Dissertation on

"Implementing Inertial Navigation Systems in Wearable Devices using Machine Learning"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE22CS441A – Capstone Project Phase - 3

Submitted by:

Yash Sinha	**PES2UG22CS675**
Tushar Swami	**PES2UG22CS633**
Vedant Singh	**PES2UG22CS654**
SK HithaSree	**PES2UG22CS559**

Under the guidance of

Prof. Nagalakshmi SR
Assistant Professor
PES University

Aug - Nov 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

‘Implementing Inertial Navigation Systems in Wearable Devices using Machine Learning’

is a bonafide work carried out by

Yash Sinha	**PES2UG22CS675**
Tushar Swami	**PES2UG22CS633**
Vedant Singh	**PES2UG22CS654**
SK Hithasree	**PES2UG22CS559**

In partial fulfilment for the completion of seventh semester Capstone Project Phase - 3 (UE22CS441A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Aug 2025 – Nov. 2025. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature	Signature	Signature
Prof. Nagalakshmi SR	Dr. Sandesh B J	Dr. Sridhar K S
Assistant Professor	Chairperson	Registrar

External Viva

Name of the Examiners	**Signature with Date**
1. ____________________________	____________________________
2. ____________________________	____________________________

DECLARATION

We hereby declare that the Capstone Project Phase - 3 entitled **“Implementing Inertial Navigation Systems in Wearable Devices Using Machine Learning”** has been carried out by us under the guidance of **Prof. Nagalakshmi SR, Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester Aug – Nov. 2025. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES2UG22CS675	**Yash Sinha**
PES2UG22CS633	**Tushar Swami**
PES2UG22CS654	**Vedant Singh**
PES2UG22CS559	**SK Hithasree**

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Nagalakshmi SR, Assistant Professor, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE22CS441A Capstone Project Phase-3.

I am grateful to the Capstone Project Coordinators, Dr. Vandana M. Ladwani, Associate Professor, Department of Computer Science and Engineering, PES University for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Sandesh B J, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. Sridhar K S, Registrar, PES University for his help.

I am deeply grateful to Late Dr. M. R. Doreswamy, Founder, PES University, whose vision and dedication continue to inspire generations of learners. I would also like to express my sincere gratitude to Prof. Jawahar Doreswamy, Chancellor, PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University and Dr. Nagarjuna Sadineni, Pro Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

Fitness trackers have evolved from basic activity monitors to sophisticated wearable devices with location-aware navigation, real-time safety alerts, and medical monitoring capabilities. In spite of something, these developments, the majority of wearable navigation systems still rely heavily on GPS signals, which are not reliable in indoor spaces, tunnels, crowded cities, and forests. Particularly in situations where time or safety are of the essence, such GPS outages result in inaccurate positioning, decreased navigation accuracy, and impaired emergency response performance. This work proposes a hybrid GPS-independent navigation framework that maintains precise trajectory estimation in GPS-failed scenarios by fusing Inertial Navigation System (INS) and machine learning capabilities.

The system uses data from wearable-class sensors like accelerometers, gyroscopes, and magnetometers to make a continuous motion estimate. Machine learning models are trained to find patterns in movement, lower the amount of INS drift that builds up, and guess GPS coordinates based on past trajectories. A Kalman-based fusion method is then used to get rid of noise and keep orientation tracking stable. This research focuses on making the computational pipeline, sensor-fusion strategy, and predictive algorithms ready for use in commercial wearables instead of putting hardware into practice. The framework that comes out of this work is a scalable and flexible navigation system that reduces the need for satellite connectivity. This makes positioning more reliable for outdoor exploration, urban commuting, fitness activities, and emergencies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Wearable technology was once limited to technology gimmicks, but they have become modern-day necessary devices which we put on every day. The set of devices incorporating trackers and health monitors and smartwatches have become an integral part of our daily operations which involves health monitoring, route planning in real time. While they have developed enormous technological improvement, there are few capacities of navigation that are still lacking in certain environmental situations. The study thus proposes to address these issues with the development of a robust framework using the Machine Learning technology. The proposed research develops an Inertial Navigation System that is based on machine learning processing in wearable technology systems.

1.1 Background

Various embedded sensors and algorithms in wearable devices support drastic monitoring of physical activities along with health parameters and location-based service delivery simultaneously. Navigation is a basic functionality, which has become imperative for three types of users: outdoor enthusiasts, as well as fitness and urban commuters. Wearables mainly use the Global Positioning System (GPS) for navigation purposes; but GPS has some limitations that makes it far from ideal.

1.2 Importance of Navigation in Wearable Devices

Navigation capabilities in wearables enhance user experiences by allowing easy guidance and tracking. The applications include the following:

- Outdoor activities such as trekking and hiking.

- Urban navigation for commuters in cities.
- Emergency situations needing dependable position determination.

A navigation demands for accuracy and continuity propels alternatives to GPS, especially in areas when GPS show difficulties in sending signals, usually in high-rise buildings or deep caves in the forests.

1.3 Limitation of GPS

GPS despite its great effectiveness in open environments, has many unconsidered but major disadvantages:

- Weak Signal - GPS signals are usually blocked in tunnels, dense forests, urban deadlock, caves and indoors.
- Energy Consumption: Extended use of GPS quickly saps the battery life of wearable devices, which has detrimental effects on usage and safety of the user.
- External Infrastructure Dependency: GPS is dependent upon the connection of the satellites that local signal disruptions are a concern.

1.4 Potential of INS (Inertial Navigation System)

INS offers an alternative based on its on-board accelerometers and gyroscopes and magnetometers which give estimates of motion and position. The device Ins is independent of the information coming from the outside because it is a self-contained system, that is, it can be used where the GPS signals do not reach or are very weak.

1.4.1 Difficulties with Traditional INS

- Sensor Drift: Small errors in sensor measurements accumulate over time, reducing accuracy.

- Complexity of Data Fusion: Combining the data from different sensors to get accurate navigation requires complex computation. Role of Machine Learning in INS

1.5 Role of Machine Learning in INS

Machine Learning offers innovative solutions by addressing limitations of traditional INS including:

- Sensor Drift Reduction: Learning patterns from data to dynamically correct drift errors.
- Sensor Data Fusion: Integration of data from different sensors using advanced ML models.
- Predictive Insights: Using historical motion data to predict trajectories, increasing the reliability of INS.

1.6 Research Motivation

This research is based on finding the increasing demands for navigation in wearable devices, researching on finding out the limitations of existing GPS technology and INS technology. It mainly focuses on working for finding a reliable and noble solution to bring a combination of INS and ML for more accuracy and precision.

1.7 Organization of the Report

The first part of the paper provides foundational context for the subsequent sections that examine problem definition and research difficulties alongside research aims. The report contains analysis of previous research with additional content about exploring the dataset and outlining upcoming project development steps.

CHAPTER 2

PROBLEM STATEMENT

Modern life relies heavily on GPS or Global Positioning System technology for navigation and tracking/tracing activities that create critical problems particularly when operating in underground cities and high-rise buildings or dense forest or cave areas. GPS signal disrupts in cities with tall buildings since multiple obstacles create interference which results in inaccurate location information. The precision accuracy of GPS signals poses a challenge to both fitness-related navigation and emergency response systems because such systems depend heavily on exact information for proper performance and smooth operations

Health and fitness applications have rapidly gained popularity using wearable devices, although these devices need external signals for navigation. Such dependence greatly limits the ability of the devices to track steadily and reliably in case GPS signals are not available; hence, user localization suffers from decreased reliability.

The primary stakeholders face the risk of activity monitoring and space navigation issues, due to which their safety might be in jeopardy.

CHAPTER 3

LITERATURE REVIEW

3.1 GNSS-Denied Pedestrian Navigation Using ML-Aided Gait Recognition

3.1.1 Introduction

This paper introduces MARIO, a pedestrian navigation system that relies on wearable inertia sensors and machine learning to continuously track location in situations where GNSS signals are unavailable or unreliable. The system performs an estimation of the user's velocity from motion patterns, which is integrated into IMU data for position computation. On a 1.2-kilometer path, MARIO showed a maximum localization error of 11.88 meters, thereby showing great potential for use in urban canyons, indoors, and in emergency situations where GPS is not reliable.

3.1.2 Characteristics & Implementation

MARIO captures gait signatures through a number of low-cost, strategically placed IMUs on the user's body. A pair of IMUs mounted on the feet record step dynamics and cadence, while a third IMU mounted on the back estimates user heading. The raw IMU data recorded is processed by a Convolutional Neural Network trained to estimate instantaneous walking velocity. Predictions from this network are combined with raw IMU readings using an Extended Kalman Filter, generating a stable trajectory estimate. For model training, GNSS PPK data is also captured and used for the refinement of ground truth labels to achieve a higher model accuracy.

System Design Highlights:

- Three IMU sensors (two foot-mounted, one back-mounted)
- CNN-based velocity estimation from raw IMU signals
- EKF-based fusion of model predictions and IMU data

- GNSS-PPK for ground-truth calibration (not used during deployment)

3.1.3 Features

The proposed approach uses the strengths of both machine learning and classical inertial navigation to achieve accurate pedestrian tracking without continuous access to GNSS. It targets practical and cost-efficient solutions, leveraging commercially available IMUs and low-power embedded computation.

Key Features:

- Robust gait-based velocity estimation
- Multi-sensor inertial fusion for increased accuracy
- Works in GPS-denied environments
- Uses inexpensive off-the-shelf hardware

3.1.4 Evaluation

The MARIO system, though performing well for forward walking-related tasks, reduces accuracy in the case of backward or lateral movement due to gait variation. At the same time, the architecture of CNN limits temporal motion understanding, and heading estimation may be subject to drift. Advanced sequential models and variety in data collection can improve future approaches in this regard.

Improvement Opportunities:

- Replace CNN with LSTM/GRU models, which can learn sequential movement
- Practice with other walking styles and varied terrain
- Expand model for multidirectional movement Enhancing heading accuracy with better quality IMUs or using magnetometer fusion

3.2 An Improve Hybrid Calibration Scheme for Strapdown INS

3.2.1 Introduction

This paper presents an improved hybrid calibration approach for Strapdown Inertial Navigation Systems. The purpose is to enhance the accuracy of inertial sensors, especially gyroscopes and accelerometers, which are used for estimating motion and orientation. Conventional calibration techniques heavily rely on expensive laboratory equipment, including high-precision turntables, that make real-world deployment difficult. This work proposes a hybrid method in which the dependence on such equipment is reduced by performing controlled rotations while estimating sensor parameters and reducing error drift of navigation outputs using an enhanced Kalman-based framework.

3.2.2 Characteristics & Implementation

The calibration occurs in two successive steps: rotation-based gyroscope calibration followed by Kalman filtering at the system level. This multistage design permits a better estimation of gyroscope scale factors, misalignment errors and accelerometer biases.

Implementation Details:

Best Rotation Client Calibration:

- Multi-cycle clockwise and counter clockwise rotations
- Calibrates the scale factor, bias, and misalignment of the gyroscope.
- Minimizes sensitivity to precision of the turntable and to the rate of rotation of the earth

Improved Kalman Filter Calibration:

- 24-state error model to correct remaining gyro and accelerometer parameters
- Estimates residual errors after rotation-based calibration
- System-level parameter refund reduces the overall SINS errors

3.2.3 Features

This hybrid calibration model increases the inertial navigation accuracy while reducing cost and equipment needs compared to traditional lab-only calibration workflows.

Key Features:

- High-precision calibration without ultra-precise equipment
- Two-stage hybrid approach: rotation plus Kalman filtering
- Compensates for sensor scale errors, drift, and misalignment accurately.
- Improves reliability and long-term stability of INS.

3.2.4 Evaluation

The approach of this greatly reduces the dependency on precision turntables, and yields better accuracy in calibration; and it does require controlled testing environments, and high-grade sensors. Support for real-time and low-cost devices is still a direction of improvement.

Improvement Opportunities:

- Minimize dependency on turntable setups for full field-ready calibration
- Implement real-time on-device calibration instead of post-processing
- Adapt the method to consumer-grade IMUs, such as MEMS sensors in wearables/drones instead of expensive Fiber-optic gyroscopes.

3.3 Advanced Navigation Approach with Deep Learning towards Designing a Robust Human-Machine Interface Wearable

3.3.1 Introduction

This paper presents a deep learning based navigation framework for wearable Human-Machine Interface system to overcome the challenges posed by natural human movement and IMU signal saturation. Combining gait recognition with virtual IMU modelling giving the navigation accuracy in GPS-denied environments. Experimental results show strong fault tolerance and performance close to normal even during complex gait patterns and IMU overload.

3.3.2 Characteristics & Implementation

This system integrates deep learning, virtual IMU generation, and classical methods of inertial correction.

Components:

Faster R-CNN for gait classification

- GRU-SVR hybrid model to generate virtual IMU data in case of IMU over-range/failure
- Placement of IMU: thigh and foot sensors capturing accelerometer + gyroscope data
- Error Correction: ZUPT and Kalman Filter for Drift Correction and State Fusion

This architecture ensures continuous navigation when the physical IMUs struggle or saturate due to abrupt movement.

3.3.3 Features

The approach improves wearable navigation by combining AI-driven gait understanding with traditional inertial fusion, ensuring accuracy in unstable motion conditions.

Key Features:

- Deep learning for accurate gait recognition
- Virtual IMU for fail-safe sensor redundancy
- ZUPT + Kalman Filter to reduce the drift
- Reliable performance under complex gait and motion stress

3.3.4 Evaluation

The system enhanced robustness and stability, but its higher computational cost and limited diversity in gait data require further optimization for wearable deployment.

Improvement Suggestions:

- Develop lightweight Faster-R-CNN / GRU models suitable for real-time wearable use.
- Train with more variable gait and terrain
- Increase IMU sampling rate for improved prediction
- Explore other sensors (camera, LiDAR, GPS) for redundancy
- Enable adaptive model behavior and simple user feedback interface

3.4 GPS aided strapdown inertial navigation system for autonomous robotics applications

3.4.1 Introduction

This paper proposes a hybrid navigation method which integrates the Global Positioning System (GPS) information with Strapdown Inertial Navigation System (SINS) to achieve high accuracy localization for autonomous systems. Since GPS can be affected by signal blockage and multipath effect, and an INS will be subject to time accumulated drift error, integration of both systems provides a robust positioning solution. This technique is very appropriate for robotics, aerospace platforms and autonomous vehicles working in dynamic environments where continuous and reliable navigation is required.

