good predictive power. With an R^2 of 0.6680, even the DVL vertical velocity component exhibits a significant improvement over raw measurements.The accuracy of the predictions is further supported by the low RMSE values for each component.

6.6.2 Prediction Accuracy Analysis

For all six sensor components, scatter plots in Figure 6.5 show the predicted values compared to the actual values. The data points are close to what would be expected if the model were ideal (red dashed line).



(a) IMU Angular Velocity X (b) IMU Angular Velocity Y (c) IMU Angular Velocity Z

(d) DVL Linear Velocity X (e) DVL Linear Velocity Y (f) DVL Linear Velocity Z

Figure 6.5: Scatter plots comparing predicted versus actual values for all six sensor components. The red dashed line represents the ideal prediction.

The scatter plots reveal crucial aspects of each model's performance. The angular velocity IMU models demonstrate exceptionally tight clustering close to the ideal prediction line, which corresponds to their high R^2 values. The tight clustering is particularly evident in the IMU Z-axis model (yaw rate), which achieved the highest R^2 (0.9948) among all the components.

DVL forward and lateral velocity predictions are also in strong correlation with ground truth values, as DVL vertical velocity component predictions are more scattered but still show a decent correlation. The vertical component's lower R^2 is just a better reflection of how unfavorable factors affect prediction quality. This is expected given that vertical motion in vessels is typically smaller in magnitude and more influenced by environmental factors like waves, making it inherently more challenging to predict.

6.6.3 Model Training Efficiency

The IMU Z-axis angular velocity model achieved an unprecedented performance, making it extremely useful for estimating vessel heading—a key parameter for navigation and planning the trajectory. DVL forward (X) and lateral (Y) velocity models performed excellently, making position estimation quite solid during loss of GNSS. The vertical velocity component performed acceptably, which is reasonable considering the path taken by a vessel and how little vertical motion impacts that path. Overall training speed, a measure of how quickly we can adjust model parameters to better

fit the training data was good across the board, with the IMU models converging to a final form quite a bit faster than the DVL models. This is probably because the IMU data itself is more consistent than the DVL data, which can vary quite a bit depending on environmental conditions.

This chapter has introduced a profound approach based on deep learning for improving the precision of IMU and DVL measurements on autonomous vessels. At the core of our method is a bidirectional LSTM network, which we have found to work exceptionally well for this kind of problem, operating on data over time (i.e., in the temporal domain). The sensor specific modeling approach has delivered impressive gains in accuracy while keeping computational efficiency in mind.

The demonstrated abilities show that deep learning methods can considerably surpass conventional filtering techniques for maritime usages, meaning that these methods might well enable robust autonomous vessel operations in challenging environments. While there are room for several future refinements most notably for vertical velocity estimation, the current implementation represents a significant improvement in potential maritime sensor enhancement technology.

7

Sensor Fusion and Robot Localization

The maritime environments faces unique challenges in precise navigation, especially when signals from Global Navigation Satellite System (GNSS) are not available or are unreliable (Kinsey, Eustice, and Whitcomb 2006). This chapter present our extensive approach to vessel localization using sensor fusion techniques that aggregate the data from several different types of sensors.A two-stage system is created, that the first stage enhances raw sensor measurements through deep learning models and the second stage involves the fusion of these enhanced readings using an Extended Kalman Filter (EKF) to produce robust state estimates.

7.1 Real-Time Sensor Enhancement System

This section describes the implementation of a real-time system for enhancing the quality of Inertial Measurement Unit (IMU) and Doppler Velocity Log (DVL) sensor data. The primary objective of this system is to provide refined sensor measurements by applying pre-trained models to the raw sensor outputs as they are received

7.1.1 System Workflow

The enhancement system for the sensors can be integrated directly and easily into the vessel's existing infrastructure, providing real-time data refinement using pre-trained Bidirectional Long Short-Term Memory (BiLSTM) models(Yang et al. 2022). First, the system loads required components into memory, including the neural network configurations, model weights, and data scalers for both IMU angular velocities and DVL linear velocities. The scalers ensure consistency by normalizing incoming data in the same manner as during training.

The system receives IMU and DVL measurements by subscribing to standard ROS2 topics. The measurements are stored in dedicated sequence buffers and maintain the 20 most recent readings in a sliding window (Moreira, Vettor, and Guedes Soares 2021). When enough data has been accumulated, the sequences are normalized and passed through reconstructed BiLSTM models. The architecture of each model is dynamically rebuilt from saved configuration files, which ensures that the model is structurally

compatible with the pre-trained weights. Once the models are in place and initialized in evaluation mode, efficient inference is performed to enhance measurement accuracy.

After inference, the enhanced outputs are denormalized to their original physical units. These refined measurements are then published as standard ROS2 messages at 10 Hz, mirroring the original topic formats. By simply subscribing to the improved data streams, downstream components can take advantage of increased sensor accuracy without requiring any changes. Specialized computational resources are not required because the entire inference pipeline is designed to operate in real time on common CPU hardware.

7.1.2 Operational Integration

For autonomous vessel navigation, the sensor enhancement system offers a number of advantages. The accuracy of heading estimation is increased by improved angular velocity measurements from the IMU, which lowers deviations during autonomous transits. Similarly, improved DVL velocity measurements increase the accuracy of position estimation as seen in Figure 7.1, especially during precise maneuvering operations or in difficult environments with complex currents.

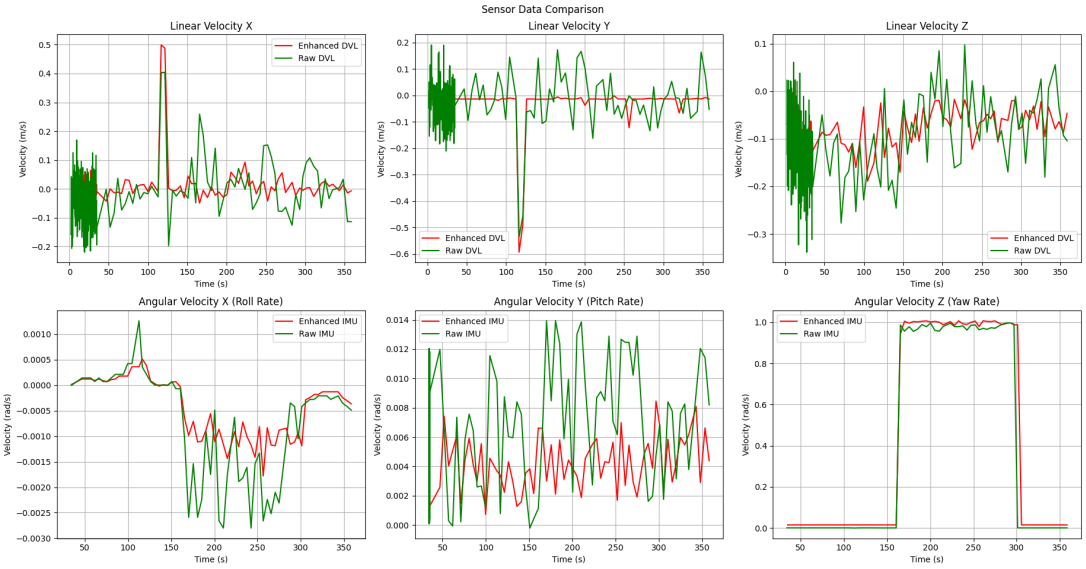


Figure 7.1: Comparison between Raw odometry vs Enhanced odometry

This system for enhancing data integrates within the existing navigation stack of the vessel through the use of standard ROS2 interfaces. The data that is enhanced maintains the same coordinate frames, timestamp conventions, and metadata as the original sources, which ensures compatibility across the entire system.

7.2 Robot Localization Through Sensor Fusion

The implementation combines data from two primary sensors: the IMU for angular velocity measurements, and the DVL for linear velocity measurements relative to the seafloor.This ROS 2 node processes the output from the two sensors at 30 Hz. It maintains the transformations between the odom and base link frames and also accounts for process and measurement noise.

