AI-DRIVEN NAVIGATION IN GNSS-DENIED ENVIRONMENTS

FOR AUTONOMOUS VEHICLES

By

Arshia Maryum

Supervisor

Dr. Suhail Akhtar

Department of Avionics Engineering

Institute of Space Technology, Islamabad

2025

AI-DRIVEN NAVIGATION IN GNSS-DENIED ENVIRONMENTS

FOR AUTONOMOUS VEHICLES

A fyp-01 report submitted to the

Institute of Space Technology in partial fulfillment of the requirements

for the degree of Bachelor of Engineering

in Avionics Engineering

by

Arshia Maryum

Supervisor

Dr. Suhail Akhtar

Department of Avionics Engineering

Institute of Space Technology, Islamabad

2025

Institute of Space Technology

Department of Avionics Engineering

AI-DRIVEN NAVIGATION IN GNSS-DENIED ENVIRONMENTS

FOR AUTONOMOUS VEHICLES

by

Arshia Maryum

APPROVAL OF BOARD EXAMINERS

_____

Prof. Dr. Suhail Akhtar

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my supervisor, **Prof. Dr. Suhail Akhtar**, for his exceptional guidance and support throughout this project. His knowledge and advice have been invaluable in helping me overcome challenges and discover innovative ideas for the navigation system. His wisdom, encouragement, and unwavering belief in me have been a constant source of motivation, shaping the direction of this project and inspiring me to achieve my goals.

I am also thankful to **Prof. Dr. Mohammad Asmat Ullah Khan**, Dean Faculty of Computing, International Islamic University Islamabad, for his invaluable support and guidance. His insights and advice have been incredibly helpful in navigating the challenges of this project. I truly appreciate his willingness to share his knowledge and expertise.

A special thank you to my father, **Dr. Sohail Iqbal**, for his unwavering support and encouragement throughout this project. I am deeply grateful for everything he has done to help me succeed. I also want to thank my family for their constant support and patience during different stages of the project.

I appreciate the resources and opportunities provided by the Institute of Space Technology, which were crucial in completing this project. Lastly, I want to thank the faculty members of the Department of Avionics Engineering for their continuous motivation and guidance.

ABSTRACT

Autonomous vehicles face significant challenges in GNSS-denied environments, where the absence or degradation of reliable Global Navigation Satellite System (GNSS) signals compromises accurate and safe navigation. This report presents the design and development of an AI-driven navigation system that delivers precise positioning in such conditions. Utilizing sensor fusion and Long Short-Term Memory (LSTM) neural networks, the system processes accelerometer and gyroscope data to estimate the vehicle's position independently of GNSS input.

Traditional inertial navigation systems rely on the integration of IMU data over time, a process prone to drift—cumulative errors that increasingly degrade positional accuracy. Kalman filters are commonly used to mitigate such drift by fusing sensor data and estimating the system's state recursively; however, they often underperform in complex, non-linear, and long-duration motion scenarios due to their reliance on predefined system dynamics and noise models.

To address this limitation, the LSTM model is trained on sequences of IMU data aligned with ground truth positions obtained from GNSS, enabling it to learn temporal patterns and understand how drift evolves. Through this training, the model minimizes prediction errors and acquires the capability to correct for drift during inference, ensuring stable and accurate trajectory estimation using only IMU inputs.

Extensive simulations, conducted on GPU servers and Google Colab, cover various motion scenarios—linear, rotational, and combined—and demonstrate the system's robustness and accuracy. Performance indicators such as positional error, noise resilience, and

computational efficiency validate the system's reliability. This research contributes to the advancement of autonomous navigation by offering a continuous, high-precision solution suitable for GNSS-denied environments. Future work will focus on integrating additional sensors and conducting real-world trials to further enhance system performance and scalability.

TABLE OF CONTENTS

TABLE OF FIGURE

TABLE OF TABLES

1 INTRODUCTION

1.1 Background and Motivation

Autonomous vehicles are revolutionizing industries such as transportation, logistics, defense, and space exploration. One of the most critical need for these systems is reliable navigation across different and challenging environments. Traditional navigation methods heavily rely on the **Global Navigation Satellite System (GNSS)** for accurate positioning. However, GNSS signals often become unavailable or unreliable in settings like urban canyons, dense forests, underground areas, or during intentional signal jamming. These GNSS-denied environments present significant challenges to autonomous navigation systems.

Fig. 1.1: GNSS Denied Environment.

Artificial Intelligence (AI) is one of the transformational tools to deal with these issues because it enables systems to mimic human-like decision-making and learn from complex

data patterns. Advanced algorithms like **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks have an advantage over others when it comes to sequential and temporal data, such as sensor readings and vehicle trajectories. By using these algorithms, autonomous systems can achieve robust navigation, predict vehicle positions, fuse multi-sensor data, and perform real-time decision-making. This study explores the application of these AI techniques to enable accurate and reliable autonomous navigation in GNSS-denied environments.

1.2 Problem Statement

GNSS-dependent autonomous systems face critical limitations in GNSS-denied environments due to signal loss, interference, and multipath errors. Traditional alternatives, such as dead reckoning or visual odometry, often suffer from cumulative errors and environmental dependencies. While AI-driven navigation solutions have shown promise, they require refinement to effectively integrate temporal data patterns for long-term stability and performance.

This study addresses the problem of designing and implementing an AI-driven navigation system that uses advanced algorithms like RNNs and LSTMs. These algorithms, combined with sensor fusion and localization techniques, aim to deliver accurate and real-time navigation capabilities in GNSS-denied environments.

1.3 Objectives of the Study

The main objectives of this study are:

- To analyze the challenges of navigation in GNSS-denied environments.
- To design an AI-based navigation framework using RNNs and LSTMs for processing temporal data.
- To implement sensor fusion techniques for integrating data from sensors such as accelerometers and gyroscopes.
- To evaluate the proposed system in simulated GNSS-denied scenarios.

1.4 Scope of the Study

This focuses on developing an AI-driven navigation system for robust operation in GNSS-denied environments.

Key aspects include:

- Reviewing existing navigation systems and the role of AI in overcoming GNSS limitations.
- Developing a system architecture incorporating RNNs, LSTMs, and sensor fusion techniques.
- Implementing and evaluating the proposed system in simulations mimicking GNSS-denied conditions.
- Concentrating on software-based simulations, with hardware implementation deferred to future work.

This study establishes a foundation for deploying advanced AI-based navigation systems in real-world autonomous vehicles.

1.5 Organization of the Study

This report is structured as follow:

- **Chapter 1:**

  Introduction – Covers the background, motivation, problem statement, objectives, and scope of the study. It also introduces the organization of the report and outlines the Sustainable Development Goals (SDGs) related to the project.

  Chapter 2:

  Literature Review – Discusses prior research on navigation systems, the limitations of GNSS, and the application of AI, including RNNs and LSTMs, in autonomous navigation.

- **Chapter 3:**

  Theoretical Framework – Explains the principles behind RNNs and LSTMs, their role in navigation, and the techniques of sensor fusion and localization used for improved accuracy.

- **Chapter 4:**

  Methodology – Describes the design and implementation of the AI-driven navigation system, including model training, data processing, system architecture, sensor fusion, and simulation setup.

- **Chapter 5:**

  Results and Discussion – Presents the simulation results and discusses the system's performance, strengths, and limitations.

- **Chapter 6:**

Conclusion and Future Work – Summarizes the findings, contributions, and suggests areas for future research and hardware implementation.

1.6 Sustainable Development Goals (SDGs)

The 2030 Agenda for Sustainable Development, adopted by the United Nations General Assembly in September 2015, includes 17 Sustainable Development Goals (SDGs) that aim to ensure a prosperous, equitable, and sustainable future for all. In alignment with this global vision, this study integrates the following SDGs, which are particularly relevant to the development and application of AI-driven navigation systems for autonomous vehicles, especially in GNSS-denied environments:

- **Goal 9: Industry, Innovation, and Infrastructure**

  Autonomous navigation systems, particularly those relying on AI and sensor fusion, are key to advancing innovation in the transportation and logistics industries. By developing advanced technologies like AI-based navigation in GNSS-denied environments, this study supports the growth of smart infrastructure and promotes technological innovation.

- **Goal 11: Sustainable Cities and Communities**

  Autonomous vehicles are a crucial component of sustainable urban mobility solutions. The development of AI-driven navigation systems that function in GNSS-denied environments can enhance urban transportation systems, reduce congestion, and improve the efficiency of traffic management, contributing to the development of sustainable cities and communities.

2 LITERATURE REVIEW

The need for autonomous navigation systems that can operate in GNSS-denied environments has gained significant attention due to the limitations of traditional GNSS-based navigation in complex, obstructed, or interference-prone areas. While GNSS is the cornerstone of modern navigation, its reliance on satellite signals makes it vulnerable in urban canyons, dense forests, and underground spaces, or in scenarios where jamming is intentional. To overcome these limitations, researchers have turned to alternative techniques, including **sensor fusion** and **AI-driven methods**, to maintain accurate and reliable positioning. This section reviews the existing literature on autonomous navigation systems, the challenges posed by GNSS-denied environments, and the application of **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM) networks** to address these challenges. It will also explore the integration of sensor fusion techniques for enhancing the performance of autonomous systems in such conditions.

2.1 Autonomous Navigation Systems

Autonomous navigation systems have seen significant advancements in recent years, with applications ranging from self-driving cars to unmanned aerial vehicles (UAVs) and space exploration. These systems rely on a combination of sensors, algorithms, and decision-making capabilities to navigate environments without human intervention. Typically, autonomous vehicles depend on technologies such as Global Navigation Satellite System (GNSS), LiDAR, radar, and vision-based systems. However, these systems often encounter challenges in GNSS-denied environments where traditional positioning methods fail.

Research in autonomous navigation focuses on enhancing the robustness and accuracy of systems, especially in areas like urban canyons, forests, tunnels, and in scenarios with signal jamming. Several approaches have been explored to achieve reliable navigation in these environments, including sensor fusion, dead reckoning, and visual odometry. While these methods offer alternatives, they come with limitations such as the accumulation of errors over time or sensitivity to environmental conditions.