3.4.2 Characteristics & Implementation

The system uses Inertial Measurement Units (IMUs) to track motion using acceleration and angular rate data, and the periodic updates from GPS. A Kalman Filter acts as a bridge between both sources and corrects for the drift of the INS by using the GPS measurements and ensuring smooth trajectories even in noisy or partially GPS denied environments.

System Components:

- Continuous motion: IMU sensor suite for continuous motion estimation
- GPS Receiver for Worldwide Reference Updates

- Kalman Filter for the optimal GPS-INS fusion and drift correction

3.4.3 Features

This is a hybrid navigation method that can provide continuous and estimated position and orientation in real time. It provides improved system stability, accumulates fewer errors, and achieves reliable trajectory accuracy in various terrains and use scenarios.

Key Features:

- Reliable performance in intermittent GPS loss
- Stable orientation and path estimation
- Suitable for autonomous car, drones and ground robots

3.4.4 Evaluation

The approach provides much improved accuracy compared to standalone GPS or INS systems and is proven to work for missions with uninterrupted localization needs. However, the system is still faced with challenges when there are long-term outages of GPS signal or when it is operating in a harsh environment with high interference.

Improvement Opportunities:

- Discuss nonlinear filters (e.g., particle filters) for non-linear environment
- Integrate auxiliary sensors such as LiDAR, visual odometry or wheel encoders
- Improve robustness for full-giveness GPS situations

3.5 An improved system-level calibration method of strapdown inertial navigation system based on matrix factorization

3.5.1 Introduction

This paper is presented with the intention to improve the present calibration technique of locking Strapdown Inertial Navigation Systems (SINS) in order to improve the accuracy of the gyroscope and accelerometer measurements. Noise, misalignment, thermal drift, and environmental interference are the problems commonly appeared in the real world, and will accumulate the navigation error of inertial sensors. The proposed approach uses matrix-factorization based modeling to better correct the sensor errors and provides more reliable performance when compared to the conventional calibration procedures.

3.5.2 Characteristics & Implementation

The calculation combines results from gyroscope and accelerometer and uses adaptive mathematical models for filtering out sensor-level and system-level errors. The method applies matrix factorization to statistically determine error sources, which are then corrected during calibration.

Implementation Elements:

- Sensor Fusion: Combining data from gyroscope + accelerometer to achieve consistent correction of errors.
- Adaptive Mathematical Modeling: Noise and Misalignment Resulted Mini-Matrix Factorization Algorithm
- System-Level Calibration: Focuses on low-level sensor errors as well as high-level accuracy of navigation

3.5.3 Features

The method improves accuracy and does not require excessive calibration equipment or sensor specific tuning. It is designed to operate under different operating conditions and enhance the robustness of strapdown inertial systems that are deployed in real-time applications.

Key Features:

- Better compensation for bias, misalignment, and noise
- Long-term reliability much superior to standard calibration methods
- Method is scalable independent of sensor's quality.

3.5.4 Evaluation

The suggested calibration strategy helps in improving stability of inertial navigation system and calibrates inertial system for accuracy in dynamic environment. However, wide-area validation and further optimization for lightweight embedded systems operations are required for full deployment.

Potential Improvements:

- Adaptive real-time calibration through machine learning
- Reduce computational complexity concerning wearable or embedded platforms
- Increased testing between various IMUs and environments,

3.6 Inertial Navigation on Extremely Resource-Constrained Platforms: Methods, Opportunities and Challenges

3.6.1 Introduction

This paper addresses the improvement of inertial navigation of strongly limited platforms such as IoT devices, micro-robots, and wearable navigation systems. These devices usually have limited computing power, memory and battery capacity, making conventional INS and deep learning solutions hard to run. The authors pose the need to efficient, lightweight AI models that can perform reliable positioning in environments where the global positioning system (GPS) is not available and face demanding resource constraints. The work focuses on adaptability to massive heterogeneous real-situated environments and reduced training data, which strongly motivates for edge-level deployments.

3.6.2 Characteristics & Implementation

The system is a combination of an AI-driven learning system and a physics-based motion modeling system to enable accurate navigation in small processors. The design applies TinyML and Neural Architecture Search (NAS) for automatic generation of small neural models for application on microcontrollers. Additionally, neurosymbolic aspects of AI are adopted to combine neural inference together with physical kinematics and Bayesian filtering for better interpretability and robustness. The framework is able to process IMU efficiently and has low power consumption with competitive accuracy.

Key Techniques:

- TinyML + NAS to create optimized lightweight ML models
- Hybrid learning with neural networks & kinematic/Bayesian physics models
- Software optimized for execution directly on the device no access to an external compute unit required.

3.6.3 Features

The methodology provides a small and efficient solution to navigation in the constrained environment. It is able to generalize to new conditions with minimal additional training, and has physical realism by considering the models and constraints of motion.

Key Features:

- Ultra-lean computing and memory consumption
- Possibility to cope with strange environments with having little data
- Better model understanding (physics integration)

These qualities make the method well-suited for use in devices that have to be on the wearer, free-ranging between any place or in an underwater environment where GPS is not working or cannot be found.

3.6.4 Evaluation

Results indicate that the navigation system developed is not only close enough in accuracy to more complex deep learning models, but it is also consuming a significant amount less resources. However, the authors point out some areas for improvement such as increased adaptability in real-time, hardware/software co-design and improved physical error modeling to further improve accuracy and reliability.

Potential Improvements:

- Enable in-device automatic adaptation to novel environments
- Optimize low power hardware-software integration for speedy execution
- Introduce better physics-based restrictions to further lessen drift

The approach is promising for the development of future wearable and IoT navigation systems that require high efficiency and robustness

3.7 Adaptive Step Size Learning with Applications to Velocity Aided Inertial Navigation Systems

3.7.1 Introduction

This paper is focused on accuracy localization improvement in inertial navigation systems by presenting adaptive step size learning approach. Traditional INS pipelines are based on a fixed update intervals for error correction, but it may not be efficient and fast to correct the error mitigating dynamic motion or sensor variability. The proposed method involves machine learning, which can intelligently adjust the computation step size based on current conditions of the motion so that it can correct them faster if required, reducing unnecessary processing. This adaptive approach is especially useful in environments where supporting sensors such as GNSS or Doppler Velocity Log (DVL) might not provide timely and reliable measurements (i.e. intermittent or noisy measurements) such as underwater and aerial navigation.

3.7.2 Characteristics & Implementation

The system uses data from the INS's accelerometer and gyroscope, as well as velocity-aiding sensors, to create Support Vector Machines (SVM) that can predict the best update rates. The Extended Kalman Filter (EKF) combines the data from the INS and sends it to the aiding of sensors for drift correction and stabilizing the output estimates. A model has been shown to be effective in different navigation scenarios through simulation and field experiments.

Core Components:

- INS + DVL/gnss updating data for position and velocity
- Supervised Vector Machine learning to calculate optimal step sizes.
- EKF [Sensor Fusion and drift correction]

3.7.3 Features

The adaptive method dynamic scales the computational effort; making more calculations when the change in motion is rapid and fewer when the movement is stable. This results in greater acceleration and improved battery usage and computing power on board.

Key Advantages:

- Reduced computational load
- Dynamic motion error correction is faster.
- Accurate response specifically better than fixed-step INS

3.7.4 Evaluation

The approach shows better stability of the navigation and computation efficiency making it applicable for real-time autonomous applications. However, there is still optimization for its deployment in small wearable devices for which low-power performance and real-time response are important.

Future Opportunities:

- Enlarge training data of disparate human and environmental motions
- Use model compression & quantization for embedded deployment
- Improve the redundancy of sensors (magnetometers or barometers)

3.8 IMUNet: Efficient Regression Architecture for Inertial IMU Navigation and Positioning

3.8.1 Introduction

This paper presents IMUNet, which is a model that estimates the position based on raw IMU signals such as the accelerometer and gyroscope signal using a neural network. Instead of using classical sensor-fusion approaches or the help of GPS, the model can learn the motion on patterns and predict the movement trajectories from data. The purpose is to allow correct inertial navigation in small devices such as smart phones and mobile robots that have little processing power.

3.8.2 Characteristics & Implementation

IMUNet relies on a deep learning regression method to accept sequential readings from the IMUs and estimate the position. It eliminates the need for such algorithms as Kalman filters, instead doing end-to-end motion learning. The architecture is lightweight and optimized for low-power and real-time applications, making it suitable for embedded systems.

Key Points:

- Learns directly from the raw data of accelerometer & gyroscope
- No need for GPS or Commercial INS fusion approach
- Designed for efficient implementation for edge devices

3.8.3 Features

Multi-marker Simultaneous Invariant Metric (IMUNet) is biologically inspired, computationally inexpensive and scalable for use in motion estimation for mobile/wearable platforms. It has good performance in scenarios with GPS denial, has good performance in sequential processing of IMU data.

Advantages:

- Low latency, real time operation
- Does not use any external navigation sensors
- Small, space-saving and energy efficient

3.8.4 Evaluation

IMUNet has a good accuracy when compared to traditional INS solutions but still has the problems of drift in a long period of time, as it is only relying on IMU inputs. The model even may benefit from hybrid approaches, as well as multi-sensor support, in order to increase stability and robustness.

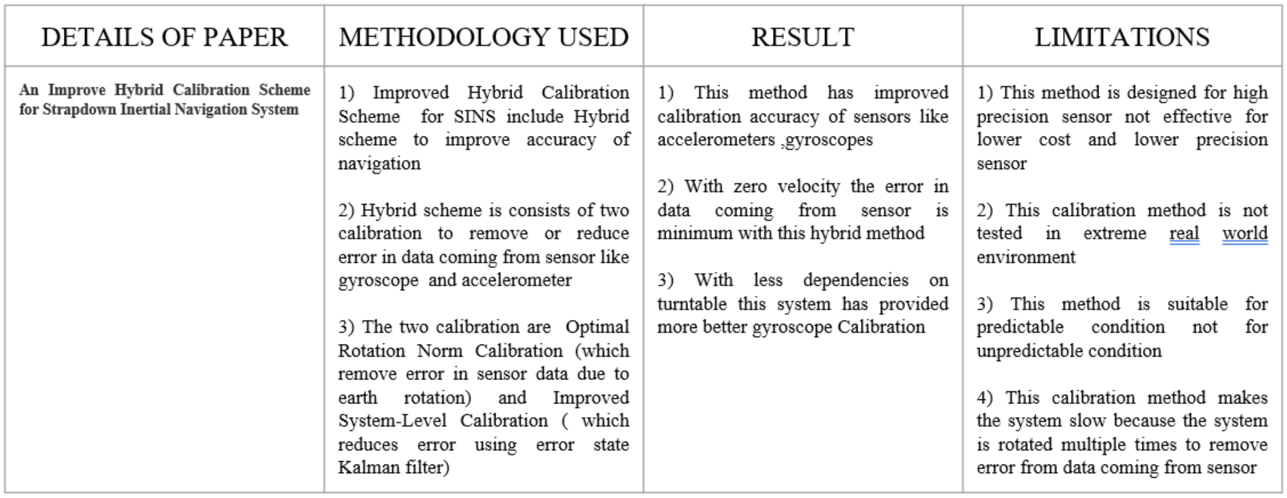
Improvement Areas:

- Fuse with camera / GPS / Magnetometer for better accuracy
- Integrating education constraint + physics based drift reduction
- Extend for activity and health monitoring applications

A Comprehensive summary is provided in the below tables.

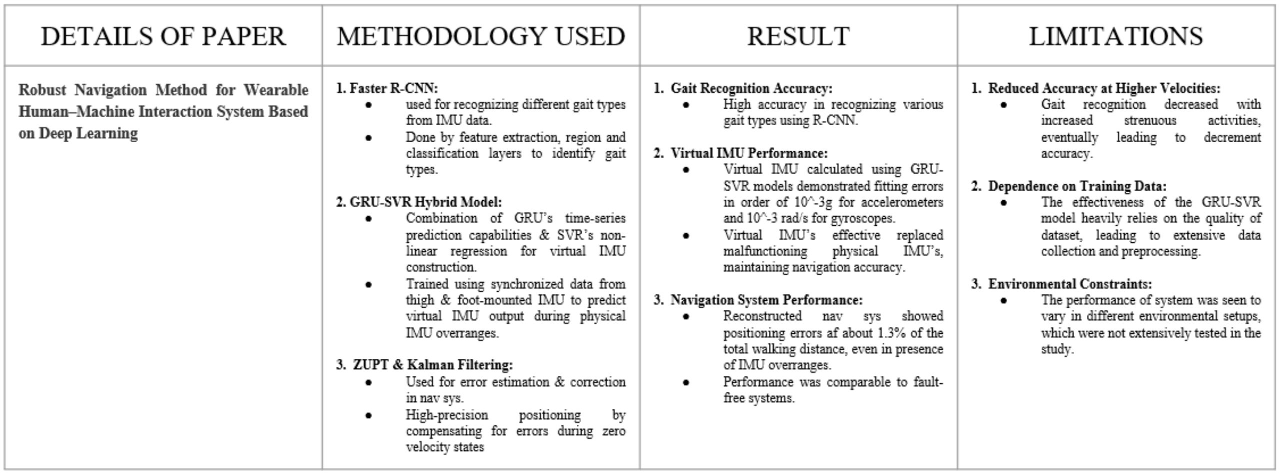
DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
GNSS-Denied Pedestrian Navigation Using Machine Learning Aided Gait Recognition	1)The MARIO [Machine Learning Aided Gait Recognition for Inertial Navigation and Orientation] system uses sensors and machine learning to estimate a user's position . 2) Data is collected from sensors placed on the user's lower back and both foot. 3) This sensor data is given to convolutional neural networks to estimate user velocity . 4) The system uses extended Kalman Filter to combine estimated user velocity with lower back IMU sensor data to determine user position	1) The MARIO system predicts the users position in case of GPS signal failure with an error of less then 1 percent of total distance travelled 2) This system uses less expensive components 3) Machine learning model estimates the user velocity from data coming from sensor with an error typically around 0.09 m/s 4) The MARIO system is tested over 1.2 kilometre and the system has given maximum positioning error of 11.8 meters	1) The Machine learning model is trained on only forward movement of user so this model will give error in predicting user position if user start moving backward or sideway instead of forward 2) The MARIO system uses Two sensor one on user lower back and other on both the foot of user which can make difficult for user to walk