7.2.1 Extended Kalman Filter Framework

The Extended Kalman Filter (EKF) is a recursive algorithm that estimates the state of a dynamic, nonlinear system. It consists of two main steps: prediction and update. The prediction step uses the system dynamics model to predict the state evolution, while the update step uses sensor measurements to refine this prediction.
There are twelve elements in the state vector that together reflect the vessel's whole pose and motion range:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T \tag{7.2.1}$$

where, (x, y, z) represent the position in the global frame,(ϕ, θ, ψ) represent the roll, pitch, and yaw angles,(v_x, v_y, v_z) represent the linear velocities in the body frame, $(\omega_x, \omega_y, \omega_z)$ represent the angular velocities around the body axes
The uncertainty in this state is represented by a 12×12 covariance matrix $\mathbf{P}$, which changes over time as new predictions and measurements are processed.

7.2.2 Prediction Step

The system state is updated in the prediction step according to the current state and the amount of time that has passed since the last update. The process consists of several components:
Orientation Update: Using the relevant kinematic differential equations, the orientation (roll, pitch, and yaw) is updated by integrating angular velocities over time:

$$\dot{\phi} = \omega_x + \omega_y \sin(\phi) \tan(\theta) + \omega_z \cos(\phi) \tan(\theta) \tag{7.2.2}$$
$$\dot{\theta} = \omega_y \cos(\phi) - \omega_z \sin(\phi) \tag{7.2.3}$$
$$\dot{\psi} = \omega_y \sin(\phi) / \cos(\theta) + \omega_z \cos(\phi) / \cos(\theta) \tag{7.2.4}$$

As 3D rotations are non-commutative, these rates take into consideration the coupling between angular rates in various axes.
Position Update: Using the current orientation, the body frame's linear velocities are converted to the world frame and then integrated to update the position:

$$\mathbf{v}_{\text{world}} = \mathbf{R} \cdot [v_x, v_y, v_z]^T \tag{7.2.5}$$
$$x+ = \mathbf{v}_{\text{world}}[0] \cdot \Delta t \tag{7.2.6}$$
$$y+ = \mathbf{v}_{\text{world}}[1] \cdot \Delta t \tag{7.2.7}$$
$$z+ = \mathbf{v}_{\text{world}}[2] \cdot \Delta t \tag{7.2.8}$$

where $\mathbf{R}$ is the rotation matrix derived from the current roll, pitch, and yaw.

Covariance Update: The state covariance matrix $\mathbf{P}$ is updated according to the state transition model and process noise:

$$\mathbf{P} = \mathbf{F} \cdot \mathbf{P} \cdot \mathbf{F}^T + \mathbf{Q} \cdot \Delta t \tag{7.2.9}$$

where $\mathbf{F}$ is the state transition matrix that incorporates the rotation matrix for position updates and the kinematic coupling for orientation updates.

7.2.3 Update Step

The update step incorporates newly available sensor measurements into the state estimate (Simon 2006). The update equations for a measurement $\mathbf{z}$ with a known measurement model $h(\mathbf{x})$ are as follows:

$$\mathbf{y} = \mathbf{z} - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1} \tag{7.2.10}$$

$$\mathbf{S} = \mathbf{H} \cdot \hat{\mathbf{P}}_{k|k-1} \cdot \mathbf{H}^T + \mathbf{R} \tag{7.2.11}$$

$$\mathbf{K} = \hat{\mathbf{P}}_{k|k-1} \cdot \mathbf{H}^T \cdot \mathbf{S}^{-1} \tag{7.2.12}$$

$$\mathbf{x}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K} \cdot \mathbf{y} \tag{7.2.13}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K} \cdot \mathbf{H}) \cdot \hat{\mathbf{P}}_{k|k-1} \cdot (\mathbf{I} - \mathbf{K} \cdot \mathbf{H})^T + \mathbf{K} \cdot \mathbf{R} \cdot \mathbf{K}^T \tag{7.2.14}$$

where $\mathbf{y}$ is the innovation (the difference between measured and predicted values),$\mathbf{H}$ is the measurement matrix that maps the state to the measurement space, $\mathbf{R}$ is the measurement noise covariance matrix, $\mathbf{K}$ is the Kalman gain,The final expression for $\mathbf{P}_{k|k}$ uses the Joseph form for numerical stability

7.2.4 Adaptations from Previous Work

The implementation is built upon the work of Pazarci (2024), with several key adaptations made to better align with the requirements of the localization system. One major change is the use of direct DVL measurements for velocity estimation which provides improved accuracy, replacing the earlier approach that relied on a laser scanner as a soft sensor. In order to replicate the behaviour and noise properties of particular simulated sensors, process and measurement noise parameters were also adjusted. Some elements have been reinterpreted to better reflect the dynamics of our vessel, but the overall state structure is still in line with the original design.

These modifications guarantee that the robot localization component provides accurate state estimation for the SLAM system while blending in perfectly with our simulation environment.

8

Feature Detection, Motion Compensation and SLAM Implementation

This chapter presents the development of basic components for sonar-based navigation, focusing on feature detection, motion compensation, and pose-graph based SLAM. The feature detection module utilizes the SOCA-CFAR algorithm to reduce noise and retain important environmental features. The motion compensation system corrects for vessel movement during sonar scans, ensuring consistent data for SLAM. Finally, the pose-graph based SLAM system uses a probabilistic occupancy grid map, updating with Bayesian methods and optimized by ISAM2, to enable improved mapping and localization.

8.1 Feature Detection Implementation

The Feature Detection module uses a SOCA-CFAR (Smallest-Of Cell-Averaging Constant False Alarm Rate) algorithm to denoise sonar data while keeping essential features that are needed for SLAM. The theoretical framework is from Pazarci (2024), which was adapted for use with our ROS2. The module itself is a ROS2 node that subscribes to raw sonar echo data and publishes filtered output, which is consistent with the ROS2 communication standard.

The module performs CFAR processing using a circular buffer that holds recent sonar echoes. This buffer enables an analysis of the leading and trailing training cells for each cell undergoing testing. For reliable noise estimation, the setup includes 40 training cells along with 10 guard cells on each side; this configuration therby serves to prevent any sort of feature distortion in the output, as emphasized by Richards (2022).

For every intensity value from the sonar, averages are computed from the leading and trailing training cells. The smaller of the two is multiplied by a threshold factor (0.8) to generate an adaptive threshold, effectively adjusting to local noise variations,a method that aligns with Rohling (1983) recommendations for sonar processing.Any value higher than this threshold is retained, while others are suppressed to zero. This selective filtering removes background noise while preserving meaningful features

such as walls and obstacles.The comparison between raw sonar data and filtered features is shown in Figure 8.1

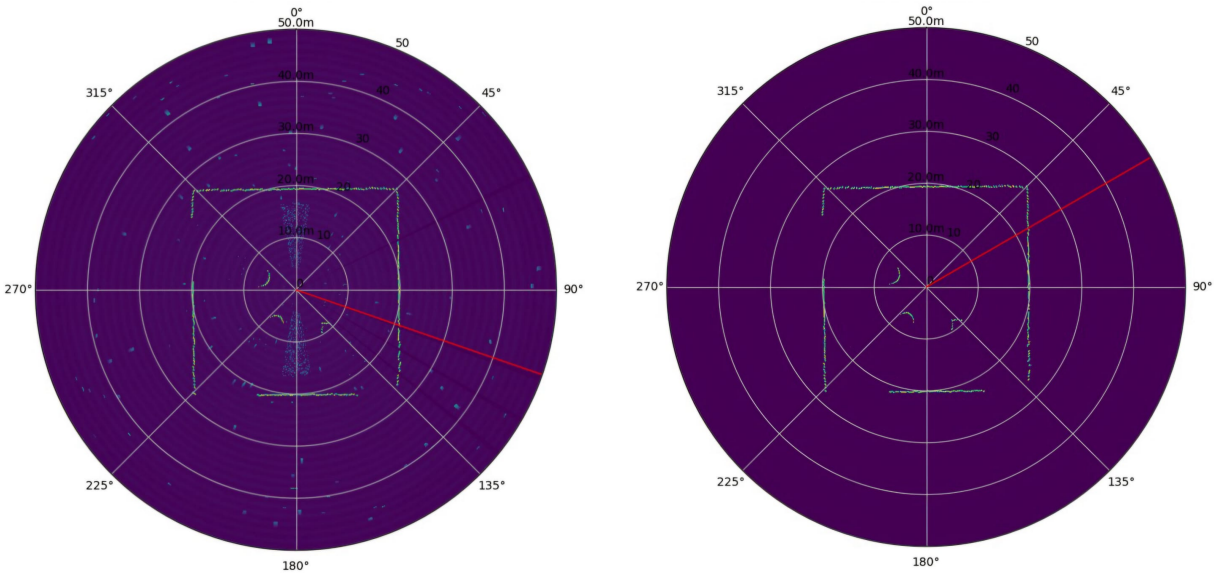


Figure 8.1: Feature Detection Process: Raw sonar data (left) and filtered features (right)

Overall, the implementation of SOCA-CFAR deals very well with the complicated nature of sonar data—such as changing noise levels across space, reflections, and distortions due to motion.Westman, Hinduja, and Kaess (2018) have done several comparison studies that show how beneficial it is to extract the right features from the sonar data. When the data is filtered in the right way, it presents a cleaner version of the environment. This filtered data greatly helps the Motion Rectification and SLAM components, which are the next logical steps in the overall autonomous navigation pipeline.

