Investigation of Magnetometer-Inertial SLAM for Autonomous Railway Robot Navigation

Master's thesis in Complex Adaptive Systems

Xiaoyue Ma
Vicky Che

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis in Complex Adaptive Systems

Investigation of Magnetometer-Inertial SLAM for Autonomous Railway Robot Navigation

Xiaoyue Ma
Vicky Che

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Investigation of Magnetometer-Inertial SLAM for Autonomous Railway Robot Navigation
Xiaoyue Ma
Vicky Che

Supervisor: Vivien Lacorre, Department of Mechanics and Maritime Sciences
Examiner: Krister Wolff, Department of Mechanics and Maritime Sciences

Master's Thesis 2025
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Investigation of Magnetometer-Inertial SLAM for Autonomous Railway Robot Navigation
Xiaoyue Ma
Vicky Che
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

In GPS-denied railway environments, such as tunnels or underground stations, estimating the speed of a moving platform remains a challenge. This thesis presents a sensor-based approach using magnetometers and inertial measurement units (IMUs) to estimate velocity without relying on external infrastructure.
We first simulate realistic magnetometer and IMU data based on motion profiles, adding noise such as Gaussian jitter, temperature drift, spatial jitter, and random spikes. The synthetic magnetic field is created using a combination of sinusoidal functions to reflect real-world magnetic variations.
Velocity is then estimated by comparing signals from front and rear magnetometers. By measuring the time delay between them, speed can be calculated using the known distance between sensors. The method includes filtering and window-based lag detection for stability under dynamic conditions.
The estimated speed is fused with IMU acceleration using an Extended Kalman Filter (EKF). Results show that IMU-only fusion performs well, but direct use of noisy magnetometer speed may degrade accuracy. To improve this, we propose a complementary filter as a pre-fusion step before EKF.
This work demonstrates a lightweight and effective method for speed estimation in low-GPS or GPS-free railway scenarios. Future work will include improving signal quality, implementing adaptive fusion, and hard-in-loop testing in real robots.

Keywords: Sensor Fusion, Data Simulation, Velocity Estimation, Extended Kalman Filter, Railway Environment.

Preface

This report presents the outcome of our master's thesis project Investigation of Magnetometer-Inertial SLAM for Autonomous Railway Robot Navigation, carried out at the Department of Mechanics and Maritime Sciences at Chalmers University of Technology during the spring of 2025.

Traditional Visual-Inertial Navigation methods face significant challenges in GPS-denied railway environments, especially in tunnels or underground stations. These environments have poor lighting conditions and monotonous visual features, which significantly decrease the performance of camera-based localization systems. Our research explores an innovative solution: using unique magnetic field characteristics generated by railway infrastructure, combined with inertial measurement unit data to implement reliable passive localization.

The core contributions of this project include developing a complete framework for sensor data simulation, designing a velocity estimation method based on the time delay of magnetometer signals, and implementing an Extended Kalman Filter for multi-sensor fusion strategy. Through systematic simulation validation, we demonstrate the effectiveness of the magnetometer-inertial fusion method under low-speed dynamic scenarios, which establishes the theoretical foundation for future railway applications.

This research not only deepens our understanding of sensor fusion technology, but also provides new approaches to address robot navigation problems in infrastructure-independent environments. The challenges faced and insights gained during the project will have important implications for the subsequent implementation and deployment of integrated SLAM systems.

Acknowledgement

We would like to express our sincere gratitude to our supervisor, Professor Krister Wolff , for the continuous guidance, thoughtful feedback, and academic support throughout this thesis project. Their experience and advice have been essential at every stage of the work.
We also warmly thank our supervising assistant, Vivien Lacorre, for the technical insights, helpful suggestions, and support during the development and evaluation phases of this work.
We further thank our project teammates and peers for their collaboration and shared efforts during this research.
We gratefully acknowledge the Department of Mechanics and Maritime Sciences at Chalmers University of Technology for providing the resources and environment that made this thesis possible.
Lastly, we thank our families and friends for their unwavering encouragement and support throughout our studies.

Xiaoyue Ma, Vicky Che, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

GNSS	Global Navigation Satellite System
VIO	Visual- Inertial Odometry
VI-SLAM	Visual-Inertial SLAM
SLAM	Simultaneous localization and mapping
GPS	Global Positioning System
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
SQUID	Superconducting Quantum Interference
NMR	Nuclear Magnetic Resonance
MEMS	Microelectromechanical Systems
DoF	Degrees of Freedom
KF	Kalman Filter
UKF	Unscented Kalman Filter
PF	Particle Filter
MMSE	Minimum Mean Squared Error
CSV	Comma-Separated Values
SNR	Signal-to-Noise Ratio
CF	Complementary Filter
Hz	Hertz
µT	Microtesla

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i	Index for track points and spatial elements
t	Index for continuous time
n	Index for discrete time samples
k	Index for recursive EKF steps
ℓ	Lag candidate in velocity estimation

Parameters

d	Distance between front and rear magnetometers [m]
Δt	Sampling interval [s]
L	Lag search window size
W	Window size for velocity estimation
σ	Standard deviation of Gaussian smoothing filter
σ_d	Noise density for white noise [units/$\sqrt{Hz}$]
σ_b	Random walk coefficient for bias instability
σ_{acc}	Accelerometer noise density [$m/s^2/\sqrt{Hz}$]
σ_{gyro}	Gyroscope noise density [rad/s/$\sqrt{Hz}$]
A_{rail}, A_{wheel}	Vibration amplitudes for rail and wheel [m/s^2]
f_{rail}, f_{wheel}	Vibration frequencies for rail and wheel [Hz]
α	Horizontal damping factor (0.3)
j_{max}	Maximum jerk limit [m/s^3]
a_{max}	Maximum acceleration [m/s^2]

a_{thresh}	Acceleration threshold for velocity estimation $[m/s^2]$
$\mathbf{Q}$	Process noise covariance matrix in EKF
$\mathbf{R}$	Measurement noise covariance matrix in EKF
α_{CF}	Fusion weight in complementary filter ($\in [0, 1]$)

Variables

s	Cumulative distance along the trajectory [m]
κ_{xy}, κ_{xz}	Curvature in the xy- and xz-plane [1/m]
$\|v\|[n]$, $v(t)$	Velocity profile in discrete and continuous time [m/s]
$\mathbf{p}$	Position vector [m]
$\mathbf{v}$	Velocity vector [m/s]
$\mathbf{a}$	Acceleration vector $[m/s^2]$
$\boldsymbol{\omega}$	Angular velocity vector [rad/s]
$\mathbf{x}_k$	State vector at time step k
v_k	Estimated velocity at time step k [m/s]
b_k	Estimated accelerometer bias at time step k $[m/s^2]$
a_k^{meas}	Measured acceleration from IMU $[m/s^2]$
z_k	Observation vector at time step k
v_k^{mag}	Velocity observation from magnetometer [m/s]
$B_x[n]$, $B_y[n]$, $B_z[n]$	Simulated magnetic field components $[\mu T]$
$\tilde{B}_x[n]$	Noisy magnetic field signal along x-axis $[\mu T]$
B_{total}	Total magnetic field magnitude $[\mu T]$
$\tau[n]$	Estimated lag (in samples) at time n
$\hat{v}[n]$	Estimated velocity from lag-based correlation [m/s]
$\mathbf{e}_t$, $\mathbf{e}_n$, $\mathbf{e}_b$	Tangential, normal, and binormal direction vectors
$\mathbf{F}_k$	State transition matrix in EKF
$\mathbf{H}_k$	Observation matrix in EKF
$\mathbf{K}_k$	Kalman gain matrix
$\mathbf{P}_k$	Estimation error covariance matrix
$\mathbf{R}$	Rotation matrix for coordinate transformation
ψ, θ, ϕ	Euler angles (yaw, pitch, roll) [rad]

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Due to the limitations of traditional sensors such as cameras and GNSS, the autonomous navigation in railway environments presents many challenges. Visual-Inertial Odometry (VIO) and its extension, Visual-Inertial SLAM (VI-SLAM), are commonly used in robotics for pose estimation and mapping. However, these methods struggle in visually degraded or feature-poor environments, such as tunnels, low-light areas, or long stretches of uniform railway tracks.

Magnetometers, which can detect variations in magnetic flux density along the railway infrastructure, offer a promising alternative [1]. The distinct magnetic signatures found in railway environments, caused by the track materials and surrounding electromagnetic interference, provide opportunities for passive localization even in GPS-denied conditions. By integrating magnetometer and inertial measurement data, a more robust SLAM system can be developed to enhance navigation performance in challenging railway scenarios [2].

1.2 Purpose

The purpose of this thesis is to investigate the feasibility of magnetometer-inertial sensor fusion for improving localization accuracy in railway environments. This research focuses specifically on the sensor fusion component—particularly addressing velocity estimation challenges in low-speed robotic motion. In such scenarios, traditional fusion methods often suffer from drift and temporal misalignment, leading to degraded localization performance. To overcome these problems, we developed a new sensor fusion strategy, incorporating Extended Kalman Filter (EKF) techniques to more accurately integrate magnetometer and IMU data [3]. This enhanced fusion approach significantly improves velocity estimation precision and thereby strengthens the robustness and potential performance of SLAM systems operating under GPS- and vision-denied conditions, such as tunnels and visually repetitive tracks [4].

1.3 Goals

The primary goal of this thesis is to design, implement, and evaluate a novel magnetometer-inertial sensor fusion algorithm tailored for low-speed railway robot navigation. In such conditions, traditional fusion methods often introduce significant drift, resulting in inaccurate velocity estimation. To address this, the project focuses on:

- Modeling and simulating realistic sensor data, including various noise characteristics, motion profiles, and environmental magnetic field disturbances typical of railway scenarios.
- Developing a new fusion approach that effectively integrates IMU and magnetometer data using an Extended Kalman Filter (EKF)-based framework, with specific adjustments to mitigate low-speed drift.
- Validating the fusion algorithm in simulation, through accuracy analysis of velocity estimation and its impact on localization performance.

To support the design and implementation of this fusion algorithm, the following chapter presents the theoretical foundation of the sensors and algorithms involved. These include the principles of magnetometer and IMU measurements, the sources of sensor noise, and the Extended Kalman Filter technique used for data fusion.

1.4 Limitations / Demarcations

1.4.1 Idealization in Simulated Data

Although real-world sensor data is available, this study primarily uses simulated data to enable controlled, repeatable experiments and large-scale testing. The simulation framework includes carefully designed noise models and motion profiles; however, idealized assumptions are inevitably introduced. Real-world data often contains complex, unstructured noise patterns and disturbances that are difficult to fully replicate. This discrepancy may affect the generalizability of the algorithm when exposed to field conditions, especially in terms of unpredictable magnetic anomalies and mechanical vibrations [5].

1.4.2 Simplified Environmental Modeling

The magnetic field and motion environment in the simulation are modeled under structured, noise-filtered conditions. While these simplifications facilitate the design and evaluation of the fusion algorithm, they do not account for factors such as ferromagnetic interference from surrounding infrastructure, transient field disturbances, or unexpected vehicle dynamics. These environmental complexities may introduce challenges not fully captured in the current testing setup [6].

1.4.3 Algorithm Assumptions and Scope

The proposed sensor fusion method is tailored for low-speed, continuous motion under typical railway trajectories. It assumes relatively smooth velocity transitions and consistent sensor alignment. Scenarios involving abrupt maneuvers, severe slippage, or sensor failures are beyond the current scope [7]. Furthermore, although the fusion method is intended to enhance SLAM performance, this study focuses specifically on improving velocity estimation and does not implement full SLAM functionalities such as loop closure or map optimization.

2

Theory

2.1 Magnetometer

2.1.1 Principle of Magnetometers

Magnetometers are sensors that measure the strength and direction of magnetic fields and are widely used in navigation, geophysical exploration, railroad monitoring and other fields. The working principle of magnetometers is based on different physical effects, such as the Hall Effect, Fluxgate Effect, Superconducting Quantum Interference (SQUID), and Nuclear Magnetic Resonance (NMR).

Common types of magnetometers used in railroad applications include Hall magnetometers and Fluxgate magnetometers. Among them, Hall magnetometers calculate the magnetic field strength by measuring the effect of the magnetic field on the current carriers, and are characterized by simple structure and low cost for real-time inspection tasks. Fluxgate magnetometers, on the other hand, rely on the magnetization properties of ferromagnetic materials to provide highly sensitive magnetic field measurements and are widely used for precise positioning and orbit monitoring[12].

Magnetic fields measured by magnetometers are typically measured in units of Tesla (T) or Gamma (γ), where $1\gamma = 10^{-9}T = 1$ nT . In a railroad environment, magnetometers can not only measure the geomagnetic field, but also detect localized changes in the magnetic field around the track, such as residual magnetism inside the track, electromagnetic fields generated by train electromagnetic fields generated by motors, and interference signals from power systems along the line. By analyzing these magnetic field data, applications such as train positioning, track condition monitoring, and optimization of railroad signaling systems can be realized [13].

In this project, we did not choose a specific magnetometer, but used a general method for simulating magnetometer data. The method is based on an ideal magnetic field model that calculates the magnetic field distribution and incorporates various types of noise to simulate magnetometer measurements in a real environment.

2.1.2 Geomagnetic Field and Its Characteristics

The geomagnetic field is a general term for the magnetic field around the Earth, which originates mainly from the Earth's rotation inside the Earth's core and the convective motion of liquid metals. The geomagnetic field is not homogeneous globally, and is influenced by the magnetic material of the earth's crust, the space environment, and external currents [15].

In the railroad environment, the geomagnetic field is not only affected by natural geomagnetic effects, but also disturbed by the railroad itself and its surrounding facilities. For example, factors such as rails, electrified railroad equipment, and train operation can cause local disturbances to the magnetic field. Therefore, these magnetic field characteristics need to be taken into account in applications such as railway inspection and rail vehicle positioning, and effectively measured and analyzed using magnetometer sensors.

2.1.3 Applications of Magnetometers in Railways

Magnetometers have a wide range of applications in the railroad system, mainly used for train positioning, track condition monitoring, geomagnetic navigation and anomaly detection.

- Train Positioning and Navigation

  In the railroad system, accurate train positioning is crucial for operational safety. Magnetometers can be used to measure the geomagnetic field distribution around the track and combined with track maps to achieve passive positioning. Compared to GNSS (Global Navigation Satellite System), magnetometers are unaffected by factors such as weather and occlusion (e.g. tunnels and underground railroads), thus providing stable and reliable position information in complex environments.

- Geomagnetic Navigation

  The geomagnetic features along the railroad are relatively stable and can be used as an important reference for autonomous navigation of railcars. By measuring the geomagnetic features with magnetometers and matching them with pre-stored geomagnetic maps, trains can realize high-precision inertial navigation and improve the accuracy of autonomous driving. This technology is particularly suitable for scenarios that do not easily rely on GNSS signals, such as underground railroads or complex urban environments.

- Track condition monitoring

  Track materials (e.g. rails, fasteners, etc.) cause localized disturbances to the magnetic field. By analyzing changes in the magnetic field measured by a magnetometer, the track can be monitored for deformation, fracture or abnormal wear.

Taken together, the application of magnetometers in railroad systems is characterized by non-contact, low cost and high accuracy, and can be combined with other sensors to further improve the robustness of railroad detection and navigation systems.

2.1.4 Noise Sources in Magnetometer Measurements

- Measurement Error

  Due to factors such as manufacturing tolerances of the sensor, temperature variations, noise in the electronics, etc., random errors may occur in the measurement values of the magnetometer. Typically, this error manifests itself as Gaussian White Noise, which affects the accuracy of the measurement [16].

- Low-Frequency Drift

  Low-Frequency Drift refers to the slow change in magnetometer measurements over longer time scales and can be caused by temperature drift, power supply fluctuations, or aging effects within the sensor. In a railroad environment, long periods of operation may cause baseline drift of the sensor, making measurements drift over long periods of time[17].

- High-Frequency Jitter

  Due to uneven railroad tracks, train vibrations, and rapid changes in external magnetic fields, magnetometer measurement data may be disturbed by high-frequency noise. This noise usually appears in the form of small periodic oscillations, affecting the stability of the measurement data[18] .

- Transient Spike Noise

  In some cases, magnetometers may be disturbed by transient but large amplitude bursts of noise, such as electromagnetic interference, magnetic objects on railroad tracks, and sudden changes in strong external magnetic fields. This noise is usually random and may cause anomalies in the measurement data[19] .

During the magnetometer data simulation in this project, we designed corresponding mathematical models for each of these noises to more realistically simulate the measurement characteristics of the magnetometer in a railroad environment. For example, Gaussian white noise is used to model measurement errors, low-frequency drift uses a sine wave function to simulate long-term baseline variations, high-frequency jitter represents the effects of the track and the environment through short-period oscillations, and transient spike noise embodies sudden disturbances through random outliers. The superposition of these noises makes the simulated data closer to the real railroad magnetometer measurements.

2.1.5 Measurement Errors and Calibration

In practice, the measurement data of the magnetometer will be affected by a variety of errors, including the measurement error of the sensor itself, the interference of the ambient magnetic field, and the noise brought about by the movement of the vehicle. These errors may cause the magnetometer data to drift, jitter or sudden outliers, thus affecting the measurement accuracy. In order to improve the reliability of the data, we not only simulate the magnetometer measurement data in our experiments, but also combine the data from the inertial measurement unit (IMU), and correct the magnetometer data through the fusion algorithm[20]. The IMU can provide more stable attitude information, which helps to compensate for the deviation of the magnetic field measurements due to the vehicle motion, and improves the accuracy of the overall data.

2.2 Inertial Measurement Unit

2.2.1 Principles of IMU

Inertial measurement units (IMUs) are one of the most common types of integrated sensor used to acquire navigation data. They typically contain accelerometers and gyroscopes to measure linear acceleration and angular velocity, respectively, and do not accept an external reference[21]. In modern rail applications, microelectromechanical system (MEMS)-based inertial measurement units are widely used due to their small size, low power consumption, and relatively low cost[22].