2.2 GNSS in Navigation: Capabilities and Limitations

The GNSS plays a crucial role in modern navigation by providing real-time, accurate positioning data. GNSS systems, such as GPS, Glonass, Galileo, and BeiDou, offer global coverage and are integral to applications in transportation, logistics, and defense. GNSS-based navigation systems are typically reliable in open areas with unobstructed satellite signals. However, GNSS faces significant limitations in certain environments. In GNSS-denied environments, such as urban canyons, dense forests, underground spaces, and areas with intentional jamming or interference, the satellite signals may become weak or completely unavailable. This leads to positioning errors, which may worsen due to multipath effects, where the signal bounces off structures before reaching the receiver. These challenges necessitate alternative navigation strategies that can function independently or in conjunction with GNSS.

In response to these limitations, research has focused on developing integrated navigation solutions that combine multiple sensors, such as accelerometers, gyroscopes, and vision-based sensors, to maintain accurate positioning without relying solely on GNSS.

2.3 Challenges in GNSS-Denied Environments

Navigating in GNSS-denied environments presents numerous challenges, particularly in maintaining accuracy and long-term reliability.

Some of the primary challenges include:

- **Signal Loss and Interference**: GNSS signals can be blocked or attenuated in environments with high-rise buildings or dense foliage, leading to signal loss. Interference from jammers can also degrade the quality of GNSS signals.
- **Error Accumulation**: Traditional dead reckoning and **Inertial Navigation Systems (INS)** often accumulate errors over time due to sensor drift, especially in the absence of external references like GNSS. Without correction from GNSS, these errors can grow significantly, leading to large deviations from the true position.
- **Environmental Dependence**: Techniques such as visual odometry and LiDAR-based localization are sensitive to changes in the environment, such as lighting conditions, weather, and the availability of clear landmarks. These systems can struggle in dynamic or cluttered environments, further complicating navigation in GNSS-denied regions.

To address these challenges, sensor fusion techniques, which combine data from multiple sensors (accelerometers, gyroscopes, cameras, etc.), are commonly employed. These methods seek to mitigate the limitations of individual sensors by providing more accurate and reliable estimates of the vehicle's position and orientation.

2.4 Artificial Intelligence in Navigation

Artificial Intelligence (AI), particularly Machine Learning (ML) and Deep Learning (DL), has shown great promise in overcoming the limitations of traditional navigation techniques. AI-driven approaches can learn patterns from data, allowing systems to adapt to varying conditions and make real-time decisions.

One of the most significant contributions of AI to navigation systems is the application of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. RNNs are particularly suited for sequential data analysis, making them ideal for processing time-series sensor data in autonomous navigation tasks. Unlike traditional models, RNNs can maintain internal states and learn from past sequences of data, allowing them to predict future sensor readings and vehicle positions.

LSTMs, a specialized type of RNN, further enhance the capabilities of standard RNNs by overcoming the issue of vanishing gradients in long sequences. LSTMs can retain information over extended time periods, making them highly effective for applications requiring long-term memory, such as tracking the position of autonomous vehicles over long distances.

AI-based techniques, such as reinforcement learning, can also be employed for real-time decision-making, where the system learns to optimize its actions based on feedback from the environment. By training on vast amounts of data, AI models can improve their prediction accuracy and generalize better to unknown environments, including GNSS-denied areas.

2.5 Summary of Related Works

Many researchers have explored AI and sensor fusion techniques to improve navigation in GNSS-denied environments. Some notable contributions include:

- **Deep-Learning for Positioning**: Recent studies have employed deep learning algorithms to process sensor data and predict vehicle positions in challenging environments. These models are trained on large datasets to recognize complex patterns and make more accurate predictions than traditional techniques.
- **Sensor Fusion for Robust Navigation**: Various approaches combine data from accelerometers, gyroscopes, and vision-based sensors to reduce reliance on GNSS. Studies have shown that sensor fusion techniques can enhance the accuracy of position estimates by compensating for the limitations of individual sensors, such as the drift in gyroscopes or the sensitivity of vision-based systems to lighting changes.
- **LSTM-based Navigation Systems**: Some studies have successfully implemented LSTM networks to predict the trajectories of autonomous vehicles in GNSS-denied conditions. By learning from historical sensor data, LSTMs can predict the future vehicle state, providing reliable navigation even when GNSS signals are unavailable.
- **Real-World Applications**: Several projects have tested AI-driven navigation systems in real-world scenarios, including urban navigation, autonomous drones, and space exploration. These studies highlight the potential of AI to enable

autonomous systems to operate reliably and efficiently in GNSS-denied environments.

While these studies have shown promising results, challenges remain in integrating multiple sensors, handling sensor noise, and ensuring the robustness of AI models in dynamic environments. Future research will likely focus on improving the integration of AI with advanced sensor technologies and fine-tuning models to optimize real-time performance in GNSS-denied environments.

3 THEORETICAL FRAMEWORK

3.1 Introduction

Autonomous navigation in GNSS-denied environments demands robust alternatives to conventional satellite-based positioning. To maintain accurate localization in such conditions, the proposed system combines sensor fusion techniques with AI-driven algorithms. The primary goal is to estimate the vehicle's position and orientation using data from inertial sensors—accelerometers, gyroscopes, and magnetometers—while mitigating the effects of GNSS signal loss. This section outlines the core navigation methodology underpinning the AI-based system, focusing on the use of sensor fusion and recurrent neural network architectures, particularly Long Short-Term Memory (LSTM) networks.

Traditional inertial navigation systems (INS) estimate position and orientation by integrating IMU data over time. While this method enables navigation without external inputs, it is highly susceptible to sensor drift—where small measurement errors compound progressively, leading to significant inaccuracies. In GNSS-denied scenarios, the lack of corrective input further exacerbates this problem. While Kalman filters are commonly applied to manage sensor noise and estimation errors, they often fall short in dynamic, non-linear, or long-duration contexts due to their reliance on fixed models and assumptions.

To overcome these limitations, the proposed system leverages LSTM networks, which are well-suited for modeling sequential data with temporal dependencies. During training, the LSTM is fed sequences of IMU data paired with accurate position references, allowing it to learn the underlying relationships between sensor patterns and actual motion. Rather

than applying rigid equations, the LSTM builds an internal model of how sensor noise and drift evolve over time, enabling it to correct for these effects during real-time inference.

Once trained, the LSTM network processes only live IMU data to infer the vehicle's trajectory, effectively compensating for drift and maintaining accurate localization even in the absence of GNSS. This learning-based approach offers a flexible and powerful solution for resilient navigation in challenging environments, enhancing the autonomy and reliability of the vehicle's guidance system.

3.2 Sensor Fusion and Localization

Sensor fusion is essential in autonomous navigation systems, particularly when GNSS signals are unavailable. It involves combining data from multiple sensors to produce a more accurate estimate of the system's state, such as position, velocity, and orientation. Commonly used sensors for this purpose include **Inertial Measurement Units (IMUs)**, which consist of accelerometers, gyroscopes, and magnetometers.

A fundamental tool for sensor fusion is the **Kalman Filter (KF)**, which is a recursive algorithm designed to estimate the state of a dynamic system from noisy sensor measurements. It works in two steps: prediction, where the system's future state is estimated, and correction, where the estimate is updated using new sensor measurements.

Localization involves determining the vehicle's position and orientation within a specific environment. Techniques like Simultaneous Localization and Mapping (SLAM) enable real-time mapping of the environment while simultaneously estimating the vehicle's position relative to the map.

3.3 Sensors Used for Navigation

In GNSS-denied environments, a variety of sensors are integrated into the navigation system to estimate the position and orientation of the vehicle. The primary sensors employed are:

3.3.1 Global Positioning System (GPS)

While GNSS is not available in all scenarios, it still plays a significant role in aiding navigation when signals are available. GPS provides position and velocity information with high accuracy, which can be fused with other sensor data when GNSS signals are weak or lost.

3.3.2 Accelerometer

An accelerometer measures acceleration forces in one or more directions, which can be integrated over time to estimate velocity and, in combination with initial position estimates, provide information on displacement. However, errors due to drift accumulate over time, which is why accelerometer data is combined with other sensor inputs.

3.3.3 Gyroscope

A gyroscope measures angular velocity, providing information on the orientation or attitude of the vehicle. Gyroscopes are crucial for estimating the change in the vehicle's orientation over time. Like accelerometers, gyroscope data tends to drift, which is mitigated by combining it with other sensors.

The sensor fusion of accelerometers, gyroscopes, and magnetometers allows for accurate estimation of position, velocity, and orientation in environments where GNSS signals cannot be relied upon.

3.4 AI Algorithms for Navigation

While sensor fusion plays a vital role in improving the accuracy of navigation in GNSS-denied environments, **Artificial Intelligence (AI)**, and specifically **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks, provide additional capabilities to handle the temporal nature of sensor data.

3.4.1 Recurrent Neural Network (RNN)

RNNs are a class of neural networks designed for sequential data processing. They are particularly suited for time-series data, such as the sensor readings over time in autonomous vehicles.

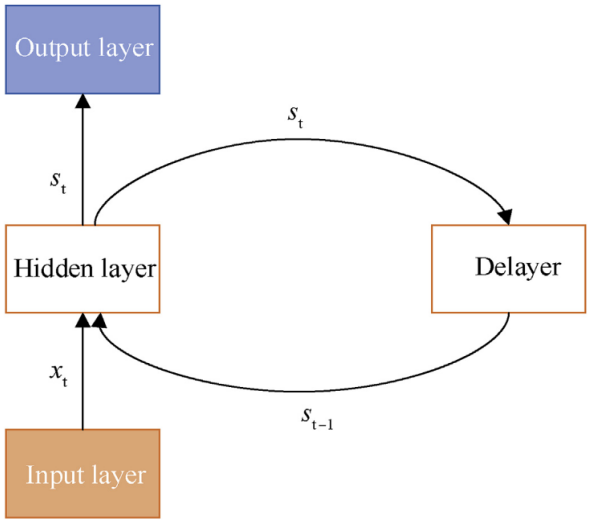

Fig. 3.1: The network structure of RNN.