8.2 Motion Compensation

An incrementally scanning mechanical sonar requires time to perform a full scan of the environment. This process takes considerable time approximately 35 seconds for a full 360° scan. When the vessel moves during scanning, the resulting data becomes distorted because different parts of the scan are obtained from different positions and orientations (Ribas et al. 2006). This creates a significant issue for SLAM algorithms because it causes the same environment to appear differently based on the vessel's movement (Folkesson et al. 2007).

To implement SLAM effectively, one must obtain a reliable reading of the environment under any conditions. Building upon the groundwork developed by Pazarci (2024), this step addresses the challenge by compensating for vessel movement during the scanning process. The solution references each sonar beam to a fixed reference frame, thereby effectively undistorting the scan.

As the first beam of a new scan comes in, the current pose (position and orientation) of the vessel is recorded as the reference frame. Each subsequent beam is then transformed to align with this reference, taking into account the movement of the vessel since the scanning started.

For each sonar beam with bearing α and range r, we first translate to Cartesian coordinates:

$$x_{\text{sonar}} = r \cdot \cos(\alpha) \tag{8.2.1}$$
$$y_{\text{sonar}} = r \cdot \sin(\alpha) \tag{8.2.2}$$

We then apply the rigid transformation to express the coordinates in the scan reference frame:

$$x_{\text{ref}} = x_{\text{sonar}} \cdot \cos(\theta_t - \theta_{\text{ref}}) - y_{\text{sonar}} \cdot \sin(\theta_t - \theta_{\text{ref}}) + (x_t - x_{\text{ref}}) \tag{8.2.3}$$
$$y_{\text{ref}} = x_{\text{sonar}} \cdot \sin(\theta_t - \theta_{\text{ref}}) + y_{\text{sonar}} \cdot \cos(\theta_t - \theta_{\text{ref}}) + (y_t - y_{\text{ref}}) \tag{8.2.4}$$

where (x_t, y_t, θ_t) represents the vessel's current pose and $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}})$ is the reference pose recorded at the start of the scan (Mallios et al. 2014). The transformed point $(x_{\text{ref}}, y_{\text{ref}})$ represents the beam data in the fixed reference frame. The comparison between the uncompensated image and the motion-compensated image is shown in Figure 8.2.

Figure 8.2: Motion Compensation: Distorted uncompensated sonar image (left) and Compensated Sonar Image(right)

The implementation has several important improvements that increase both robustness and accuracy. It has a much more refined frame transformation system, which uses the TF2 library to correctly handle the transformations needed between the coordinate frames of the vehicle's body and the odometry. Another improvement is the implementation of a reliable odometry buffer system that synchronizes vessel movement

data with sonar timestamps. These enhancements together result in a more reliable motion rectification system that assures consistent environmental perception regardless of vessel movement patterns, laying a solid foundation for the subsequent SLAM processes.

8.3 Sonar based SLAM Implementation

A pose-graph based SLAM algorithm for inland environments is implemented building on the pioneering work of Pazarci (2024), This methodology represents both the trajectory of the vessel and environmental observations as a factor graph structure where the nodes represent the poses of the vehicle at different times, and the edges represent the constraints between these poses (Thrun et al. 2006). These constraints come from odometry measurements and loop closure detections, both of these are modeled with appropriate uncertainty factors to account for measurement reliability.

At the center of the system lies the ISAM2 (Incremental Smoothing and Mapping) backend, which in real time optimizes the factor graph (Kaess et al. 2012). Unlike batch optimization methods, which recalculate the entire solution with each new measurement, ISAM2 updates just the affected portion of the graph and so maintains efficiency in computational operations that extend over time. The backend manages the incremental factorization of the information matrix and relinearizes nonlinear factors when necessary.

The system manages computational complexity by using a keyframe-based approach that selectively adds vessel poses to the factor graph (Strasdat, Montiel, and Davison 2010). New keyframes are generated only when the vessel has moved a significant distance or rotated beyond specified thresholds. This prevents the graph from growing unnecessarily while ensuring it has sufficient spatial coverage. Each keyframe, then, represents a snapshot of the environment, when associated with the necessary sonar measurements, a keyframe serves as the basis for both map construction and the detection of loop closures.

Crucially, the long-term accuracy of the mapping system relies on loop closure detection. Revisited locations are identified when the system matches the current pose against historical keyframes. The matches are based on spatial proximity and orientation similarity. When a clear match is made, a new constraint is added to the factor graph, tying the current pose to the historical pose with a relatively tight uncertainty model. These loop closure constraints makes the optimization to correct accumulated drift throughout the trajectory, maintaining global consistency in both the vessel's path and the resulting map.

The probabilistic occupancy grid map is created by integrating sonar measurements according to the optimized vessel trajectory (Elfes 1989). Each cell in the grid indicates a small area of the environment and maintains a log-odds value depicting the prob-

ability of occupancy. When sonar data is processed, cells corresponding to detected obstacles are marked as occupied, while cells that are along the ray between the vessel and obstacles are marked as free using a modified Bresenham's line algorithm. This approach creates a consistent environmental representation that creates differences between obstacles, navigable areas, and unexplored regions.

Specific accommodations for sonar characteristics encompass adaptive cell marking strategies that are based on measurement confidence and specialized handling of beam patterns to account for the wide aperture typical in sonar sensors. The sonar map created from the simulation environment is shown in Figure 8.3

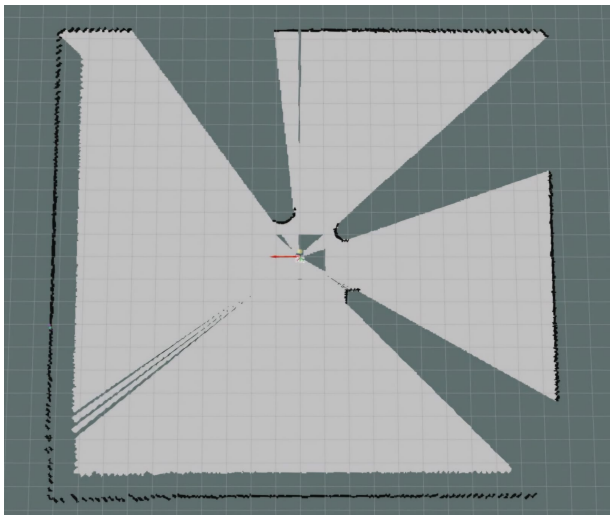


Figure 8.3: SLAM Map of a simulated environment

In simulation settings, the implementation yielded encouraging outcomes with decent mapping precision and pose assessment under regulated conditions.The system, however, may encounter challenges in real-world deployment scenarios, especially with complex sonar noise patterns, rapidly changing environments, and computational limitations on embedded platforms. Future work should address these constraints by enhancing feature extraction techniques and creating a more efficient loop closure methods, while building upon the current foundation for autonomous mapping in inland waterways.