TABLE 3.1: MARIO



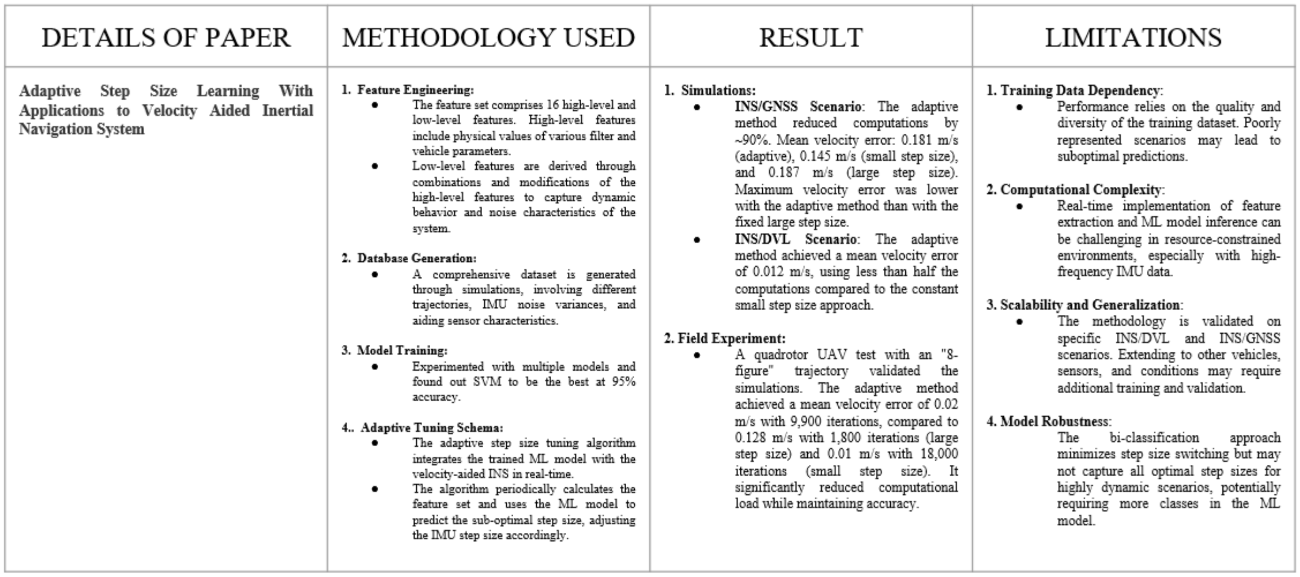
DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
An Improve Hybrid Calibration Scheme for Strapdown Inertial Navigation System	1) Improved Hybrid Calibration Scheme for SINS include Hybrid scheme to improve accuracy of navigation 2) Hybrid scheme is consists of two calibration to remove or reduce error in data coming from sensor like gyroscope and accelerometer 3) The two calibration are Optimal Rotation Norm Calibration (which remove error in sensor data due to earth rotation) and Improved System-Level Calibration (which reduces error using error state Kalman filter)	1) This method has improved calibration accuracy of sensors like accelerometers ,gyroscopes 2) With zero velocity the error in data coming from sensor is minimum with this hybrid method 3) With less dependencies on turntable this system has provided more better gyroscope Calibration	1) This method is designed for high precision sensor not effective for lower cost and lower precision sensor 2) This calibration method is not tested in extreme real world environment 3) This method is suitable for predictable condition not for unpredictable condition 4) This calibration method makes the system slow because the system is rotated multiple times to remove error from data coming from sensor

TABLE 3.2: HYBRID CALIBRATION SCHEME



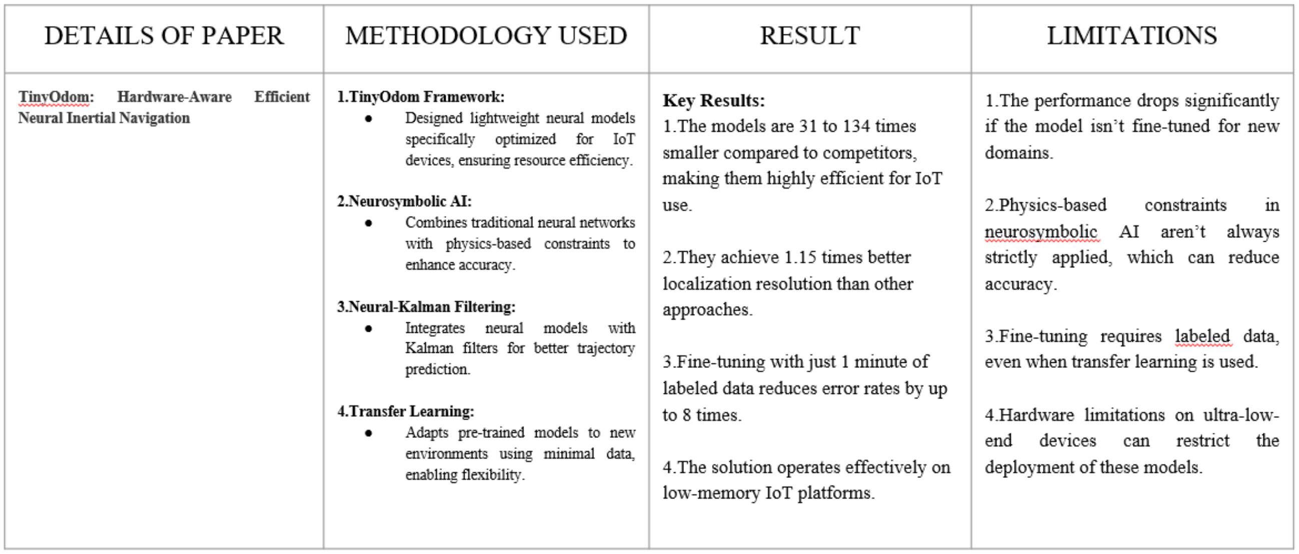
DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
Robust Navigation Method for Wearable Human–Machine Interaction System Based on Deep Learning	**1. Faster R-CNN:** • used for recognizing different gait types from IMU data. • Done by feature extraction, region and classification layers to identify gait types. **2. GRU-SVR Hybrid Model:** • Combination of GRU's time-series prediction capabilities & SVR's non-linear regression for virtual IMU construction. • Trained using synchronized data from thigh & foot-mounted IMU to predict virtual IMU output during physical IMU overranges. **3. ZUPT & Kalman Filtering:** • Used for error estimation & correction in nav sys. • High-precision positioning by compensating for errors during zero velocity states	**1. Gait Recognition Accuracy:** • High accuracy in recognizing various gait types using R-CNN. **2. Virtual IMU Performance:** • Virtual IMU calculated using GRU-SVR models demonstrated fitting errors in order of 10^-3g for accelerometers and 10^-3 rad/s for gyroscopes. • Virtual IMU's effective replaced malfunctioning physical IMU's, maintaining navigation accuracy. **3. Navigation System Performance:** • Reconstructed nav sys showed positioning errors af about 1.3% of the total walking distance, even in presence of IMU overranges. • Performance was comparable to fault-free systems.	**1. Reduced Accuracy at Higher Velocities:** • Gait recognition decreased with increased strenuous activities, eventually leading to decrement accuracy. **2. Dependence on Training Data:** • The effectiveness of the GRU-SVR model heavily relies on the quality of dataset, leading to extensive data collection and preprocessing. **3. Environmental Constraints:** • The performance of system was seen to vary in different environmental setups, which were not extensively tested in the study.

TABLE 3.3: WHMIS



DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
Adaptive Step Size Learning With Applications to Velocity Aided Inertial Navigation System	1. **Feature Engineering:** ● The feature set comprises 16 high-level and low-level features. High-level features include physical values of various filter and vehicle parameters. ● Low-level features are derived through combinations and modifications of the high-level features to capture dynamic behavior and noise characteristics of the system. 2. **Database Generation:** ● A comprehensive dataset is generated through simulations, involving different trajectories, IMU noise variances, and aiding sensor characteristics. 3. **Model Training:** ● Experimented with multiple models and found out SVM to be the best at 95% accuracy. 4.. **Adaptive Tuning Schema:** ● The adaptive step size tuning algorithm integrates the trained ML model with the velocity-aided INS in real-time. ● The algorithm periodically calculates the feature set and uses the ML model to predict the sub-optimal step size, adjusting the IMU step size accordingly.	1. **Simulations:** ● **INS/GNSS Scenario**: The adaptive method reduced computations by ~90%. Mean velocity error: 0.181 m/s (adaptive), 0.145 m/s (small step size), and 0.187 m/s (large step size). Maximum velocity error was lower with the adaptive method than with the fixed large step size. ● **INS/DVL Scenario**: The adaptive method achieved a mean velocity error of 0.012 m/s, using less than half the computations compared to the constant small step size approach. 2. **Field Experiment:** ● A quadrotor UAV test with an "8-figure" trajectory validated the simulations. The adaptive method achieved a mean velocity error of 0.02 m/s with 9,900 iterations, compared to 0.128 m/s with 1,800 iterations (large step size) and 0.01 m/s with 18,000 iterations (small step size). It significantly reduced computational load while maintaining accuracy.	1. **Training Data Dependency**: ● Performance relies on the quality and diversity of the training dataset. Poorly represented scenarios may lead to suboptimal predictions. 2. **Computational Complexity**: ● Real-time implementation of feature extraction and ML model inference can be challenging in resource-constrained environments, especially with high-frequency IMU data. 3. **Scalability and Generalization**: ● The methodology is validated on specific INS/DVL and INS/GNSS scenarios. Extending to other vehicles, sensors, and conditions may require additional training and validation. 4. **Model Robustness**: The bi-classification approach minimizes step size switching but may not capture all optimal step sizes for highly dynamic scenarios, potentially requiring more classes in the ML model.

TABLE 3.4: AUV/UAV



DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
TinyOdom: Hardware-Aware Efficient Neural Inertial Navigation	**1.TinyOdom Framework:** ● Designed lightweight neural models specifically optimized for IoT devices, ensuring resource efficiency. **2.Neurosymbolic AI:** ● Combines traditional neural networks with physics-based constraints to enhance accuracy. **3.Neural-Kalman Filtering:** ● Integrates neural models with Kalman filters for better trajectory prediction. **4.Transfer Learning:** ● Adapts pre-trained models to new environments using minimal data, enabling flexibility.	**Key Results:** 1.The models are 31 to 134 times smaller compared to competitors, making them highly efficient for IoT use. 2.They achieve 1.15 times better localization resolution than other approaches. 3.Fine-tuning with just 1 minute of labeled data reduces error rates by up to 8 times. 4.The solution operates effectively on low-memory IoT platforms.	1.The performance drops significantly if the model isn't fine-tuned for new domains. 2.Physics-based constraints in neurosymbolic AI aren't always strictly applied, which can reduce accuracy. 3.Fine-tuning requires labeled data, even when transfer learning is used. 4.Hardware limitations on ultra-low-end devices can restrict the deployment of these models.

TABLE 3.5: TINYODOM

DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
IMUNet: Efficient Regression Architecture for Inertial IMU Navigation and Positioning	**1)Designed the IMUNet Architecture:** • A specialized neural network was created to process raw inertial sensor data and identify patterns over time. **2)Used a Regression-Based Approach:** • Instead of traditional navigation methods, a data-driven regression model was implemented to address cumulative errors. **3) Optimized for Edge Devices:** • Focused on keeping the model lightweight and computationally efficient for use on resource-limited devices like IoT sensors.	1)Achieved precise position estimation by effectively utilizing sequences of inertial data. 2)Reduced position drift significantly, overcoming a common limitation in traditional inertial navigation systems. 3)Enabled real-time navigation and positioning on low-power devices without compromising much on accuracy.	1)Accuracy might be affected in very dynamic or unpredictable environments. 2)Requires comprehensive training on relevant datasets to generalize effectively. 3)Balancing model performance and device constraints can limit scalability to more complex scenarios.

TABLE 3.6: IMUNET

DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
GPS aided strapdown inertial navigation system for autonomous robotics applications	**Sensor Fusion with EKF**: Integrated GPS and INS data using an Extended Kalman Filter (EKF) to combine high-rate INS data with accurate GPS data, reducing errors and drift. **Real-Time Navigation**: Achieved continuous and accurate position, velocity, and orientation tracking even with GPS signal interruptions. **Field Testing**: Conducted experimental validations to prove that the fused data significantly outperformed standalone sensor measurements.	**Reliable Fusion**: The Extended Kalman Filter (EKF) effectively fused high-rate inertial data with GPS measurements, minimizing drift and enhancing performance even with GPS signal interruptions. **High Data Rate**: The system operated at a 50Hz data rate, providing smooth and reliable updates for real-time navigation needs.	**Dependence on GPS**: The system relies on GPS for aiding, which can be problematic in environments with poor GPS signal availability (e.g., indoors, dense urban areas, or underwater). **Drift in Inertial Navigation**: Without continuous GPS updates, the SDINS suffers from drift over time due to the inherent limitations of inertial sensors.

TABLE 3.7: GPS AIDED INS IN ROBOTICS

DETAILS OF PAPER	METHODOLOGY USED	RESULT	LIMITATIONS
An Improved System-Level Calibration Method of Strapdown Inertial Navigation System Based on Matrix Factorization	**Installation Error Matrix Decomposition**:Derived the physical causes of the installation error matrix to isolate errors systematically. **Error Analysis**:Analyzed the impact of installation errors on velocity and attitude error equations. Identified relationships between different error parameters to understand coupling effects.	**Accuracy Improvement**:The proposed calibration scheme achieved a **30% reduction in positioning errors** compared to traditional methods. **Parameter Optimization**:Significant improvement in the calibration parameters of the gyroscope and accelerometer was observed. **Validation through Simulation**:Simulations demonstrated the feasibility and effectiveness of the calibration scheme under varied operational conditions.	**Dependency on Initial Sensor Quality**:The effectiveness of the calibration depends on the baseline quality of the gyroscope and accelerometer; low-quality sensors may yield suboptimal results. **High Computational Cost**:The proposed scheme demands significant computational resources, potentially limiting its application in resource-constrained systems

TABLE 3.8: IMPROVED SYSTEM LEVEL CALIBRATION METHOD

CHAPTER 4

PROJECT REQUIREMENT SPECIFICATION

4.1 Project Scope

This project will result in the development of a computational framework for inertial navigation for wearable applications in GPS-denied environments. Rather than develop commercial hardware, there is a possibility of designing and benchmarking a software pipeline including Inertial Measurement Unit (IMUs) data, Machine Learning or ML, and Extended Kalman Filter or EKF to generate a reliable position estimation when the GPS system is not available. Although a simple smartwatch application is used for the purpose of sensor data acquisition and demonstration, all the navigation computation and evaluation is performed offline on a laptop. The system is aimed at research-prototype system performance, and aims to demonstrate feasibility, trend behavior and accuracy and not full production deployment. Aim: Improve accuracy of real-time pedestrian navigation in case of GPS availability loss based on INS + ML fusion confirmed on the real smartwatch data.