IMUs typically detect and send back data at a fixed frequency, and these outputs are used in navigation algorithms such as SLAM to estimate the attitude, linear velocity and position through integration[22].

Accelerometers are primarily responsible for measuring linear acceleration in three orthogonal axes based on Newton's second law of motion. When acceleration is applied to the sensor, the proof mass inside generates a displacement that is proportional to the applied acceleration. The system detects the measurable signal generated by this displacement and converts it to the corresponding acceleration value through a signal conditioning circuit[23]. The measurement range of modern MEMS accelerometers typically ranges from $\pm$2g to $\pm$16g. where for navigation, accelerometers must have an operational range of at least $\pm$10g[21].

Gyroscopes are mainly used to detect the angular velocity around three orthogonal axes. Commonly, gyroscopes are categorized into optical gyroscopes and vibrating gyroscopes, and MEMS IMUs usually use the latter, which operates on the principle of the Coriolis effect[21, 23]. When a vibrating element (usually a proof mass vibrating along the main axis) is subjected to rotation, it induces secondary vibrations orthogonal to the primary vibrations, the amplitude of which is proportional to the angular velocity, which in turn produces a change in capacitance between the vibrating structure and the fixed sensing electrodes[21]. Typical measurement ranges of modern MEMS gyroscopes range from $\pm$250 degrees/s to $\pm$2000 degrees/s,

with lower ranges providing better resolution for precise motion tracking and higher ranges suitable for detecting fast rotational motion in applications such as gaming or drone stabilization.

The standard IMU integrates accelerometers and gyroscopes into a microelectromechanical system and therefore outputs a value containing 6-DoF of linear acceleration and angular velocity. As technology evolves, advanced IMU integrates magnetometers as well, eventually outputting 9-DoF containing magnetic field vectors on 3 orthogonal axes[21].

It should be noted that this study focuses on data characterization as well as sensor fusion, so the differences caused by the structure and design of different sensors are not the focus of this paper. Specific sensors were also not chosen in this study, but rather a simulation was performed using sensor data features, and more generalized simulated data was used instead of actual data from specific sensors. This approach reduces the cost of data collection while avoiding sensor-related specificities, which enhances the general applicability of the research and allows the results to be more broadly applicable to a variety of IMU-based implementations.

2.2.2 Measurements and Representation

The IMU data mainly contains linear acceleration detected by accelerometer: $a = [a_x, a_y, a_z]^T$ (unit:m/s^2) and angular velocity detected by gyroscope:$\omega = [\omega_x, \omega_y, \omega_z]^T$ (unit: rad/s). In rail environment, acceleration measurements reflect changes in vehicle operating conditions, including changes in linear motion caused by actions such as starting, braking, and cornering. Angular velocity can be used to monitor the rotational motion of the vehicle caused by cornering, sideways tilting, and track irregularities, which is an important indicator for assessing the dynamic performance of the vehicle and the quality of the track[24]. Since the sensors detect and send data based on a fixed frequency, the sensor data are also all time-series matrices, which are usually analyzed in the time domain.

Specifically, in railway environments, IMU data applying to an vehicle is usually related to the actual motion state of the vehicle, such as a train. For example, the longitudinal acceleration (a_x) reflects the traction and braking process of the train, the lateral acceleration (a_y) reflects the centrifugal force generated by the train during a turn, and the vertical acceleration (a_z) reflects the smoothness and irregularity of the track. The angular velocities in different directions reflect more the relationship between the influence of the environment on the vehicle motion, for example, the roll angular velocity (ω_x, rotating around the longitudinal axis) reflects the track superelevation and the train lateral inclination, the pitch angular velocity (ω_y, rotating around the transverse axis) reflects the track gradient change, and the yaw angular velocity (ω_z, rotating around the vertical axis) reflects the train steering process together with the linear acceleration. Obviously, there is a correspondence between the angular velocity and the curvature of the track in the three directions, and the train passes through the track segments with changing curvature with a corresponding angular velocity response.

Since IMU data is derived from sensor perception within a vehicle, the measurements are represented in a local coordinate system defined by the sensor's orientation, rather than in a geographic coordinate system. Therefore, there is a relative transformation required between the IMU data and the geographic reference frame when applying it to methods such as SLAM for map building. For vector quantities such as velocity and acceleration, this transformation is achieved through rotation matrices, as shown in Equation. 2.1:

$$\mathbf{v}_{local} = \mathbf{R}(\alpha, \beta, \gamma) \ \cdot \ \mathbf{v}_{geographic} \tag{2.1}$$

Where $\mathbf{v}_{geographic}$ is the vector in the geographic coordinate system, $\mathbf{v}_{local}$ is the vector in the local coordinate system of the sensor, $\mathbf{R}(\alpha, \beta, \gamma)^{-1}$ is the inverse of the rotation matrix based on Euler angles (roll angle α, pitch angle β, and yaw angle γ).

For position transformations, an additional translation component would be required, as shown in Equation. 2.2:

$$\mathbf{p}_{local} = \mathbf{R}(\alpha, \beta, \gamma) \ \cdot \ (\mathbf{p}_{geographic} - \mathbf{T}) \tag{2.2}$$

Where $\mathbf{T}$ represents the origin of the local coordinate system in the geographic frame. The rotation matrix $\mathbf{R}(\alpha, \beta, \gamma)$ can be further expressed as a combination of the three basic rotation matrices:

$$\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R}_z(\gamma) \ \cdot \mathbf{R}_y(\beta) \ \cdot \ \mathbf{R}_x(\alpha) \tag{2.3}$$

Which maps the measurements in the sensor's local coordinate system to the geographic coordinate system[25].

2.2.3 Error Characteristics

There are two main types of errors in IMU sensors: deterministic errors and stochastic errors. Among them, deterministic errors (e.g., bias, scale factor, and cross-coupling) can usually be measured and compensated for by calibration methods, for example, measurement errors caused by sensor temperature changes due to the external environment can be reduced by temperature compensation algorithms [26]. Of course, in high-precision applications, further compensation needs to be considered since this part of the deterministic error cannot be completely eliminated and is negligible in most cases.

Stochastic errors have received more attention in SLAM research because these types of errors are difficult to be completely eliminated by simple calibration. In the railroad environment, common stochastic errors typically include the following categories:

1. **Sensor Inherent White Noise**
   This manifests itself as high-frequency random fluctuations in the measured value, originating from electronic noise and mechanical Brownian motion within the sensor. Although the effect of a single measurement is limited, it can lead to integration errors during the navigation solving process[21].

2. **Environmental and Mechanical Noise**
   These are structured disturbances captured during high-frequency sampling (typically 100-300Hz) of the IMU, primarily originating from the surrounding environment and mechanical systems. In railroad scenarios, this includes rail-wheel interactions, engine vibrations, and track irregularities[27]. These disturbances have specific frequency characteristics related to the vehicle's speed and track conditions, and can impact measurement accuracy, especially at low speeds.

3. **Long-term Bias Instability**
   This manifests itself primarily as random walk and bias instability and accumulates over time [28]. This error is proportional to the square root of the integration time and is the main cause of long-term bias in position estimation in SLAM [21].

2.3 Sensor Fusion

2.3.1 Principles of Sensor Fusion

Sensor Fusion refers to the use of data from multiple sensors, which are synthesized and processed by specific algorithms and models to obtain more accurate, stable and reliable information. This method is widely used in the fields of autonomous driving, robot navigation, inertial navigation, etc[29].

The basic principle of sensor fusion is to utilize the complementary characteristics of multiple sensors in order to reduce the uncertainties and errors of a single sensor. For example, Inertial Measurement Unit (IMU) is able to provide higher accuracy motion information in a short period of time, but with drift error, while GPS (Global Positioning System) is able to provide absolute position information, but the accuracy is greatly affected by the environment. Through sensor fusion, the advantages of various types of sensors can be combined to improve the accuracy and reliability of the overall system [14].

Common sensor fusion methods include[30]:

- Data-level fusion: direct fusion of raw data from multiple sensors, such as the combination of accelerometer and gyroscope data.

- Feature-level fusion: fusion of features extracted from different sensors, such as using image feature points combined with IMU data for localization.

- Decision-level fusion: high-level fusion based on independent judgments provided by multiple sensors, such as multiple sensors to determine the existence of obstacles, and ultimately give a comprehensive decision.

2.3.2 Application of Sensor Fusion in Train Track Localization

In train track localization, sensor fusion can make up for the limitations of a single sensor and improve positioning accuracy and reliability. The track environment is complex, and GPS signals may be obstructed by viaducts, tunnels and buildings, thus affecting the positioning accuracy. Meanwhile, inertial navigation systems (INS) generate drift errors after a long period of operation. Magnetometers can provide additional heading information, but are more affected by ferromagnetic interference. Therefore, a single sensor cannot maintain stable accuracy in all environments, and sensor fusion can effectively deal with these problems[31].

In this study, we use a sensor fusion method that combines IMU (inertial measurement unit), magnetometer, and track information to achieve high-precision track positioning. The main advantages of this method include:

Improved positioning accuracy: the combination of IMU, magnetometer and track information enables the system to maintain high positioning accuracy even if the GPS signal is lost for a short period of time.

Enhanced Robustness: Utilizes data from multiple sensors and relies on other sensors to maintain normal operation when some sensors are disturbed or data is lost.

Reducing error accumulation: the drift error of IMU can be corrected by GPS or orbit information, while the heading information provided by magnetometer can further improve the accuracy of attitude estimation.

2.3.3 Types and Applications of Sensor Fusion

The algorithms for sensor fusion mainly include the following:

Table 2.1: Comparison of Sensor Fusion Algorithms

Sensor Fusion Algorithm	**Description**	**Advantages**	**Disadvantages**

Sensor Fusion Algorithm	Description	Advantages	Disadvantages
Kalman Filter (KF)[32]	Applicable to linear systems; state estimation is realized through a prediction-update mechanism. Commonly used in inertial navigation, GNSS/IMU fusion, and autonomous driving.	Small computational effort, suitable for real-time systems.	Only applicable to linear systems, difficult to handle nonlinear states.
Extended Kalman Filter (EKF)[32]	Applicable to nonlinear systems; linearizes the state equation and then filters. Widely used in GPS/IMU fusion, trajectory estimation, and other applications.	Can handle certain degrees of nonlinearity.	Errors may be introduced in the linearization process, potentially affecting accuracy.
Unscented Kalman Filter (UKF)[33]	Avoids linearization errors of EKF using Unscented Transform, improving estimation accuracy for nonlinear systems. Commonly used in UAVs, robots, and autopilot systems.	Performs better than EKF in nonlinear systems.	Higher computational complexity, may not be suitable for real-time systems.
Particle Filter (PF)[34]	Uses particle sampling to estimate the state, suitable for highly nonlinear and non-Gaussian noise systems. Commonly used in vision SLAM, target tracking, and navigation.	Applicable to arbitrary nonlinear and non-Gaussian systems.	High computational cost, poor real-time performance.

Sensor Fusion Algorithm	Description	Advantages	Disadvantages
Neural Network-Based Fusion[35]	Uses deep learning methods to automatically learn the relationship between sensor data through neural networks. Suitable for multi-sensor fusion in complex environments.	Effective for complex sensor fusion problems, adaptable to various sensor types.	Requires a large dataset for training, may not be efficient for real-time applications.

2.4 Extended Kalman Filter

In multi-sensor systems, filtering techniques are essential to mitigate measurement noise and model uncertainty. The classical Kalman Filter (KF) provides optimal state estimation for *linear systems with Gaussian noise*; however, most practical systems are nonlinear in nature, particularly when involving inertial sensors and external disturbances. To address this, the **Extended Kalman Filter (EKF)** linearizes the nonlinear models around the current estimate, enabling recursive Bayesian state estimation in nonlinear domains.

2.4.1 Motivation for Using EKF

In this study, we aim to estimate the velocity of a rail vehicle using a fusion of:

- Accelerometer data from an Inertial Measurement Unit (IMU), and
- Velocity proxies derived from simulated magnetometer signals.

The IMU provides high-frequency dynamic information but suffers from drift over time. In contrast, magnetometer-based signals can indirectly reflect changes in velocity and motion direction but are highly sensitive to environmental noise. By fusing these data sources, the system benefits from both high responsiveness and drift compensation.

Several alternative filtering approaches exist:

- **Kalman Filter (KF)** – assumes linear models; insufficient for our nonlinear setup.
- **Unscented Kalman Filter (UKF)** – better for high nonlinearity but more computationally intensive.
- **Particle Filter (PF)** – highly flexible but computationally expensive and less efficient for low-dimensional systems.

Considering the moderate degree of nonlinearity inherent in the system dynamics and the stringent requirements for real-time processing, the Extended Kalman Filter (EKF) emerges as a highly suitable approach. Unlike the standard Kalman Filter, which is limited to linear systems, the EKF approximates nonlinear state and measurement models through first-order Taylor series expansions, enabling effective state estimation in nonlinear contexts.

While UKF offers improved handling of strong nonlinearities, its higher computational demand may impede real-time implementation. Particle Filters provide greater flexibility in modeling arbitrary distributions but at the cost of significant computational overhead, particularly inefficient for systems with relatively low-dimensional state spaces.

Moreover, the EKF benefits from well-established theoretical foundations and extensive practical applications, which facilitate implementation, tuning, and integration into embedded systems. Therefore, in light of its ability to provide reliable nonlinear state estimation within reasonable computational resources, the EKF is selected as the primary filtering framework for sensor fusion and velocity estimation in this study [36].

2.4.2 Mathematical Formulation

The EKF is based on a nonlinear discrete-time state-space model:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1} \tag{2.4}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{2.5}$$

where:

- $\mathbf{x}_k$ is the hidden state vector at time step k (e.g., velocity),
- $\mathbf{u}_{k-1}$ is the control input (e.g., IMU acceleration),
- $\mathbf{z}_k$ is the measurement (e.g., magnetometer-derived velocity),
- $f(\cdot)$ is the nonlinear process model,
- $h(\cdot)$ is the nonlinear observation model,
- $\mathbf{w}_{k-1} \sim \mathcal{N}(0, Q)$ is process noise,
- $\mathbf{v}_k \sim \mathcal{N}(0, R)$ is measurement noise.

Since $f(\cdot)$ and $h(\cdot)$ are nonlinear, the EKF linearizes them at each timestep using Jacobians:

$$F_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}}, \quad H_k = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}} \tag{2.6}$$

These Jacobians are then used in the standard Kalman update equations for prediction and correction:

- **Prediction:**

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}) \tag{2.7}$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^\top + Q \tag{2.8}$$

- **Update:**

$$\mathbf{y}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}) \tag{2.9}$$
$$S_k = H_k P_{k|k-1} H_k^\top + R \tag{2.10}$$
$$K_k = P_{k|k-1} H_k^\top S_k^{-1} \tag{2.11}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \mathbf{y}_k \tag{2.12}$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2.13}$$

Here, K_k is the Kalman gain, and P represents the estimation error covariance matrix [37].

2.4.3 Applications in Sensor Fusion

EKF has seen extensive use in sensor fusion scenarios, particularly in GPS-denied or magnetometer-compromised environments such as indoor navigation, aerospace systems, and railway tunnels [38]. By incorporating IMU data (typically noisy but high-frequency) and magnetometer readings (which can provide heading or orientation), EKF helps mitigate individual sensor weaknesses and enhances overall system accuracy [39].

In our work, the EKF framework enables the fusion of simulated magnetometer-derived velocity estimates with IMU-derived acceleration data. This approach is expected to yield more accurate and robust velocity estimations compared to using either sensor modality independently.

2.5 Literature review

The use of magnetometer and inertial sensor fusion for railway localization has gained increasing attention due to its potential to provide accurate and infrastructure-independent navigation. Various research efforts have explored different strategies for estimating train position and speed using on-board sensor data.

A prominent study by Hammar and Hedberg (2015) investigated the feasibility of train localization and speed estimation using bogie-mounted inertial and magnetic sensors [8]. Their work introduced two primary methods: one based on estimating the fundamental frequency of magnetic field variations (wheel-turn extraction), and another leveraging mechanical vibrations detected by IMU sensors (signature matching). By integrating these signals through a Kalman filter, they achieved promising results—speed estimates within 0.5 m/s and localization accuracy in the range of 3–5 meters, particularly highlighting the advantage of magnetic signatures for passive localization on fixed tracks.

Engelberg and Mesch (2000) proposed a non-contact eddy current sensor system for precise speed and distance estimation of rail vehicles [9]. The method uses two eddy current sensors mounted on the bogie to detect rail irregularities. These signals are cross-correlated to determine transit time, from which speed is calculated. The system proved robust in lab and field tests, with relative distance measurement errors below 0.2% even in challenging environments. Notably, the sensors maintained high accuracy in the presence of dirt, snow, and varying rail surface conditions.

In a more recent approach focused on infrastructure-free localization, Wendel et al. (2011) introduced a sensor fusion algorithm combining IMU sensors with wheel tachometers [10]. This approach avoids reliance on external infrastructure such as GNSS or track-side transponders and offers robust dead reckoning for trains. The fusion method uses an Extended Kalman Filter to estimate position and heading, achieving improved long-term stability by accounting for wheel slip and IMU drift. This work highlights the importance of well-modeled sensor characteristics and bias handling when fusing low-cost sensors for real-world railway applications.

Beyond these sensor-specific implementations, recent work has also emphasized tightly-coupled fusion strategies integrating magnetometers with visual-inertial systems. For instance, an enhanced SLAM framework incorporating magnetic measurements within a sliding-window optimization has been shown to significantly improve localization in environments with limited visual features [11].

These existing studies provide strong motivation for further exploration into low-speed sensor fusion enhancements. Building on this foundation, the current thesis proposes a new EKF-based fusion framework specifically designed to improve velocity estimation in low-dynamic scenarios, thereby laying the groundwork for more robust SLAM integration in visually and spatially degraded railway environments.

3

Methods

3.1 Overview

The methodology of this study is mainly divided into two parts: the first part is the characterization and simulation of sensor data, and the second part is the optimization and performance evaluation of sensor fusion algorithms. The specific flow is shown in Figure. 3.1.