RNNs have the ability to maintain hidden states, allowing them to remember past information, which is essential for navigation where past sensor measurements are crucial for predicting future states. The standard RNN model is represented as:

$$h_t = f(W_h h_{t-1} + W_x x_t + b_h)$$

$$y_t = W_y h_t + b_y$$

where:

- $\boldsymbol{h_t}$ is the hidden state at time t
- $\boldsymbol{x_t}$ is the input at time t
- $\boldsymbol{W_h}$, $\boldsymbol{W_x}$, and $\boldsymbol{W_y}$ are the weight matrices for the hidden state, input, and output respectively.
- $\boldsymbol{f}$ is the activation function,
- $\boldsymbol{b_h}$ and $\boldsymbol{b_y}$ are the biases.

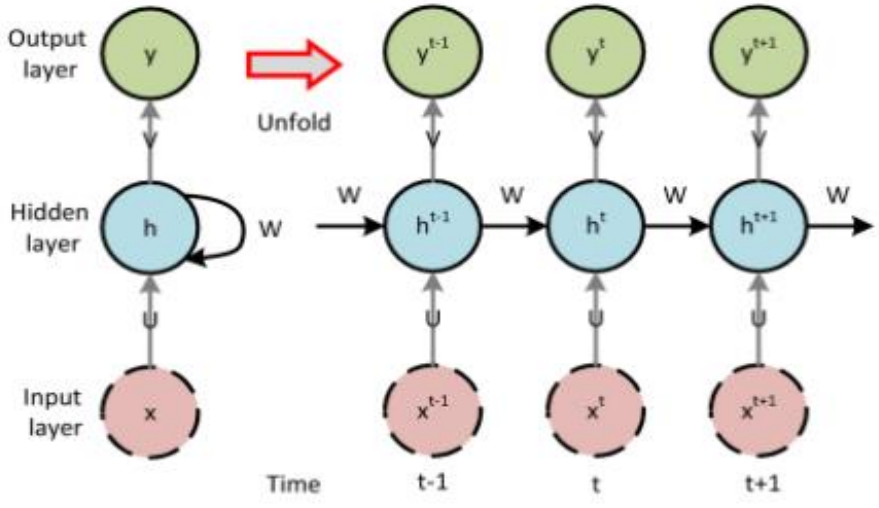

Fig. 3.2: RNN expand structure

The key challenge with RNNs is their inability to remember long-term dependencies, which is where **LSTM networks** come in.

3.4.2 Long Short-Term Memory (LSTM)

LSTM networks are an extension of RNNs that are designed to better capture long-term dependencies in sequential data. LSTMs address the vanishing gradient problem inherent in traditional RNNs by incorporating memory cells that can maintain information over extended periods.

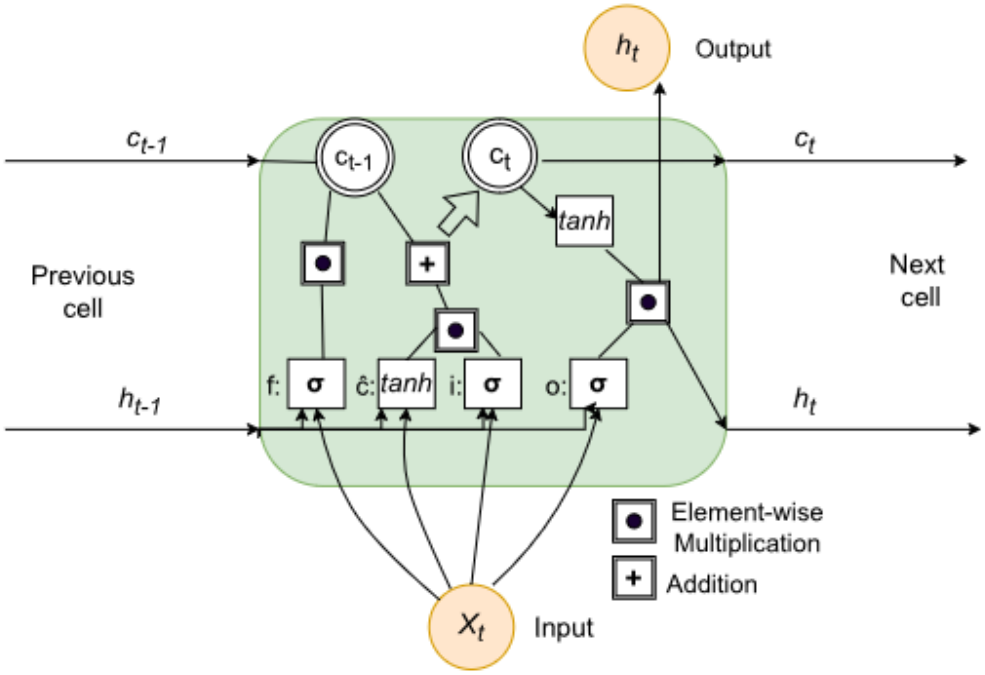

Fig. 3.3: Long Short Term Memory Cell

Fig. 3.3. shows, cell state and hidden state, which is also the output of the cell, are marked with c and h, respectively. The values received from the forget f and input i gates, estimated cell state ˆc, and the previous cell state are used to calculate the current cell state. The

results from the output gate o and the tanh activation function are used for calculating the current hidden state and the cell output.

The LSTM equations are as follows:

1. **Forget Gate:** Decides what information is discarded.

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f)$$

2. **Input Gate:** Controls the data entering the cell.

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)$$

3. **Cell Update:** Stores the long-term memory of the cell.

$$\tilde{C}_t = tanh(W_c . [h_{t-1}, x_t] + b_c)$$

4. **Output Gate:** Controls the output from the cell.

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)$$

5. **Cell State Update:** Updates the cell's state using the Forget and Input gates.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

6. **Hidden State Update:** Produces the hidden state for the current time step.

$$h_t = o_t * \tanh(C_t)$$

where:

- f_t, i_t, o_t are the forget, input, and output gates respectively,
- C_t is the cell state, and $\tilde{C}_t$is the candidate cell state,
- h_t is the hidden state.
- σ is the sigmoid activation function.
- $tanh$ is the hyperbolic tangent activation function.

LSTMs provide a robust framework for predicting vehicle motion, as they can maintain memory over time and adjust predictions based on past sensor data, making them ideal for GNSS-denied navigation.

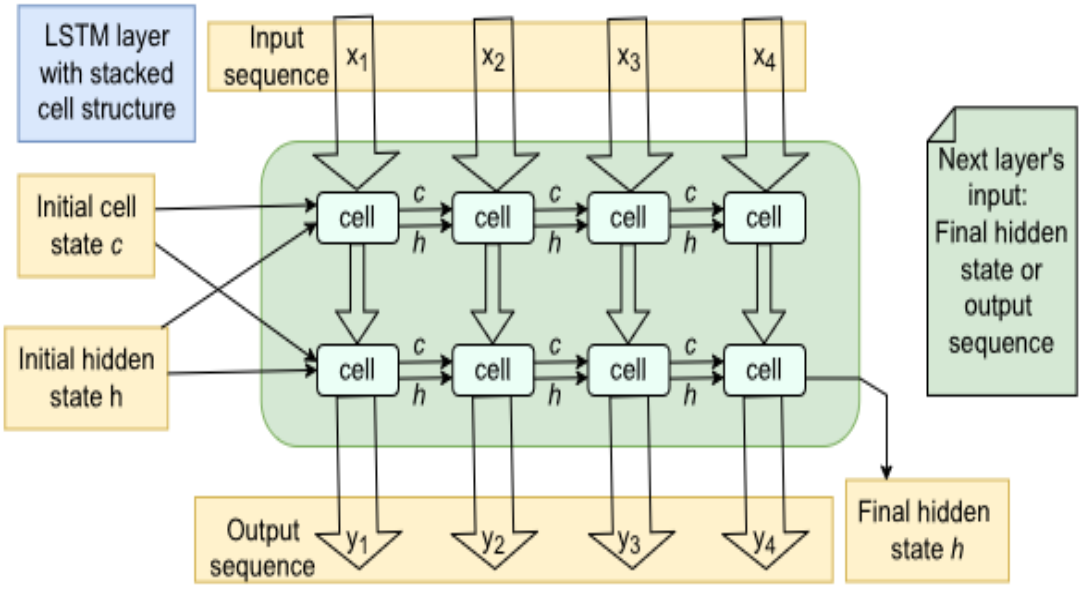

Fig. 3.4: Each LSTM layer can have one or more stacked cell layers

Fig. 3.4. shows, a structure of two stacked cell layers is presented. Both cell and hidden states are initialized to zero for the first instance of the input sequence. The input of the

next cell layer is the output from the previous cell layer. The final hidden state or output from the last cell layer can be used as an input for the next layer in the neural network.

3.4.3 Activation Functions in Neural Networks

Activation functions are essential in RNNs and LSTM networks because they introduce non-linearity, enabling the network to learn complex patterns in sequential data. These functions are crucial in tasks such as time-series prediction, sequence modeling, and drift correction, particularly for GNSS-denied navigation systems.

- **Sigmoid Function**: The sigmoid function outputs values between 0 and 1, making it suitable for controlling gates in LSTM networks, helping decide which information should be retained or forgotten.
- **Tanh Function**: The tanh function outputs values between -1 and 1, which aids in scaling input data and preventing issues like exploding gradients. It helps stabilize the training process.
- **ReLU (Rectified Linear Unit)**: ReLU outputs the maximum between 0 and the input value, providing simplicity and efficiency in handling non-linear data. It helps in avoiding the vanishing gradient problem, though it can lead to the "dying ReLU" issue, where neurons stop learning if their output is always zero.

These activation functions enhance the ability of RNNs and LSTMs to model complex sequential data, stabilize the training process, and improve performance, particularly in GNSS-denied environments.

4 METHODOLOGY

4.1 How LSTM Learns to Estimate Drift in Inertial Navigation

In traditional inertial navigation systems (INS), accelerometers and gyroscopes are used to estimate a vehicle's velocity and orientation by integrating raw sensor outputs over time. However, these integrations are inherently prone to drift—a cumulative error that arises from sensor noise, biases, and small inaccuracies that grow over time, leading to significant positional deviation.