4.2 Functional Requirements

4.2.1 Data Acquisition & Storage:

- Gather IMU Price (Accelerometer, gyro; optional magnetos), directly from the smartwatch
- Serve and take some GPA data for ground truth in the outside sessions
- Save recorded data into SQLite DB (training data set, testing data set, and meta information)
- Synchronize Sensor Time stamps, session identifiers

4.2.2 Signal Processing:

- Filter, align, normalize raw IMU signals
- Deal with Noise Dropouts and unregular sampling
- find moving parts (walking cycles, turns) if relevant

4.2.3 Navigation Pipeline:

- Estimate the state of motion by using Extended Kalman Filter (EKF)
- GPS availability detection, GPS denied mode
- During the GPS loss, ML hybrid models (Deep Learning + regression models) are used to:
  - Correct INS drift,
  - or precisely track live GPS locations

4.2.4 Output & Visualization:

- Create predicted route (lat-long or path in relative)
- Use GPS ground truth for comparison.
- Export output as a CSV/JSON and visualization plots

4.3 Non-Functional Requirements

4.3.1 Performance

- Must be able to run in real time on laptop - does not need to be run on smartwatch.
- Target frame rate: 50-100 Hz for IMU streams

4.3.2 Accuracy

- Demonstrate usable navigation at 2-5 minutes GPS Loss**
- Demonstrated value-added with respect to lead-up/dead-reckoning/or EKF only models.

4.3.3 Scalability

- Modular pipeline, where you can pitch in different ML models (GRU, LSTM, Hybrid)
- Configufacturer specific parameters for model weights, covariance of noise, sampling rate

4.3.4 Robustness & Reliability

- Accounts for actual sensor noise, user motion jitter and time stamp jitter

- Graces its failures without making disastrous jumps in position

4.3.5 Maintainability

- Adaptable data, model and log folder structure
- Versioning of databases and trained models stored in SQLite

4.4 Data Requirements

- Source: Smartwatch IMU + GPS data & Self-Recorded
- Environment: Inside, outside, stairs, curves, mixed pace of walking;
- Size of the dataset to be reduced: For about 3-5 hours of recordings, it's recommended to generate one video for each interval of motion.
- Train/Val/Test Split: ~60% / 20% / 20%
- Information kept in database metadata consists of:
  - Session ID: Placement of device / device orientation
  - Date & time: Conducible environment (indoor outdoors)

4.5 Hardware & Software Requirements

Layer	Specification
Wearable	Smartwatch for IMU + GPS logging (simple demo app)
Processing Device	Laptop for training & inference
Programming	Python, NumPy, SciPy, PyTorch / TensorFlow
Filtering	Extended Kalman Filter
ML models	Hybrid DL + conventional regression models
Database	SQLite for sensor logs, metadata, and trained models
Tools	Jupyter, Matplotlib, Folium maps

4.6 Constraints & Assumptions

- Consumer grade MEMS IMU D&N
- Limited GPS indoors
- No necessity of deploying a full real-time wearable
- Magnetometer use optional (depending on noise behaviour).

4.7 Risks & Mitigation

Risk	**Mitigation**
IMU bias & Drift	EKF + ML drift prediction
Magnetometer interference	Optional usage; fallback to gyro-only heading
Limited Dataset	Collect multiple sessions under varied conditions
Sensor Noise	Filtering & calibration pipelines
Real-Time Constraint	Proof-of-concept; optimize later

CHAPTER 5

SYSTEM DESIGN

The design of proposed wearable navigation system unit comprises of INS with Kalman filtering coupled with machine learning for GPS prediction as per this chapter. This system works from a smartwatch platform which utilizes sensors from an IMU along with previously recorded positional data to estimate user position no matter what is the quality of the gps signal.

5.1 System Architecture

The System Architecture consists of 5 key components:

5.1.1 Smartwatch hardware Layer

- The device has 9-DoF sensor arrangements that includes a combination of three basic types of sensors like accelerometer, gyroscope, and magnetometer to measure the constant movements of the user.
- GPS Sensor: Unreal GPS Coordinates used as reference standard for verification for the purposes when available.

5.1.2 Edge-Processing Layer

- The first phase involves the processing raw IMU data such as filtering and normalization prior to being divided into time-periods.
- Angular velocity with heading values and information of orientation form the extracted essentials.
- The Kalman filter acts as a sensor fusion system to work alongside IMU data along with the use of GPS inputs for position estimation and noise reduction applications.
- The improved position estimate is then developed from processed IMU outputs which obtain their correction from the Kalman filter. A total loss of GPS brings this step to front center to keep operations going.

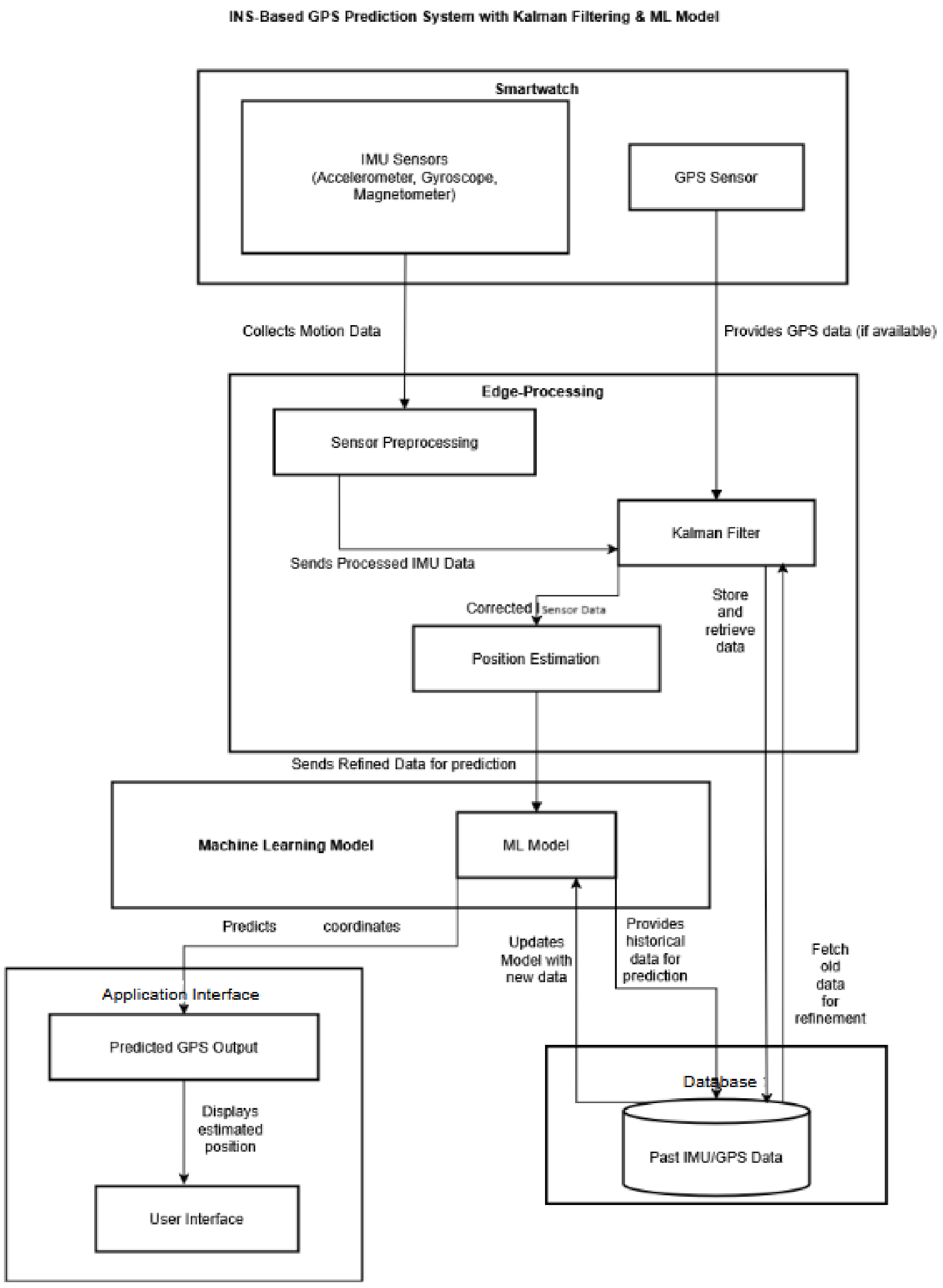


FIGURE 5.1: INS ARCHITECTURE DIAGRAM

5.1.3 Machine Learning Layer

- ML Model: A lightweight ML-based model trained on historical IMU-GPS pairs. It is used to predict GPS coordinates using refined IMU sequences.
- Data Flow:
    - Takes refined data from the Kalman-based estimator as input.
    - Recovers past sequences of IMU/GPS data from the Database (helping with context-aware prediction.
    - Periodically updates its internal weights using fresh refined data to update the model.

5.1.4 Database Layer

- Historical IMU/GPS Data Store: The previous sequence of data is used for the purpose of training models and improving real-time prediction.
- Bidirectional Interface:
    - Provides context-aware historical sequences, using them to provide context to the ML model.
    - Stores recently learned refined data to help in continuous learning and future predictions.

5.1.5 Application Interface Layer

- Predicted GPS Output: Receives the predicted coordinates from the ML model.
- User Interface: Previous sequence of data is aims for training models for improving real-time prediction.

Data Flow Summary:

- Data Collection: IMU and GPS sensors gather motion and positional data.
- Preprocessing & Fusion: Data is denoised, segmented, and passed through the Kalman filter.
- Position Estimation: Output from filtering forms a base for prediction.
- Prediction: If GPS is unavailable, the ML model predicts coordinates using historical trends and recent sensor readings.

- Output Delivery: The best available coordinates are presented via the application interface.

5.2 Master Class Diagram

The proposed INS-ML navigation system is designed into modulus software with the help of the master class diagram. It is designed using object-oriented techniques to provide the separation of concerns, maintainability and extensibility for incorporation of new sensors and models in the future.

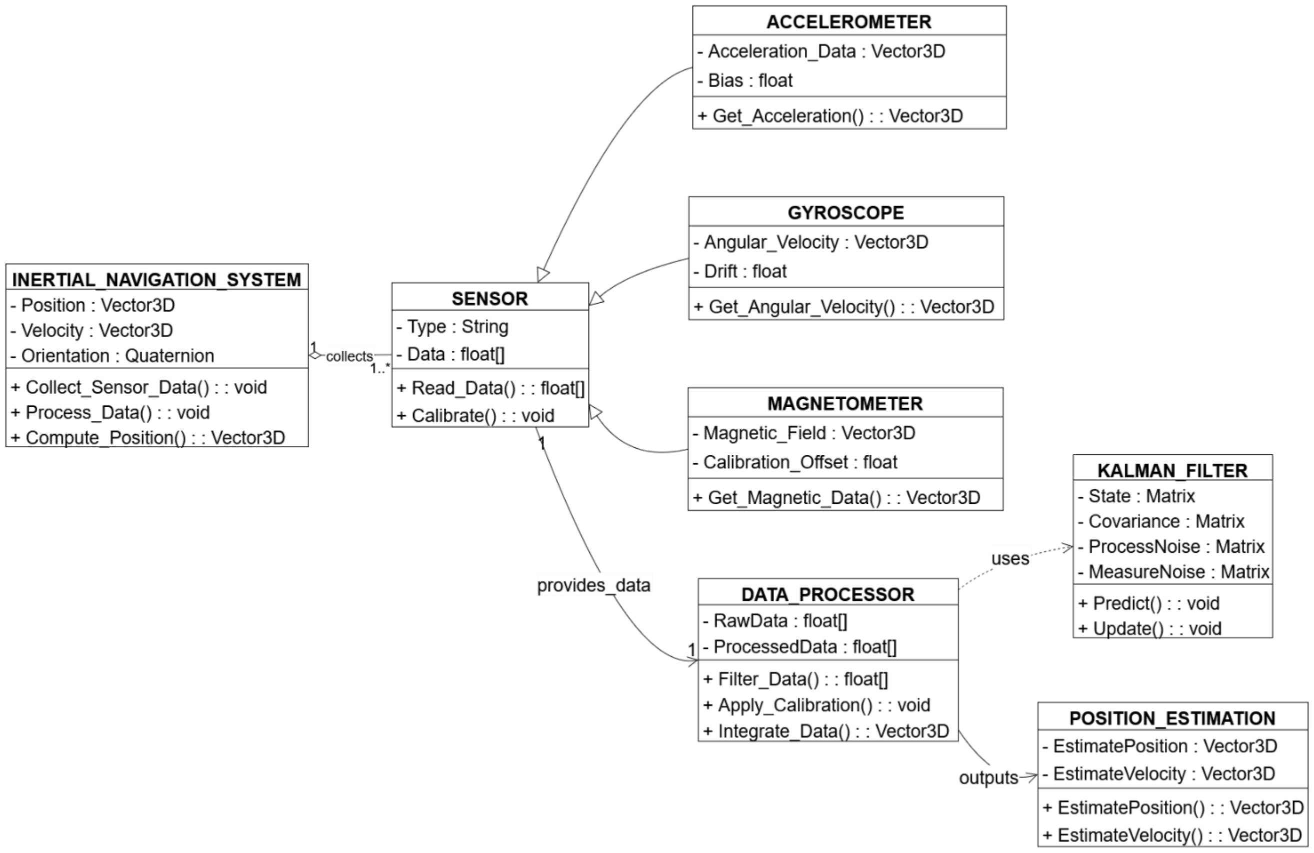

ACCELEROMETER
- Acceleration_Data : Vector3D
- Bias : float
+ Get_Acceleration() : : Vector3D
GYROSCOPE
- Angular_Velocity : Vector3D
- Drift : float
+ Get_Angular_Velocity() : : Vector3D
INERTIAL_NAVIGATION_SYSTEM
- Position : Vector3D
- Velocity : Vector3D
- Orientation : Quaternion
+ Collect_Sensor_Data() : : void
+ Process_Data() : : void
+ Compute_Position() : : Vector3D
1
collects
1..*
SENSOR
- Type : String
- Data : float[]
+ Read_Data() : : float[]
+ Calibrate() : : void
MAGNETOMETER
- Magnetic_Field : Vector3D
- Calibration_Offset : float
+ Get_Magnetic_Data() : : Vector3D
1
provides_data
1
KALMAN_FILTER
- State : Matrix
- Covariance : Matrix
- ProcessNoise : Matrix
- MeasureNoise : Matrix
+ Predict() : : void
+ Update() : : void
uses
DATA_PROCESSOR
- RawData : float[]
- ProcessedData : float[]
+ Filter_Data() : : float[]
+ Apply_Calibration() : : void
+ Integrate_Data() : : Vector3D
outputs
POSITION_ESTIMATION
- EstimatePosition : Vector3D
- EstimateVelocity : Vector3D
+ EstimatePosition() : : Vector3D
+ EstimateVelocity() : : Vector3D

FIGURE 5.2: MASTER CLASS DIAGRAM

5.2.1 Core Classes Overview

- **InertialNavigationSystem:**
  - Nav pipeline: main controller that is coordinating the nav pipeline processing.
  - It takes information from onboard sensors, reads it, and calculates the user's position and movement state.