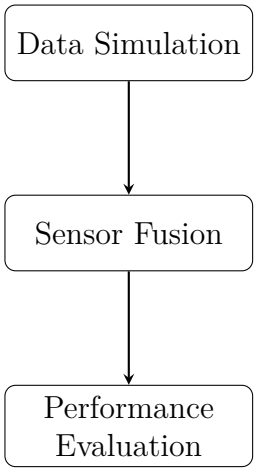

Figure 3.1: Overview of the research methodology

Building upon the theoretical foundations, this study adopts a simulation-based approach to validate the effectiveness of sensor fusion techniques. By first characterizing the sensor data and modeling its behavior, we establish a basis for optimization strategies that enhance fusion accuracy. These optimizations, in turn, contribute to the design and evaluation of integrated SLAM algorithms, ensuring robust performance in real-world applications.

3.2 Data Simulation

Based on the study of the sensor data, there is correlation between the magnetometer data and the IMU data. Specifically, in the process of estimating speed using a magnetometer, two magnetometers are required to measure the magnetic flux

density, and theoretically these two magnetometers will produce similar waveforms. However, the acceleration or deceleration of the robot causes these waveforms to vary in "width". Although these variations are not identical to the IMU data (because this difference is used to compensate for the errors in generating speeds from the IMU data), the two sensor data should show similar trends during the time when acceleration and deceleration occur. Therefore, before generating simulated data from both sensors, the changes in the robot's motion need to be modeled. In order to obtain a kinematic model of the robot, it is first necessary to model the track and obtain the corresponding trajectory data, for example the curvature and coordinates of each point of the trajectory. The specific flow is shown in Figure. 3.2.

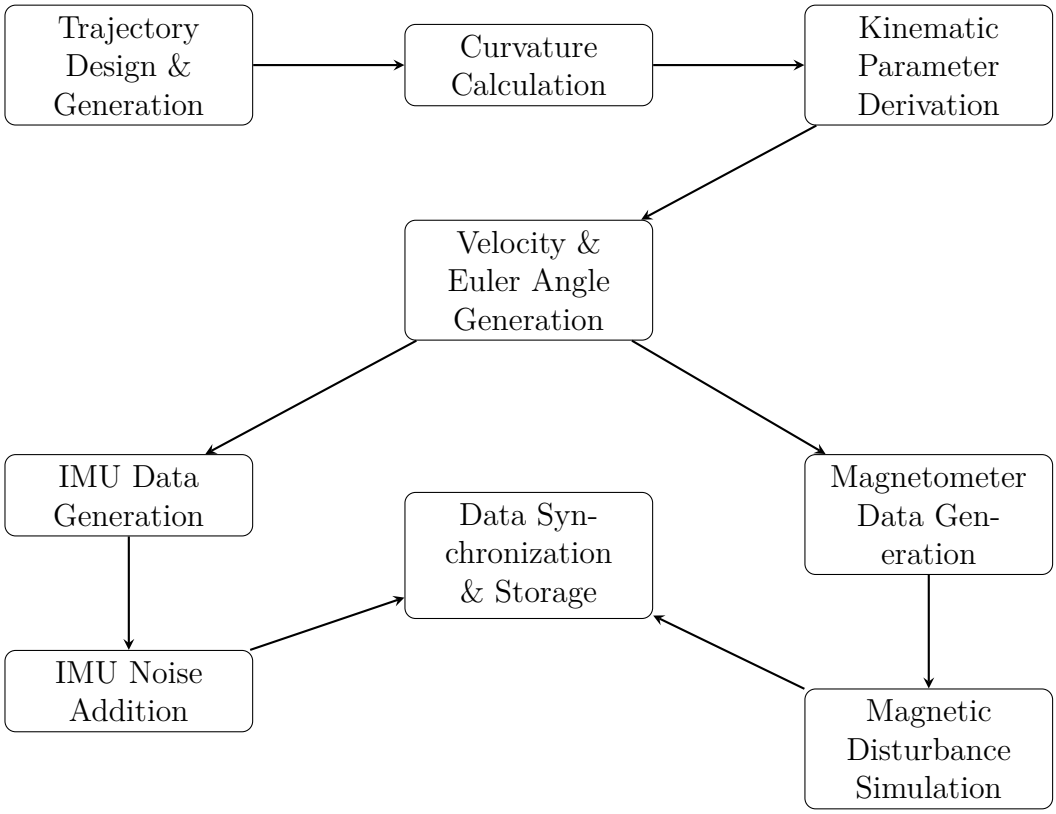

Figure 3.2: Sensor Data Simulation System Workflow

3.2.1 Trajectory Generation

Realistic railways are usually between tens and thousands of kilometers, but excessively long railways and the corresponding generated data may have an impact on subsequent algorithm runs, such as performance degradation due to excessively long running times. Considering that this study focuses on speed estimation of railway robots at low speeds, excessively long railways also suffer from underutilization.

Therefore, the length of the railway should not be designed to be too long, and at the same time, it should try to satisfy the terrain that may be encountered in the physical world that will cause various changes in motion, such as curves with different curvatures, appropriate gradients, necessary straight-line segments

and transition sections, and starting and ending points where speed is zero to design the acceleration and deceleration phases. According to the above requirements, the design of the railway track in this study is shown in Table. A.1. The track is designed by 13 sections, including 4 straight sections, 6 transition sections and 3 curves.

The curve radii are set to 200-300 meters, which is not based on fixed standards of any specific country, but rather represents a consensus among railway engineering professionals. Radii within this range allow the robot to navigate turns normally. For sensor fusion algorithm testing purposes, curve radii of 200-300 meters can generate moderate centrifugal force variations, providing meaningful test scenarios for magnetometer-IMU data fusion while avoiding extreme curvature changes that could potentially cause data anomalies.

To simulate realistic railway terrain variations, elevation changes are added to the base track design using Gaussian-based mathematical models. The elevation profile combines two main features: a hill and a valley, as detailed in Table. A.2. The gradient is designed to no more than 2% of the railway length to ensure that the robot can travel along the railway [40].

The combined elevation profile is given by Equation. 3.1:

$$z(d) = h_{\text{hill}}(d) + h_{\text{valley}}(d) \tag{3.1}$$

where d represents the cumulative distance along the track, d_{peak} is the distance to the peak of the hill, and d_{bottom} is the distance to the bottom of the valley.

The final schematic of the track is shown in Figure. 3.3. And the xy-plane and xz-plane curvature of the track are shown in Figure. 3.4 and 3.5. And the elevation profile is shown in Figure. 3.6.

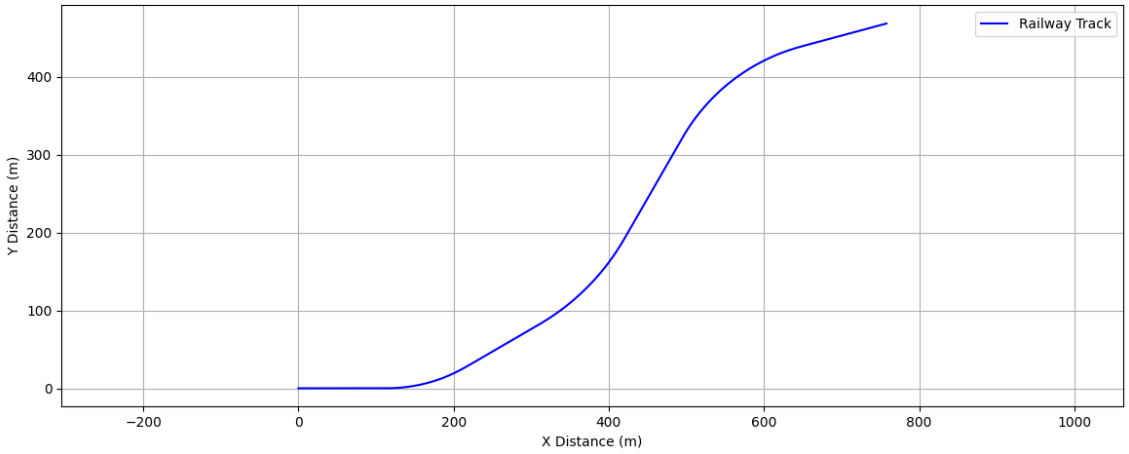

Figure 3.3: Schematic of the final generated track

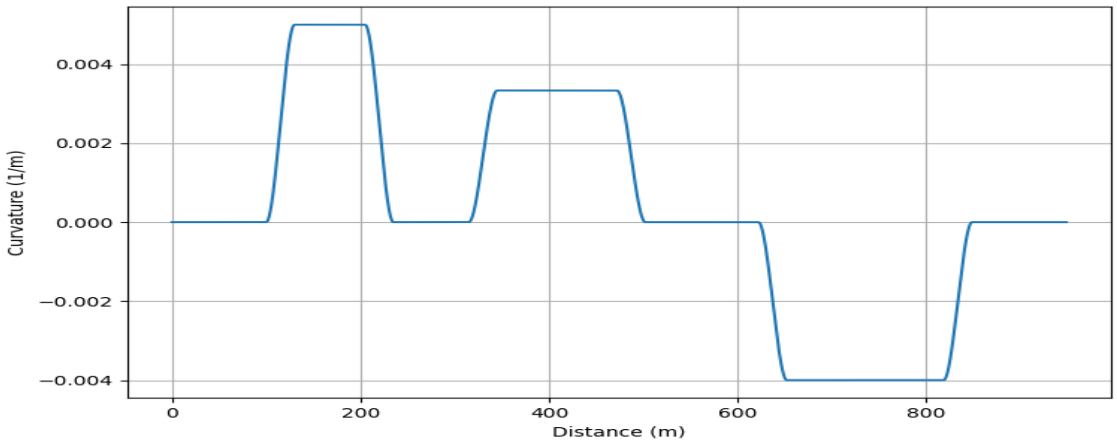

Figure 3.4: Curve of the xy-plane curvature of the track

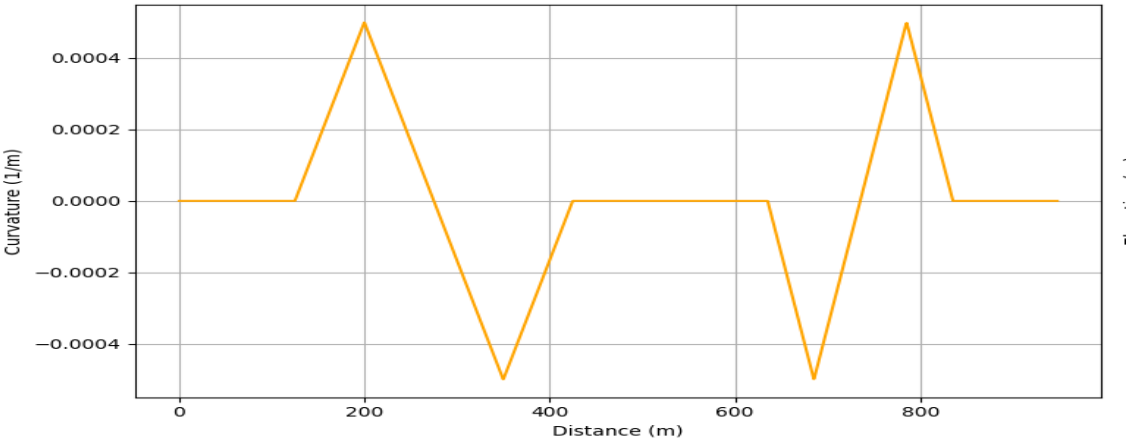

Figure 3.5: Curve of the xz-plane curvature of the track

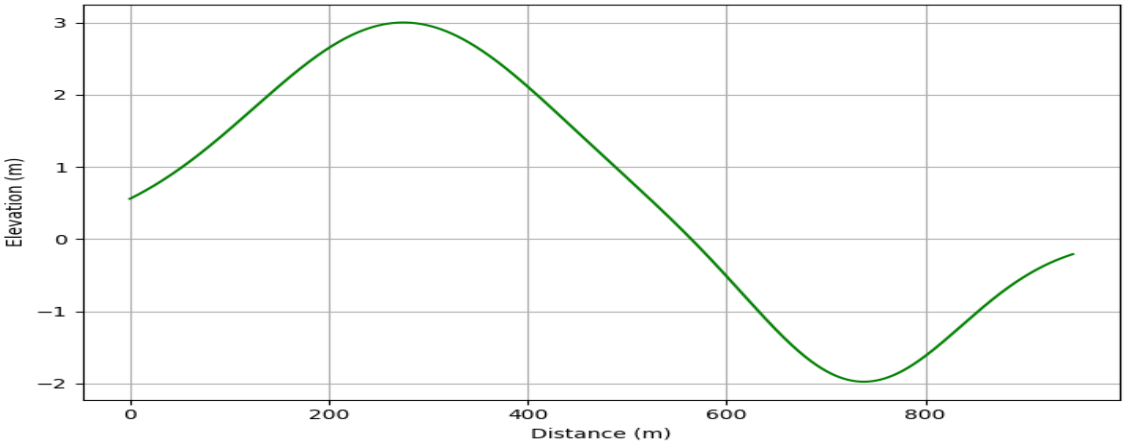

Figure 3.6: Curve of the elevation profile of the track

3.2.2 Kinematics Modeling

Based on the study objectives, the kinematics model needs to handle various motion modes including forward motion, backward motion, acceleration, and deceleration. Therefore, the kinematics model requires receiving a command or command list to determine the overall motion of the robot.

All the motions mentioned above follow similar generation logic. The robot adjusts its motion based on target acceleration and target velocity, and determines position based on the current motion state. The model generates kinematic states by first calculating velocity from time and target acceleration, while controlling acceleration and velocity changes through configured jerk limitations. Although the system is designed for three-axis operation, the analysis primarily focuses on the x-axis data, which represents the direction of motion, to concentrate on velocity estimation studies.

3.2.2.1 State Management and Command-Driven Architecture

To handle the traveling process with various motion patterns, the model uses a command structure including the following parameters:

- **Motion type:** Determines the motion behavior of this command, whether to move or stop.
- **Target velocity:** Specifies the desired velocity after acceleration or deceleration.
- **Maximum acceleration:** Defines the maximum acceleration for velocity changes, which also serves as a condition for jerk configuration.
- **Duration:** Specifies the time duration for executing this command.
- **Maximum centripetal acceleration:** Provides velocity constraints during steering maneuvers.

These command parameters determine the motion profile generated by each command. The model also maintains internal state variables that track the robot's current condition and provide continuity between commands:

- **Current speed:** Records the instantaneous velocity after completing one command, serving as the initial condition for the next command.
- **Current distance:** Tracks the cumulative traveled distance along the trajectory, marking the position for command transitions.
- **Stop state flag:** Indicates whether the robot has come to a complete stop. When the robot is detected to be stopped, the model terminates data generation after the velocity reaches zero to avoid generating meaningless static data.

- **Current timestamp:** Maintains the current time reference for command sequencing and synchronization.
- **Maximum jerk:** Constrains the maximum rate of acceleration change, which is a fundamental parameter determined by the robot's motor system characteristics.

3.2.2.2 Velocity Profile Generation

The model calculates the velocity based on time-domain sampling. This method can handle time synchronization properly, providing a synchronized time standard for the following sensor data simulation.

For direction change scenarios:

$$\text{Deceleration: } |v(t)| = |v_0| - a_{\max}(t - t_0) \quad \text{for } t \leq t_0 + T_1 \tag{3.2a}$$
$$\text{Acceleration: } |v(t)| = a_{\max}(t - t_0 - T_1) \quad \text{for } t > t_0 + T_1 \tag{3.2b}$$

For same-direction motion:

$$|v(t)| = |v_0| + a \cdot (t - t_0) \tag{3.2}$$

where

$$a = \text{sgn}(|v_t| - |v_0|) \cdot a_{\max} \tag{3.2c}$$

For jerk-limited segments:

$$\text{Jerk up: } v(t) = v_{\text{start}} + \text{sgn}(\Delta v) \cdot 0.5 \cdot j_{\max} \cdot t^2 \tag{3.3a}$$
$$\text{Constant acc: } v(t) = v_{\text{phase1,end}} + \text{sgn}(\Delta v) \cdot a_{\max} \cdot (t - t_{\text{jerk}}) \tag{3.3b}$$
$$\text{Jerk down: } v(t) = v_{\text{phase2,end}} + \text{sgn}(\Delta v) \cdot (a_{\max} t_{\text{down}} - 0.5 j_{\max} t_{\text{down}}^2) \tag{3.3c}$$

To achieve more realistic mechanical motion characteristics, the model introduces a jerk-limited velocity profile generation algorithm. This algorithm implements three-phase acceleration control:

- **Acceleration ramp-up phase**: Gradually increase acceleration through limited jerk values
- **Constant acceleration phase**: Maintain maximum acceleration for linear velocity changes
- **Acceleration ramp-down phase**: Gradually decrease acceleration until reaching target velocity

Position calculation then follows:

$$s(t) = s(t - \Delta t) + |v(t)| \cdot \Delta t \tag{3.4}$$

where $s(t)$ is the cumulative distance at time t. This approach is easier for both calculating coordinates and applying to the following data simulation and velocity estimation.

Since the calculated cumulative distance may not correspond to discrete track points, coordinate interpolation is required to obtain the actual position:

1. Distance increment calculation: $\Delta s(t) = |\mathbf{v}(t)| \cdot \Delta t$
2. Cumulative distance update: $s_{\text{cum}}(t) = s_{\text{current}} + \sum \Delta s(t)$
3. Track index mapping: $i_{\text{track}}(t) = s_{\text{cum}}(t) \cdot \frac{N_{\text{points}}-1}{L_{\text{total}}}$
4. Coordinate interpolation: $\mathbf{x}_{\text{coords}}(t) = \text{interp}(i_{\text{track}}(t), [0, 1, ..., N_{\text{points}}-1], \mathbf{x}_{\text{track}})$

where $s_{\text{cum}}(t)$ is the total cumulative distance including contributions from all previous commands, and s_{current} represents the cumulative distance traveled up to the beginning of the current command.