9 Evaluation

This chapter provides a thorough evaluation of the proposed LSTM enhanced EKF method for estimating odometry. It is sectioned into three parts, and each part serves to critically assess the performance and limits of the newly proposed method. Initially, a comparative analysis takes place between the LSTM-enhanced EKF and the conventional EKF-only method to evaluate the improvements caused by integrating deep learning techniques into the odometry pipeline. Second, the robustness of the LSTM enhanced odometry under various motion dynamics is assessed by comparing its precision to ground truth data across various velocity profiles. Third, the quality of the SLAM maps generated by both techniques are compared with a reference map, offering valuable information about how well they work in actual mapping situations. Each of these assessment phases contributes to a comprehensive understanding of the suggested method's capabilities by providing contrasting viewpoints on its performance (Kümmerle et al. 2009). The metrics and datasets employed in these evaluations are carefully selected to cover a diverse range of operational conditions relevant to autonomous navigation.

9.1 Evaluation Methodology

This section explains the methodology used to conduct this comprehensive assessment. The evaluation is sectioned into parts aimed at critically assessing the performance and limits of the proposed method.

9.1.1 Data Collection Process

Different data collection techniques that were specific to each assessment goal were used for our three-part evaluation.

For the comparative analysis of LSTM enhanced EKF versus EKF-only odometry, the system was setup to continuously process sensor inputs through both pipelines, ensuring they received same input data streams. This method allowed direct performance comparison and eliminated any differences resulting from sensor reading variations. The vessel was guided through trajectories while recording odometry estimates from

both methods alongside ground truth data.

For the speed-dependent performance evaluation, data specifically designed to isolate velocity as a variable was collected. Three distinct datasets were recorded with the vessel operating at controlled speeds of 1 m/s, 2 m/s, and 3 m/s, while maintaining similar trajectory patterns. This setup allowed to directly investigate how increasing velocity affects the LSTM enhanced EKF performance, independent of other factors.

For the map quality assessment, three distinct maps using identical sensor data was generated: one with the EKF-only odometry, one with LSTM enhanced EKF odometry, and a reference ground truth map. By maintaining consistency in the mapping algorithm and input data, any differences in map quality could be attributed solely to the odometry method employed.

9.1.2 Evaluation Metrics

To comprehensively evaluate the odometry estimation method, two complementary error metrics along with statistical analysis were used.

Absolute Position Error (APE)

At each timestamp, APE determines the direct Euclidean distance between the estimated positions and the ground truth.

$$APE(t) = |p_{est}(t) - p_{gt}(t)| \tag{9.1.1}$$

Where $p_{est}(t)$ is the estimated position and $p_{gt}(t)$ is the ground truth position at time t. With no trajectory alignment, this metric measures the accuracy of the instantaneous positioning.

Absolute Trajectory Error (ATE)

ATE first uses a rigid transformation to align the estimated path with ground truth in order to assess the overall trajectory's global consistency:

$$ATE = \min_{T \in SE(3)} \sum_{i=1}^{n} |T \cdot p_{est}(t_i) - p_{gt}(t_i)|^2 \tag{9.1.2}$$

Where T is the rigid transformation that minimizes the overall error. As ATE prioritizes trajectory shape preservation over precise positioning, it is especially useful for SLAM applications. (Zhang and Scaramuzza 2018).

Statistical Measures

The error distributions for both APE and ATE are thoroughly characterized using the following statistical measures:

Root Mean Square Error (RMSE): RMSE is sensitive to outliers because it assigns more weight to larger errors. In navigation applications, this feature may be vital for detecting techniques that occasionally result in large positioning errors.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{9.1.3}$$

Mean Error: A measure of central tendency that represents average performance is provided by the arithmetic average.

$$Mean = \frac{1}{n}\sum_{i=1}^{n} e_i \tag{9.1.4}$$

Median Error: A robust measure of central tendency that is less affected by outliers than the mean is provided by the median, which is the middle value when all errors are arranged.

$$Median = e_{\frac{n+1}{2}} \text{ when sorted} \tag{9.1.5}$$

Standard Deviation: By measuring error dispersion, the standard deviation provides insight into the reliability and consistency of positioning estimates.

$$StdDev = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(e_i - Mean)^2} \tag{9.1.6}$$

For map quality assessment, additional specialized metrics were utilized:
F1 Score: The map's ability to accurately identify occupied areas while minimizing false positives and negatives is assessed using the harmonic mean of precision and recall.

Mean Squared Error (MSE): Calculates the average squared difference between the ground truth and predicted occupancy values.

Root Mean Squared Error (RMSE): The average magnitude of error, expressed in the same units as the original data, is provided by the square root of MSE.

These thorough statistical analysis allows to not only compare average performance but also understand error distributions, consistency, and reliability bounds for both odometry methods.

9.2 Comparison of LSTM enhanced EKF and EKF-only Odometry

This section presents a direct comparison of the odometry estimation performance between the proposed LSTM enhanced Extended Kalman Filter (EKF) method and a conventional EKF-only approach

9.2.1 Case 1: Evaluation Results

The first evaluation dataset majorly represents linear motion. Figure 9.1 shows the trajectory comparison between ground truth, LSTM enhanced EKF, and EKF only approaches.

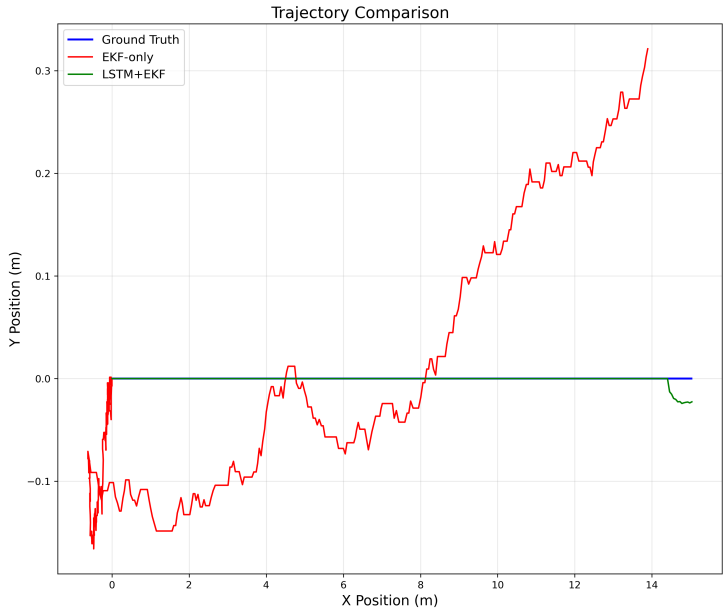


Figure 9.1: Case 1:Trajectory comparison between ground truth, LSTM+EKF, and EKF-only approaches

As shown in Figure 9.1, the EKF only trajectory gradually departs from ground truth; in the later portions of the path, errors surpass two meters. Even after prolonged travel, the LSTM enhanced EKF trajectory remains remarkably aligned with ground truth, with deviations staying under 0.4 meters. This graphic comparison demonstrates how the drift accumulation that impacts conventional odometry techniques is successfully avoided by the LSTM enhanced EKF approach.

Table 9.1 presents the detailed statistical measures for both APE and ATE metrics, comparing EKF-only and LSTM enhanced EKF approaches.The LSTM enhanced EKF approach significantly improves this dataset, reducing the ATE RMSE by 84.01% and the APE RMSE by 83.64%. All statistical measures show these notable improvements, but the standard deviation (87.64% for APE) and maximum error (83.60% for APE)

Table 9.1: Case 1: Error Metrics

Metric	Method	RMSE (m)	Mean (m)	Median (m)	Std (m)
APE	EKF-only	1.554287	1.382257	1.437374	0.710755
	LSTM+EKF	0.254324	0.238674	0.277371	0.087835
	Improvement	83.64%	82.73%	80.70%	87.64%
ATE	EKF-only	1.549128	1.377103	1.423218	0.709498
	LSTM+EKF	0.247692	0.233223	0.271034	0.083416
	Improvement	84.01%	83.06%	80.96%	88.25%

show especially striking decreases.