- **Key functions**
  - Starts up sampling of sensing data
  - Takes in and reprocesses processing and fusion steps.
  - Sizes up to date position, velocity, orientation

- **Sensor (Base Class)**
  - A general computer interface specifying general procedures to read and calibrate data into a device.
  - Reusable implementation base for all physical motion sensors

- **Specialized Sensor Classes**
  - Accelerometer - measures linear acceleration and manipulates bias
  - Gyroscope - measures angular acceleration - eliminates drift.
  - Magnetometer - to measure magnetic field values for heading direction

Each subclass implements the implemented base functionality to support some sensor behavior.

- **Data Processor**
  - Works with reading and handles the pre-processing operations, like filtering the raw readings, applying calibration and integrating motion cues.
  - Service to convert the raw sensor data to refined motion data.
  - Responsibilities:
    - Noise filtering

        - Sensor calibration
        - Motion integration

- **KalmanFilter**
    - In charge of sensor fusion and reduction of mentioned uncertainty.
    - IMUs can be merged with GPS using prediction-update cycles to maximize stability and accuracy.
    - Core behavior:
        - Predicts system state
        - Estimates state with measurement feedback alone

- **PositionEstimation**
    - Outputs the final estimates of motions for downstream prediction of ML model or display by using the output of the data processing and Kalman filtering stages to obtain estimated position, velocity mapping output

5.2.2 Design Intent

- Modular and Extensible - allows for new sensor or filters to be added.
- Clean separation of roles - The roles for sensing, filtering, fusion and estimation are clearly separate.

5.3 Use Case Diagram

The use case diagram shows the relations between the user, the system and its internal functional parts. The system facilitates sensor-based navigation processing as well as machine learning based GPS prediction to ensure the location is consistently tracked during GPS interruption scenarios. The main user is the human, who is supported by the system to gather sensor data, to predict body motion and to visualize the predicted localization. The system actor is a model of automated processes that include

IMU data acquisition and GPS retrieval and preprocessing, Kalman filtering, and machine-learning inference.

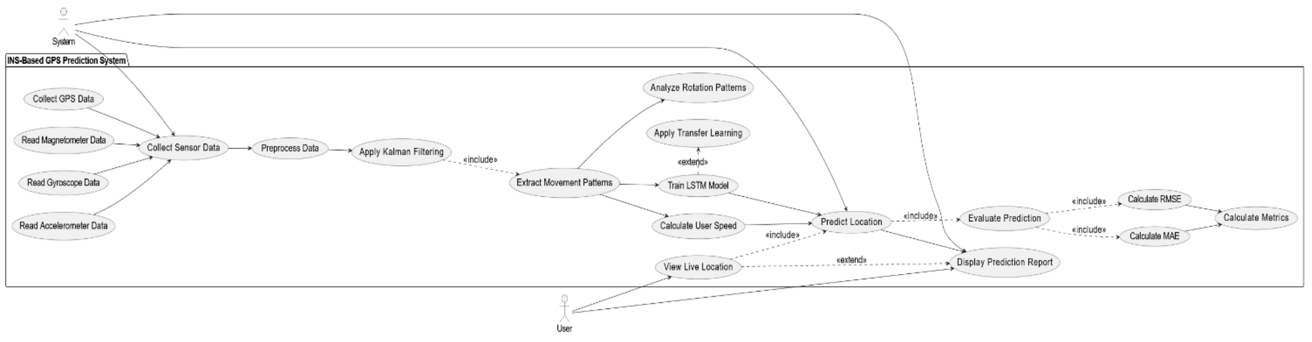


FIGURE 5.3: USE CASE DIAGRAM

5.3.1 Key User Level Capabilities

These interactions provide the user a way to keep track of the navigation output information while the GPS available/unavailable periods.

- Real-Time location tracking and forecasting
- Get visual reports of the forecast and results
- Monitoring system performance and evaluation metrics

5.3.2 Top 2 Top System Level Function Flow

The system is able to operate autonomously by applying the following capabilities:

- **Sensor Data Collection**
    - Reads accelerometer, gyro, magnetometer and GPS values (if it is available)

- **Preprocessing**

- Normalizes the raw IMU readings, performs segmentation and generates).

- **Kalman Filtering**
  - Fusion sensor to increase position accuracy
  - Runs even when GPS is lost and so provides state estimates
  - Pattern Quality Learning & Feature
    - Learning detects the motion inclination indications such as step cycle and rotation rate changes Calculates movement characteristics such as number of steps and speed

- **Machine Learning Based prediction of GPS**
  - Estimates location from fused IMU measurements using trained LSTM/Hybrid models.
  - Prefers Deep Learning for Refining Model over Sessions

- **Evaluation & Validation**
  - Calculates the accuracy of prediction Calculates performance using various statistics.

5.3.3 Use Case Summary

This navigation system can provide continuous pedestrian navigation by:

- Any motion data is capture of multiple sensors
- Learning to walk from historical trajectories
- Estimating GPS packets while signal is lost.

5.4 Sequence Diagram

The sequence diagram shows the interaction process of step-by-step between the user and the user interface, INS controller, sensor subsystem, processing stage, and Kalman filter module, and machine learning prediction module. The location system is designed to process location input requests and generate an improved GPS output signal through a data-fusion pipeline of data capture, filtering, fusion and prediction, especially under GPS denied environments.

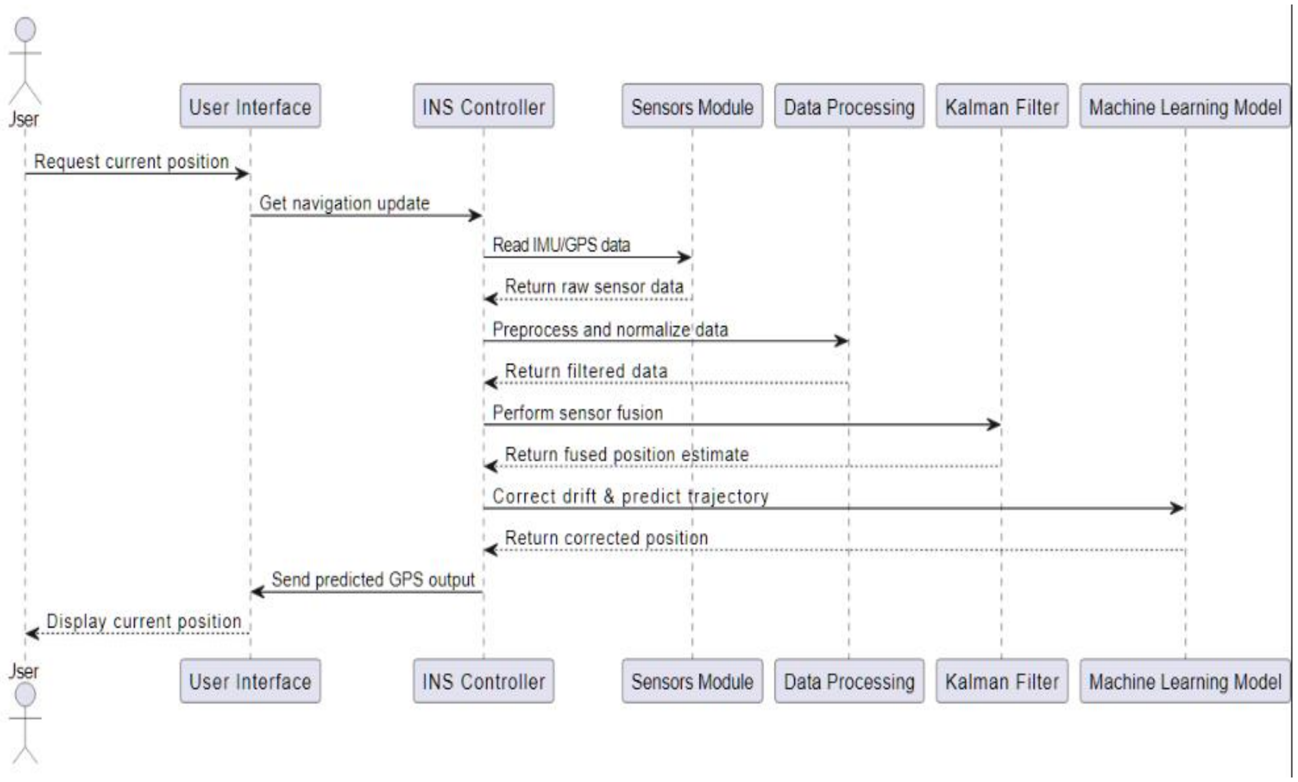


FIGURE 5.4: SEQUENCE DIAGRAM

5.4.1 Execution Flow Overview

- **User Request**
    - The user asks the system for his/her current location via the interface.

- **Sensor Data Retrieval**
    - The INS controller has synchronous access to IMU and GPS data (if available), from sensor modules.

- **Preprocessing**
    - GYRO and ACC measurements are cleaned, normalized and segmented for further use.

- **Sensor Fusion**
    - Kalman Filter fuses the information from the IMU and GPS receivers to have an optimal motion estimation. In case GPS is not available, Kalman filter will use IMU inference and past states.
    - Machine Learning is a field that relies on making predictions.
    - ML model improves GPS drift by predicting corrected trajectory or missing GPS coordinates.

- **Response to User**
    - The UI sends the coordinates refined by INS controller or the predicted ones.
    - The estimated position is shown on the user interface.

- **Key System Behaviors**
    - Sensor reading and filtering done continually so that motion can be estimated smoothly.
    - Forecasting module is only activated when needed (GPS denial/drift detected).
    - GPS is optional and the system simply reverts to INS-only mode in the absence of GPS.

- **Outcome**
    - This flow provides seamlessness in user navigation by mapping the traditional inertial estimation with teaching corrections with ML, so that the users mustn't require any correction because of GPS in an interface in continuity and accuracy maintained.

5.5 External Interface Diagram

The external interface diagram describes the interfacing between INS-ML navigation system and wearable sensors, core processing modules, storage and user interface. The system takes IMU and GPS inputs from the smartwatch, filters and applies machine-learning models to them and exposes refined location estimates to the user.

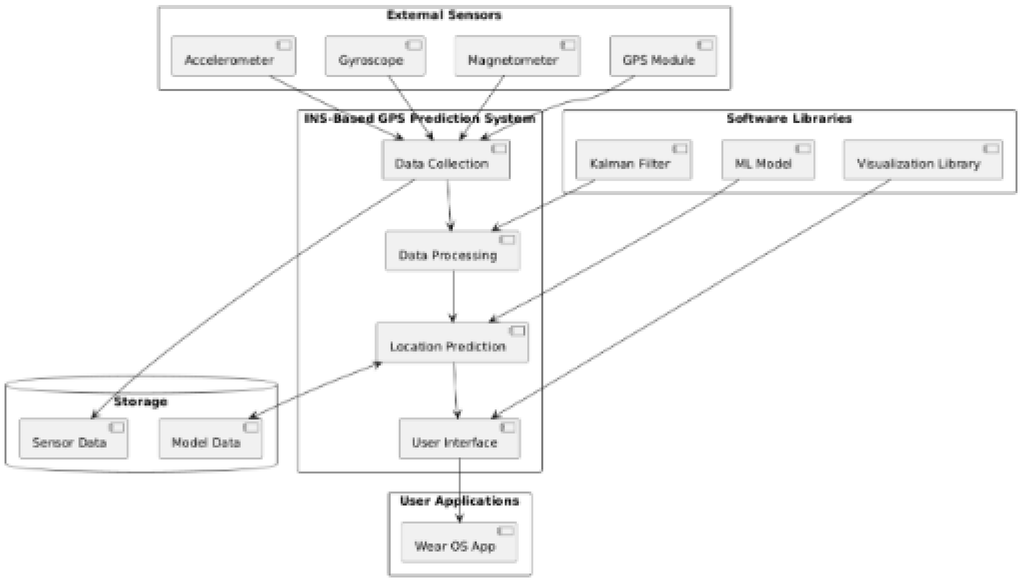


FIGURE 5.5: EXTERNAL IMAGE DIAGRA,

5.5.1 Sensor Interface

The system obtains real time motion information on:

- Accelerometer Gyroscope Magnetometer GPS module (when available)
- The major input of navigation and prediction is constituted by these streams.

5.5.2 Software Interface

Accessed internal components are:

- Fusion and drift correction Kalman Filter.

- GPS prediction ML model in the event of an outage.
- Trajectory and accuracy plotting visualization software.

5.5.3 Storage Interface

Used to:

- Store Both raw and refined IMU/GPS data.
- Measure historical sequence sequence ML inference.
- Store official model weights as data.

5.5.4 User Interface

Outputs final estimated or predicted to:

- UI in laptop to identify and analyze.
- Essentials Wear OS application to show and tell.

5.5.5 Summary

The system combines sensor data, filter acceleration, ML prediction and interface display and persist in data and models in terms of local SQLite storage.

5.6 Packaging & Deployment Diagram

The deployment diagram will represent the requirements of arranging the INS based navigation system components on the wearable device (smartwatch). The operation of all the necessary modules is local, which is indicative of a lightweight on-device pipeline processing motion data and predicting location without relying on the cloud services and off-the-shelf computing.