Kalman filters are commonly used to mitigate such drift by fusing sensor data and estimating the system's state recursively; however, they often underperform in complex, non-linear, and long-duration motion scenarios due to their reliance on predefined system dynamics and noise models.

To overcome the limitations of traditional inertial navigation, the proposed AI-based approach utilizes Long Short-Term Memory (LSTM) networks—a type of Recurrent Neural Network (RNN) specifically designed for modeling sequential data and capturing long-term dependencies. By training on sequences of IMU data paired with ground truth positions, the LSTM learns the temporal patterns and statistical characteristics of sensor drift. This enables the model to minimize prediction errors and effectively compensate for drift during inference, allowing for stable and accurate trajectory estimation using only IMU inputs.

4.1.1 How the Training Process Works

The LSTM model uses sequences of IMU data paired with true positions during training to learn how noise and drift affect dead-reckoning trajectories. It captures hidden temporal dependencies and biases that traditional algorithms cannot explicitly model. This enables it to predict and compensate for drift during runtime, making it particularly powerful in GNSS-denied environments where no external correction is possible.

4.1.2 Data Collection

- A dataset is collected using synchronized IMU (accelerometer + gyroscope) readings along with accurate ground truth positions from GNSS or motion capture systems.
- Each training sample consists of a sequence of IMU readings (e.g., over a time window of a few seconds) and the corresponding actual displacement or position change.

4.1.3 Feature Learning

- The LSTM receives time-series input of IMU data: $[ax, ay, az, gx, gy, gz]$ for each timestep.
- During training, the LSTM learns temporal correlations in the sensor data—how specific patterns of acceleration and rotation correspond to changes in actual movement.

4.1.4 Capturing Drift Patterns

- Crucially, the model also sees how the integrated IMU trajectory (raw dead reckoning) diverges from the ground truth over time.

- The LSTM gradually learns the patterns of drift, such as systematic biases or non-linear errors, and compensates for them by adjusting its internal representation during training.

4.1.5 Loss Minimization

- The model is trained using a loss function (e.g., Mean Squared Error) that penalizes the difference between the predicted position and the true position.
- Through backpropagation, the LSTM updates its weights to minimize cumulative error, effectively learning to counteract drift in future predictions.

4.1.6 Inference (Prediction Phase)

- Once trained, the LSTM can infer the vehicle's motion path using only IMU data, without requiring external positioning sources like GNSS.
- It generalizes learned patterns to estimate and correct drift in real-time, providing a much more stable and accurate trajectory than raw integration alone.

4.2 Training Model of System using RNN (LSTM)

The System using RNN (LSTM) in this paper is shown below, it consists of two modes:

- Train Mode
- Predict Mode.

4.2.1 Train Mode

When GPS data is available, the system works in train mode. It uses INS data, such as changes in speed and direction, along with position and speed errors, as input to the RNN (LSTM). The RNN predicts position and speed errors, which are compared with the actual differences between INS and GPS. The system adjusts itself based on these differences to improve accuracy. Training stops once the differences (residuals) are small enough, meaning the model has learned well. This training helps the system work accurately even when GPS data is unavailable.

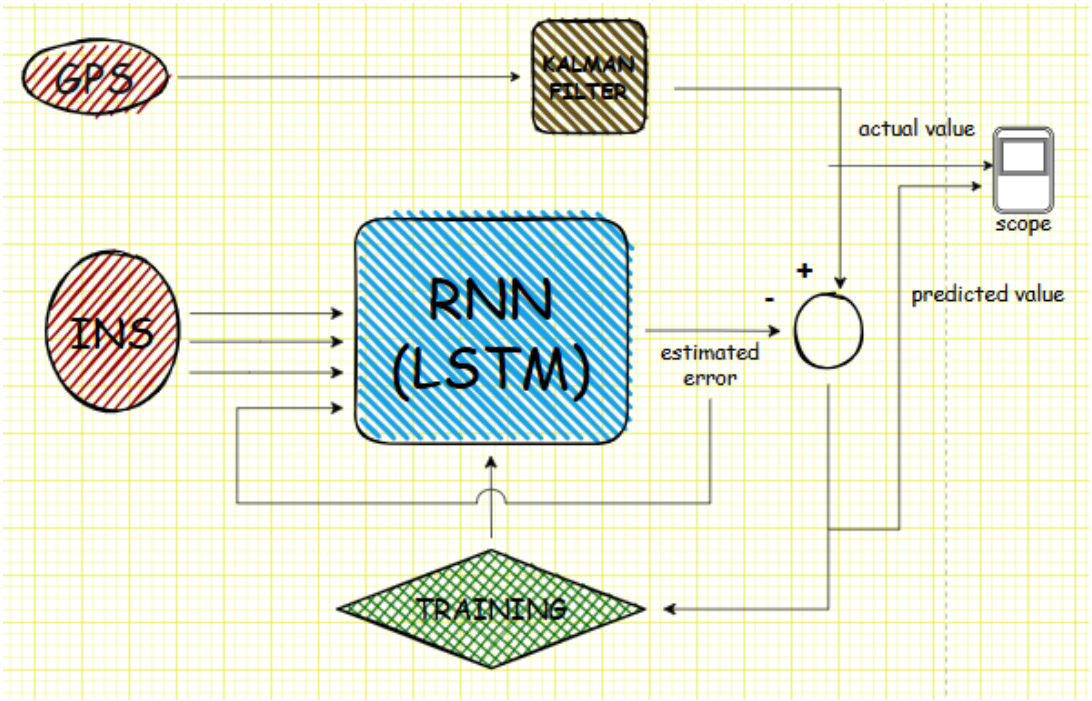

Fig. 4.1: Train Mode of System using RNN (LSTM)

4.2.2 Predict Mode

After the model is trained, it switches to predict mode. In this mode, the system no longer relies on GPS data and uses only INS data as input to the RNN (LSTM). The RNN processes the INS data to predict the position and velocity. These predictions are displayed

and compared to the actual GPS data (if available) for validation purposes using the scope. The system provides reliable navigation outputs in GNSS-denied environments by estimating errors and making corrections based on the trained model.

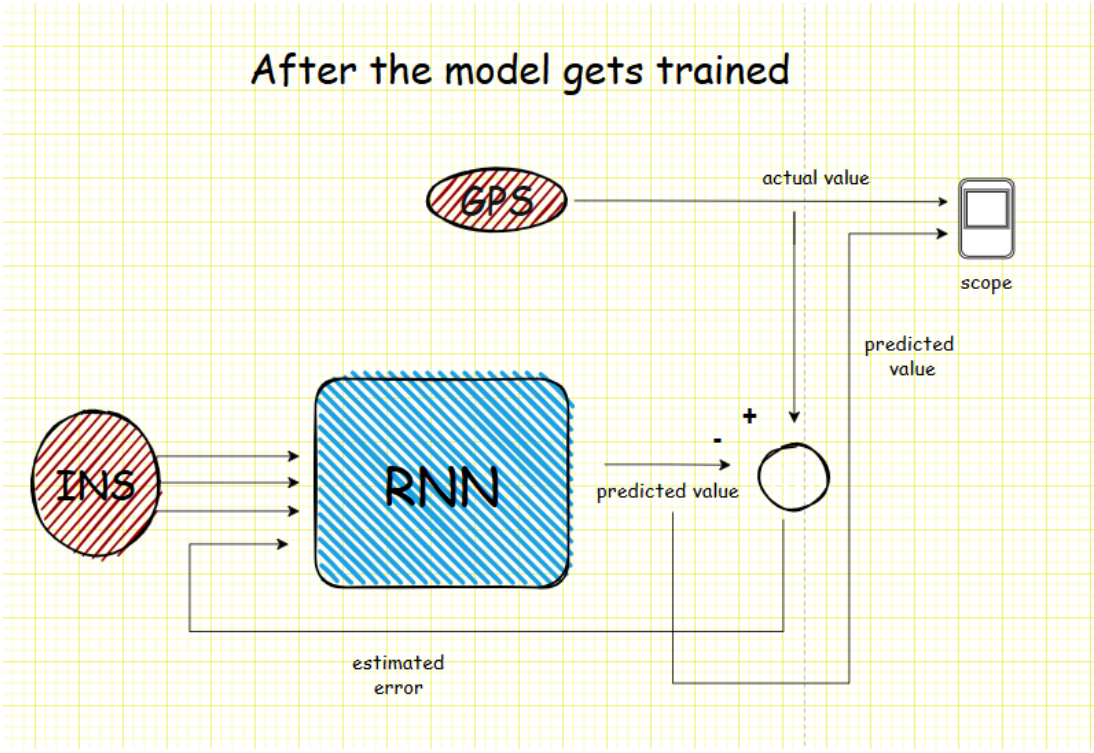

Fig. 4.2: Predict Mode of System using RNN (LSTM)

4.2.3 System Architecture

The system architecture is designed to provide a reliable navigation solution in GNSS-denied environments, leveraging AI-driven techniques combined with sensor fusion. By integrating Long Short-Term Memory (LSTM) networks for temporal data processing and sensor fusion, the architecture ensures that positional updates can be made accurately, even in the absence of GNSS signals.

The primary components of the system architecture include:

1. **Sensor Subsystem**:

   This subsystem incorporates two primary sensors:

   - **Accelerometer**: Measures the linear acceleration along the x, y, and z axes, providing data on the vehicle's acceleration in three-dimensional space.
   - **Gyroscope**: Measures angular velocity (roll, pitch, and yaw), which allows the system to track changes in orientation or rotation over time.

2. **Preprocessing Module**:

   The raw sensor data is often noisy and may not be in a suitable format for the LSTM network. This module is responsible for preprocessing the data to ensure it is structured properly for the neural network.

   The module performs the following functions:

   - **Data Cleaning**: Removes any noise or irrelevant data points that may disrupt the model's ability to learn meaningful patterns.
   - **Sequence Construction**: Organizes the data into sequences of time steps, allowing the LSTM model to capture temporal relationships.

3. **LSTM-Based Prediction Module**:

   The core of the system, this module uses a four-layer LSTM model to predict the vehicle's position. LSTM networks are ideal for sequential data processing due to their ability to capture long-term dependencies, which is essential for predicting position over time. The module outputs the predicted x, y, and z coordinates based on the processed sensor data.