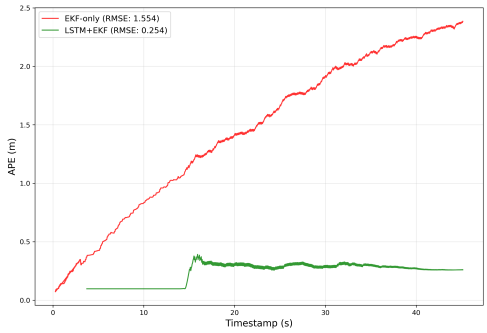


Figure 9.2: Absolute Position Error (APE) over time for case 1 dataset, comparing LSTM+EKF and EKF-only approaches

Figure 9.2 displays the Absolute Position Error (APE) for both approaches on the case 1 dataset over time. With the APE gradually rising as the trajectory advances, the plot clearly shows how the EKF-only method accumulates error over time. On the other hand, even after prolonged operation, the LSTM enhanced EKF method consistently maintains low error over the course of the trajectory, showing no apparent drift.

9.2.2 Case 2: Evaluation Results

The second evaluation dataset includes more complex navigation scenarios mainly circular motion. Figure 9.3 shows the trajectories for this dataset.
Figure 9.3 indicates that both approaches face more difficulties when dealing with the more intricate maneuvers in this dataset. Significant distortion can be seen in the EKF-only trajectory, especially when turning and on longer path segments. Even though there are more obvious deviations than in case 1, the LSTM enhanced EKF trajectory still maintains a significantly higher degree of alignment with ground truth than EKF-only.

Table 9.2 presents the detailed error metrics for this more challenging dataset.

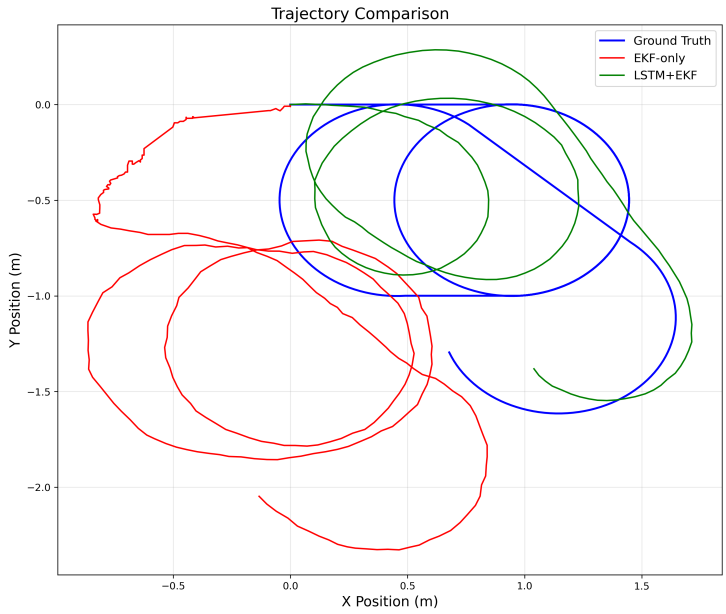


Figure 9.3: Case 2: Trajectory comparison between ground truth, LSTM+EKF, and EKF-only approaches

Table 9.2: Case 2: Error Metrics

Metric	Method	RMSE (m)	Mean (m)	Median (m)	Std (m)
APE	EKF-only	1.826196	1.696600	1.937658	0.675676
	LSTM+EKF	0.416429	0.377026	0.372314	0.176818
	Improvement	77.20%	77.78%	80.79%	73.83%
ATE	EKF-only	1.831249	1.705815	1.940865	0.666085
	LSTM+EKF	0.420032	0.380526	0.370664	0.177840
	Improvement	77.06%	77.69%	80.90%	73.30%

Even though case 2's improvement percentages are marginally lower than case 1's, they are still significant, with ATE RMSE and APE RMSE decreased by 77.20% and 77.06%, respectively. Although the performance improvement over EKF-only is still significant, the slightly larger error magnitudes seen in case 2 compared to case 1 for the LSTM enhanced EKF approach (APE RMSE of 0.416429m vs. 0.254324m) indicate that more complex motion patterns remain difficult to handle even with the enhanced method.

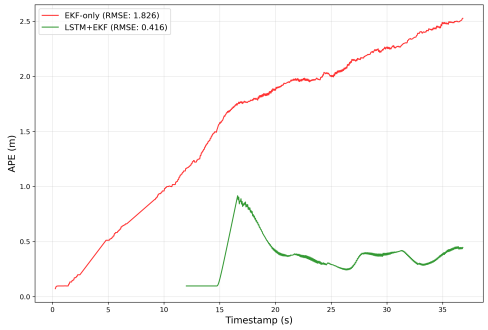


Figure 9.4: Absolute Position Error (APE) over time for case 2 dataset, comparing LSTM+EKF and EKF-only approaches

Figure 9.4 displays the APE for the more difficult case 2 dataset over time. While the EKF-only approach shows significant drift, the LSTM enhanced EKF approach maintains significantly lower error throughout, much like case 1. But compared to case 1, case 2's LSTM enhanced EKF error shows a little more variability, indicating that the trajectory is more complex because of its frequent turns and speed variations.
With particularly impressive results in environments with predictable motion patterns, these results show that the LSTM enhanced EKF approach consistently and significantly enhances navigation in a range of scenarios.

9.3 Performance Evaluation at Varying Speeds

A thorough analysis was carried out across three speed regimes—1 m/s, 2 m/s, and 3 m/s—to evaluate the robustness of our LSTM enhanced EKF approach under various motion dynamics. Testing at 3 m/s was done to understand performance boundaries and high-speed limitations, even though the approach was mainly trained on data from the 1-2 m/s range.

Table 9.3: LSTM+EKF APE RMSE at Different Speeds

Speed	LSTM+EKF RMSE (m)
1 m/s	0.327511
2 m/s	0.436407
3 m/s	0.759705

As illustrated in Table 9.3 and illustrated in Figure 9.5, the LSTM enhanced EKF approach clearly exhibits a pattern of increasing error with higher speeds. With an APE RMSE of 0.327511 m, our method achieves excellent accuracy at 1 m/s. Despite doubling the speed, this performance is still strong at 2 m/s with an RMSE of 0.436407 m, indicating only a 33.25% increase in error. We find a more significant decrease in performance at 3 m/s, with the RMSE increasing to 0.759705 m, a 74.08% increase over the 2 m/s scenario. Given that our training data was concentrated on the more realistic 1-2 m/s range, this suggests that higher speeds pose more difficulties for the LSTM enhanced EKF approach.

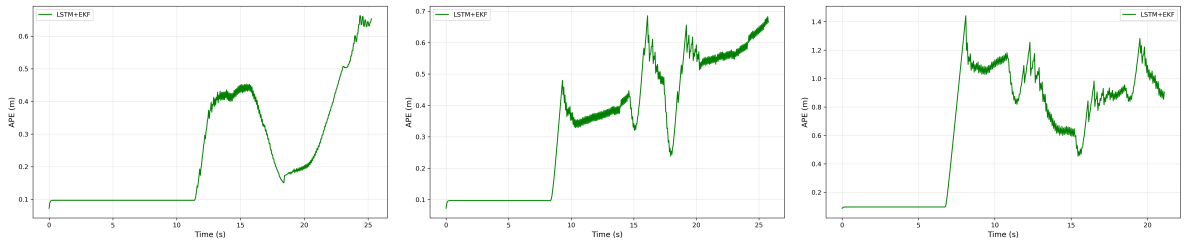


Figure 9.5: LSTM+EKF performance across different speeds: 1 m/s (left), 2 m/s (middle), and 3 m/s (right)

Table 9.4: Detailed LSTM+EKF APE Metrics Across Speed Ranges

Speed	Mean (m)	Median (m)	Std (m)
1 m/s	0.280333	0.237842	0.169342
2 m/s	0.396391	0.397063	0.182552
3 m/s	0.690641	0.710337	0.270000

A more thorough analysis of the error metrics in Table 9.4 provides more details on the performance traits at various speeds. The method performs exceptionally well at 1 m/s, with a median error of 0.237842 m and a mean error of 0.280333 m. The extremely consistent positioning estimates are indicated by the low standard deviation (0.169342 m). All error metrics show a moderate increase at 2 m/s, with the median error increasing to 0.397063 m (66.94% increase) and the mean error increasing to 0.396391 m (41.40% increase). Even though overall accuracy slightly declines, positioning consistency is still strong, as evidenced by the standard deviation, which only slightly increases to 0.182552 m (7.80% increase).