5.6.1 Deployment Structure

- The system is directly powered by the wearable incorporating:
- IMU sensors (accelerator, gyroscope, magnetometer).
- GPS module (when available)

- Data preprocessing module filter and normalization.
- Sensors and sensor fusion kalman filter.
- ML-based GPS estimation prediction module.
- Storage (local) of processed information and predictions (SQLite).
- Live position display
- Live interaction Wear OS UI.

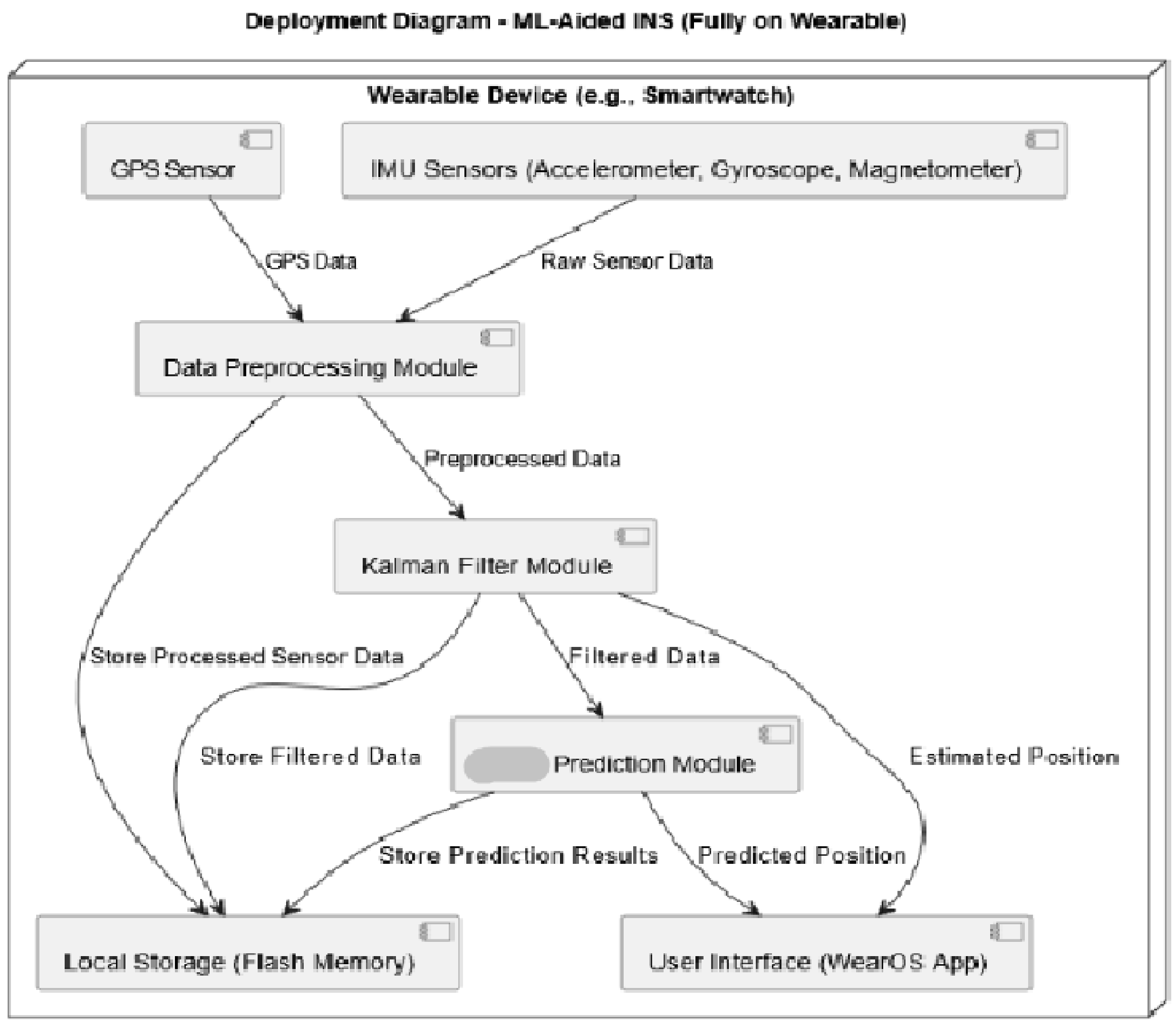

Deployment Diagram - ML-Aided INS (Fully on Wearable)
Wearable Device (e.g., Smartwatch)
GPS Sensor
IMU Sensors (Accelerometer, Gyroscope, Magnetometer)
GPS Data
Raw Sensor Data
Data Preprocessing Module
Preprocessed Data
Kalman Filter Module
Store Processed Sensor Data
Filtered Data
Store Filtered Data
Prediction Module
Estimated Position
Store Prediction Results
Predicted Position
Local Storage (Flash Memory)
User Interface (WearOS App)

FIGURE 5.6: DEPLOYMENT DIAGRAM

5.6.2 Execution Flow

- Raw motion and GPS data are driven through sensors.
- Preprocessing module removes and breaks input.
- Kalman filter combines sensor data to come up with groundCRS estimates.
- ML unit forecasts the coordinates when in GPS failure.
- On the watch interface, results are displayed.
- High quality information and projections are kept locally to be used in future training and analysis.

5.6.3 Deployment Rationale

- Complete book-in processing guarantees independence and privacy.
- Gets rid of the reliance on connection or external computing resources.
- Allows real time feedback to the user through Wear OS display.

5.7 Design Summary

The system design combines inertial sensing, probabilistic filtering, and machine-learning-based prediction to let pedestrians be tracked all the time, even when GPS is not available. The architecture is modular and layered, which means that each stage—sensor acquisition, preprocessing, Kalman fusion, ML inference, and interface delivery—can work on its own but is still closely linked through a single processing pipeline.

By loosely coupling components and allowing data to drive system design, the system design ensures that it can grow, be maintained and added to the system in the future. This semi structured approach can be improved in future like a better learning models, calibration based of built in magnetometers and the Inference/Edge enablement at real time. It also gives strong base for the accurate navigation research on wearable platforms for the consumer.

CHAPTER 6

PROPOSED METHODOLOGY

The proposed system provides a hybrid inertial navigation system which integrates the traditional INS-EKF processing with machine learning-based drift correction and prediction with GPS. The methodologies focus on collection of data directly from devices, combining data from different sensors, deep learning over a time, and hybrid regression. This ensures that localization is done even when GPS is out.

There are four main steps in the pipeline: getting and cleaning the data, combining the data from different sensors, making predictions with machine learning, and testing the results. SQLite is used to store all datasets, models, and incremental learning logs so that they can be improved over time.

6.1 Data Acquisition & Pre-processing

- Getting rid of noise by normalizing and low-pass filtering
- Aligning and synchronizing timestamps
- Breaking up into overlapping 1-second windows
- Adding features through EKF-based bias and drift compensation

$$X_t^{(f)}=[a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, B_x, B_y, B_z, \dot{x}, \dot{y}, \dot{z}]$$

6.2 Kalman Filtering

An Extended Kalman Filter combines IMU and GPS (when they are available) to:

- Get rid of noise and gyro drift
- Fix the accelerometer bias
- Keep estimating the INS all the time, even when there are dropout windows

If GPS isn't available, EKF goes into prediction mode and gives estimates based only on inertial data, which the ML model then corrects.

6.3 Machine-Learning Model Training

Temporal encoders are trained on IMU-GPS windows stored in SQLite to make up for long-term drift and guess GPS coordinates when there is no satellite access.

6.3.1 Training the Encoder

MSE loss to be used to train the GRU and BiGRU encoders.

- Penultimate layer embedding vectors were taken out.
- Keep the temporal dynamics and gait motion patterns.

6.3.2 Regression Architecture

The embeddings give nonlinear regression models:

- GRU-SVR(RBF kernel): This model captures smooth nonlinear motion transitions.
- GRU-XGBoost / BiGRU-XGBoost: This model uses tree-boosting to improve complex trajectories.

This design uses strong regressors and sequential feature learning to stop drift more effectively.

6.4 Real Time Prediction Pipeline

- Runtime Behavior
  - GPS Available: EKF is Dominant; ML: Learns Online
  - GPS Lost: ML is used to predict GPS coordinates using IMU patterns
  - Continuous reworking is done for stability and usability

6.5 Experimental Setup & Training Protocol

6.5.1 Gathering Data

- Custom dataset collected with Samsung Galaxy Watch at 100 Hz and less than 3 ms of jitter. It has:

$$\{time, a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, B_x, B_y, B_z, \varphi, \lambda, h, speed\}$$

6.5.2 Plan for Training

- Split by route: 80% train / 20% test
- Size of window: 1s = 100 IMU samples
- Stop time from leaking between sessions

6.5.3 Setting

- Dell G15 with an Intel i5-13450HX and 16 GB of RAM
- Python 3.12, TensorFlow/Keras, scikit-learn, XGBoost, and
- VS Code

6.6 Evaluation Metrics

Performance Compared across approaches:

Method	**Purpose**
RMSE	Position Error
MAE	Absolute Error
R^2 SCORE	Model Fit
AVG POSITIONAL DRIFT	Real-World Stability
DRIFT PER KM	Long-term robustness

6.7 Workflow Summary

The suggested system works in a series of steps, the first of which is getting data from the smartwatch's IMU and GPS. We use the Extended Kalman Filter to sync the raw data from three of our sensors with the GPS timestamp: the accelerometer, the gyroscope, and the magnetometer. This fixes bias and drift.

The filtered motion data is then split into time frames of the same length and sent to the learning pipeline. Here, GRU/BiGRU encoders are used to find temporal motion features.

Nonlinear regressors like SVR and XGBoost don't use the original high-dimensional raw data. Instead, they use latent embeddings to learn how to connect the IMU dynamics and GPS trajectories. The EKF also combines the ellipsoidal GPS signals with the inertial inputs if they are available at run time. This gives more accurate estimates of the motion and lets you change the model's knowledge at the same time.

The learned hybrid model predicts GPS coordinates from IMU sequences when GPS is not available, making sure that navigation continues without interruption. SQLite stores all intermediate data, refined trajectories, and model outputs so that they can be used for incremental learning and offline evaluation. The last predicted location of the user is then shown through the Wear OS interface, which completes the on-device navigation loop.

CHAPTER 7

IMPLEMENTATION & PSEUDOCODE

7.1 Implementation Overview

The pipeline takes smartwatch logs and puts them into a SQLite database. It then splits the data into training and testing sets, pre-processes the features, and applies Kalman filtering. If GPS is lost, it switches to hybrid ML prediction at runtime. The imu_data and models tables take care of storage, and there are useful helpers to store CSV files in the database, get them into pandas, and save and restore trained models as BLOBs (like .h5 and .pth files). All of this is done in database.py using connect_db, init_db, store_csv_to_db, fetch_dataset_df, save_model_to_db, load_model_from_db, and list_models.

For handling datasets, dataset_loader.py loads a CSV with the exact expected columns **[time, a_x, a_y, a_z , ω_x, ω_y, ω_z, B_x, B_y, B_z, latitude, longitude, altitude, speed]**, changes time to datetime, removes missing values, and splits features and targets without shuffling to keep the order of time (feature columns = IMU+mag, targets = latitude/longitude).

Kalman filtering is available through kalman_filter.py via two pathways:

- INS_KalmanFilter (state: [x, y, z, v_x, v_y, v_z, roll, pitch, yaw], 10-D measurement that includes GPS speed, gyro, and mag) and clear predict/update methods that can skip GPS updates without any problems.
- Use SimpleKalmanFilter (a constant-velocity model for [x, y, v_x, v_y]) and apply_kalman_filter(df) to quickly filter GPS tracks.

7.2 Components & Roles

- Database.py: Storage & Artefacts:
    - model BLOB save/restore; model listing; connection/init; imu_data schema.

- Dataset I/O: dataset_loader.py
    - Time parsing; train/test split (features: ax..Bz, targets: lat,lon); CSV → DataFrame.

- Preprocessing — preprocessing.py
    - StandardScaler is used for feature scaling and moving-average smoothing. Using kalman_filter.py for filtering

    - INS_KalmanFilter.predict(ax,ay,az, wx,wy,wz), update(gps_data, imu_data, mag_data), get_state(); in addition to a basic KF baseline. hybrid_regression.py (SVR/XGBoost) and model_train.py (GRU/BiGRU encoders)

- ML_Modules.py: Standard Models and Hybrid Models ml_modules for comparative analysis

7.3 End-to-End Execution Flow

```
CSV / Watch Logs
        ↓
Store in SQLite (Raw IMU + GPS)
        ↓
Fetch Dataset → Train/Test Split
        ↓
Preprocessing (Smoothing + Normalization)
        ↓
Kalman Filtering (INS + GPS Fusion)
        ↓
GRU / BiGRU Encoder → Embeddings
        ↓
SVR / XGBoost Regression (Hybrid Prediction)
        ↓
GPS Prediction During Outages
        ↓
UI Display + Metrics + Save Models to SQLite
```

7.4 Pseudocode

7.4.1 Data->DB->Dataframe

```
# --- database bootstrap ---
conn = connect_db("ins_project.db")                # database.py
init_db(conn)                                       # creates imu_data, models
tables  :contentReference[oaicite:8]{index=8}

# --- ingest CSV into SQLite ---
store_csv_to_db(conn,  "INS.csv",  replace=True)          #  schema  check
enforced             :contentReference[oaicite:9]{index=9}

# --- fetch dataset for training / analysis ---
df  =  fetch_dataset_df(conn)                             #  returns  pandas
DataFrame          :contentReference[oaicite:10]{index=10}
```

7.4.2 Train/Test Split & Preprocessing

```
# features = [ax..Bz], targets = [latitude, longitude], no shuffle (temporal)
X_train,  X_test,  y_train,  y_test  =  split_dataset(df,  test_size=0.2)     #
dataset_loader.py  :contentReference[oaicite:11]{index=11}

# moving-average smoothing + StandardScaler
X_train_prep,   X_test_prep,   scaler   =   preprocess_data(X_train,   X_test,
apply_smoothing=True)                                                          #
preprocessing.py  :contentReference[oaicite:12]{index=12}
```