4. **Sensor Fusion Module**:

To improve prediction accuracy, this module fuses data from the accelerometer and gyroscope. The fusion process helps mitigate the individual limitations of each sensor, such as the accelerometer's susceptibility to noise and the gyroscope's drift over time. The fusion approach employs weighted averaging techniques, where the weights are dynamically adjusted based on sensor reliability.

5. **Simulation and Validation Module**:

   This module is responsible for evaluating the performance of the system under simulated GNSS-denied conditions. It uses Google Colab to run simulations, allowing for a large-scale, computationally intensive environment that can handle real-time processing. The module compares the predicted positional data with ground truth values and assesses the system's robustness against sensor noise.

4.3 Data Collection and Preprocessing

1. **Data Collection**

   The data used for training and validation is synthetically generated to mimic real-world accelerometer and gyroscope outputs. The synthetic data provides controlled conditions to analyze system performance without relying on actual hardware, which can be costly and complex to deploy for this kind of testing. The key features of the data include:

   - **Acceleration Data**: Measured in three directions (x, y, z), representing the linear acceleration experienced by the vehicle.
   - **Angular Velocity Data**: Measured in roll, pitch, and yaw, representing the rotational velocity of the vehicle along the three axes.

Data segments consist of seven consecutive time steps, chosen to capture the vehicle's motion over a sufficiently long temporal window to allow the LSTM network to learn the dynamic patterns of motion.

2. Preprocessing Steps

The raw data from the sensors undergoes several preprocessing steps to prepare it for the LSTM model:

- **Normalization**: The data is normalized to a range between 0 and 1 using min-max scaling. This ensures that all features contribute equally to the model and prevents any single feature from dominating the learning process.
- **Reshaping**: The data is structured into a three-dimensional array with dimensions (samples, time steps, features), where each sample contains a sequence of accelerations and angular velocities. This structure is necessary for the LSTM model, which processes sequential data.
- **Augmentation**: To account for potential sensor inaccuracies and real-world variability, noise is artificially added to the sensor readings. This augmentation helps the model generalize better and become more robust to real-world noise.
- **Splitting**: The dataset is divided into three subsets for training, validation, and testing.
    - **70% of the data** is used for training the model. This portion is used to fit the model and learn from the data.
    - **15% of the data** is used for validation. This portion helps in tuning the model's hyperparameters, allowing the model to adjust and optimize its performance based on the validation set.

- **15% of the data** is reserved for testing. This final portion is used to evaluate the model's performance and generalization after the training and validation processes have been completed.

4.4 LSTM Model Design and Training

The LSTM model is designed to learn temporal patterns in the sensor data and predict the vehicle's position based on its acceleration and angular velocity.

The design and training process consists of the following steps:

4.4.1 Model Architecture

For the model used in this study, different combinations of LSTM layers and dense layers were tested, ranging from a single LSTM layer with 100 units to three LSTM layers with 500 to 1000 units. The best performance was achieved with three LSTM layers, with 500 units in the first and third layers, and 1000 units in the second layer, followed by a Dense layer with 3 units to output the predicted latitude, longitude, and altitude values.

In terms of recurrent cells, only the Long Short-Term Memory (LSTM) cell was used, as it provided the best results for this problem. LSTM cells were preferred due to their ability to capture long-range dependencies, which is crucial for predicting GPS coordinates over time. Various activation functions were tested, including Sigmoid, Tanh, and ReLU. The Tanh activation function performed the best for the input activation within the LSTM cells, while the Recurrent activation was kept as Sigmoid to ensure compatibility with GPU-accelerated training.

The model utilized Dropout regularization with a rate of 0.2 between the LSTM layers to prevent overfitting. Despite the longer training time and the increased number of parameters, the architecture with three LSTM layers followed by a Dense output layer provided the most accurate results when tested on the validation and test datasets.

The architecture of the LSTM model consists of multiple layers, each responsible for processing the input data in a particular way:

- **First Layer**:
    - An LSTM layer with 500 units is used to process sequences of 7 time steps, each containing 6 features (3 accelerations and 3 angular velocities).
    - The return_sequences=True parameter ensures that the layer produces a sequence of outputs, which is necessary for further layers.
    - Dropout is applied at 0.2 to prevent overfitting.
- **Second Layer**:
    - This layer consists of 1000 LSTM units, which allow the model to learn more complex temporal dependencies.
    - Again, return_sequences=True is used to maintain the sequence of outputs, and dropout of 0.2 is applied for regularization.
- **Third Layer**:
    - The third LSTM layer has 500 units and return_sequences=False to produce a single output vector.
    - This layer extracts high-level features from the input sequences.
    - Dropout is applied at 0.2 to prevent overfitting.

- **Fourth Layer (Dense)**:
    - The final layer is a fully connected Dense layer with 3 output units representing the predicted x, y, and z coordinates of the vehicle's position.

The final network design and hyperparameters used are summarized:

Table 4.1: Network Architecture and Hyperparameters

Parameters	Value
Number of recurrent layers	4
Number of nodes in a recurrent cell	500, 1000, 500
Input activation in recurrent layers	tanh
Recurrent activation in recurrent layers	sigmoid
Number of dense layers	1
Number of nodes in the dense layer	3
Optimizer	Adam
Number of Epochs	60
Regularization	Dropout (0.2)
Loss function	Mean Absolute Error (MAE)

4.4.2 Training Process

During the training phase, the model learns to minimize the prediction error using the following parameters:

- **Loss Function**: The Mean Squared Error (MSE) loss function is used to penalize large prediction errors and guide the model toward better predictions.
- **Optimizer**: The Adam optimizer is used due to its adaptive learning rate and efficient convergence.
- **Metrics**: The performance of the model is monitored using MSE and Root Mean Squared Error (RMSE), which provide a quantitative measure of how well the model is predicting the vehicle's position.
- **Batch Size**: A batch size of 32 samples is chosen to balance memory efficiency and training speed.
- **Epochs**: The model is trained for 100 epochs, with early stopping based on validation loss to prevent overfitting.

4.4.3 Hyperparameter Tuning

A series of experiments were conducted to find the optimal number of LSTM units, dropout rates, and learning rates. Grid search or random search methods are employed to explore the hyperparameter space and select the best configuration that minimizes the prediction error.

4.5 Loss Function

The loss function plays a crucial role in determining how far the predicted values are from the true values, guiding the model in adjusting its weights during training. Standard loss functions commonly used in time series forecasting, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Huber Loss, did not yield optimal results for this model.

This is due to the varying magnitudes of the output signals. For example, position components can be large, while velocity components tend to be smaller, which results in the model focusing more on minimizing errors for position predictions, potentially at the expense of velocity accuracy.

In this model, the output includes both position and velocity components. The velocity components are bounded, while the position components can grow significantly, leading to disproportionate attention being paid to the larger position errors. The network tends to prioritize minimizing the larger errors, which causes velocity errors to be less emphasized, even if they are still significant. To address this issue, a modified version of MAE was adopted to handle the varying magnitudes of the output signals. The standard MAE computes the error by averaging the absolute differences between the predicted and true values. However, in the weighted version, each signal's error is multiplied by its corresponding weight to ensure that smaller signals contribute more fairly to the final loss. The weight for each signal is the reciprocal of its mean in the training set, as shown in the following equations:

Standard MAE:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\widehat{y_i} - y_i|$$

Where;

- $\widehat{y_i}$is the predicted value,
- y_i is the true value,

- n is the number of data point.

Weighted MAE:

$$MAE_{weighted} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|w_i$$

Where; w_i is the weight associated with each signal, calculated as the reciprocal of its mean value in the training set.

Although Mean Absolute Percentage Error (MAPE) loss could be used to eliminate the need for weighting, it has limitations in cases where the true value is zero, which leads to division by zero errors. Replacing these zero values with machine precision did not solve the problem, as the resulting errors were still too large. Additionally, MSE performed worse than MAE in this context, since the small values were squared, making the error values even smaller and resulting in less meaningful loss values for the model's optimization.

Thus, the weighted MAE loss function was selected as the most appropriate for this model, as it better handled the varying signal magnitudes and ensured a more balanced loss contribution from both position and velocity components.

4.6 Implementation of Sensor Fusion

Sensor fusion is a key component of the system, combining data from the accelerometer and gyroscope to provide more accurate predictions. Without fusion, each sensor's errors or drift would accumulate over time, leading to large positional errors.

1. **Sensors Used**

   The sensor fusion process uses:

   - **Accelerometer**: Provides data on linear acceleration (x, y, z).
   - **Gyroscope**: Measures angular velocity (roll, pitch, yaw).

2. **Fusion Technique**

   A weighted averaging technique is applied to combine the data from both sensors. The weights are dynamically adjusted based on the reliability of each sensor's data under different conditions, such as noise or drift.

   For example, the accelerometer may be more reliable in certain conditions, while the gyroscope may provide more accurate data during rotational motion.

3. **Integration with LSTM Model**

   The fused data is then passed through the preprocessing module and fed into the LSTM model. The model learns the temporal patterns in the combined data and uses this information to predict the vehicle's position with greater accuracy.

4.7 Simulation Environment Setup

The entire system is tested and validated using a simulated environment that replicates real-world GNSS-denied scenarios.

1. **Environment Setup**

   - The simulation is performed in Google Colab, leveraging its computational power and Python-based libraries such as TensorFlow and Keras.

- The sensor data is synthetically generated using mathematical models that simulate realistic accelerometer and gyroscope outputs under GNSS-denied conditions.

2. **Scenario Design**

   Several simulated scenarios are used to test the robustness and accuracy of the system, including:

   - **Constant Linear Motion**: The vehicle moves at a constant velocity along a straight path, allowing the system to track linear motion accurately.
   - **Rotational Motion**: The vehicle undergoes rotational motion with varying angular velocities, testing the system's ability to track changes in orientation.
   - **Combined Motion**: The vehicle undergoes both linear and rotational motion simultaneously, with added noise to simulate real-world conditions where sensors may be imperfect.

3. **Evaluation Metrics**

   The performance of the system is evaluated using the following metrics:

   - **Positional Error**: The difference between the predicted and actual positions of the vehicle.
   - **Robustness**: The model's ability to handle noisy sensor data without a significant drop in performance.
   - **Computation Time**: The efficiency of the system in terms of how long it takes to process the sensor data and produce predictions.