All error metrics show a more noticeable increase at 3 m/s. The mean error rises to 0.690641 m (74.23% increase from 2 m/s), while the median error rises to 0.710337 m (78.90% increase). The standard deviation increases by 47.90% to 0.270000 m, indicating less consistent positioning performance. This shows that high speed operation leads to higher worst case errors, even though the increase is not as large as one might expect given the significant speed change.

The observed speed-dependent performance has several important practical implications. Our LSTM enhanced EKF approach clearly demonstrates optimal performance at low to moderate speeds (1-2 m/s), achieving sub-half-meter accuracy fit for precise navigation tasks in indoor and urban environments. Despite the fact that the system was primarily trained on lower-speed data, it still maintains sub-meter average precision (mean error of 0.690641 m) at 3 m/s, even though accuracy decreases.

9.4 SLAM Map Quality Comparison

To evaluate the impact of improved odometry on mapping applications, three distinct maps were produced: one with LSTM enhanced EKF odometry, one with EKF-only odometry, and a reference ground truth map. All maps were created using an occupancy grid representation that covered the same physical areas and was aligned to a common reference frame in order to enable direct comparison.

9.4.1 Visual Map Comparison

Figure 9.6 presents a side-by-side comparison of the ground truth map alongside the maps produced using LSTM+EKF and EKF-only odometry.

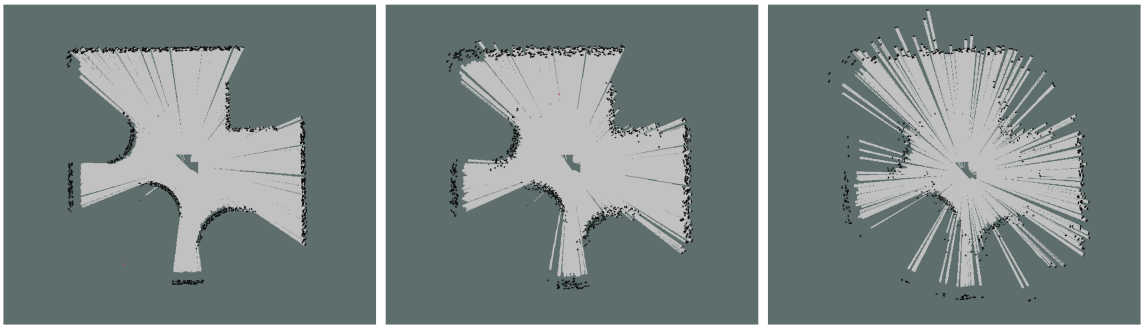


Figure 9.6: Comparison of maps generated using ground truth (left), LSTM+EKF (middle), and EKF-only odometry (right)

The superiority of the LSTM enhanced EKF map over the EKF-only map is confirmed by visual analysis. The LSTM enhanced EKF approach exhibits noticeably better wall alignment and feature preservation with more accurate representations of walls and obstacles.Difference maps demonstrate that LSTM improved EKF errors are more localized and smaller than EKF-only errors, which are larger and more common.

9.4.2 Quantitative Map Comparison

To ensure a comprehensive assessment and cover various aspects of map quality, a number of complementary measures were employed, as shown in Table 9.5.
The findings show that the LSTM enhanced EKF model performs better than the EKF-only model on every metric:

Table 9.5: Comparison of Map Quality Metrics

Metric	GT vs LSTM+EKF	GT vs EKF-only	Improvement (%)
F1	0.9757	0.9459	3.2%
MSE	0.009614	0.019504	50.7%
RMSE	0.098049	0.139657	29.8%

With an F1 score of 0.9757 compared to 0.9459 for the EKF-only map, the LSTM enhanced EKF map shows a better balance between recall and precision when identifying occupied and free space. The F1 score has improved by 3.2%.

With an MSE of 0.009614 compared to 0.019504 for the EKF-only map (50.7% reduction) and an RMSE of 0.098049 compared to 0.139657 (29.8% reduction), the LSTM enhanced EKF map significantly reduces error metrics. This shows that the method creates more accurate cell-level maps.

These enhancements result from the more accurate position estimates provided by the LSTM enhanced EKF, which lower drift and improve spatial accuracy. Better mapping quality leads to practical benefits like increased operational safety, resource economy, better path planning, and improved localization.

9.5 Discussion and Limitations

The evaluation results show that the LSTM enhanced EKF approach performs better than traditional EKF-only methods in a variety of metrics. Among its primary benefits are efficient drift reduction, dependable error performance, enhanced map feature preservation, practical real-world applicability at normal operating speeds, and graceful degradation in challenging circumstances.

However, it is crucial to be aware of several restrictions. The method's performance deteriorates at higher speeds (3 m/s) due to a lack of high-speed training data. Additionally, performance is somewhat reduced in complex navigation scenarios with frequent directional changes. Any learning-based approach's effectiveness depends on the distribution of its training data, and our system performed best in the 1-2 m/s range that dominated our training set.

Future research could overcome these limitations by means of expanded training datasets, adaptive confidence weighting between LSTM and EKF components, online learning capabilities, and improved environmental adaptation mechanisms. These improvements could further increase the system's adaptability to different operating environments.

10 Conclusion

This thesis introduces an innovative approach to improve the navigation precision in environments where a Global Navigation Satellite System fails for autonomous vessels. The approach combines the use of deep learning with traditional sensor fusion and SLAM methods. The research centers on improving odometry estimation by the application of deep learning technique known as Long Short-Term Memory (LSTM) networks to enhance data from Inertial Measurement Units (IMUs) and Doppler Velocity Logs (DVLs).

10.1 Summary of Contributions

The research has made multiple impactful contributions to autonomous vessel navigation in environments lacking GNSS:

First, we developed a comprehensive simulation environment using ROS 2 and Gazebo that realistically models vehicle dynamics and sensor behaviors.This environment facilitated data collection, algorithm development, and the thorough evaluation of performance (Macenski et al. 2022).

Second, we designed and implemented LSTM-based sensor enhancement models for IMU angular velocity and DVL linear velocity measurements (Hochreiter and Schmidhuber 1997).These models displayed wonderful performance in rectifying sensor errors. R^2 values surpassed 0.93 for five of the six targeted sensor components. The models learned the intricate patterns in the behaviors of the sensor's errors and made substantial improvements in measurement accuracy.

Third, we integrated the enhanced sensor outputs with an Extended Kalman Filter (EKF) based sensor fusion approach (Moore and Stouch 2013). Substantial improvements were shown by the LSTM enhanced EKF over traditional EKF-only methods. The EKF-LSTM reduced position error by up to 83.64% and performed consistently across different operational scenarios.

Fourth, we evaluated the SLAM performance with the enhanced odometry. The approach demonstrated clear gains in map quality metrics compared to traditional methods, with a 50.7% reduction in Mean Squared Error (MSE) and a 29.8% reduction in Root Mean Squared Error (RMSE) when compared to maps created using EKF-only odometry.

10.2 Key Findings

A thorough assessment showed some key results:

The LSTM enhanced odometry consistently outperformed approaches that only used traditional EKF across all testing scenarios. This performance advantage held up under various operational conditions and was particularly impressive in environments where the motion patterns were predictable.

The method not only surpasses the average accuracy of other systems, but also demonstrates more consistent performance with lower standard deviations in error metrics. This reliability makes the system suitable for use in autonomous vehicles.

The approach yielded a performance that is evidently related to vehicle speed. At practical operating speeds (1-2 m/s), the system maintained nearly ideal accuracy, with sub-half-meter position accuracy. Higher speeds like 3 m/s led to some degradation in performance, though the system still maintained sub-meter average accuracy.

The improved odometry directly resulted a enhanced mapping capability, with notable improvements in map quality metrics. This proves that better odometry leads to more reliable environmental representations, which is important for autonomous navigation and path planning.

10.3 Limitations and Challenges

Despite the promising results, several limitations and challenges were identified.

10.3.1 Technical Limitations

Data Dependency: The performance of the LSTM models is much conditioned on the quality and distribution of the training data. The approach showed optimal performance within the speed range (1-2 m/s) represented in the training data, with less effectiveness at higher speeds.