3.2.2.3 Enhanced Motion Processing

Direction Change Processing: Based on the above theoretical fundamentals, two important challenges are addressed:

1. **Difference between geographical coordinate calculations and physical motion intentions**

   For example, when a robot moves backward (negative velocity in geographical coordinates) and needs to decelerate to stop, the required acceleration is also negative in the coordinate system. However, direct mathematical computation would result in a positive velocity increment, which contradicts the physical intention of reducing speed to zero. Therefore, the velocity calculation needs to consider speed and direction separately rather than relying solely on coordinate signs, with different approaches for different motion scenarios.

2. **Dependency between motion segments**

   The model uses the relationship between the cumulative distance and the goal acceleration. However, the beginning and the end of the motion in this certain region might vary based on the last region, where the region and the related motion are determined based on the command(s). Therefore, the state of last motion also needs to be considered for the beginning of this state.

Therefore, when a direction change is detected, the system adopts a two-phase approach:

- Phase 1: Use jerk-limited deceleration algorithm to reduce current speed to zero

- Phase 2: Use jerk-limited acceleration algorithm to accelerate from zero to target speed

3.2.2.3.1 Gaussian filtering : In addition to the above design, Gaussian filtering is also applied in the model to obtain smoother data. The tangent vectors, which represent the direction of motion at each track point, are smoothed to eliminate numerical discontinuities that may occur during discrete differentiation between adjacent track points. The filtering is also applied to the final velocity data and orientation data to ensure physically realistic motion characteristics. The equation is shown in Equation 3.2:

$$(G_\sigma \cdot v)_i = \sum_{k=-m}^{m} v(i-k) \cdot G(k) \tag{3.2}$$

where v is the speed profile of the points: $v = |v| = \sqrt{v_x^2 + v_y^2 + v_z^2}$, and $G(k)$ is the Gaussian kernel function:

$$G(k) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{k^2}{2\sigma^2}\right) \tag{3.3}$$

The parameter σ is set to 1.0, and the Gaussian kernel is truncated to $\pm 3\sigma$, which is a commonly used configuration in signal processing and robotics applications based on empirical practice. This configuration helps reduce the noise introduced by discretization while preserving the essential characteristics of the velocity and orientation profiles.

The kinematics model is generated based on the logic above, and the details are shown in Algorithm 3.1. The model also requires physical constraints including maximum acceleration and centripetal acceleration limits to be applied to ensure realistic motion profiles, particularly when navigating curved sections.

Algorithm 3.1: Robot Kinematics Model

Input	Track points $\mathbf{p}_i$, command list $\mathcal{C}$, sampling frequency f_s
Output	Time-synchronized kinematics data $\{\mathbf{v}(t), \lvert\mathbf{v}\rvert(t), \mathbf{p}(t)\}$
Step 1	**Initialize State:** Set current speed: $v_{\text{current}} = 0$, current distance: $s_{\text{current}} = 0$ Set maximum jerk: $j_{\max} = 0.3$ m/s^3, sampling interval: $\Delta t = 1/f_s$
Step 2	**Speed Profile Generation:** **For each command** $c \in \mathcal{C}$**:** Extract parameters: $v_{\text{target}} = c[\text{max\_v}]$, $a_{\max} = c[\text{max\_a}]$, $T = c[\text{duration}]$ **IF** direction change ($v_{\text{current}} \cdot v_{\text{target}} < 0$): Generate two-phase jerk-limited profile: Phase 1: Decelerate to zero using jerk-limited deceleration Phase 2: Accelerate to target using jerk-limited acceleration **ELSE** same direction motion: Generate single jerk-limited acceleration profile
Step 3	**Jerk-Limited Segment Generation:** Calculate jerk time: $t_{\text{jerk}} = a_{\max}/j_{\max}$ Calculate velocity change during jerk phases: $\Delta v_{\text{jerk}} = 0.5 \cdot j_{\max} \cdot t_{\text{jerk}}^2$ **IF** $\lvert\Delta v\rvert \leq 2\Delta v_{\text{jerk}}$: Short segment: $t_{\text{jerk,actual}} = \sqrt{\lvert\Delta v\rvert / j_{\max}}$, $t_{\text{const}} = 0$ **ELSE** long segment: $t_{\text{jerk,actual}} = t_{\text{jerk}}$, $t_{\text{const}} = (\lvert\Delta v\rvert - 2\Delta v_{\text{jerk}})/a_{\max}$ Generate three-phase profile: Phase 1: $\lvert\mathbf{v}(t)\rvert = \lvert v_0\rvert + \text{sgn}(\Delta v) \cdot 0.5 j_{\max} t^2$ Phase 2: $\lvert\mathbf{v}(t)\rvert = \lvert v_{\text{phase1}}\rvert + \text{sgn}(\Delta v) \cdot a_{\max}(t - t_{\text{jerk}})$ Phase 3: $\lvert\mathbf{v}(t)\rvert = \lvert v_{\text{phase2}}\rvert + \text{sgn}(\Delta v) \cdot (a_{\max} t_{\text{down}} - 0.5 j_{\max} t_{\text{down}}^2)$
Step 4	**Position Calculation and Stop Handling:** Calculate distance increments: $\Delta s(t) = \lvert\mathbf{v}(t)\rvert \cdot \Delta t$ Update cumulative distance: $s(t) = s_{\text{current}} + \sum \Delta s(t)$ Interpolate track positions: $\mathbf{p}(t) = \text{interp}(\mathbf{s}_{\text{track}}, \mathbf{p}_{\text{track}}, s(t))$ **IF** command is 'stop': Find stop index where $\lvert\mathbf{v}(t)\rvert < 10^{-6}$ Truncate data at stop point

3.2.3 IMU Data Simulation

This section focuses on the simulation of IMU data based on the kinematic model from 3.2.2. In order to be consistent with the changes in motion shown by the magnetometer data, the IMU data are also generated based on the output of the kinematic model, including the velocity and Euler angles. IMU data simulation can be divided into two main parts, one is the calculation of the ideal data, and the other is the noise simulation. The IMU data need to use an internal coordinate system based on its position related to the robot instead of a geographic coordinate system. Hence, the IMU data also needs to be transformed in this data conversion

step to ensure that the data are close to the real data received from the sensors.

3.2.3.1 IMU Data Calculation

Accelerometer data Ideal IMU data are first calculated on the basis of the velocities generated by the kinematic model. Since the velocity of a robot usually has components in three directions, and the velocity is obtained by restricting the maximum acceleration and the maximum combined velocity in the kinematic model, in order to avoid further errors caused by differentiation of the velocities in the three directions (which is not the focus of this study), the data of the speed profile, generated by the kinematic model is mainly used for the computation here. The derivative of speed profile with respect to cumulative distance (dv/ds) is first calculated using numerical differentiation [42] as shown in Equation. 3.4:

$$\frac{d|v|}{ds} \approx \frac{|v|(s+\Delta s) - |v|(s-\Delta s)}{2\Delta s} \tag{3.4}$$

Where $|v|$ is the speed profile, and s is the cumulative distance of the position, described in section 3.2.2. In Equation 3.4, the central difference method is used to calculate the derivative for interior points, where $v(s+\Delta s)$ and $v(s-\Delta s)$ represent the velocity values at the next and previous points, respectively, and $2\Delta s$ is the distance between these points. For the first point ($s = s_0$) and last point ($s = s_n$) of the trajectory where central difference cannot be applied, forward and backward difference methods are used instead:

$$\left.\frac{d|v|}{ds}\right|_{s=s_0} \approx \frac{|v|(s_0+\Delta s) - |v|(s_0)}{\Delta s}$$
$$\left.\frac{d|v|}{ds}\right|_{s=s_n} \approx \frac{|v|(s_n) - |v|(s_n-\Delta s)}{\Delta s}$$

This boundary treatment is a standard approach in numerical differentiation when data points at both sides are not available for central differencing[42]. After calculating the derivatives, Gaussian filter was applied to the results as described in Section 3.2.2, to reduce numerical noise and ensure smooth acceleration profiles.

To generate the acceleration in 3 axes, the direction vectors need to be calculated as mentioned in Algorithm **??**. For example, the tangent vector $\mathbf{e}_t$ is generated according to Equation 3.5. These vectors are different from orientation data, used to calculate the acceleration components along different axes, while the orientation data is used to transform the coordinates from geographical to local coordinates.

$$\mathbf{e}_{t,i} = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{|\mathbf{p}_{i+1} - \mathbf{p}_i|} \tag{3.5}$$

Where $\mathbf{p}_i = (x_i, y_i, z_i)$ represents the three-dimensional coordinates of the i-th point on the track, and $|\cdot|$ denotes the Euclidean norm of the vector.

Similarly, the normal direction vectors are calculated as:

$$\mathbf{e}_{n,i} = \frac{[-e_{t,i,y}, e_{t,i,x}, 0]}{|[-e_{t,i,y}, e_{t,i,x}, 0]|} \tag{3.6}$$

$$\mathbf{e}_{b,i} = \mathbf{e}_{t,i} \times \mathbf{e}_{n,i} \tag{3.7}$$

where $\mathbf{e}_{n,i}$ represents the normal direction vector in the xy-plane and $\mathbf{e}_{b,i}$ represents the binormal direction vector in the xz-plane.

The calculation of acceleration applies the physical method based on direction vectors. This approach utilizes the intrinsic physical characteristics of robot motion to reduce errors that would arise from direct numerical differentiation. The simulation can thus focus more on modeling physical noise caused by the environment, rather than introducing mathematical errors.

Based on kinematics theory, the acceleration can be separated into three components: tangential acceleration, which reflects the process of acceleration and deceleration along the traveling direction; xy-plane normal acceleration, which represents the centripetal acceleration during steering; and xz-plane normal acceleration, which describes the acceleration component caused by elevation changes.

The tangential acceleration is calculated based on the derivative of speed profile with respect to cumulative distance, as shown in Equation. 3.8:

$$a_{tan,i} = |v_i| \cdot \left| \frac{dv}{ds} \right|_i \tag{3.8}$$

And the normal accelerations are calculated for both horizontal and vertical planes:

$$a_{norm,xy,i} = |v_i|^2 \cdot \kappa_{xy,i} \tag{3.9}$$

$$a_{norm,xz,i} = |v_i|^2 \cdot \kappa_{xz,i} \tag{3.10}$$

where κ_{xy} and κ_{xz} represent the curvatures in the xy-plane and xz-plane respectively.

In the specific calculation process, the profiles of these acceleration components are calculated first, and the three acceleration components are transformed into vectors by the direction vectors. The final three-axis acceleration data is generated by summing these three components, where each axis contains contributions from all three physical processes:

$$\mathbf{a}_{tan,i} = a_{tan,i} \cdot \mathbf{e}_{t,i} \tag{3.11}$$

$$\mathbf{a}_{norm,xy,i} = a_{norm,xy,i} \cdot \mathbf{e}_{n,i} \tag{3.12}$$

$$\mathbf{a}_{norm,xz,i} = a_{norm,xz,i} \cdot \mathbf{e}_{b,i} \tag{3.13}$$

The complete acceleration calculation method is shown in Algorithm. 3.2.

Algorithm 3.2: Accelerometer Data Calculation

Input: speed profile $|v|$, track points $\mathbf{p}_i$, curvature κ
Output: **a**

Step 1 Calculate Spatial Derivatives:
Calculate distances: $\Delta s_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
Apply minimum threshold: $\Delta s_i = \max(\Delta s_i, \epsilon)$
Compute cumulative distance: $s = \sum_{j=0}^{i-1} \Delta s_j$
Calculate speed derivative using central difference:
$\frac{dv}{ds}|_i = \frac{v(s+\Delta s)-v(s-\Delta s)}{2\Delta s}$ for interior points
$\frac{dv}{ds}|_{s=s_0} = \frac{v(s_0+\Delta s)-v(s_0)}{\Delta s}$ for first point
$\frac{dv}{ds}|_{s=s_n} = \frac{v(s_n)-v(s_n-\Delta s)}{\Delta s}$ for last point

Step 2 Calculate Direction Vectors:
Calculate tangential direction: $\mathbf{e}_{t,i} = \frac{\mathbf{p}_{i+1}-\mathbf{p}_i}{|\mathbf{p}_{i+1}-\mathbf{p}_i|}$
Calculate normal direction (xy-plane): $\mathbf{e}_{n,i} = \frac{[-e_{t,i,y}, e_{t,i,x}, 0]}{|[-e_{t,i,y}, e_{t,i,x}, 0]|}$
Calculate normal direction (xz-plane): $\mathbf{e}_{b,i} = \mathbf{e}_{t,i} \times \mathbf{e}_{n,i}$

Step 3 Calculate Acceleration Components:
for each point i
Calculate tangential acceleration magnitude:
$a_{tan,mag}[i] = \text{clip}(v[i] \cdot \frac{dv}{ds}[i], -3.0, 3.0)$
Calculate normal acceleration magnitudes:
$a_{norm,xy,mag}[i] = \text{clip}(v[i]^2 \cdot \kappa_{xy}[i], -1.5, 1.5)$
$a_{norm,xz,mag}[i] = \text{clip}(v[i]^2 \cdot \kappa_{xz}[i], -1.5, 1.5)$
Calculate acceleration vectors:
$\mathbf{a}_{tan}[i] = a_{tan,mag}[i] \cdot \mathbf{e}_{t,i}$
$\mathbf{a}_{norm,xy}[i] = a_{norm,xy,mag}[i] \cdot \mathbf{e}_{n,i}$
$\mathbf{a}_{norm,xz}[i] = a_{norm,xz,mag}[i] \cdot \mathbf{e}_{b,i}$
end for

Step 4 Combine Components:
$\mathbf{a} = \mathbf{a}_{tan} + \mathbf{a}_{norm,xy} + \mathbf{a}_{norm,xz}$

Gyroscope Data The gyroscope data also has three axis components. These angular velocities can be calculated using the curvature and velocity profile directly. The yaw angular velocity ω_z is determined by the xy-plane curvature, reflecting the steering motion in the horizontal plane. The pitch angular velocity ω_y is determined by the xz-plane curvature, reflecting the motion caused by elevation changes. The roll angular velocity ω_x considers the banking effect to simulate the influence from track superelevation on the robot. These angular velocity rates are computed by Equation 3.14.

$$\begin{aligned} \omega_z &= |v_i| \times \kappa_{xy,i} \\ \omega_y &= |v_i| \times \kappa_{xz,i} \\ \omega_x &= \omega_{0,i} + 0.1 \times |\omega_z| \end{aligned} \tag{3.14}$$

where $\omega_{0,i}$ is the base noise component.

Coordinate Transformation As mentioned before, the data detected by IMU are usually based on its internal coordinate system rather than a geographical coordinate system, but the simulations described above were generated based on a geographical coordinate system. Therefore, in order to avoid the transformation of the coordinate system from producing data distortions that would unnecessarily affect this study, the transformation of the coordinate system was performed at the end of the data generation.

The specific conversion process is mainly completed by calculating the rotation transformation matrix using Euler angles. The rotation transformation matrix is calculated as shown in Equation. 3.15:

$$\mathbf{R} = \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi) \tag{3.15}$$

Where R_x, R_y, and R_z denote rotations around the x, y, and z axes, respectively. In this section, since it is mainly the IMU data that need to be used, the transformations are mainly performed on the acceleration and angular velocity as shown in Equation. 3.16:

$$\begin{aligned} \mathbf{a}_{\text{IMU}} &= \mathbf{R} \cdot \mathbf{a}_{\text{geo}} \\ \boldsymbol{\omega}_{\text{IMU}} &= \mathbf{R} \cdot \boldsymbol{\omega}_{\text{geo}} \end{aligned} \tag{3.16}$$

Both acceleration and angular velocity can be transformed using the same rotation matrix because they are both vectors that follow the same transformation rules when changing coordinate systems. In this simplified model, we assume that the origin of the IMU coordinate system coincides with the robot coordinate system, eliminating the need for translation components in the transformation. This transformation ensures that the simulated IMU data accurately reflects the way the real sensor perceives motion within its own frame of reference.

In addition, to align with the actual IMU coordinate system convention, the y and z axes data are negated before coordinate transformation.

3.2.3.2 Noise Simulation

After generating the desired IMU data, noise needs to be added to the data. IMU data possesses different kinds of noise as described in the Theory chapter. In this

study, in addition to white noise and long-term bias instability, which can have a major impact on IMU data, simplified track vibration noise is added as environmental vibration.

Sensor Inherent White Noise The inherent white noise is modeled as a Gaussian process of zero mean. For a sensor with sampling frequency f and noise density σ_d, the discrete-time white noise $w[n]$ added to each measurement is formulated as:

$$w[n] \sim \mathcal{N}(0, \sigma_d^2 \cdot f) \tag{3.17}$$

Where $\mathcal{N}(\mu, \sigma^2)$ represents a normal distribution with mean μ and variance σ^2. For this implementation, we selected noise density values of $\sigma_{acc} = 0.002\ m/s^2/\sqrt{Hz}$ for the accelerometer and $\sigma_{gyro} = 0.0001\ rad/s/\sqrt{Hz}$ for the gyroscope.

Long-term bias instability Long-term bias instability was modeled as a random walk process, where the bias at each time step accumulates small random increments. This effectively simulates the drift characteristic of MEMS sensors where errors gradually accumulate over time. The mathematical model is given by:

$$b[n] = b[n-1] + \Delta b[n] \tag{3.18}$$

$$\Delta b[n] \sim \mathcal{N}(0, \sigma_b^2 \cdot \Delta t) \tag{3.19}$$

Where $b[n]$ is the bias at time step n, $\Delta b[n]$ is the random increment, σ_b is the random walk coefficient, and Δt is the time interval between consecutive samples.

The bias instability coefficients were set to $\sigma_{b,acc} = 0.0001\ m/s^2$ for the accelerometer and $\sigma_{b,gyro} = 0.00001\ rad/s$ for the gyroscope, reflecting typical values for medium-grade MEMS IMUs.