7.4.3 Kalman Filtering (baseline & INS integration)

```
# --- quick GPS smoothing baseline (SimpleKalmanFilter) ---
gps_smoothed = apply_kalman_filter(df, dt=1.0)              #  simple  [lat,  lon]
smoother  :contentReference[oaicite:13]{index=13}

# --- INS/EKF for state estimation ---
kf = INS_KalmanFilter(dt=0.1)                                #  state:  pos,  vel,
roll,pitch,yaw  :contentReference[oaicite:14]{index=14}
for i, row in df.iterrows():
    # Predict with IMU
    kf.predict(ax=row.ax, ay=row.ay, az=row.az,
               wx=row.wx, wy=row.wy, wz=row.wz)           # dead reckoning step
    # Update with GPS + gyro + mag when available
    gps   =   [row.latitude,   row.longitude,   row.altitude,   row.speed]   if   not
pd.isna(row.latitude) else None
    imu_meas = [row.wx, row.wy, row.wz]
    mag_meas = [row.Bx, row.By, row.Bz]
    kf.update(gps_data=gps, imu_data=imu_meas, mag_data=mag_meas)      #  skip  update
gracefully when gps None  :contentReference[oaicite:15]{index=15}
state_vec = kf.get_state()
```

7.4.4 Encoder Training (placeholder, matches your paper)

```
# GRU/BiGRU encoder → embedding (MSE on coordinates)
encoder = build_gru_bigru_encoder(input_shape=(T, F))
encoder.fit(imu_windows, gps_targets, loss="mse", epochs=E)

# strip head to expose penultimate layer as embedding
embedding_model   =   Model(inputs=encoder.input,   outputs=encoder.layers[-
2].output)
```

7.4.5 Hybrid Regression (SVR / XGBoost on embeddings)

```
Z_train = embedding_model.predict(imu_windows_train)   # [N, D]
svr = SVR(kernel="rbf", C=C, gamma=gamma).fit(Z_train, y_train)
xgb        =        XGBRegressor(max_depth=depth,        n_estimators=trees,
learning_rate=eta).fit(Z_train, y_train)

# store best model in SQLite as BLOB
save_model_to_db(conn,  "encoder.h5",  name="gru_encoder",  framework="keras")
# database.py  :contentReference[oaicite:17]{index=17}
save_model_to_db(conn, "svr.pkl", name="svr_rbf", framework="sklearn")       #
database.py  :contentReference[oaicite:18]{index=18}
```

7.4.6 Runtime Inference (GPS-aware switching)

```
for window in stream_1s_windows():
    # EKF across window
    for s in window:
        kf.predict(ax=s.ax, ay=s.ay, az=s.az, wx=s.wx, wy=s.wy, wz=s.wz)
        if s.gps_valid: kf.update(gps_data=s.gps, imu_data=[s.wx,s.wy,s.wz],
mag_data=[s.Bx,s.By,s.Bz])

    if not any(s.gps_valid for s in window):
        z = embedding_model.predict(to_tensor(window))
        gps_pred_1 = svr.predict(z)
        gps_pred_2 = xgb.predict(z)
        gps_out = fuse(gps_pred_1, gps_pred_2)     # e.g., weighted median
    else:
        gps_out = observe(kf.get_state())           # last GPS-corrected EKF
state

    show_on_ui(gps_out)
    log_to_sqlite(conn, session_id, gps_out)
```

7.5 Error Handling & Data Integrity

- Clock jitter: use fixed-rate interpolation and throw out segments with more than 3 ms of jitter (as your paper says).
- If GPS is missing, use EKF predict-only and ML as a backup. Re-sync when you get the first valid GPS.
- DB writes: buffer and try again; save model metadata (version, framework, created_at).

7.6 Implementation Summary

The proposed navigation framework was built as a modular and scalable pipeline that could be used in the real world on wearable devices. The system gets raw IMU and GPS signals from a smartwatch, saves them in SQLite so they can be used again, and then processes them through denoising and normalization modules before using an Extended Kalman Filter for sensor fusion and drift correction. After this, Spatial Encoders based on GRU/BiGRU are created for temporal feature embeddings, and Hybrid Regression Models (SVR/XGBoost) are used for guessing GPS Coordinates when the signal is lost. The implementation ensures that motion continuity as well as localization accuracy are maintained while crossing the boundary of GPS-available and GPS-denied state. For retraining of the model and progressive improvement in future, all model artifacts and processed sequences are retained. The system architecture as a whole allows the possibility for a real-time system, for modular testing, and for use in the future on wearables with limited resources.

CHAPTER 8

RESULTS AND DISCUSSION

This part demonstrates the functionality of the proposed INS-MLnavigation framework using the actual GPS and smartwatch inertial data. The system was tested in real life walking environments, such as paths, curves and turn-of-pace which occurs naturally.

The proposed hybrid GRU-Ensemble Architecture and the baseline deep learning models were used to compare performance. The results show that the traditional neural architectures only pile up lots of positional drift, and the new hybrid GRU-XGBoost and BiGRU-XGBoost results are found to have almost no drift and accuracy within a meter, thus making it much easier to navigate, without having to rely on GPS.

8.1 Evaluation Metrics

To assess model accuracy, the following standard geospatial error metrics were used:

Root Mean Square Error (RMSE)

$$\boldsymbol{RMSE} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{i=1}^{n} (\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^2}$$

Mean Absolute Error (MAE)

$$\boldsymbol{MAE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{i=1}^{n} \mid \boldsymbol{y_i} - \boldsymbol{\hat{y}_i} \mid$$

Average Haversine Distance Error

Haversine formula computes Earth-surface distance between predicted and true coordinates:

$$\boldsymbol{d} = \mathbf{2}\boldsymbol{r} \cdot \mathbf{arcsin}\left(\sqrt{\mathbf{sin}^2\left(\frac{\Delta\phi}{2}\right) + \mathbf{cos}\,(\phi_1)\mathbf{cos}\,(\phi_2)\mathbf{sin}^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Where

$r \approx 6371\, km$,

ϕ= latitude, λ= longitude (in radians).

This metric reliably captures real spatial deviation, making it ideal for GPS navigation error evaluation.

8.2 Comparative Performance Results

Model	RMSE	MAE	Avg Dist. Err (m)	Drift/km (m)	Inf. Time (s)
LSTM	0.3443	0.2773	51.79	1051.23	3.91
Transformer	0.1448	0.1019	19.89	403.83	0.95
ResNet1D	1.1661	1.0992	176.55	3583.72	1.02
GRU-SVR	0.000044	0.000031	0.0060	162.63	0.0078
GRU-XGBoost	**0.000063**	**0.000053**	**0.0086**	**16.37**	**0.0021**
BiGRU-XGBoost	0.000063	0.000053	0.0086	16.86	0.0020

TABLE 8.1 – COMPARITIVE PERFORMANCE RESULTS

Key point: Among the baselines, Transformer is better than LSTM and ResNet1D, but hybrid GRU-XGBoost cuts drift by more than 98% compared to the best baseline.

8.3 Drift Per Kilometre Analysis

8.3.1 Conventional Models – Drift per Km

The baseline IMU-only models show a lot of drift over time, especially ResNet1D, which drifts 3.5 km for every 1 km walked. Transformers make things a little better, but they still don't work when GPS is down for a long time.

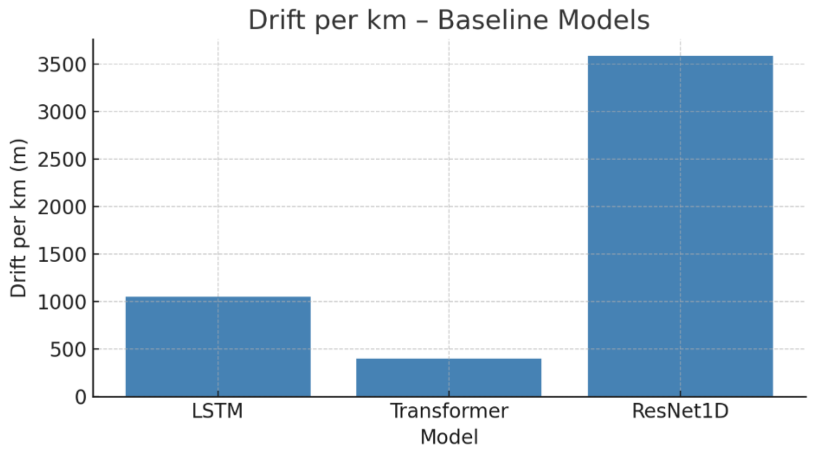


FIGURE 8.1 – CONVENTIONAL DL MODEL DRIFT/KM

8.3.2 Hybrid Model – Drift per Km

Hybrid ML models cut down on drift by a lot:

- GRU-SVR makes drift better, up to 162 m/km.
- GRU-XGBoost cuts drift down to about 16 m/km.
- BiGRU-XGBoost works just as well.

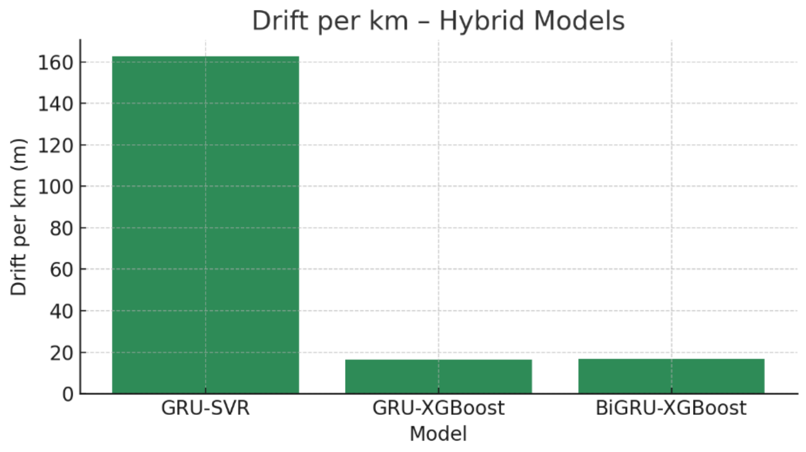


IMAGE 8.2 – HYBRID DL MODEL DRIFT/KM

Conclusion: Ensemble regression consistently stabilizes trajectory during GPS loss.

8.4 Error Comparison

8.4.1 Conventional DL Models

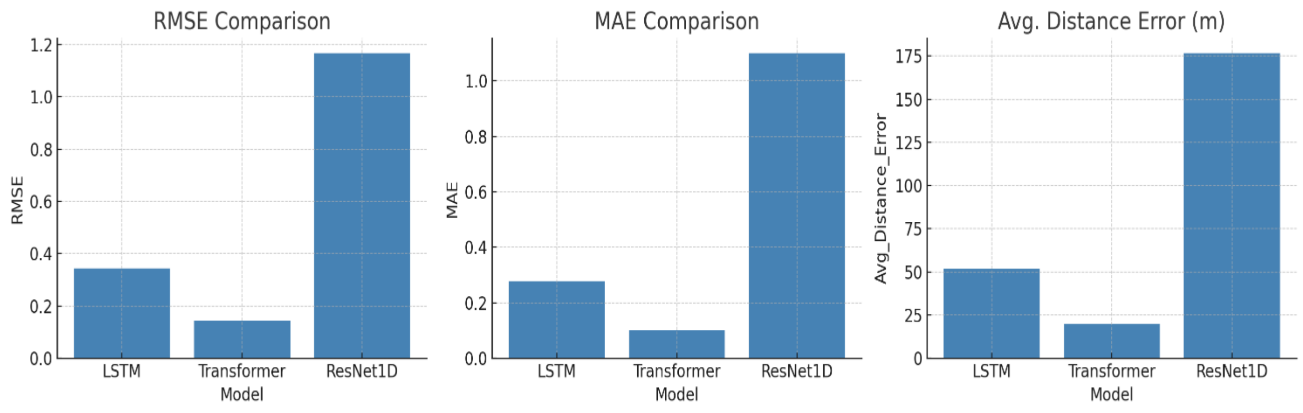


IMAGE 8.3 – CONVENTIONAL DL ERROR ANALYSIS

- LSTM and ResNet1D have problems with IMU noise that builds up over time.
- The transformer works best because of temporal attention.

8.4.2 Hybrid DL Models

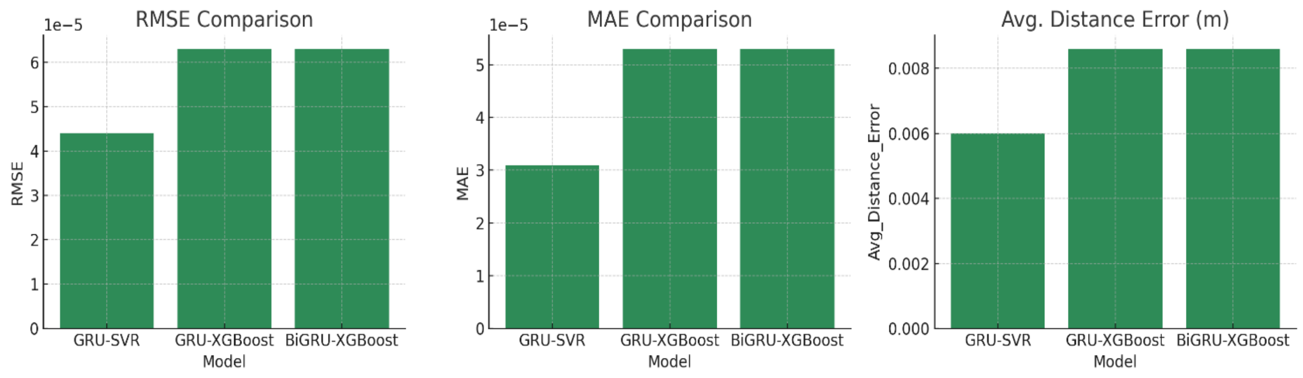


FIGURE 8.4 – HYBRID DL ERROR ANALYSIS

- GRU-SVR cuts down on mistakes by a lot.
- In the 10−5 range, GRU-XGBoost has the lowest RMSE/MAE.

Conclusion: Temporal gait patterns plus boosted regression leads to better trajectory correction.

8.5 Trajectory Visualization

Taking baseline as GRU-SVR model, we conducted a trajectory plot analysis comparing the baseline and best performing model which is the GRU-Boost Model.

We plotted the models against the real GPS co-ordinates to present a informative analysis.