Computational Requirements: The model based enhancement brings additional computational overhead compared to existing traditional methods. While it justifies its use with the performance improvements, this could be a consideration during deployment on platforms where computational resources are limited.

Model Generalization: The present application may encounter difficulties in extrapolating to sharply distinct vessel kinds or to drastically different working conditions from those found in the training data.

Sensor Limitations: The methodology relies on the basic functions of the fundamental sensors beneath it. It can supplement the basic readings of those sensors if they are operating, but it cannot make up for a sensor that has failed in a fundamental way or that has degraded to an extreme level.

10.3.2 Implementation Challenges

Calibration and Training: To implement the system on a new vessel, the appropriate training data would have to be collected, and the LSTM models may need to be retrained to account for specific vessel dynamics and sensor characteristics.

Parameter Tuning: Both LSTM model hyperparameters and EKF component hyperparameters represent options that must be selected or tuned before the model can be demonstrated. The hyperparameters have a range that is optimal for different vessels and different kinds of operating environment. Thus, a systematic process of tuning them for hyperparameter selection is required.

Integration Complexity: Combining the improved odometry system with current navigation frameworks takes careful attention to coordinate transformations, timing synchronization, and data flow management.

Real-time Performance: Although the simulation showed real time capability, deployment on physical hardware may need optimization to ensure that processing delays do not negatively impact performance of the navigation system.

10.4 Future Work

Based on the findings and limitations identified in this research, This research identifies several potential avenues for future work.

10.4.1 Technical Enhancements

Extended Training Regimes: The dataset used for training can be further enlarged by including more variety of motion patterns, higher speeds, and varying environmental conditions could improve the ability of the system to work efficiently across a broader operational envelope.

Adaptive Confidence Weighting: Dynamic weighting between LSTM and EKF components can be implemented based on motion characteristics and sensor reliability met-

rics. This could optimize performance across many different conditions.

Online Learning Capabilities: Online adaptation mechanisms incorporated into the system would help it keep improving during operations, lessen its reliance on pre-training, and sharpen its adaptability to fresh environments.

Multi-modal Sensor Fusion: Incorporating other sensing modalities, such as vision-based systems, into the approach could further enhance navigation robustness in difficult environments.

10.4.2 Real-world Implementation

Several main steps would be needed to actualize the LSTM-enhanced navigation system in real-world autonomous ships.

Hardware Adaptation: The specific configurations of sensors and computational resources on targeted vessels would necessitate adaptations of the system. These adaptations could involve either optimizing the architecture of the neural network or implementing hardware acceleration.

Robust Error Handling: Deploying in the real world would need comprehensive error detection and handling mechanisms to ensure safety in the event of sensor failures or unexpected environmental conditions.

Environmental Testing: To validate performance in real-world conditions, systematic testing would be required across a range of maritime environments. These include varying water conditions,weather patterns and operational scenarios.

Training Data Acquisition: To train the models, accurate ground truth data need to be collected to pair with raw sensor readings. In places where GNSS is available, we could collect training datasets by recording sensor data alongside GNSS positional data, then use these learned patterns to navigate when GNSS is unavailable. For the types of environments where GNSS is permanently unavailable, temporary, laser-based tracking systems or landmark-based approaches with surveyed reference points could provide the necessary ground truth during training. Sim-to-real transfer techniques could also reduce the amount of real-world data if paired data of real and simulated sensor responses could be gathered from similar, but not identical, scenarios (Tobin et al. 2017). In either approach,the ground truth is only needed to train the models; once trained, the vessel would navigate using just its onboard sensors and the enhancement models.

Human-Machine Interface: Creating appropriate interfaces for human supervision and intervention will be crucial for safe and effective operation in mixed autonomy maritime environments.

10.4.3 Broader Applications

This research has developed techniques that could be applied in many areas beyond autonomous vessels.

Underwater Vehicles: The enhanced odometry method could be tailored for use in underwater vehicles working in places where both GNSS and acoustic positioning systems are unavailable.

Indoor Navigation: Identical techniques might be utilized for terrestrial robots functioning in GNSS-denied indoor settings, especially in situations where conventional visual or LiDAR-based methods are challenging.

Aerial Vehicles: This strategy may aid aerial platforms trying to navigate in environments with unreliable GNSS, such as urban canyons or indoor spaces.

10.5 Concluding Remarks

This research has shown that deep learning methods, particularly LSTM networks, can increase the precision of odometry estimation for autonomous vessels working in GNSS-denied environments. When combined with traditional sensor fusion methods, these new techniques yield a far more reliable and accurate navigation solution, which serves a dual purpose: solving a pressing problem in maritime autonomy and demonstrating the potential of deep learning to revolutionize sensor fusion.

The thorough assessment carried out in our simulation environment gives us plenty of strong evidence to show that this approach is likely to work in the real world, even if some aspects make it less than ideal for certain kinds of autonomous vessels. On the other hand, the comprehensive evaluation has also hinted at some important considerations that we need to iron out before this approach can make the leap to real vessel.

The path to completely autonomous vessels is still hard and needs developments in safety systems, regulatory frameworks, human-machine cooperation, and navigation technology. However, this study is a significant step in addressing one of the fundamental problems of maintaining accurate positioning in the absence of GNSS signals and sets the foundation for future advancements in this rapidly evolving sector.

List of Figures

List of Tables

Bibliography

Ba, J. L., J. R. Kiros, and G. E. Hinton (2016): "Layer Normalization". In: *arXiv preprint arXiv:1607.06450*.

Caballero, F., L. Merino, J. Mancisidor, and A. Ollero (2009): "Vision-Based Odometry and SLAM for Medium and High Altitude Flying UAVs". In: *Journal of Intelligent & Robotic Systems* 54.1-3, pp. 137–161. DOI: 10.1007/s10846-008-9234-z.

Christ, R. D. and R. L. W. Sr (2007): *The ROV Manual: A User Guide for Observation-Class Remotely Operated Vehicles*. Boston, MA: Elsevier Butterworth-Heinemann.

Cohen, N. and I. Klein (2023): *Inertial Navigation Meets Deep Learning: A Survey of Current Trends and Future Directions*. DOI: 10.48550/arXiv.2307.00014. arXiv: 2307.00014 [cs.RO].

Deep Learning Basics Part 8: Convolutional Neural Network — Sasirekha Rameshkumar (Apr. 2025). Medium Blog Post. URL: https://medium.com/@sasirekharameshkumar/deep-learning-basics-part-8-convolutional-neural-network-cnn-4cff567bad46.

Elfes, A. (1989): "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6, pp. 46–57. DOI: 10.1109/12.24247.

Folkesson, J., J. Leonard, J. Leederkerken, and R. Williams (2007): "Feature Tracking for Underwater Navigation Using Sonar". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2959–2964. DOI: 10.1109/IROS.2007.4399486.

Foote, T. (2013): "tf: The transform library". In: *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pp. 1–6. DOI: 10.1109/TePRA.2013.6556373.

Fossen, T. I. (2011): *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley Series on Ocean Engineering. John Wiley & Sons.

Fukuda, G., D. Hatta, X. Guo, and N. Kubo (2021): "Performance Evaluation of IMU and DVL Integration in Marine Navigation". In: *sensors* 21.4, p. 1196. DOI: 10.3390/s21041196.

Gal, Y. and Z. Ghahramani (2016): "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*, pp. 1019–1027.

Gu, Y. and S. W. Wallace (2021): "Operational benefits of autonomous vessels in logisticsâ€"A case of autonomous water-taxis in Bergen". In: *Transportation Research Part E* 154, p. 102456. DOI: 10.1016/j.tre.2021.102456.

Hochreiter, S. and J. Schmidhuber (1997): "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Kaess, M., H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert (2012): "iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree". In: *The International Journal of Robotics Research* 31.10, pp. 1223–1235. DOI: 10.1177/0278364911430415.

Kinsey, J. C., R. M. Eustice, and L. L. Whitcomb (2006): "A Survey of Underwater Vehicle Navigation: Recent Advances and New Challenges". In: *IFAC Proceedings Volumes* 39.15. Proceedings of the 7th IFAC Conference on Manoeuvring and Control of Marine Craft, pp. 1–12.

Koenig, N. and A. Howard (2004): "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3, pp. 2149–2154. DOI: 10.1109/IROS.2004.1389727.