Transition Vibration Noise In addition to sinusoidal track-induced vibration, this study introduces a new noise model termed Transition Vibration Noise. It targets the transient oscillatory behavior observed during speed changes, which are typical in mechanical systems experiencing acceleration or deceleration [43].

The method first detects transition regions by computing the rate of change in acceleration, i.e., the derivative of acceleration magnitude. If the value exceeds a predefined threshold δ_{trans}, the system generates a decaying sinusoidal response as follows:

$$a_{\text{trans}}(t) = A_{\text{trans}} \cdot e^{-kt} \cdot \sin(2\pi f_{\text{trans}} t + \phi) \tag{3.20}$$

Where A_{trans} is the amplitude of vibration, k is a damping factor, and f_{trans} is the resonance frequency of the mechanical structure. This noise is added to all three axes proportionally during detected transition intervals.

Realistic Spike Noise To simulate sudden mechanical disturbances typical of braking impacts or track surface irregularities, a new noise type named Realistic Spike Noise is introduced [44]. These spikes occur at locations of high deceleration—detected via acceleration thresholding—and are implemented as sharp transient impulses:

$$a_{\text{spike}}(n) = \begin{cases} A_{\text{spike}} \cdot \mathcal{N}(1, \sigma^2), & \text{if } |a[n]| > \delta_{\text{brake}} \text{ and rand}() < p_{\text{spike}} \\ 0, & \text{otherwise} \end{cases} \tag{3.21}$$

Here, p_{spike} is the probability of spike occurrence and A_{spike} denotes the base magnitude. This simulates real-world IMU outputs during emergency stops or track feature transitions.

Track Vibration To add diversity to the noise, we also included mechanical vibrations specific to railways in our simulations. These vibrations are modeled primarily as sinusoidal disturbances with frequencies corresponding to two primary sources:

1. **Rail Joint Vibrations:** Periodic disturbances caused by rail joints occurring at regular intervals along the track:

$$a_{rail}(t) = A_{rail} \sin(2\pi f_{rail} \cdot t) \tag{3.22}$$

   Where $f_{rail} = \frac{v_{robot}}{d_{joint}}$, with v_{robot} being the robot velocity and d_{joint} the distance between joints.

2. **Wheel Rotation Vibrations:** Higher-frequency disturbances caused by wheel rotation:

$$a_{wheel}(t) = A_{wheel} \sin(2\pi f_{wheel} \cdot t) \tag{3.23}$$

   Where $f_{wheel} = \frac{v_{robot}}{2\pi r_{wheel}}$, with r_{wheel} being the wheel radius.

Since these vibrations are eventually converted into inputs to the accelerometer, the noise generated here is shown in terms of the vibration acceleration measured by the accelerometer, as shown in Equation. 3.24:

$$\begin{bmatrix} a_{vib,x}(t) \\ a_{vib,y}(t) \\ a_{vib,z}(t) \end{bmatrix} = \begin{bmatrix} \alpha \cdot a_{rail}(t) \\ \alpha \cdot a_{rail}(t) \\ a_{rail}(t) + a_{wheel}(t) \end{bmatrix} \tag{3.24}$$

Where $\alpha = 0.3$ represents the reduced impact factor on horizontal axes.

Timestamp Perturbations In addition, timestamp perturbation is introduced to simulate real-world sensor synchronization issues. While the same timestamp is maintained for both accelerometer and gyroscope data within the IMU, a small random perturbation ($\pm 0.5\,ms$) is added to the magnetometer timestamps relative to the IMU. This simulates the temporal misalignment that commonly occurs between different sensor types in practical implementations. The perturbed timestamps are stored alongside the sensor data to facilitate subsequent synchronization studies in the sensor fusion algorithms.

Table. 3.3 summarizes the overall noise addition process for IMU data simulation.

Algorithm 3.3: IMU Noise Simulation

Input: $\mathbf{a}[n]$, $\boldsymbol{\omega}[n]$, f
Output: $\tilde{a}[n]$ (Noisy accelerometer data), $\tilde{\omega}[n]$ (Noisy gyroscope data)

1: Generate white noise
  $w_a[n] \leftarrow \mathcal{N}(0, \sigma_{acc} \cdot \sqrt{f})$ for all n
  $w_\omega[n] \leftarrow \mathcal{N}(0, \sigma_{gyro} \cdot \sqrt{f})$ for all n
2: Generate bias instability
  for n from 1 to length(a):
    $b_a[n] \leftarrow b_a[n-1] + \mathcal{N}(0, \sigma_{b,acc} \cdot \sqrt{dt})$
    $b_\omega[n] \leftarrow b_\omega[n-1] + \mathcal{N}(0, \sigma_{b,gyro} \cdot \sqrt{dt})$
3: Generate vibration noise
  for n from 0 to length(a):
    $t \leftarrow n \cdot dt$
    $a_{rail}[n] \leftarrow A_{rail} \cdot \sin(2\pi \cdot f_{rail} \cdot t)$
    $a_{wheel}[n] \leftarrow A_{wheel} \cdot \sin(2\pi \cdot f_{wheel} \cdot t)$
    $a_{vib,x} \leftarrow \alpha \cdot a_{rail}[n]$
    $a_{vib,y} \leftarrow \alpha \cdot a_{rail}[n]$
    $a_{vib,z} \leftarrow a_{rail}[n] + a_{wheel}[n]$
4: Combine all noise sources
  $\tilde{a}[n] \leftarrow a[n] + w_a[n] + b_a[n] + a_{vib}[n]$ for all n
  $\tilde{\omega}[n] \leftarrow \omega[n] + w_\omega[n] + b_\omega[n]$ for all n
5: **return** $\tilde{a}$, $\tilde{\omega}$

3.2.4 Magnetometer Data Simulation

This section describes the methodology used to generate high-frequency synthetic magnetometer data based on real-world kinematic inputs. The simulation aims to reproduce realistic magnetic field variations experienced by sensors mounted on a moving platform, including spatial disturbances, noise characteristics, and sensor geometry.

3.2.4.1 Kinematic Data Preprocessing and Interpolation

The original input consists of discrete position and velocity measurements sampled at relatively low frequency. To match the target magnetometer sampling rate (e.g.,

100 Hz), cubic spline interpolation is applied to all kinematic variables (position, velocity, and speed profile). This interpolation step generates a high-resolution trajectory with uniformly spaced timestamps, enabling more detailed and temporally consistent magnetic field simulation.

Synthetic timestamps are derived from the cumulative distance traveled divided by the instantaneous speed, creating a realistic temporal scale that captures variable velocity profiles. The interpolation and timestamping together ensure smooth and dense motion data input, which is critical for spatially coherent magnetic field and noise simulation.

3.2.4.2 Sensor Placement Modeling

Two sensors are simulated with fixed spatial offsets along the velocity vector to emulate front and rear installations on a rail vehicle. This arrangement captures time-phased magnetic field variations between sensors, a common approach in magnetic localization systems [49]. A fixed spatial offset of ± 0.5 meters is applied to emulate front and rear sensor placement, assuming forward direction without trajectory-based orientation.

3.2.4.3 Spatial Magnetic Field Modeling

The magnetic field at each sensor is computed using a synthetic base model composed of multiple sinusoidal functions of position. Each axis is defined as a weighted sum of sine and cosine terms with varying frequencies. This allows the generated field to exhibit rich spatial variation over time.

This formulation helps simulate complex but repeatable spatial magnetic features such as infrastructure-induced field distortions without directly modeling the global geomagnetic field.

Algorithm 3.4: Magnetic Field Synthesis (Updated Model)

Input: $x[n]$ (1D position array)
Output: $B_x[n]$, $B_y[n]$, $B_z[n]$

1: Compute position-dependent signal:
   $B_x = \sum_{i=1}^{3} A_i \cdot \sin(2\pi x[n]/\lambda_i) + A_i \cdot \cos(2\pi x[n]/\lambda_i)$
   $B_y =$ similar multi-sine expression
   $B_z =$ similar multi-sine expression
2: Apply directional correction for rear sensor:
   $B_y^{\text{rear}} \leftarrow -B_y^{\text{rear}}$, $B_z^{\text{rear}} \leftarrow -B_z^{\text{rear}}$

3.2.4.4 Noise Model for Sensor Emulation

To replicate realistic magnetometer sensor behavior, four types of noise are independently added to each magnetic field axis:

- Gaussian noise: Represents high-frequency random measurement fluctuations and electronic sensor noise.
- Temperature drift: Modeled as a sinusoidal function with a 24-hour period to simulate diurnal temperature effects on sensor output.
- Aging drift: A low-frequency sinusoid with an annual period, representing long-term sensor characteristic changes.
- Spatial jitter: High-frequency spatially dependent fluctuations modeled as a sinusoidal function of cumulative traveled distance, simulating fine-scale environmental variability.
- Spike noise: Occasional, discrete amplitude jumps with low probability, simulating transient measurement outliers often caused by external electromagnetic interference.

These noise components combine to reflect the composite nature of real magnetometer signals, which exhibit both temporal and spatial variability across multiple frequency bands [47, 48].

Algorithm 3.5: Magnetometer Noise Simulation

Input: $B_x[n], B_y[n], B_z[n]$, $t[n]$, $d[n]$ (time, cumulative distance) **Output:** $\tilde{B}x[n]$, $\tilde{B}y[n]$, $\tilde{B}z[n]$

1: Add Gaussian noise: $\mathcal{N}(0, \sigma)$
2: Add temporal drift:
Temperature drift $\propto \sin(2\pi t[n]/Tday)$ Aging drift $\propto \sin(2\pi t[n]/Tyear)$ 3: Add spatial jitter: $\propto \sin(2\pi d[n]/\lambda jitter)$
4: Add random spike noise with small probability
5: Combine: $\tilde{B}_i[n] = B_i[n] +$ noise components for $i \in x, y, z$

3.2.4.5 Field Magnitude Calculation and Visualization

After adding noise, the total magnetic field magnitude is computed using the Euclidean norm:

$$B_{\text{total}} = \sqrt{B_x^2 + B_y^2 + B_z^2} \tag{3.25}$$

3.2.4.6 Data Export

The complete simulated dataset, including noisy vector components and magnitude at the target sampling frequency, is exported in CSV format for use in subsequent algorithm development, testing, and visualization.

3.2.5 Velocity Estimation from Magnetometer Signals

To obtain a velocity estimate independent of inertial data, we implemented a magnetometer-based estimation approach that relies on the time delay between signals from front and rear sensors. When the robot moves forward, both sensors detect similar magnetic features at different times. By computing the delay between matched patterns in these signals, the velocity can be estimated based on known sensor spacing.

3.2.5.1 Estimation Principle

Let $B_f[n]$ and $B_r[n]$ be the magnetometer signals from the front and rear sensors, respectively. In our implementation, we use only the x-axis component of the magnetic field, as it sufficiently captures the forward-direction variations.

The lag $\tau[n]$ is computed by minimizing the mean squared error (MMSE) between a trailing window of front and rear signals. This approach improves temporal locality and avoids artifacts from using future information, making it suitable for real-time applications.

If the lag is successfully estimated, the velocity is calculated as:

$$v[n] = \frac{d}{\tau[n] \cdot \Delta t} \tag{3.26}$$

where d is the sensor spacing, and Δt is the sampling interval.

Algorithm 3.6: 1D Magnetometer-Derived Velocity Estimation (MMSE Method)

Input: $B_{xf}[n]$, $B_{xr}[n]$ (x-axis signals), d, Δt, window size W, lag range $[\ell_{\min}, \ell_{\max}]$
Output: $\hat{v}[n]$ (estimated velocity)

For each time step $n \geq W$:
  Extract trailing window: $S_f = B_{xf}[n - W : n]$, $S_r = B_{xr}[n - W : n]$
  For $\ell = \ell_{\min}$ to $\ell_{\max}$:
    Align S_f and S_r by lag ℓ
    Compute MMSE: $E[\ell] = \text{mean}[(S_f[: -\ell] - S_r[\ell :])^2]$
  Select $\tau[n] = \arg\min_\ell E[\ell]$
  If estimated lag is valid and acceleration $< a_{\text{thresh}}$:
    $\hat{v}[n] = \frac{d}{\tau[n] \cdot \Delta t}$
  Else:
    $\hat{v}[n] =$ None (skip estimation)

3.2.5.2 Parameter Configuration

In this implementation, the sensor distance d is set to 1.0,m, and the sampling interval $\Delta t = 1/100$,s. A trailing window of 800 samples is used, with a sliding step size of 20. The lag is searched over a range from 5 to 130 samples.

To suppress unreliable estimates during rapid acceleration or deceleration, a low-pass filtered acceleration signal is used to conditionally disable estimation when the absolute acceleration exceeds 0.1,m/s^2. This improves the robustness of velocity detection during dynamic events.

In scenarios involving backward motion, the signal ordering must be reversed before lag computation, i.e., the rear signal should be treated as leading. In this work, we focus on forward motion, and leave bidirectional extension to future implementations.

3.3 EKF-Based Sensor Fusion Implementation

Based on the velocity estimation described in the previous section, an EKF is employed for sensor fusion using simulated accelerometer data and velocity estimates derived from magnetometer measurements. This approach generates velocity through acceleration integration and refines these estimates through fusion with magnetometer-derived velocity corrections.

3.3.1 System Model and State Definition

The EKF design is based on traditional navigation systems, such as SLAM, since the sensor fusion module is intended to serve as a component of such navigation systems in GPS-denied environments. Therefore, the state vector follows the conventional SLAM system architecture, although this study primarily focuses on the estimation of the velocity, as shown in Equation. 3.27.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \\ \mathbf{a}_k \end{bmatrix} \tag{3.27}$$

where $\mathbf{p}_k$ represents the position of the robot, $\mathbf{v}_k$ is the velocity along the motion direction, and $\mathbf{a}_k$ is the the acceleration measured at time step k. The initial settings for position, velocity, and accelerometer bias are all set to zero.

Considering that the magnetometer estimation primarily analyzes the robot motion along the traveling direction, the system employs a simplified linear state transition model as the process model. This simplification avoids complex nonlinear Jacobian calculations while maintaining sufficient modeling accuracy to capture the motion characteristics.

The linear process model is based on classical kinematic integration equations, where position is obtained through velocity integration, velocity is calculated through acceleration integration, and the acceleration is assumed to remain the same as the previous time step since we have no information about how the acceleration changes in the prediction step. The process model is shown as Equation. 3.28.

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{v}_{k-1} \\ \mathbf{a}_{k-1} \end{bmatrix} \tag{3.28}$$

Specifically, the prediction phase implements:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, u_{k-1}) \tag{3.29}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{Q} \tag{3.30}$$

where $\mathbf{F}_k = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1}, u_{k-1}}$ is the state transition matrix of the process model, $\mathbf{P}$ is the covariance matrix, and $\mathbf{Q}$ is the noise matrix of the process.

And in the update phase:

$$\mathbf{y}_k = z_k - h(\hat{\mathbf{x}}_{k|k-1}) \tag{3.31}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R} \tag{3.32}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1} \tag{3.33}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{y}_k \tag{3.34}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{3.35}$$

where z_k is the observation measurement, $\mathbf{H}_k = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k|k-1}} = [1 \quad 0]$ is constant, and $\mathbf{K}_k$ is the Kalman gain.

3.3.2 Dual-Mode Observation Strategy

Since the magnetometer velocity estimation does not consistently provide better results, the EKF implements two operational modes to evaluate the performance of different sensor combinations for observation: IMU-only mode and dual-sensor fusion mode.

Mode 1 - IMU-only Operation: The observation model focuses solely on accelerometer measurements:

$$\mathbf{z}_k = h(\mathbf{x}_k) = \mathbf{a}_k \tag{3.36}$$

with observation matrix:

$$\mathbf{H}_{\text{single}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{3.37}$$

Mode 2 - IMU+Magnetometer Fusion: The observation model incorporates both magnetometer-derived velocity and IMU acceleration:

$$\mathbf{z}_k = h(\mathbf{x}_k) = \begin{bmatrix} \mathbf{v}_k \\ \mathbf{a}_k \end{bmatrix} \tag{3.38}$$

with observation matrix:

$$\mathbf{H}_{\text{bi}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.39}$$

In addition, adaptive switching between different modes must be addressed. The magnetometer-derived velocity estimation may not provide valid results at each time step, so the fusion mode incorporates a data validation mechanism. When the estimated velocity value is NaN or None, both the observation Jacobian matrix **H** and the observation noise matrix **R** are switched to the IMU-only configuration, preventing accuracy degradation that could cause filter instability or failure. This adaptive mechanism is primarily controlled by temporal synchronization constraints.

3.3.3 Parameter Configuration and Tuning

Table. 3.7 summarizes the key parameters used in the EKF implementation.

Table 3.7: EKF Parameter Configuration

Type	Parameter	Final Value	Tuning Range
Initial Covariance $\mathbf{P}_0$	σ_p^2	1.0 m^2	0.1 - 10.0 m^2
	σ_v^2	0.3 $(m/s)^2$	0.1 - 1.0 $(m/s)^2$
	σ_a^2	0.2 $(m/s^2)^2$	0.05 - 0.5 $(m/s^2)^2$
Measurement Noise R	R_{mag}	0.5 $(m/s)^2$	0.1 - 4.0 $(m/s)^2$
	R_{acc}	0.05 $(m/s^2)^2$	0.01 - 0.1 $(m/s^2)^2$
Process Noise Q	q_{pos}	1×10^{-4}	$10^{-5} - 10^{-3}$
	q_{vel}	1×10^{-1}	$10^{-2} - 1$
	q_{acc}	1×10^{0}	$10^{-1} - 10$
Synchronization Parameters	$T_{\max}$	0.01 s	0.005 - 0.02 s
	Δt	0.01 s	Fixed (100 Hz)

The parameter tuning process followed systematic design principles to balance the accuracy of the estimation and the computational efficiency:

- The **initial covariance matrix $\mathbf{P}_0$** represents prior uncertainty about the initial state, with higher values that allow faster adaptation, but potentially cause instability during filter initialization.
- The **measurement noise R** parameters directly influence the fusion weights between the sensors, where higher magnetometer noise (R_{mag}) reduces the reliance on estimates of magnetic velocity during dynamic motion phases.
- The **process noise Q** components control the trade-off between the responsiveness of the tracking and the smoothness, the noise from the position process (q_{pos}) kept minimal to prevent drift accumulation, while the noise from the velocity and acceleration allow adaptation to motion changes.
- The **temporal synchronization threshold** ($T_{\max}$) ensures data quality by rejecting misaligned sensor measurements that could degrade fusion performance.