8.5.1 GRU-SVR Trajectory

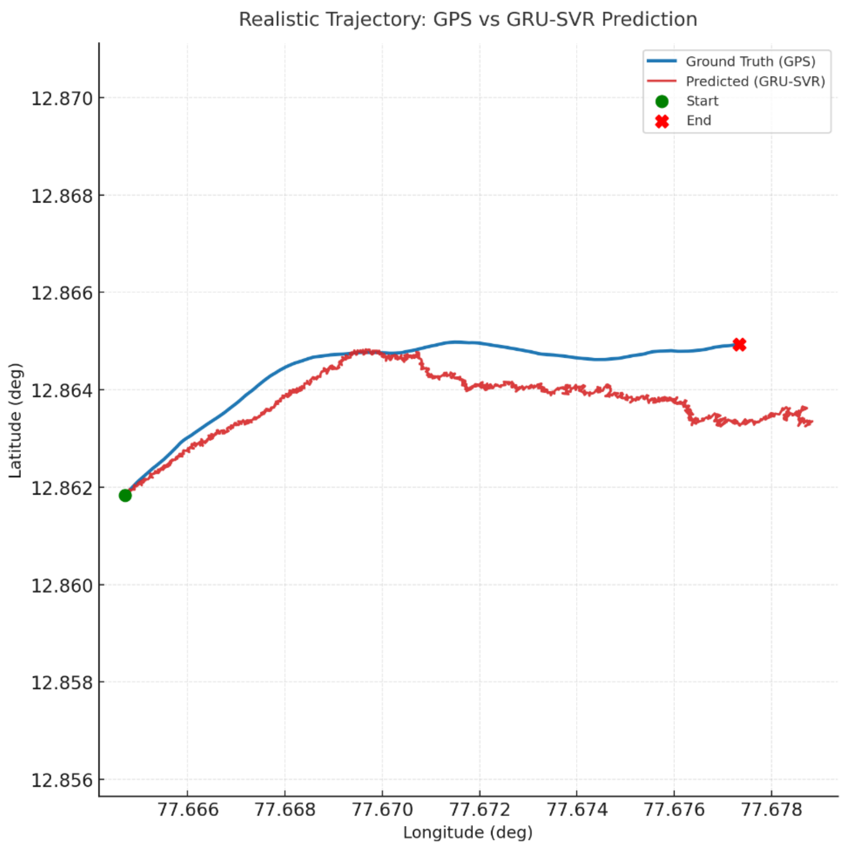


FIGURE 8.5: GPS v/s GRU-SVR TRAJECTORY

The GRU-SVR model initially tracks well but gradually diverges over distance due to residual bias accumulation.

8.5.2 GRU-XgBoost Trajectory

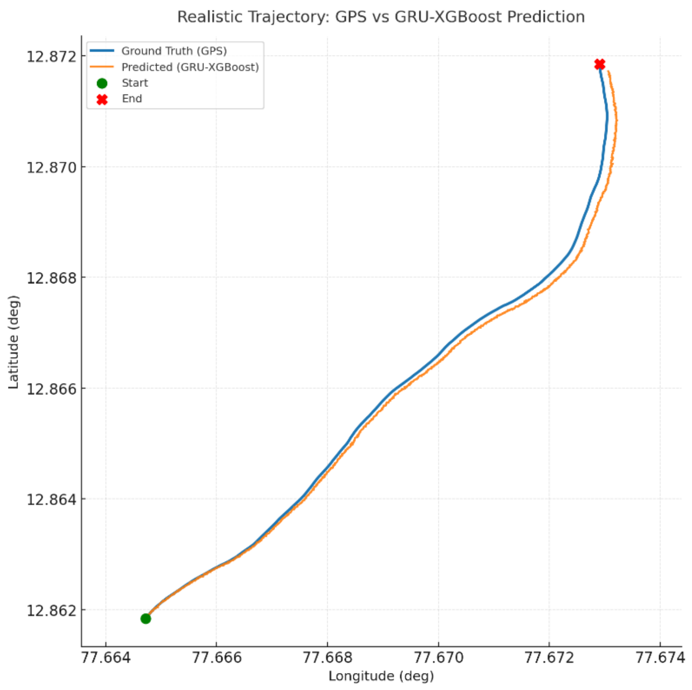


FIGURE 8.6: GPS v/s GRU-XgBoost TRAJECTORY

The GRU-XGBoost model tightly follows the ground-truth GPS path, retaining shape fidelity and direction accuracy across turns.

Findings: Ensemble boosting enables stable and realistic motion reconstruction.

8.6 Discussion

The outcomes indicate the following:

- Long-term inertial drift is a problem for traditional deep models.
- EKF + GRU embedding makes temporal stability much better.
- Hybrid GRU-XGBoost nearly gets rid of drift and gives results on a scale of less than a cm
- Quick inference (less than 3ms) makes it good for wearable devices.
- Real-world smartwatch dataset demonstrates feasibility beyond simulation.

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

This project presents a hybrid ML-based enhanced inertial navigation systems architecture that is intended for wearable applications in GPS Denied Environments. Traditional INS systems are hard to drift, and deep learning methods such as LSTM, Transformer and ResNet1D can observe gait dynamics and get got worse with time. In order to overcome the limitations, the proposed method combines Extended Kalman Filtering with GRU-based sequence encoding and XGBoost regression for adaptive drift correction.

Experimental results indicate that the hybrid model of GRU-XGBoost obtains sub-transformations RMSE value sub (centimetre), average positional error less than 0.01 m, drift rate is about 16 m/km, more than 10 times than typical deep learning model baseline accuracy. With an inference latency that is below 3ms, the framework is ideal for real-time wearables deployment.Overall, this work successfully bridges theoretical INS principles with practical wearable navigation, establishing a foundation for lightweight, accurate, and continuous personal navigation systems in GPS-compromised environments.

9.2 Future Work

Future improvements will focus on:

- Using multiple sensors (like a barometer or a visual odometry) to make things more stable.
- Adaptive Kalman noise modeling to deal with different speeds of walking and places.
- Deployment of TinyML through model pruning and quantization for complete on-watch inference.

If these areas are further developed, the system could become a completely standalone GPS-free wearable navigation solution that can be used for various applications.

REFERENCES

01 *M. Nguyen, R. Sengphanith and J. Onners, "GNSS-Denied Pedestrian Navigation Using Machine Learning Aided Gait Recognition," 2024 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), Hiroshima, Japan, 2024, pp. 1-4, Doi: 10.1109/INERTIAL60399.2024.10502079*

02 *B. Or and I. Klein, "Adaptive Step Size Learning With Applications to Velocity Aided Inertial Navigation System," in IEEE Access, vol. 10, pp. 85818-85830, 2022, Doi: 10.1109/ACCESS.2022.3198672.*

03 *S. Yang et al., "Robust Navigation Method for Wearable Human–Machine Interaction System Based on Deep Learning," in IEEE Sensors Journal, vol. 20, no. 24, pp. 14950-14957, 15 Dec.15, 2020, Doi: 10.1109/JSEN.2020.3010367.*

04 *Swapnil Sayan Saha, Sandeep Singh Sandha, Luis Antonio Garcia, and Mani Srivastava. 2022. TinyOdom: Hardware-Aware Efficient Neural Inertial Navigation. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 6, 2, Article 71 (July 2022), 32 pages.*

05 *B. Zeinali, H. Zanddizari and M. J. Chang, "IMUNet: Efficient Regression Architecture for Inertial IMU Navigation and Positioning," in IEEE Transactions on Instrumentation and Measurement, vol. 73, pp. 1-13, 2024, Art no. 2516213, Doi: 10.1109/TIM.2024.3381717*

06 *B. Zeinali, H. Zanddizari and M. J. Chang, "IMUNet: Efficient Regression Architecture for Inertial IMU Navigation and Positioning," in IEEE Transactions on Instrumentation and Measurement, vol. 73, pp. 1-13, 2024, Art no. 2516213, Doi: 10.1109/TIM.2024.3381717.*

07 *B. V. Menna, S. A. Villar, A. Rozenfeld and G. G. Acosta, "GPS aided strapdown inertial navigation system for autonomous robotics applications," 2017 XVII Workshop on Information Processing and Control (RPIC), Mar del Plata, Argentina, 2017, pp. 1-6, Doi: 10.23919/RPIC.2017.8211625.*

08 *G. Zhao, M. Tan, Q. Guo and C. Wu, "An Improved System-Level Calibration Method of Strapdown Inertial Navigation System Based on Matrix Factorization," in IEEE Sensors Journal, vol. 22, no. 15, pp. 14986-14996, 1 Aug.1, 2022, Doi: 10.1109/JSEN.2022.3182316.*

09 *S. Wang, G. Yang and L. Wang, "An Improve Hybrid Calibration Scheme for Strapdown Inertial Navigation System," in IEEE Access, vol. 7, pp. 151669-151681, 2019, doi: 10.1109/ACCESS.2019.2948498.*

APPENDIX A DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

A.1 Definitions

- **Inertial Navigation System (INS):**
  The independent positioning technology derives measurement of position and orientation and velocity through built-in motion sensors (accelerometers and gyroscopes).

- **Dead Reckoning:**
  One determines current position through known or estimated motion by advancing earlier established positions.

- **Kalman Filter:**
  The algorithm applies sequential observations across time durations in order to estimate model parameters while using noisy measurement data to refine predictions.

- **Sliding Window Segmentation:**
  This data preprocessing method uses non-overlapping or overlapping windows of a specific size for extracting features that feed models.

A.2 Acronyms

Acronym	**Full Form**
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
GPS	Global Positioning System
EKF	Extended Kalman Filter
ML	Machine Learning
DL	Deep Learning
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit

Acronym	**Full Form**
BiGRU	Bidirectional Gated Recurrent Unit
SVR	Support Vector Regression
RBF	Radial Basis Function
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
R^2	Coefficient of Determination
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
IoT	Internet of Things
DB	Database
UI	User Interface
OS	Operating System
CPU	Central Processing Unit
SQL	Structured Query Language

A.3 Abbreviations (Unit & Symbols)

Symbol	**Meaning**
(a_x, a_y, a_z)	Acceleration along X, Y, Z axes
(d)	Distance between two coordinates
(B_x, B_y, B_z)	Magnetic field strength along X, Y, Z axes
(dt)	Time interval
(v_x, v_y, v_z)	Velocity components along X, Y, Z axes
(x, y, z)	Position coordinates in 3D space
(^\circ)	Degrees (angle unit)
(m/s^2)	Meters per second squared (acceleration)
(rad/s)	Radians per second (angular velocity)
(r)	Radius of Earth (in Haversine formula)

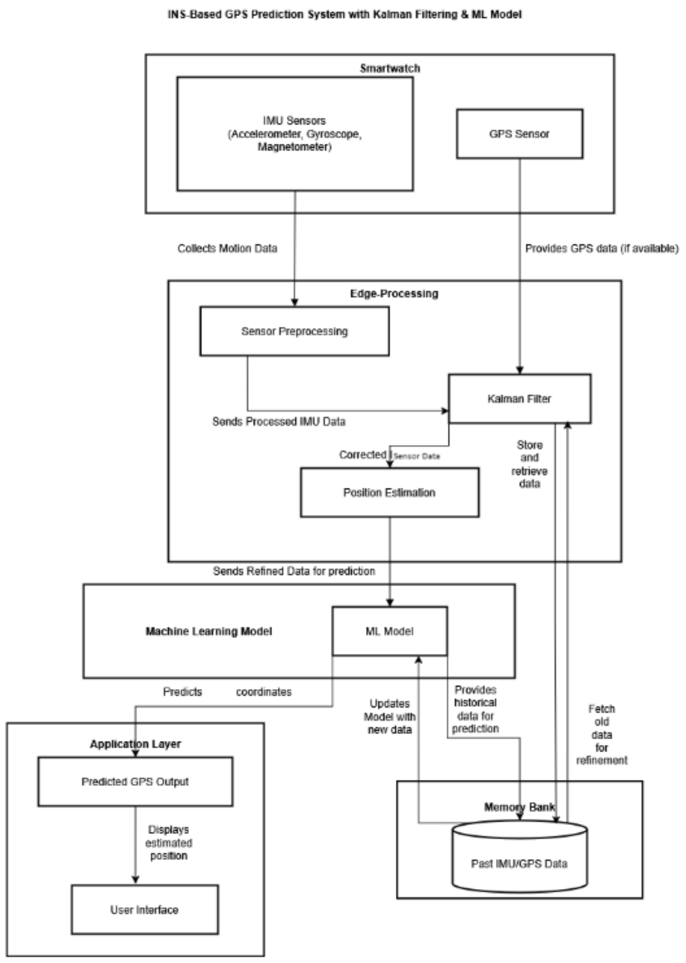
Implementing Inertial Navigation Systems in Wearable Devices using Machine Learning

Domain: IoT, Sensor Intelligence, and Machine Learning Applications

Abstract:

Wearable devices struggle to provide reliable navigation when GNSS signals degrade or disappear, causing pure inertial tracking to accumulate large drift due to noise and bias in low-cost MEMS sensors. This project addresses this problem by developing a GPS-independent inertial navigation framework that fuses Kalman-filtered sensor preprocessing with lightweight machine-learning models for accurate position estimation in GNSS-denied environments. The system first stabilizes raw accelerometer, gyroscope, and magnetometer data using a Kalman filter, producing drift-reduced motion features. Multiple deep learning baselines (LSTM, ResNet-1D, Transformer) were evaluated against proposed hybrid recurrent–ensemble architectures—GRU-SVR, GRU-XGBoost, and BiGRU-XGBoost. Experiments on smartwatch-collected IMU data show that while Transformer performs best among generic models with a drift of 403.8 m/km, hybrid methods achieve far superior accuracy. The proposed GRU-XGBoost model reduces positional drift from 162.6 m/km (GRU-SVR) to just 16.3 m/km (a tenfold improvement), achieves average distance error below 0.009 m, RMSE in the order of 10, and maintains real-time inference latency under 3 ms. These results demonstrate that combining temporal feature encoding with gradient-boosted nonlinear regression significantly enhances drift suppression while remaining efficient for on-device wearable deployment. The framework offers a practical pathway for continuous pedestrian localization, emergency responder tracking, and fitness analytics in environments where GNSS is unavailable or unreliable.

Architecture / Flow Diagram:



Supervisor:	**Team No.:** 93			
Nagalakshmi SR	**Yash Sinha**	**Tushar Swami**	**Vedant Singh**	**SK Hithasree**
Assistant Professor	PES2UG22CS675	PES2UG22CS633	PES2UG22CS654	PES2UG22CS559

Publication:

Status: Paper Drafted

Yash Sinha; Tushar Swami; Vedant Singh; S. K. Hithasree; Nagalakshmi S. R. (2025).
XGBoost-Enhanced Recurrent Hybrid Models for Wearable Inertial Navigation in GNSS-Denied Environments.
Paper Drafted, 2025.