Kümmerle, R., B. Steder, C. Dornhege, G. Grisetti, M. Ruhnke, C. Stachniss, and A. Kleiner (2009): "On Measuring the Accuracy of SLAM Algorithms". In: *Autonomous Robots* 27.4, pp. 387–407. DOI: 10.1007/s10514-009-9156-z.

Längkvist, M., L. Karlsson, and A. Loutfi (2014): "A review of unsupervised feature learning and deep learning for time-series modeling". In: *Pattern Recognition Letters* 42.1, pp. 11–24. DOI: 10.1016/j.patrec.2014.01.008.

Macenski, S., T. Foote, B. Gerkey, C. Lalancette, and J. Langsfeld (2022): *Robot Operating System 2: Design, Architecture, and Uses In The Wild*. arXiv: 2211.07752 [cs.RO].

Mallios, A., P. Ridao, D. Ribas, and E. Hernández (2014): "Scan Matching SLAM in Underwater Environments". In: *Autonomous Robots* 36.1-2, pp. 1–18. DOI: 10.1007/s10514-013-9359-5.

Manhães, M. M. M., S. A. Scherer, M. Voss, and L. R. Douat (2016): "UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation". In: *OCEANS 2016 MTS/IEEE Monterey*, pp. 1–8. DOI: 10.1109/OCEANS.2016.7761080.

McKinney, W. (2017): *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. 2nd. O'Reilly Media.

Moore, T. and D. Stouch (2013): "A Generalized Extended Kalman Filter Implementation for the Robot Operating System". In: *International Advanced Robotics Programme's 7th International Workshop on Robotics for Risky Environments Extreme Robotics (IARP RISE-ER)*, pp. 1–7.

Moreira, L., R. Vettor, and C. Guedes Soares (2021): "Neural Network Approach for Predicting Ship Speed and Fuel Consumption". In: *Journal of Marine Science and Engineering* 9.2, p. 119. DOI: 10.3390/jmse9020119.

Nair, V. and G. E. Hinton (2010): "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814.

New to subsea navigation? — Nortek (Apr. 2025). URL: https://www.nortekgroup.com/knowledge-center/wiki/new-to-subsea-navigation.

Pascanu, R., T. Mikolov, and Y. Bengio (2013): "On the difficulty of training Recurrent Neural Networks". In: *Proceedings of the 30th International Conference on Machine Learning*, pp. 1310–1318.

Paull, L., S. Saeedi, M. Seto, and H. Li (2014): "AUV Navigation and Localization: A Review". In: *IEEE Journal of Oceanic Engineering* 39.1, pp. 131–146. DOI: 10.1109/JOE.2013.2278891.

Pazarci, U. (2024): "SLAM with Sonar Data for Autonomous Inland Vessels". Master Thesis. Dortmund, North Rhine-Westphalia, Germany: Technical University Dortmund.

Ping360 Sonar — bluerobotics.com (Apr. 2025). URL: https://bluerobotics.com/store/sonars/imaging-sonars/ping360-sonar-r1-rp/.

Quigley, M., B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng (2009): "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*.

Ribas, D., P. Ridao, J. Neira, and J. D. Tardós (2006): "SLAM Using an Imaging Sonar for Partially Structured Underwater Environments". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1776–1781. DOI: 10.1109/IROS.2006.282514.

Ribas, D., P. Ridao, J. D. Tardós, and J. Neira (2014): "Underwater SLAM in Man-Made Structured Environments". In: *Underwater Robotic Manipulation*. Ed. by H.-S. Shin and K.-H. Sohn. Intelligent Systems, Control and Automation: Science and Engineering. Springer, pp. 13–38. DOI: 10.1007/978-3-319-08338-4_2.

Richards, M. A., ed. (2022): *Fundamentals of Radar Signal Processing*. Ed. by M. A. Richards. Third.

Rohling, H. (1983): "Radar CFAR Thresholding in Clutter and Multiple Target Situations". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-19.4, pp. 608–621. DOI: 10.1109/TAES.1983.309350.

Schmidhuber, J. (2015): "Deep learning in neural networks: An overview". In: *Neural Networks* 61, pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

Schuster, M. and K. K. Paliwal (1997): "Bidirectional Recurrent Neural Networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

Sendobry, A., S. Kohlbrecher, J. Meyer, P. Geyer, C. von Drach, U. Klingauf, and J. RoBmann (2012): "Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo". In: *Lecture Notes in Computer Science*. Vol. 7538, pp. 548–562. DOI: 10.1007/978-3-642-34327-8_36.

Simon, D. (2006): *Optimal State Estimation: Kalman, H, and Nonlinear Approaches*. Hoboken, New Jersey: John Wiley & Sons.

Snyder, J. (2010): "Doppler Velocity Log (DVL) navigation for observation-class ROVs". In: *OCEANS 2010 MTS/IEEE SEATTLE*. DOI: 10.1109/OCEANS.2010.5664561.

Strasdat, H., J. Montiel, and A. J. Davison (2010): "Real-time Monocular SLAM: Why Filter?" In: *2010 IEEE International Conference on Robotics and Automation*, pp. 2565–2572. DOI: 10.1109/ROBOT.2010.5509965.

Thrun, S., M. Montemerlo, H. Dahlkamp, D. Hähnel, W. Burgard, A. Angermann, D. Fox, W. Whittaker, and J. Leonard (2006): "GraphSLAM: A unifying framework for online and offline SLAM". In: *Robotics: Science and Systems*.

Tobin, J., R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel (2017): "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. DOI: 10.1109/IROS.2017.8202179.

Unlocking the Power of LSTM, GRU, and the Struggles of RNN in Managing Extended Sequences — Murad Atcorvit (Apr. 2025). Medium Blog Post. URL: https://medium.com/@muradatcorvit23/unlocking-the-power-of-lstm-gru-and-the-struggles-of-rnn-in-managing-extended-sequences.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017): "Attention is All you Need". In: *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Westman, E., A. Hinduja, and M. Kaess (2018): "Feature-based SLAM for Imaging Sonar with Under-constrained Landmarks". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 381–388. DOI: 10.1109/ICRA.2018.8460598.

What is LSTM (Long Short-Term Memory)? — MathWorks (Apr. 2025). Website. URL: https://www.mathworks.com/discovery/lstm.html.

Woodman, O. J. (2007): *An introduction to inertial navigation*. Tech. rep. UCAM-CL-TR-696. University of Cambridge, Computer Laboratory.

Wu, Y., H. Tan, L. Qin, B. Ran, and Z. Jiang (2018): "A hybrid deep learning based traffic flow prediction method and its understanding". In: *Transportation Research Part C* 90, pp. 166–180. DOI: 10.1016/j.trc.2018.03.005.

Xsens MTi-100 IMU — movella.com (Apr. 2025). URL: https://www.movella.com/products/sensor-modules/xsens-mti-100-imu.

Yang, C.-H., C.-H. Wu, J.-C. Shao, Y.-C. Wang, and C.-M. Hsieh (2022): "AIS-Based Intelligent Vessel Trajectory Prediction Using Bi-LSTM". In: *IEEE Access* 10, pp. 24933–24946. DOI: 10.1109/ACCESS.2022.3154812.

Ye, X., J. Wang, D. Wu, Y. Zhang, and B. Li (2023): "A Novel Adaptive Robust Cubature Kalman Filter for Maneuvering Target Tracking with Model Uncertainty and Abnormal Measurement Noises". In: *sensors* 23.15, p. 6966. DOI: 10.3390/s23156966.

Yuh, J., G. Marani, and D. R. Blidberg (2011): "Applications of marine robotic vehicles". In: *Intel Serv Robotics* 4, pp. 221–231. DOI: 10.1007/s11370-011-0096-5.

Zhang, Z. and D. Scaramuzza (2018): "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3903–3910. DOI: 10.1109/IROS.2018.8594463.

Eidesstattliche Versicherung

(Affidavit)

GOVINDARAJ , SUSHMITHA

Name, Vorname
(surname, first name)

242969

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

EXTENDED SONAR-BASED SLAM FOR GNSS-FREE LOCALIZATION OF AUTONOMOUS VESSELS

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 19.05.2025

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 19.05.2025

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**