3.3.4 Implementation Summary

The EKF implementation follows the standard prediction update framework, but is optimized for railway application environments. The linear state transition model avoids real-time Jacobian matrix calculations, reducing computational complexity and enabling real-time execution on embedded systems.

The observation matrix employs dynamic switching mechanisms that allow a single filter to handle different sensor configurations, providing flexibility for incorporating additional sensor types in future implementations.

Details of the EKF implementation are presented in Algorithm. 3.8.

Algorithm 3.8: Extended Kalman Filter for Velocity Estimation

Input: $\mathbf{a}_k$, $\mathbf{v}_k^{\text{mag}}$, Δt, `BI_FUSION` flag
Output: $\hat{\mathbf{x}}_k$ (estimated state $[\mathbf{p}, \mathbf{v}, \mathbf{a}]^T$)

1: Initialize: $\hat{\mathbf{x}}_0 = [0, 0, 0]^T$, $\mathbf{P}_0 = \text{diag}(\sigma_p^2, \sigma_v^2, \sigma_a^2)$
2: For each time step $k = 1, 2, ..., N$
Prediction Step:
$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$
$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1}$
$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$
Update Step:
If (`BI_FUSION` and $\mathbf{v}_k^{\text{mag}}$ is valid):
$\mathbf{z} = [\mathbf{v}_k^{\text{mag}}, \mathbf{a}_k]^T$
$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$\mathbf{R}_{\text{use}} = \mathbf{R}_{\text{bi}}$
Else:
$\mathbf{z} = [\mathbf{a}_k]$
$\mathbf{H} = [0, 0, 1]$
$\mathbf{R}_{\text{use}} = \mathbf{R}_{\text{single}}$
$\mathbf{y} = \mathbf{z} - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}$ (Innovation)
$\mathbf{S} = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_{\text{use}}$ (Innovation covariance)
$\mathbf{K} = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}^{-1}$ (Kalman gain)
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}\mathbf{y}$
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_{k|k-1}$

3.4 Proposed Complementary Fusion Strategy

In conventional pipelines, velocity estimated from the magnetometer and acceleration from the IMU are fused directly via an Extended Kalman Filter (EKF). However, we propose a lightweight intermediate fusion method based on a complemen-

tary filter, to be used prior to the EKF update. This method aims to enhance stability and reduce the effect of transient errors in either source.

The fused velocity is computed by blending the magnetometer-based and IMU-based velocity estimates using a weighted sum:

$$v_{\text{fused}} = \alpha \cdot v_{\text{mag}} + (1 - \alpha) \cdot v_{\text{imu}}, \quad \alpha \in [0, 1] \tag{3.40}$$

where:

- v_{mag} is the velocity obtained from magnetometer lag correlation (Section 3.2.5),
- v_{imu} is obtained by integrating IMU acceleration (Section 3.2.3),
- α is a tunable fusion gain that controls the contribution of each source.

The proposed implementation can be summarized as follows:

- Step 1: Compute $v_{\text{imu}}(t)$ by integrating filtered acceleration data.
- Step 2: Estimate $v_{\text{mag}}(t)$ from the lag-based method.
- Step 3: Apply the complementary fusion formula with gain α (empirically set or adaptive).
- Step 4: Use v_{fused} as the input velocity observation for the EKF.

This strategy serves to reduce noise propagation into the EKF and smooth out erratic measurements in scenarios with temporary magnetic distortion or IMU drift.

Note: This fusion module was not implemented in the current experiment but is proposed as a potential future extension to improve the robustness and computational efficiency of the system.

4

Results

4.1 Simulation Results and Validation

This section presents the simulated magnetometer outputs described in the methodology. Visualizations of total field magnitude and axis-specific components are provided for both front and rear sensors. These results illustrate the spatial and temporal variation of the synthetic magnetic field and demonstrate how noise modeling contributes to realistic signal dynamics during vehicle motion.

4.1.1 Magnetometer data simulation

Field Magnitude Calculation and Visualization

After adding noise, the total magnetic field magnitude is computed using the Euclidean norm:

$$B_{\text{total}} = \sqrt{B_x^2 + B_y^2 + B_z^2} \tag{4.1}$$

Figure 4.1 shows the total magnetic field strength from both front and rear sensors over time. As expected, the signals exhibit high-frequency variations, with observable lag between front and rear curves. The overall amplitude remains within the tens to hundreds of μT, matching typical indoor or near-track field magnitudes.

The synthetic signal includes complex multi-frequency components and injected noise, simulating a realistic magnetic environment. The repeating patterns enable temporal correlation, which is essential for downstream velocity estimation.

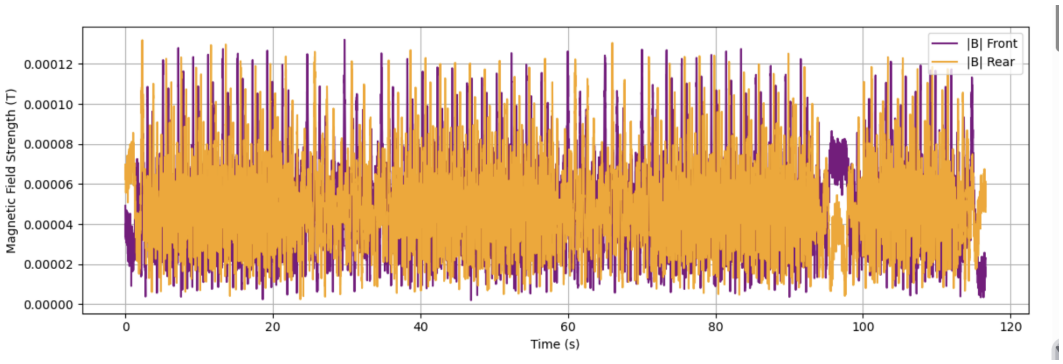

Figure 4.1: Total Magnetic Field: Front and Rear Sensors

Due to the high sampling rate and dense signal structure, the full-length waveform appears visually crowded, making it difficult to observe the local alignment between the front and rear signals. To better illustrate the phase difference and repeating patterns, a zoomed-in segment of the magnetic field is shown in Figure 4.2.

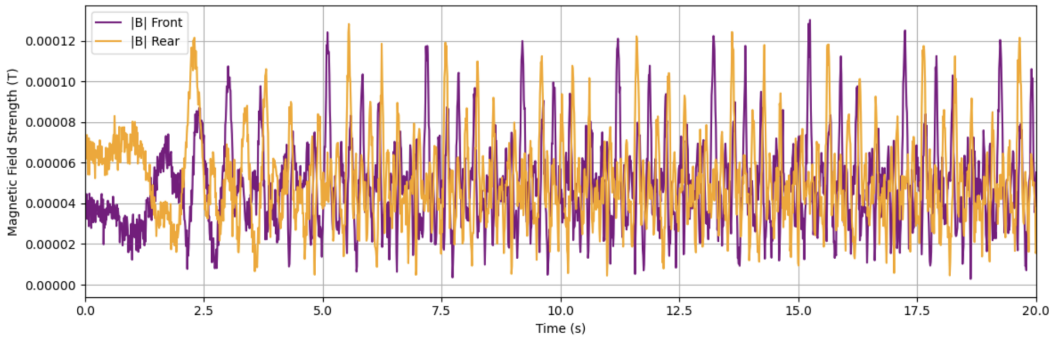

Figure 4.2: Zoomed-In Segment of Total Magnetic Field

Magnetometer Signal Components

Figures 4.3 and 4.4 display the individual x, y, and z components of the simulated magnetic field. The waveforms show significant local variation and complex oscillatory structure, as a result of high-frequency spatial modulation and synthetic noise injection.

Importantly, the y and z components of the rear sensor appear inverted compared to the front sensor. This is consistent with our simulation setup, where the rear magnetometer's local coordinate frame is mirrored to reflect typical sensor mounting on railway vehicles.

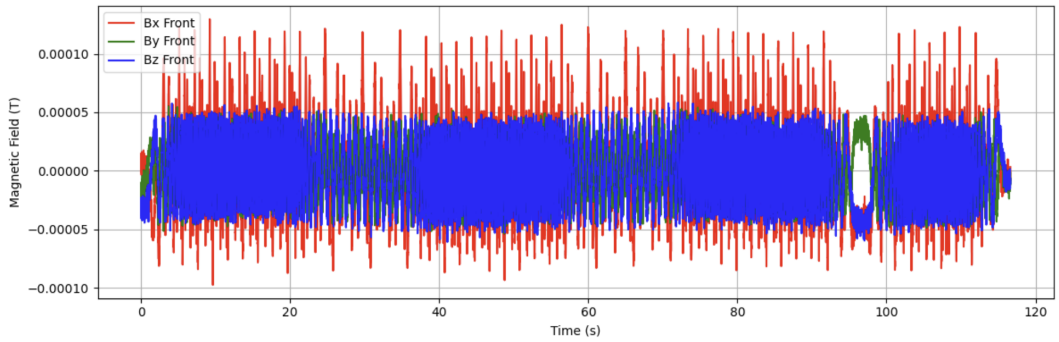

Figure 4.3: Front Sensor Magenetic Field Components

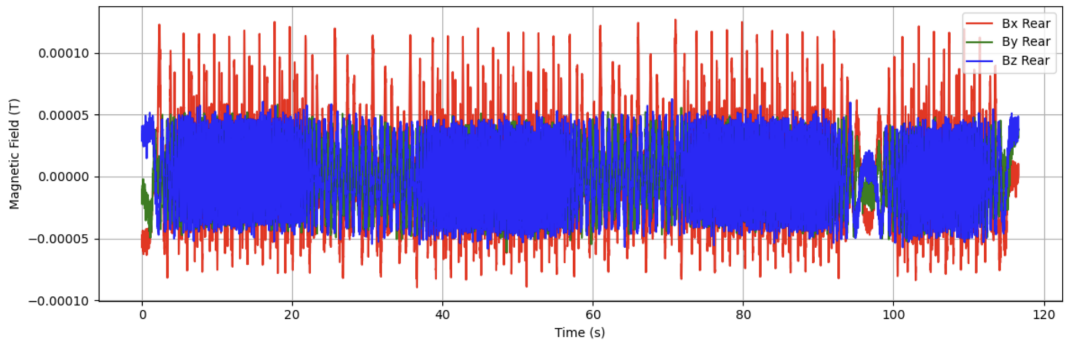

Figure 4.4: Rear Sensor Magnetic Field Components

Comparison with Real Magnetometer Data

To validate the simulated magnetic field behavior, the real-world magnetometer readings from front and rear sensors were also plotted for the x, y, and z axes.

As shown in Figures 4.5 and 4.6, the overall waveform patterns and value ranges of the real signals closely match the simulated ones. The front sensor shows smooth oscillatory patterns, while the rear sensor exhibits a clear inversion in the y and z axes—consistent with the mirrored coordinate configuration modeled in simulation.

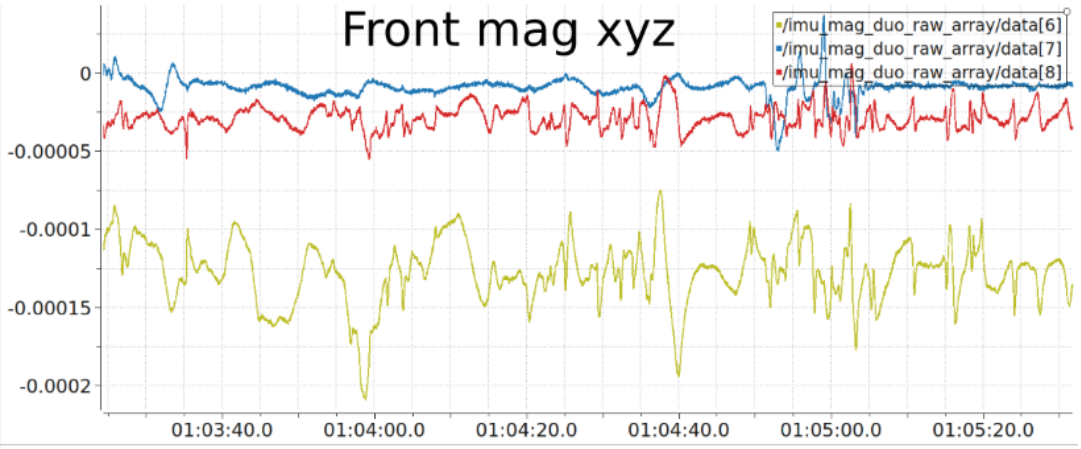

Figure 4.5: Real Magnetometer Components: Front Sensor

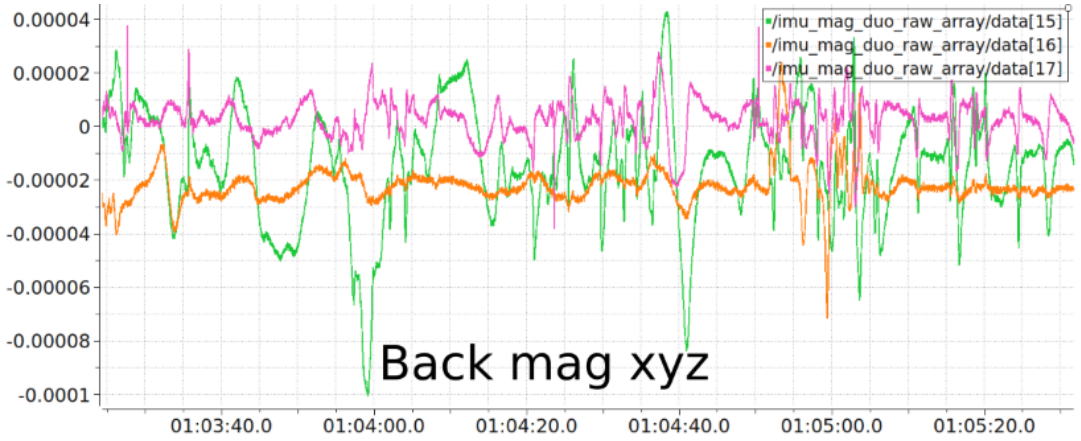

Figure 4.6: Real Magnetometer Components: Rear Sensor

4.1.2 IMU data simulation

To validate the simulated inertial signals, two sets of plots were generated for angular velocity (gyroscope) and linear acceleration. These illustrate the behavior of the synthetic IMU under the prescribed motion trajectory.

Gyroscope Signals

Figure 4.7 shows the angular velocity along the x, y, and z axes. While the overall angular rates are relatively low due to the predominantly linear motion, small oscillations and noise patterns are visible throughout the sequence.

Notably, a mild upward trend appears in the x-axis during the latter portion of the trajectory, corresponding to a simulated curved segment in the track. Distinct vibration spikes are also present shortly after each speed transition, intended to simulate the impact-like events observed during real-world deceleration.

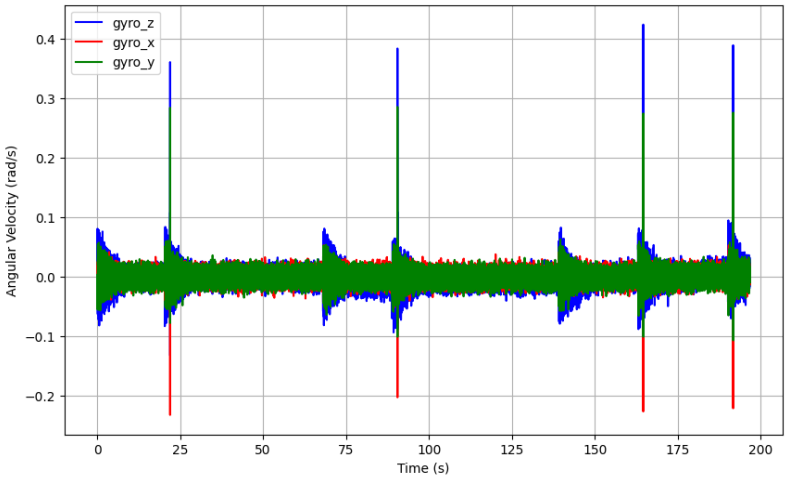

Figure 4.7: Angular Velocity simulation

Acceleration Signals

Figure 4.8 presents the linear acceleration on all three axes. The z-axis exhibits values oscillating around 9.8 m/s², consistent with gravity plus added vertical vibration noise. These fluctuations fall within the expected range of 9.8–10 m/s².

The x-axis clearly reflects the robot's motion phases: acceleration and braking appear as sustained positive and negative peaks, respectively. This axis also shows significant friction-induced noise, likely due to modeled mechanical contact during movement.

The y-axis exhibits less prominent variations but still indicates some lateral dynamics, possibly caused by minor trajectory deviations or simulated rail unevenness.

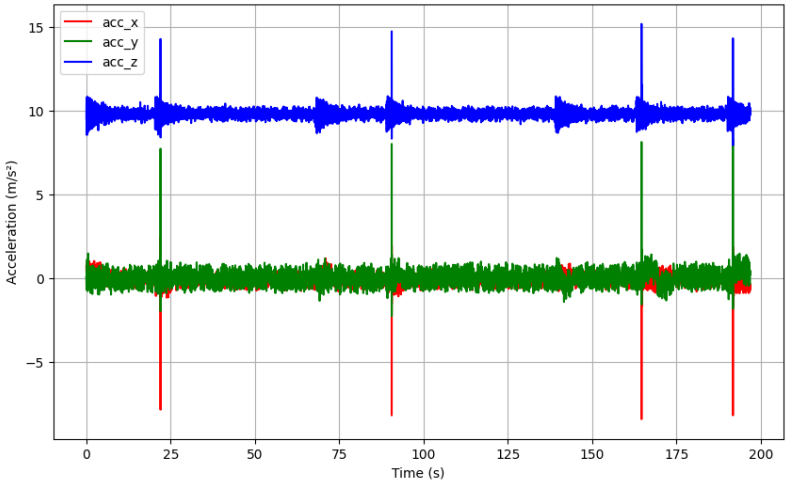

Figure 4.8: Acceleration Simulation

Comparison with Real IMU Data

To further assess the realism of the simulated inertial signals, real-world IMU measurements were recorded along the same motion trajectory and plotted for both angular velocity and linear acceleration.