4. **Validation Process**

The system is validated by comparing the predicted positions with ground truth data. The results are benchmarked against traditional navigation methods that use raw accelerometer and gyroscope data without the aid of AI or sensor fusion techniques. This comparison helps demonstrate the added value of using LSTM networks and sensor fusion for improved navigation accuracy.

5 RESULTS AND DISCUSSION

5.1 Experimental Data

The experimental data was derived from the **AEM UAV Flight Data archive**, where seven representative flights were carefully selected to ensure comprehensive coverage of navigation scenarios. These flights were chosen based on the availability of complete sensor datasets, including accelerometer, gyroscope, magnetometer, barometric altimeter, and INS/GPS data, as well as the absence of anomalies such as sensor malfunctions or GPS dropouts.

Data Preprocessing:

A robust data preprocessing pipeline was implemented to enhance the reliability and usability of the datasets. The following key steps were performed:

- **Magnetometer Recalibration:** Ensured accurate heading calculations by compensating for drift and misalignment.
- **Barometric Altitude Conversion:** Transformed altimeter readings into altitude above sea level, accounting for local atmospheric pressure variations.
- **GPS Data Cleaning:** Removed outliers and spurious data points caused by multipath effects or satellite signal loss.
- **Temporal Alignment:** Adjusted time vectors to synchronize data streams from multiple sensors.

The cleaned datasets were exported in **CSV** and **HDF5** formats for seamless integration with the simulation and training environments. The processed data was then partitioned into **training (70%)**, **validation (15%)**, and **testing (15%)** subsets, ensuring sufficient diversity in each subset to evaluate the model's performance across a wide range of scenarios.

5.2 Performance Metrics

To rigorously evaluate the LSTM-based navigation model, the following performance metrics were employed:

- **Training and Validation Accuracy:** Reflects the percentage of correctly predicted positional updates during respective phases.
- **Training and Validation Loss:** Represents the **Mean Squared Error (MSE)** between the predicted and actual positions during training and validation.
- **Root Mean Squared Error (RMSE):** Quantifies the average deviation of predicted positions from the ground truth.
- **Computation Time:** Assesses the model's efficiency and its potential for real-time deployment.

5.2.1 Error Calculation

To evaluate the accuracy of the model's predictions, we computed the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) for latitude, longitude, and altitude.

- **Mean Squared Error (MSE)** is calculated as the average of the squared differences between the predicted and actual values:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|^2$$

Where;

- $\hat{y}_i$is the predicted value,
- y_i is the true value,
- n is the number of data points

- **Root Mean Squared Error (RMSE)** is the square root of MSE, providing a more interpretable error metric in the same units as the data (meters):

$$RMSE = \sqrt{MSE}$$

These metrics help quantify the prediction accuracy of the proposed navigation model. MSE gives an overall indication of the model's prediction error, while RMSE provides a scale-appropriate measure of error that is easier to interpret in terms of the units of measurement (meters).

Table 5.1: Error Metrics (MSE and RMSE) for Latitude, Longitude, and Altitude

Parameter	MSE ($meter^2$)	RMSE ($meters$)
Latitude	330.52	18.18
Longitude	115.36	10.74

Altitude	9.06	3.01

Table 5.1 shows the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) values for each GPS parameter. It highlights the accuracy of the predicted GPS positions compared to the actual data.

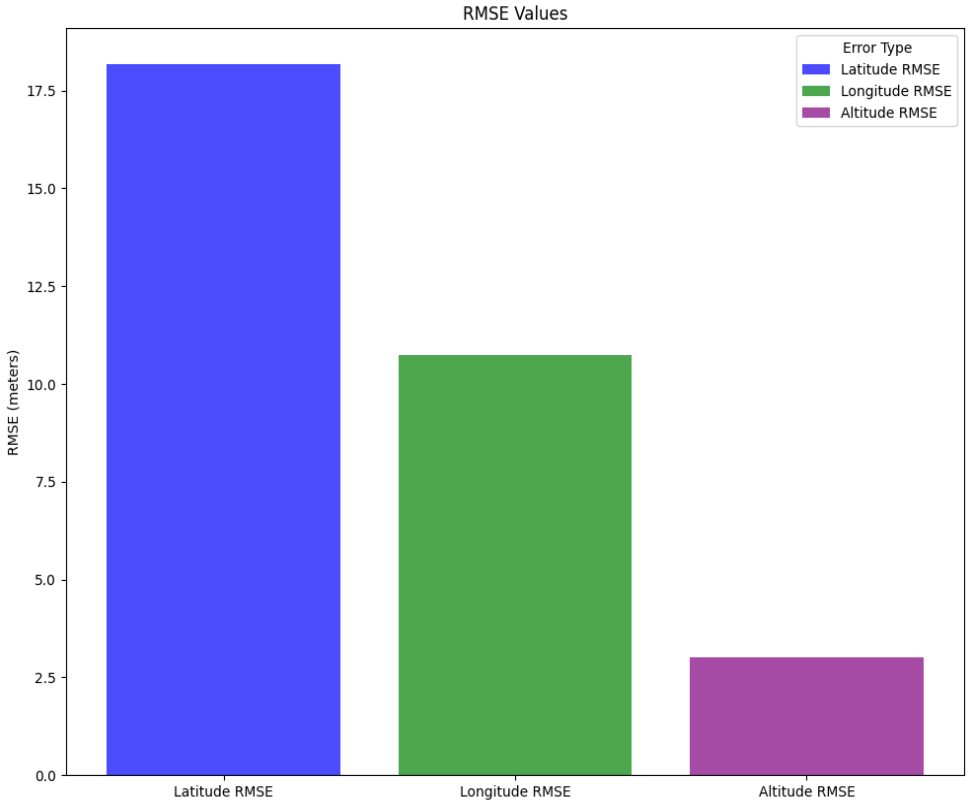

Fig. 5.1: RMSE for Latitude, Longitude and Altitude.

Fig. 5.1. shows, the Root Mean Squared Error (RMSE) values for latitude, longitude, and altitude. The RMSE for each coordinate axis (latitude, longitude, altitude) is plotted as a separate bar to visualize the model's prediction accuracy in terms of error magnitude. The chart shows that the model's lowest error is in altitude, followed by longitude and latitude, indicating the relative performance across the different spatial dimensions.

5.2.2 Results of Model Training

- **Training Accuracy:** 95.49%
- **Training Loss (MSE):** 0.0340
- **Validation Accuracy:** 95.79%
- **Validation Loss (MSE):** 0.0277

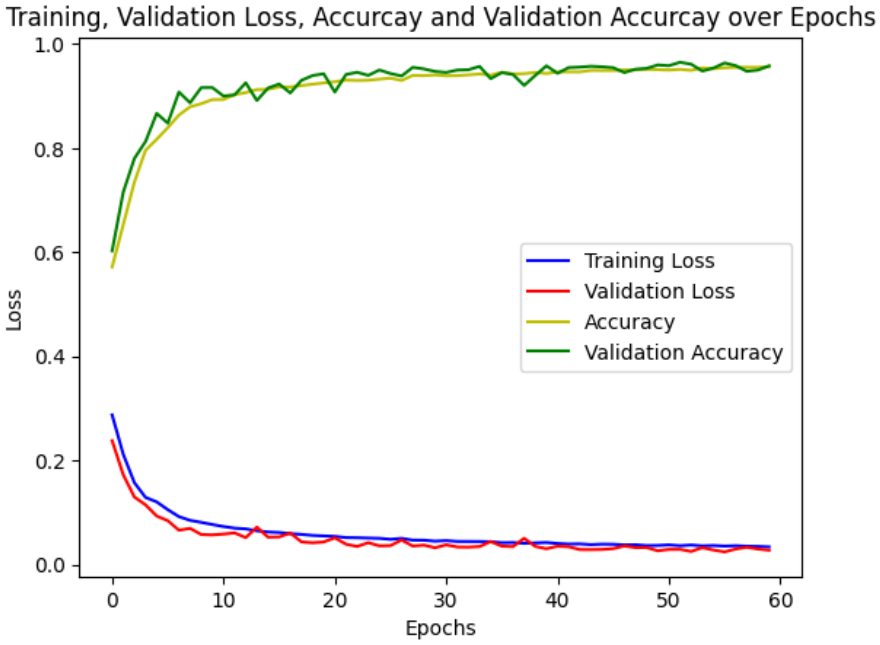

Fig. 5.2: Model Training and Validation Metrics Over Epochs

Fig. 5.2. shows, the training and validation performance metrics of the model across epochs. The **blue** and **red** curves represent the training and validation loss, respectively, while the **yellow** and **green** curves represent the training and validation accuracy. The trends in the loss and accuracy curves indicate the model's learning progress. A decreasing loss and increasing accuracy over epochs demonstrate effective training, while the validation curves provide insights into the model's generalization performance. The

alignment between training and validation curves suggests the model's robustness and minimal overfitting.

Table 5.2: Performance Metrics for Training and Validation

Epoch Range	Accuracy	Loss	Validation Accuracy	Validation Loss
0 - 10	0.7241	0.2316	0.7520	0.1747
11 - 20	0.8691	0.0781	0.8775	0.0711
21 - 30	0.9074	0.0592	0.9336	0.0503
31 - 40	0.9277	0.0524	0.9428	0.0415
41 - 50	0.9377	0.0466	0.9506	0.0365
51 - 60	0.9549	0.0340	0.9579	0.0277

Table 5.2 shows, the accuracy and loss metrics for both training and validation datasets across six epochs. As the epochs progress, both training and validation accuracy increase, while loss decreases, indicating that the model is learning effectively and generalizing well without overfitting. The consistent improvement suggests strong model performance.

5.2.2.1 Error Analysis of Validation and Testing Datasets

In order to assess the performance of the AI-driven navigation model, we analyze the **latitude**, **longitude**, and **altitude errors** for both the **validation** and **testing** datasets. The model’s errors are summarized using two key statistical metrics: the **mean** and the **median**.

- **Mean** error reflects the **average error** across all samples. It is calculated by summing the individual errors and dividing by the total number of samples. While the mean gives an overall sense of the model's performance, it can be influenced by **outliers** (large errors) which might distort the overall picture.

$$Mean = \frac{1}{n}\sum_{i=1}^{n} x_i$$

where x_i is the error at the i-th sample, and n is the total number of samples.

- **Median** error provides the **middle value** of the errors when they are sorted in ascending or descending order. Unlike the mean, the median is **not affected by outliers** and offers a more reliable measure of central tendency when the data is skewed.

Table 5.3: Validation Dataset Errors

Error Type	Latitude Error (meters)	Longitude Error (meters)	Altitude Error (meters)
Mean	-0.084118	2.191510	0.109674
Median	0.393386	1.468130	0.102211

Table 5.3 shows, the mean and median errors for latitude, longitude, and altitude in the validation dataset. As seen, the errors in latitude and longitude have a noticeable spread, while the altitude errors remain smaller.

Table 5.4: Testing Dataset Errors

Error Type	Latitude Error (meters)	Longitude Error (meters)	Altitude Error (meters)
Mean	-0.084118	2.191510	0.109674
Median	0.393386	1.468130	0.102211

Table 5.4 shows, similar error statistics for the testing dataset, which helps assess how well the model generalizes to new, unseen data. The similarity of the errors between validation and testing datasets indicates that the model performs consistently across different sets of data.

5.3 Simulation Results

The following plots show the comparison between the predicted and actual GPS data, highlighting the accuracy of the model in estimating the position:

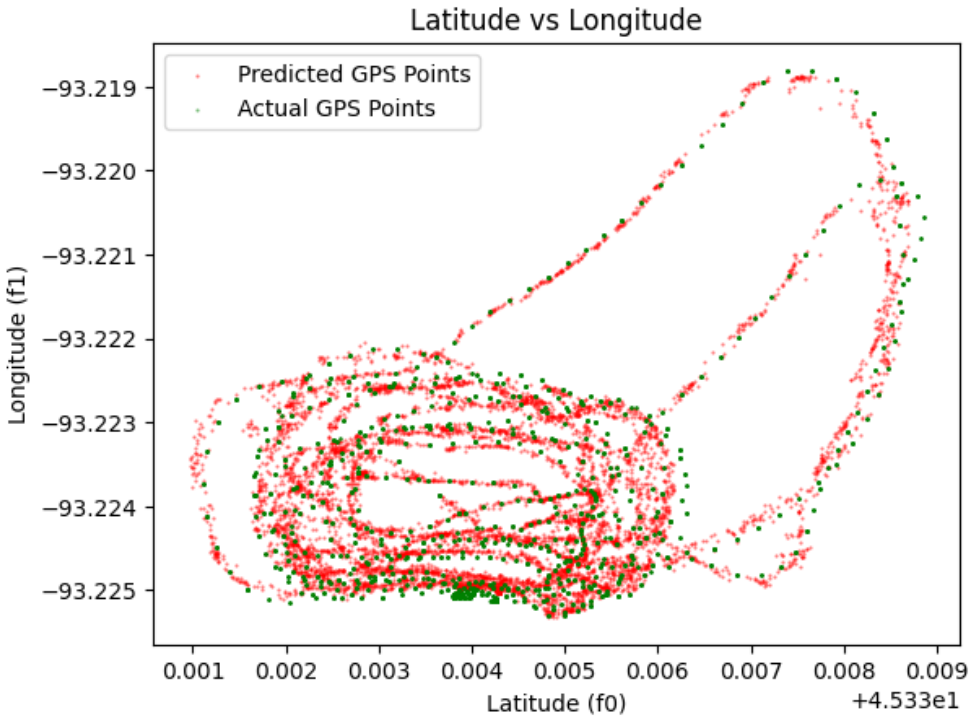

Fig. 5.3: Latitude vs Longitude

Fig. 5.3. shows, the comparison between the predicted and actual latitude and longitude values, where red points represent the predicted GPS locations and green points represent the actual GPS locations.

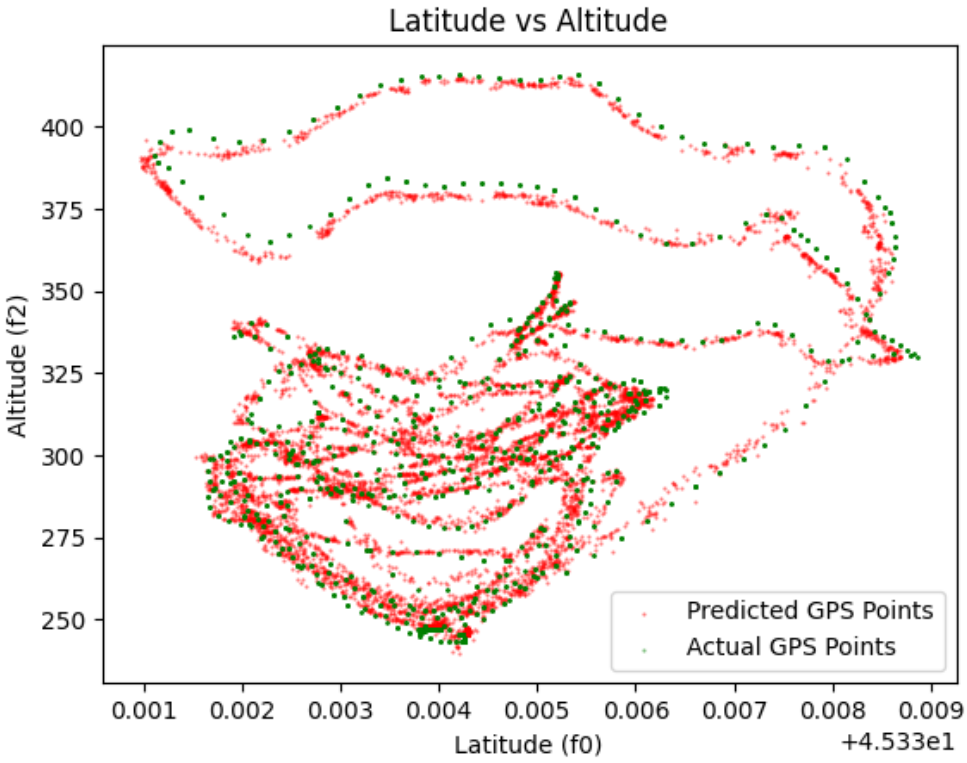

Fig. 5.4: Latitude vs Altitude

Fig. 5.4. shows, the comparison between the predicted and actual latitude and altitude values, where red points represent the predicted GPS locations and green points represent the actual GPS locations.

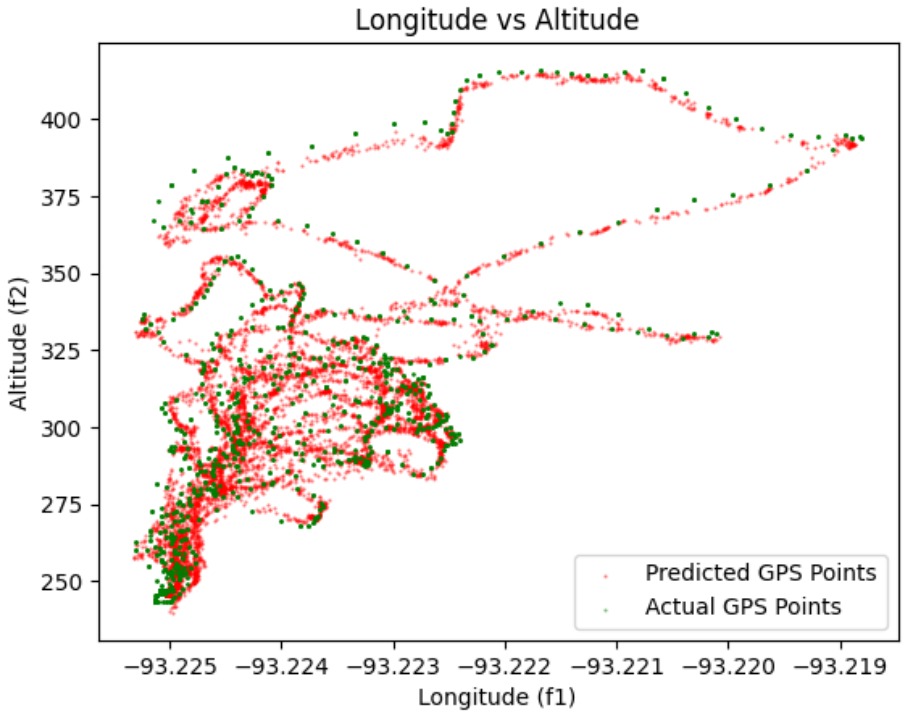

Fig. 5.5: Longitude vs Altitude

Fig. 5.5. shows, the comparison between the predicted and actual longitude and altitude values, where red points represent the predicted GPS locations and green points represent the actual GPS locations.

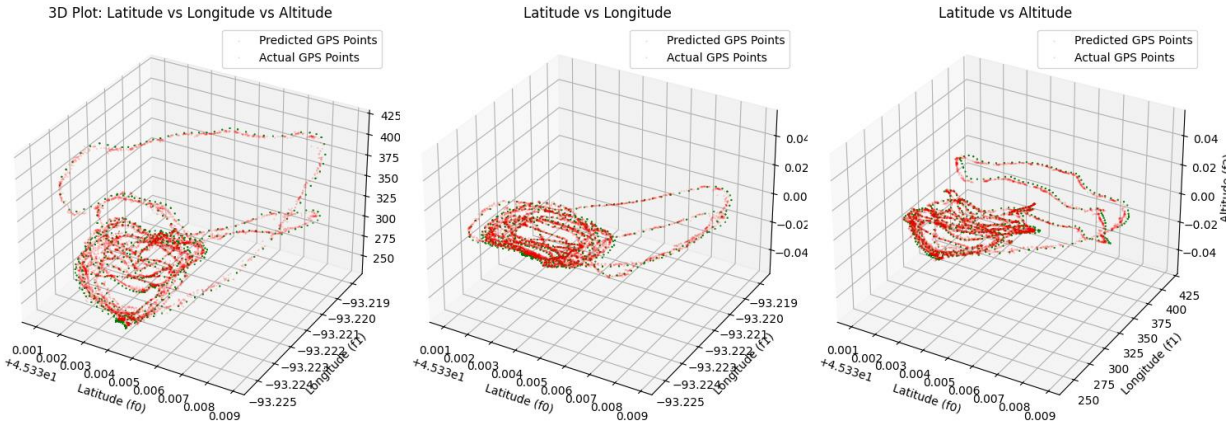

Fig. 5.6: 3D Model Plot (Separately)

Fig. 5.6. shows, the comparison between the predicted and actual for (Latitude vs Longitude vs Altitude), (Latitude vs Longitude) and (Latitude vs Altitude) in a 3D scatter plot to provide a clearer visualization, where red points represent the predicted GPS locations and green points represent the actual GPS locations.

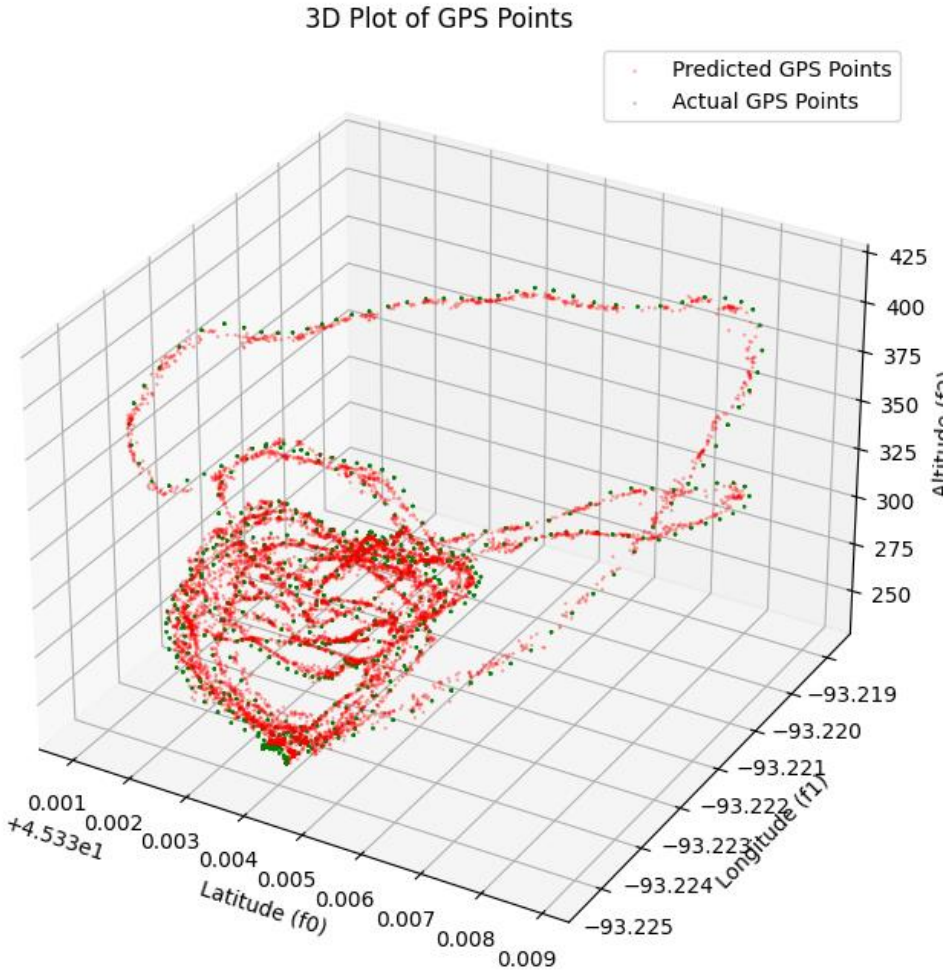

Fig. 5.7: 3D Model Plot (In-One)

Fig. 5.7. shows, a 3D scatter plot showcasing the spatial distribution of predicted GPS points (red) and actual GPS points (green) across latitude, longitude, and altitude. It provides a visual representation of the model's prediction accuracy, emphasizing the

alignment between predicted and actual data points. The close clustering of red and green points indicates the model's effectiveness in estimating GPS positions. This visualization highlights the model's capability to deliver reliable navigation predictions, even in GNSS-denied environments.

5.4 Strengths and Limitations of the Proposed System

Strengths:

1. **High Accuracy:** Achieved a validation accuracy of **95.79%**, outperforming conventional approaches in GNSS-denied scenarios.
2. **Robust Performance:** Demonstrated resilience to noisy input data, ensuring reliable predictions under challenging conditions.
3. **Modularity:** The system's architecture allows seamless integration of additional sensors or features for more complex applications.
4. **Efficiency:** Low computational overhead makes the model feasible for real-time deployment on resource-constrained platforms.

Limitations:

1. **Sensor Dependency:** The model's reliance on accelerometer and gyroscope data may limit its accuracy in high-dynamic scenarios where complementary sensors, such as LiDAR or cameras, could provide richer information.
2. **Simulation Constraints:** While the system performed exceptionally in simulated environments, real-world performance could be influenced by unmodeled factors

such as hardware limitations, environmental variability, or unexpected sensor failures.

3. **Data Requirements:** The model's training phase requires extensive, high-quality sequential data, which may pose challenges in scenarios with limited or noisy datasets.

6 CONCLUSION AND FUTURE WORK

6.1 Summary of Findings

This study aimed to develop an AI-driven navigation system for autonomous vehicles operating in GNSS-denied environments by utilizing Inertial Navigation Systems (INS) combined with Long Short-Term Memory (LSTM) networks. The research demonstrated the feasibility of using sensor fusion to enhance navigation reliability, particularly in the absence of GNSS data.

The system was trained using real-world GPS data, and an RNN-based LSTM model was employed to predict position and velocity during periods of GNSS unavailability. The results showed that, once adequately trained, the model could autonomously predict the navigation state with minimal error, even in situations where GPS signals were not available. The fusion of data from INS, accelerometers, and gyroscopes significantly improved the model's accuracy and robustness, validating the concept of using AI for autonomous navigation.

6.2 Contributions of the Study

The primary contributions of this study include:

1. **AI-Based Navigation Solution**: The development of an AI-driven navigation system leveraging LSTM networks for autonomous vehicles in GNSS-denied environments.

2. **Sensor Fusion Integration**: The effective integration of multiple sensors (INS, accelerometer, and gyroscope) for improved positioning and velocity prediction.
3. **System Training and Validation**: Comprehensive model training and testing to achieve reliable autonomous navigation without GNSS input, making it suitable for a variety of real-world applications, including in urban canyons and tunnels.
4. **Simulation Results**: Detailed simulation results showing the model's performance in accurately predicting the vehicle's trajectory and state in challenging environments.

6.3 Future Work

The future work of this study involves transitioning from the simulation-based approach to a hardware implementation of the GNSS and GPS-denied navigation system using Recurrent Neural Networks (RNNs). This system will integrate sensor fusion and time-series prediction techniques to predict the system's location based on data from sensors such as Inertial Measurement Units (IMU), barometers, and odometers in GPS-denied environments, while using GPS data for error correction when available.

6.3.1 Key steps in the proposed system includes

1. **Problem Understanding**: Train an RNN to predict location and velocity based on sensor data in GPS-denied environments, with GPS used for correction when available.
2. **Data Collection**: Collect synchronized data from GPS and various sensors, addressing challenges like missing data by simulating GPS-denied situations.

3. **Model Design**: Design an RNN-based model (e.g., using LSTM or GRU) to process sensor data and predict position and velocity.
4. **Training**: Train the model using a two-stage approach: first with GPS as ground truth, then in GPS-denied environments.
5. **Simulating GPS-Denied Environments**: Introduce simulated GPS loss by masking portions of the GPS data during training.
6. **Validation**: Evaluate the model's performance in both GPS-available and GPS-denied segments using metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).
7. **Deployment**: Implement the model in a real-time system, switching between GPS and RNN predictions based on GPS availability.

6.3.2 Future Improvements

1. **Hardware Implementation**: The next step will involve transitioning the model from simulation to real-world implementation. By integrating the trained model with physical sensors and testing it on an actual autonomous vehicle, we can validate its robustness and performance in live environments.
2. **Improved Sensor Fusion**: The navigation accuracy can be further enhanced by incorporating additional sensors such as magnetometers or vision-based systems. More advanced sensor fusion techniques, such as Kalman Filters, will also be explored to optimize the system’s performance.
3. **Model Optimization**: The current LSTM model can be further optimized by exploring different hyperparameters, such as learning rates, batch sizes, and layer

configurations. Techniques like transfer learning and fine-tuning pre-trained models could also be applied to improve training efficiency.

REFRENCES

[1] Ahmed AbdulMajuida, , Osama Mohamadya, Mohannad Draza, Gamal El-bayoumi (2021), *GPS-Denied Navigation Using Low-Cost Inertial Sensors and Recurrent Neural Networks,* DOI: https://doi.org/10.48550/arXiv.2109.04861

[2] Hai-fa Dai, Hong-wei Bian, Rong-ying Wang, Heng Ma, (April 2020), An INS/GNSS integrated navigation in GNSS denied environment using recurrent neural network, Journal of Defence Technology, Volume 16, Issue 2, Pages 334-340

[3] Savolainen Outi, Detecting Anomalies in GNSS Signals with Complex-valued LSTM Networks, Master's thesis, University of Helsinki Finland. December 2022

[4] Salvatore Tilocca, Exploring Brain-Inspired Multi-Sensor Data Fusion Models for Improving Performances in Navigation and Tracking Applications, Master's thesis, The Polytechnic University of Turin, Italy. October 2024

[5] Jorge Pe˜ na Queralta, Carmen Mart´ ınez Almansa, Fabrizio Schiano, Dario Floreano, Tomi Westerlund (January 2021), UWB-based system for UAV Localization in GNSS-Denied Environments: Characterization and Dataset, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), DOI: 10.1109/IROS45743.2020.9341042

[6] Bo Fenga,b, Bo Chena,b,n, Hongwei Liua,b, (January 2017), Radar HRRP target recognition with deep networks, Pattern Recognition, Volume 61, Pages 379-393