Figures 4.9 and 4.10 show that the simulated data align closely with the real IMU output in both magnitude and waveform. The angular velocity shows similar noise levels and overall trends, and the acceleration signals exhibit expected gravity-aligned bias on the z-axis and clear changes in the x-axis reflecting forward motion. This confirms that the trajectory in simulation closely replicated the real vehicle movement.

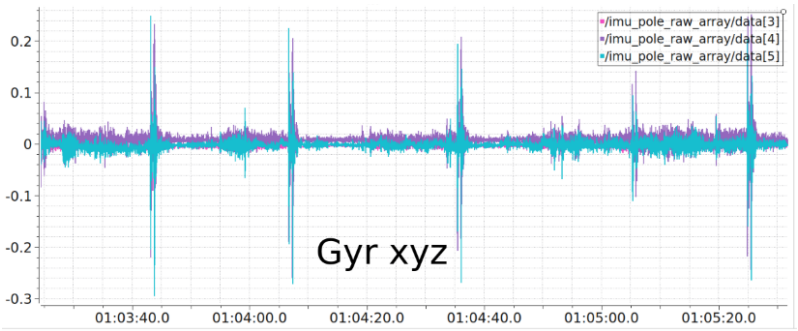

Figure 4.9: Real Angular Velocity

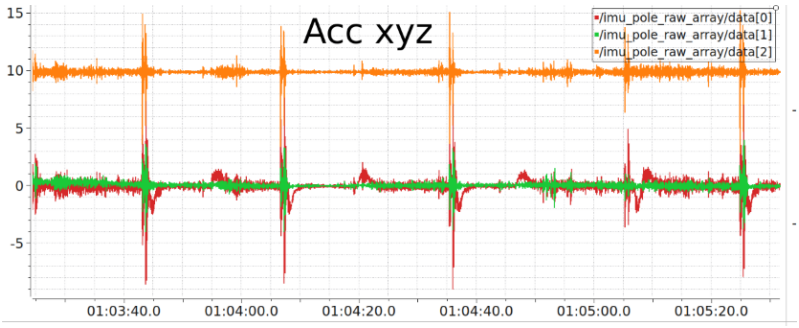

Figure 4.10: Real Linear Acceleration

4.2 Velocity Estimation from Magnetometer Data

The velocity was estimated using the 1D lag-matching method described in Section 3.3. Only the x-axis component of the magnetometer signal was used, as it sufficiently captures the magnetic variation in the direction of motion. This simplification reduces computational complexity and focuses on the axis most relevant for forward velocity.

Figure. 4.11 shows the estimated velocity (purple) alongside the true simulated velocity (black dotted) and filtered acceleration signal (orange). The acceleration curve

is included to indicate dynamic changes and help interpret where lag estimation is valid.

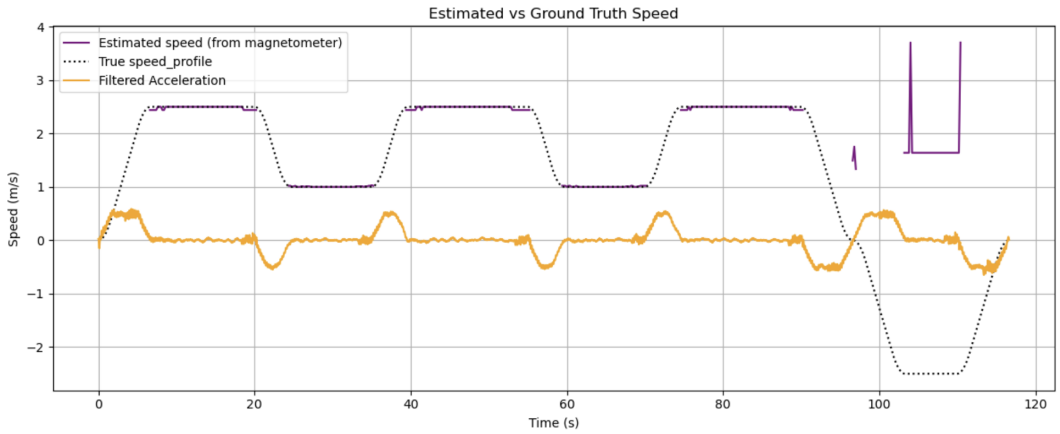

Figure 4.11: Estimated velocity from magnetometer signals (purple), filtered acceleration (orange), and ground truth velocity (dotted black).

As seen in the figure, the estimated velocity successfully captures the main structure of the true velocity waveform. In particular, the estimated values align well with the peak and valley plateaus of each motion segment. Although the estimation is not continuous—due to selective computation during low-acceleration periods—the resulting segments exhibit high accuracy in both magnitude and timing.

It is also observed that during the final deceleration phase, the true speed becomes negative (due to the robot moving backward). The estimated speed, however, does not track this reversal. This is expected, as the current implementation does not handle backward motion explicitly. A note on this limitation is included in the discussion.

4.3 Performance of EKF

4.3.1 Velocity Estimation with IMU-only EKF

Figure 4.12 shows the estimated velocity from the EKF using only IMU acceleration data, compared with the ground truth velocity.

As seen in the plot, the EKF performs well in tracking the velocity profile, successfully capturing the main motion phases including acceleration, cruising, and deceleration. The estimated velocity follows the ground truth closely, with stable plateaus during constant-speed segments and reasonable transitions between different phases.

Small fluctuations are visible throughout the estimation, particularly during the constant-speed intervals. These disturbances can be attributed to noise in the accel-

eration measurements, which primarily result from the simulated railway vibrations and mechanical effects included in our noise model.

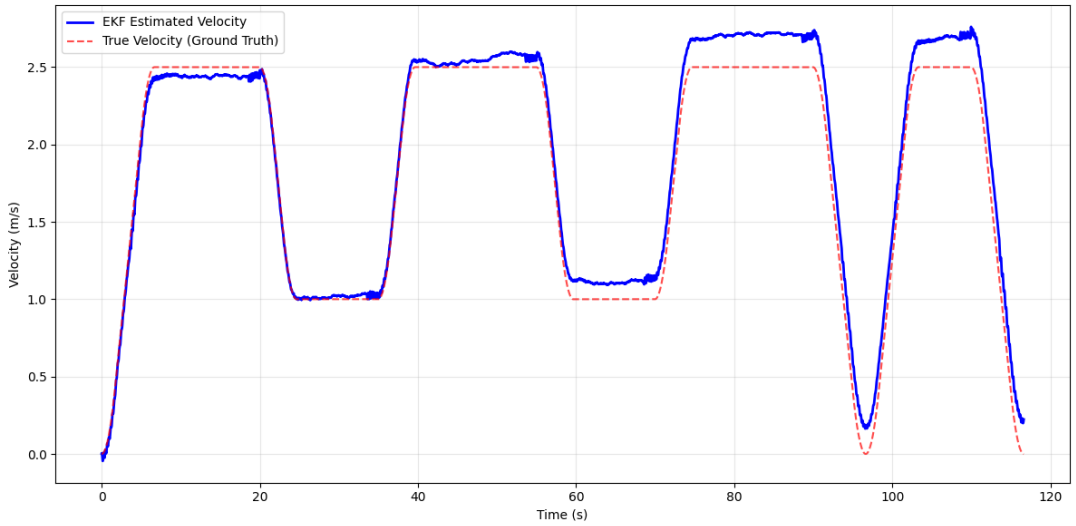

Figure 4.12: EKF Velocity Estimation using only acceleration data.

4.3.2 Velocity Estimation with IMU and Magnetometer-Fused EKF

Figure 4.13 presents the estimated velocity when magnetometer-derived velocity estimates are also included in the EKF.

Compared with the previous result, the velocity curves in this configuration show better overall performance, especially in maintaining accuracy during the later segments where the IMU-only approach suffered from significant drift. The magnetometer-fused EKF successfully keeps the velocity estimates close to the ground truth throughout the entire trajectory.

However, the inclusion of magnetometer data introduces noticeable oscillations during the constant-speed segments. These fluctuations are caused by noise in the magnetometer-derived velocity measurements being fed into the filter. Despite these oscillations, the trade-off appears worthwhile as it prevents the severe drift that occurred in the IMU-only case, particularly during the final phases of the motion.

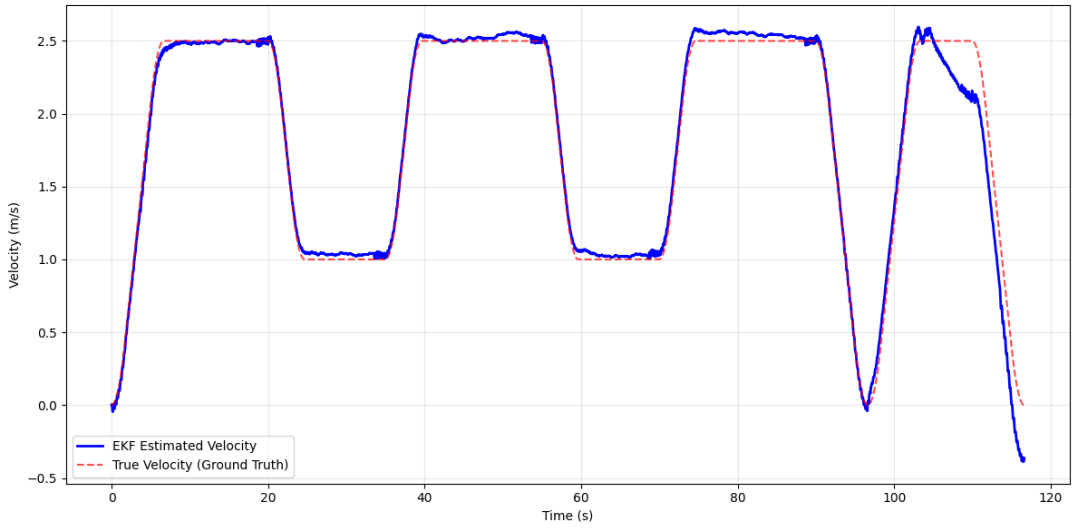

Figure 4.13: EKF Velocity Estimation using IMU data and magnetometer-derived velocities.

5

Discussion

5.1 Magnetometer Field Simulation

5.1.1 Modeling Approach and Sinusoidal Representation

The synthetic magnetic field was constructed using a weighted sum of sine and cosine functions with varying spatial frequencies. This structure simulates the spatial complexity of real magnetic environments near rail infrastructure, including oscillations caused by repetitive metallic structures and environmental anomalies.

This modeling technique is aligned with prior research leveraging Fourier-based representations of ambient magnetic fields [51]. The use of multiple frequency components allows us to generate rich field variations necessary for effective lag detection.

The parameter ranges were selected to produce distinguishable waveform shifts over the 1-meter baseline between sensors, while ensuring physical realism with total field values in the range of 30–120 μT, typical for urban or semi-indoor settings [52].

5.1.2 Noise Design vs. Realistic Disturbances

To reflect sensor and environmental imperfections, several types of noise were added to each field component:

- Gaussian white noise simulating measurement jitter
- Temperature and aging drift using long-period sine waves
- Spatial jitter induced by small-scale distance changes
- Sparse spike noise to emulate intermittent EM interference

This design produces complex yet analyzable signals suitable for algorithm validation. Notably, we avoided incorporating extreme transient spikes—such as those caused by ferromagnetic debris or abrupt hardware shocks—as these can severely disrupt lag estimation [54].

Instead, we maintained moderate, structured noise to allow stable correlation while

preserving realism. The resulting signals exhibit similar structure and amplitude to real-world measurements, as validated through visual comparison with recorded datasets.

5.2 Magnetometer-Derived Velocity Estimation

5.2.1 1D Signal Selection and Simplification

In contrast to the previous 3D correlation-based method, the revised implementation uses only the x-axis magnetic field. This simplification was made based on the observation that magnetic field variation along the motion direction is sufficient for lag estimation, while reducing noise from orthogonal axes.

The magnetic field function was also updated to include more high-frequency components, producing sharper signal transitions that improve lag resolution. These changes lead to more stable and interpretable lag matching across multiple velocity phases.

5.2.2 Improved Real-Time Compatibility

The velocity estimation now uses a trailing window and only processes lag when enough past data is collected. This ensures causal computation, suitable for real-time deployment. Unlike the original version, which used future samples, this updated version only looks backward in time, making it more physically realistic.

Moreover, to prevent corrupted estimates during high dynamics, an acceleration threshold is applied. If the filtered acceleration exceeds a set limit, lag estimation is skipped at that frame. This explains the intermittent nature of the estimated velocity curve and avoids spurious matches.

The filtered acceleration plot, shown in Figure 4.11, highlights periods of low dynamic change where estimation is reliable. It also helps interpret why the velocity curve appears segmented—because estimation is suppressed during rapid acceleration or deceleration.

5.2.3 Accuracy of Plateau Matching

While the estimated velocity lacks smooth transitions between motion segments, it very accurately aligns with the constant-speed plateaus in the ground truth signal. This confirms that the lag-based method effectively captures stable motion intervals. However, since it does not interpolate between those segments, transitions are missing in the final output.

This segmented behavior results in a staircase-like appearance in the estimated velocity. Each flat region corresponds to a successfully matched lag during a low-acceleration interval.

5.2.4 Limitations in Reversing Motion

As seen in the final segment of the velocity plot, the true velocity turns negative, indicating the robot is moving backward. The estimated velocity does not reflect this reversal, as the lag matching function assumes forward motion. In principle, this could be corrected by swapping the order of the signals passed to the lag estimation function. This limitation is noted but not addressed in the current implementation.

5.3 IMU Signal Behavior

The gyroscope and accelerometer plots reveal several characteristics resulting from both the underlying motion profile and the noise models introduced during simulation.

5.3.1 Gyroscope Signal Interpretation

The mild upward drift observed in the x-axis angular velocity during the second half of the trajectory corresponds to a simulated turning segment in the track. This was intentionally included to emulate the behavior of a robot entering a curved section, leading to persistent yaw-rate activity.

The sharp peaks following each speed transition are due to the *Realistic Spike Noise* model described in Section 3.2.3.2. These simulate braking shocks or mechanical jolts, typically observed in real IMU logs during deceleration. Their sparse but intense appearance is consistent with high-impact dynamics such as suspension snapback or wheel-rail shock.

5.3.2 Accelerometer Signal Interpretation

The z-axis oscillations cluster around $9.8\,\mathrm{m/s^2}$, dominated by gravitational acceleration. The slight variations result from both vertical structural vibration and the *Transition Vibration Noise* model introduced during acceleration changes. These short-lived damped sinusoidal bursts simulate the dynamic response of the chassis under force changes.

The x-axis reflects sustained dynamics from robot propulsion and braking. Unlike sharp spikes, the acceleration pattern here includes longer segments with steady values, corresponding to motion phases. Noise components added during simulation—especially vibration and frictional disturbances—cause sustained fluctuations in this axis, more visible than in y.

The y-axis signal is comparatively quieter but still shows lateral activity. This may stem from simulated lateral instability or unbalanced track geometry.

Together, these features demonstrate how realistic noise modeling and dynamic event simulation contribute to the fidelity of IMU signal generation.

5.4 Impact of Magnetometer Integration on EKF Estimation

The performance of the Extended Kalman Filter (EKF) was evaluated under two configurations: using only IMU acceleration as input, and using both IMU and magnetometer-derived velocity as measurements.

In the IMU-only setup, the EKF was able to estimate the general motion profile of the vehicle. The resulting velocity curves followed the ground truth in curvature, with only minor discrepancies in magnitude and axis orientation. The presence of smooth plateaus in the estimated velocity suggests the filter effectively integrated the accelerative behavior during constant-speed segments.

To further improve the estimation accuracy, we then incorporated the magnetometer-derived velocity as an additional observation in the EKF measurement model.

This dual-sensor configuration showed better performance in terms of velocity estimation range and overall tracking accuracy. Although the noisy characteristics of the magnetometer-derived velocity estimates introduced some oscillations into the EKF results, the overall velocity tracking was significantly improved compared to the IMU-only approach. This demonstrates that enhancing the quality of magnetometer-derived velocity estimation would directly translate to better EKF performance.

Overall, the inclusion of magnetometer measurements demonstrates both the sensitivity and adaptability of the EKF framework. While helpful in theory, the quality of the reference signal plays a critical role in determining the final estimation performance. Improvements in the magnetic speed estimation pipeline—such as denoising, confidence weighting, or gating based on dynamic conditions—could allow more reliable integration in future versions.

5.5 Complementary Fusion Strategy: Feasibility and Potential

As described in Section 3.2.7, we propose a lightweight velocity fusion strategy based on complementary filtering, intended as a future enhancement to the current EKF pipeline. Although not implemented in this study, this method presents a theoretically sound and computationally efficient approach to mitigating the limitations of direct fusion.

Complementary filtering has been widely applied in inertial navigation, particularly in attitude estimation, due to its simplicity and robustness [57]. Its extension to velocity fusion allows the system to benefit from the stable low-frequency characteristics of magnetometer-derived velocity estimation and the responsive high-frequency dynamics of IMU-derived velocity.

In practical deployment scenarios, where computational resources are limited or real-time response is required (e.g., embedded systems on rail vehicles), this approach could reduce the burden on the EKF, improve convergence stability, and suppress transient noise bursts prior to state correction. Prior studies [58] have demonstrated that such filter architectures perform well in hybrid IMU-magnetometer systems, suggesting strong potential for future implementation in our context.

6

Conclusion

This thesis presented a magnetometer-based approach to simulate and estimate velocity for low-speed railway robotic platforms. The study included a synthetic magnetic field simulation framework, a lag-based speed estimation method, and an Extended Kalman Filter (EKF) fusion scheme integrating IMU acceleration with magnetometer-derived velocity.

The magnetometer data simulation process introduced sinusoidal spatial variations and realistic multi-band noise to emulate the complexity of real railway magnetic environments. Noise components including Gaussian fluctuations, temperature and aging drift, spatial jitter, and rare spikes were carefully designed to balance realism and analytical tractability. The field simulation successfully demonstrated temporal and spatial correlations across front and rear sensors, enabling delay-based velocity inference.

Through a lag-based MMSE correlation approach, the velocity profile was extracted from front-rear magnetometer signal delays. The estimated speed followed the general shape of the ground truth velocity and reflected motion phases such as acceleration, cruising, and deceleration. Small overestimations and oscillations were observed due to noise interference and simplified modeling assumptions.

The Extended Kalman Filter evaluation revealed distinct performance differences between using acceleration data alone and combining data from both sensors. The EKF with IMU-only data successfully captured the overall motion characteristics and maintained smooth velocity estimates during cruising segments.

After incorporating magnetometer-derived velocity as an additional measurement, the filter performance showed notable improvements. For instance, the magnetometer data provided more accurate velocity values and better estimation range compared to the IMU-only approach. Although some oscillations were introduced due to magnetometer noise, the overall tracking accuracy was enhanced.

This combination of sensors' data demonstrates the potential value of magnetometer-IMU fusion, especially for applications where magnetometer signal quality can be improved through preprocessing techniques that leverage the complementary characteristics of both sensor types.

7

Future Work

While the current study demonstrates the feasibility of magnetometer-inertial fusion for velocity estimation in simulated railway scenarios, several avenues remain open for future exploration to enhance robustness, generalizability, and real-world applicability.

First and foremost, real-world validation is essential. The present work relies on simulated sensor data, which, although carefully modeled, cannot fully replicate the complexities of physical environments. Future research should involve deploying physical magnetometers and IMUs on railway robots and collecting data under varying track, weather, and operational conditions. Such field trials will help assess how noise characteristics, magnetic interference, and mechanical vibrations manifest in practice and challenge the proposed algorithms.

Secondly, the noise models used in the simulation—comprising Gaussian noise, drift, jitter, and spike artifacts—can be further refined based on empirical measurements. Collecting and analyzing raw magnetic field data from real railway systems would enable the construction of more accurate statistical and spectral noise models. These could be used to train or tune denoising filters, adaptive fusion strategies, and synthetic data generators for future algorithm testing.

Another key challenge lies in sensor configuration and calibration. The accuracy of lag-based velocity estimation and sensor fusion depends heavily on the relative orientation and placement of the magnetometers and IMU. Future work could develop automatic calibration routines or self-diagnosing alignment checks to compensate for mounting errors or dynamic shifts in sensor geometry.

Moreover, the current EKF-based fusion framework assumes fixed filter parameters and static noise covariances. In dynamic operational contexts, noise levels and signal confidence can vary significantly. Adaptive filtering techniques—such as covariance tuning, quality-weighted fusion, or neural network-assisted fusion logic—may help improve real-time robustness and error handling under changing environmental conditions.

Lastly, the velocity estimation system developed here could be integrated into a more comprehensive Simultaneous Localization and Mapping (SLAM) framework.

Such an extension would allow the system not only to estimate motion but also to build and align maps using magnetometer signals, IMU data, and possibly additional sensors like odometry or LiDAR. By incorporating loop closure detection, map optimization, and probabilistic data association, a full SLAM solution could offer infrastructure-free navigation in underground or GPS-denied railway networks.

In summary, future work should aim to transition the proposed system from simulation to real-world deployment, with particular emphasis on sensor modeling, adaptive fusion, calibration robustness, and integration with broader localization systems.

Bibliography

[1] Siebler, B., Lehner, A., Sand, S., & Hanebeck, U. D. (2024). Magnetic field mapping of railway lines with graph SLAM. In 2024 27th International Conference on Information Fusion (FUSION) (pp. 1-8). IEEE.

[2] Skog, I., Kok, M., Hendeby, G., Huang, C., & Edridge, T. (2025). On the Connection Between Magnetic-Field Odometry Aided Inertial Navigation and Magnetic-Field SLAM. In 2025 IEEE/ION Position, Location and Navigation Symposium (PLANS) (pp. 809-814).

[3] Patonis, P., Patias, P., Tziavos, I. N., Rossikopoulos, D., & Margaritis, K. G. (2018). A fusion method for combining low-cost IMU/magnetometer outputs for use in applications on mobile devices. Sensors, 18(8), 2616.

[4] Zhuang, Y., Sun, X., Li, Y., Huai, J., Hua, L., Yang, X., ... & Chen, R. (2023). Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches. Information Fusion, 95, 62-90.

[5] Mahajan, I., Unjhawala, H., Zhang, H., Zhou, Z., Young, A., Ruiz, A., ... & Negrut, D. (2024). Quantifying the sim2real gap for GPS and IMU sensors. arXiv preprint arXiv:2403.11000.

[6] Chukwurah, N., Adebayo, A. S., & Ajayi, O. O. (2024). Sim-to-real transfer in robotics: Addressing the gap between simulation and real-world performance. International Journal of Robotics and Simulation, 6(1), 89-102.

[7] Patonis, P., Patias, P., Tziavos, I. N., Rossikopoulos, D., & Margaritis, K. G. (2018). A fusion method for combining low-cost IMU/magnetometer outputs for use in applications on mobile devices. Sensors, 18(8), 2616.

[8] Hedberg, E., & Hammar, M. (2015). Train localization and speed estimation using on-board inertial and magnetic sensors.

[9] Engelberg, T., & Mesch, F. (2000). Eddy current sensor system for non-contact speed and distance measurement of rail vehicles. WIT Transactions on The Built Environment, 50.

[10] J. Wendel, C. Huber, & G. F. Trommer. (2011). Train localization using sen-

sor fusion of tachometers and IMU sensors. IEEE/ION Position, Location and Navigation Symposium, 2011, pp. 147–153.

[11] Joshi, B., & Rekleitis, I. (2024, May). Enhancing visual inertial SLAM with magnetic measurements. In 2024 IEEE International Conference on Robotics and Automation (ICRA) (pp. 10012-10019). IEEE.

[12] Tumanski, S. (2016). Handbook of magnetic measurements. CRC press.

[13] Strang, T., Lehner, A., Heirich, O., Siebler, B., & Sand, S. (2024). Train-Localization in Tunnels using Magnetic Signatures. In 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) (pp. 702-707). IEEE.

[14] Reid, T. G., Houts, S. E., Cammarata, R., Mills, G., Agarwal, S., Vora, A., & Pandey, G. (2019). Localization requirements for autonomous vehicles. arXiv preprint arXiv:1906.01061.

[15] Campbell, W. H. (2003). Introduction to geomagnetic fields. Cambridge University Press.

[16] Hausman, B. A., Michel, F. C., Espley, J. R., Cloutier, P. A., & Acuna, M. H. (2006). Measuring noise in magnetometers: An example using the Mars Global Surveyor magnetometers. Journal of Geophysical Research: Space Physics, 111(A11).

[17] Mateos, I., Patton, B., Zhivun, E., Budker, D., Wurm, D., & Ramos-Castro, J. (2015). Noise characterization of an atomic magnetometer at sub-millihertz frequencies. Sensors and Actuators A: Physical, 224, 147-155.

[18] Krzyzewski, S. P., Perry, A. R., Gerginov, V., & Knappe, S. (2019). Characterization of noise sources in a microfabricated single-beam zero-field optically-pumped magnetometer. Journal of Applied Physics, 126(4).

[19] McCuen, B. A., Moldwin, M. B., & Engebretson, M. J. (2022). Characterization of Transient Induced Current Events in Ground Magnetometer Data. Authorea Preprints.

[20] Advanced Navigation. (2023). Inertial measurement unit (IMU) – An introduction. https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/

[21] Budiyono, A. (2012). Principles of GNSS, Inertial, and Multi-sensor Integrated Navigation Systems. Industrial Robot: An International Journal, 39(3), ir.2012.04939caa.011.

[22] Farrell, J. A., & Farrell, J. A. (2008). Aided navigation: GPS with high rate sensors. McGraw-Hill.

[23] Goel, A., Ul Islam, A., Ansari, A., Kouba, O., & Bernstein, D. S. (2021). An Introduction to Inertial Navigation From the Perspective of State Estimation [Focus on Education]. IEEE Control Systems, 41(5), 104–128.

[24] Aydemir, G. A., & Saranlı, A. (2012). Characterization and calibration of MEMS inertial sensors for state and parameter estimation applications. Measurement, 45(5), 1210–1225.

[25] Noureldin, A., Karamat, T. B., & Georgy, J. (2013). Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration. Springer Berlin Heidelberg.

[26] Ali, M. (2016). Compensation of temperature and acceleration effects on MEMS gyroscope. 2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 274–279.

[27] Perrin, G., Soize, C., Duhamel, D., & Funfschilling, C. (2013). Track irregularities stochastic modeling. Probabilistic Engineering Mechanics, 34, 123–130.

[28] Park, M., & Gao, Y. (2008). Error and Performance Analysis of MEMS-based Inertial Sensors with a Low-cost GPS Receiver. Sensors, 8(4), 2240–2261.

[29] Hall, D. L., & Llinas, J. (1997). An introduction to multisensor data fusion. Proceedings of the IEEE, 85(1), 6-23.

[30] Khaleghi, B., Khamis, A., Karray, F. O., & Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. Information fusion, 14(1), 28-44.

[31] Wang, Y., Lou, Y., Zhang, Y., Song, W., Huang, F., Tu, Z., & Zhang, S. (2021). Raillomer: Rail vehicle localization and mapping with LiDAR-IMU-odometer-GNSS data fusion. arXiv preprint arXiv:2111.15043.

[32] Gao, J. B., & Harris, C. J. (2002). Some remarks on Kalman filters for the multisensor fusion. Information Fusion, 3(3), 191-201.

[33] Wan, E., & Van Der Merwe, R. (2000). Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (cat. No. 00EX373).

[34] Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on signal processing, 50(2), 174-188.

[35] Palm, G., & Schwenker, F. (2009). Sensor-fusion in neural networks. In Harbour Protection Through Data Fusion Technologies (pp. 299-306). Springer Netherlands.

[36] Rigatos, G. G. (2010). Extended Kalman and particle filtering for sensor fusion in motion control of mobile robots. Mathematics and computers in simulation,

81(3), 590-607.

[37] Maybeck, P. S. (1990). The Kalman filter: An introduction to concepts. In Autonomous robot vehicles (pp. 194-204). New York, NY: Springer New York.

[38] Marković, L., Kovač, M., Milijas, R., Car, M., & Bogdan, S. (2022, June). Error state extended kalman filter multi-sensor fusion for unmanned aerial vehicle localization in gps and magnetometer denied indoor environments. In 2022 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 184-190). IEEE.

[39] Eljaik, J., Kuppuswamy, N., & Nori, F. (2015). Multimodal sensor fusion for foot state estimation in bipedal robots using the extended kalman filter. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 2698-2704). IEEE.

[40] Esveld, C. (2001). Geometry of a railway line. In Modern railway track (Vol. 385, pp. 13). MRT-productions.

[41] Blunt, L., & Jiang, X. (2003). Advanced techniques for assessment surface topography: development of a basis for 3D surface texture standards" surfstand". Elsevier.

[42] Gautschi, W. (2011). Numerical analysis. Springer Science & Business Media.

[43] Try, P., & Gebhard, M. (2023). A vibration sensing device using a six-axis imu and an optimized beam structure for activity monitoring. Sensors, 23(19), 8045.

[44] Pérez, J., Alcázar, M., Sánchez, I., Cabrera, J. A., Nybacka, M., & Castillo, J. J. (2023). On-line learning applied to spiking neural network for antilock braking systems. Neurocomputing, 559, 126784.

[45] Solin, A., Kok, M., Wahlström, N., Schön, T. B., & Särkkä, S. (2018). Modeling and interpolation of the ambient magnetic field by Gaussian processes. IEEE Transactions on robotics, 34(4), 1112-1127.

[46] Zhang, A., Dehghanian, P., Stevens, M., Snodgrass, J., & Overbye, T. (2023). Synthetic Geomagnetic Field Data Creation for Geomagnetic Disturbance Studies using Time-series Generative Adversarial Networks. In 2023 IEEE Texas Power and Energy Conference (TPEC) (pp. 1-6). IEEE.

[47] Wahlström, N.,& Gustafsson, F. (2013). Magnetometer modeling and validation for tracking metallic targets. IEEE Transactions on Signal Processing, 62(3), 545-556.

[48] Jin, F., Tu, X., Wang, J., Yang, B., Dong, K., Mo, W., ...& Song, J. (2020). Noise modeling and simulation of giant magnetic impedance (GMI) magnetic sensor. Sensors, 20(4), 960.

[49] Siebler, B., Lehner, A., Sand, S., & Hanebeck, U. D. (2021). Evaluation of simultaneous localization and calibration of a train mounted magnetometer. In Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021) (pp. 2285-2293).

[50] Hedberg, E., & Hammar, M. (2015). Train localization and speed estimation using on-board inertial and magnetic sensors.

[51] Solin, A., Kok, M., Wahlström, N., Schön, T. B., & Särkkä, S. (2018). Modeling and interpolation of the ambient magnetic field by Gaussian processes. IEEE Transactions on robotics, 34(4), 1112-1127.

[52] Skog, I., Kok, M., Hendeby, G., Huang, C., & Edridge, T. (2025). On the Connection Between Magnetic-Field Odometry Aided Inertial Navigation and Magnetic-Field SLAM. In 2025 IEEE/ION Position, Location and Navigation Symposium (PLANS) (pp. 809-814).

[53] Mateos, I., Patton, B., Zhivun, E., Budker, D., Wurm, D., & Ramos-Castro, J. (2015). Noise characterization of an atomic magnetometer at sub-millihertz frequencies. Sensors and Actuators A: Physical, 224, 147-155.

[54] Wahlström, N., & Gustafsson, F. (2013). Magnetometer modeling and validation for tracking metallic targets. IEEE Transactions on Signal Processing, 62(3), 545-556.

[55] Gu, C., Heidrich-Meisner, V., & Wimmer-Schweingruber, R. F. (2023). Sliding-Window Cross-Correlation and Mutual Information for the analysis of solar wind measurements.

[56] Mateos, I., Patton, B., Zhivun, E., Budker, D., Wurm, D., & Ramos-Castro, J. (2015). Noise characterization of an atomic magnetometer at sub-millihertz frequencies. Sensors and Actuators A: Physical, 224, 147-155.

[57] Mahony, R., Hamel, T., & Pflimlin, J. M. (2008). Nonlinear complementary filters on the special orthogonal group. IEEE Transactions on automatic control, 53(5), 1203-1218.

[58] Kok, M., Hol, J. D., & Schön, T. B. (2017). Using inertial sensors for position and orientation estimation. arXiv preprint arXiv:1704.06053.

A

Implementation Details

A.1 Design of track parameters

Table A.1: Track Structure Design Parameters

Section	Type	Parameters	Characteristics
S_1	Straight	• Length: 100 m • Direction: Horizontal • Track points: 50	Initial straight with stable speed
T_1	Transition	• Length: 30 m • Curvature: $0 \rightarrow 1/200$ m • Track points: 50	Transition from straight to curved section
C_1	Arc	• Radius: 200 m • Arc angle: $\pi/6 - 2 \cdot 30/(2 \cdot 200)$ • Track points: 50	Large radius arc with gradual deceleration
T_2	Transition	• Length: 30 m • Curvature: $1/200$ m $\rightarrow 0$ • Track points: 50	Transition from curved to straight section
S_2	Straight	• Length: 80 m • Direction: $\sim 30^\circ$ ($\pi/6$) • Track points: 50	Inclined section with speed recovery Hill elevation: 6 m peak at middle position
T_3	Transition	• Length: 30 m • Curvature: $0 \rightarrow 1/300$ m • Track points: 50	Transition from straight to curved section
C_2	Arc	• Radius: 300 m • Arc angle: $\pi/6 - 2 \cdot 30/(2 \cdot 300)$ • Track points: 50	Medium radius arc with controlled speed
T_4	Transition	• Length: 30 m • Curvature: $1/300$ m $\rightarrow 0$ • Track points: 50	Transition from curved to straight section
S_3	Straight	• Length: 120 m • Direction: $\sim 60^\circ$ ($\pi/3$) • Track points: 50	Long straight with speed stabilization

Continued on next page

Table A.1 – Continued from previous page

Section	Type	Parameters	Characteristics
T_5	Transition	• Length: 30 m • Curvature: 0 → -1/250 • Track points: 50	Transition from straight to curved section (right turn)
C_3	Arc	• Radius: 250 m • Arc angle: $\pi/4 - 2 \cdot 30/(2 \cdot 250)$ • Direction: Right turn • Track points: 50	Final curve with speed reduction (right turn) Valley elevation: -4 m depth at end position
T_6	Transition	• Length: 30 m • Curvature: -1/250 m → 0 • Track points: 50	Transition from curved to straight section
S_4	Straight	• Length: 100 m • Direction: $\sim 105°$ ($\pi/3 + \pi/4$) • Track points: 50	Final section at reduced speed

Table A.2: Elevation Profile Parameters

Feature	Location	Peak/Depth	Width	Mathematical Model
Hill	Middle of S_2	3 m	300 m	$3 \cdot \exp\left(-\frac{(d-d_{\text{peak}})^2}{2 \cdot 150^2}\right)$
Valley	End of C_3	-2 m	200 m	$-2 \cdot \exp\left(-\frac{(d-d_{\text{bottom}})^2}{2 \cdot 100^2}\right)$

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY