CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Remote Asset Tracking Management Information System

Master's thesis in Computer Science and Engineering

BHARATH RAVICHANDRAN
VARUNPRASAD RAVI

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Remote Asset Tracking Management Information System

BHARATH RAVICHANDRAN

VARUNPRASAD RAVI

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Remote Asset Tracking Management Information System
BHARATH RAVICHANDRAN
VARUNPRASAD RAVI

Supervisor: Ahmed Ali-Eldin Hassan, Computer Science and Engineering
Advisor: Pierre Louis Bullot, Alstom
Examiner: Ahmed Ali-Eldin Hassan, Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2025

Abstract

Modern freight railways operate at a scale and complexity that make continuous visibility of assets indispensable. Trains cover vast distances, are constantly reconfigured in yards, and move through environments where conventional monitoring tools cannot provide precise or continuous information. In such conditions, the central challenge lies not only in knowing where wagons are located, but in reliably determining which wagons are attached to which locomotive at any given moment. This knowledge is critical for ensuring safe braking performance, efficient use of power, proper cargo delivery, and timely response to operational disruptions.Traditional tracking methods focus primarily on presence detection within fixed segments of track. While effective for basic occupancy monitoring, they fail to capture the dynamic and fine-grained information required for modern operations, particularly when trains are frequently rearranged or move in parallel on complex infrastructure. Emerging reliance on digitalisation and autonomous or remotely controlled freight runs further heightens the demand for real-time, accurate composition data, as human confirmation can no longer serve as a fallback.

GPS appears to offer a solution but introduces its own difficulties: noisy positioning, asynchronous reporting, coverage gaps, and missing or delayed updates. Without corrective mechanisms, these issues can obscure the true train composition, leaving operators uncertain whether wagons are properly assigned, detached, or misplaced. This thesis addresses these challenges by developing methods that clean and align incoming data, group assets into coherent trains, and apply predictive logic to bridge gaps when information is incomplete. By doing so, it transforms fragmented GPS signals into a continuous and trustworthy picture of train composition. The outcome is a proof-of-concept system that strengthens safety, enhances logistical reliability, and establishes a digital foundation for the future of efficient, automated freight railway operations.

Keywords: Railways, GPS, asset tracking, wagon–locomotive assignment, train composition, Kalman filter, connected components, missing data recovery, streaming data, logistics management

Acknowledgements

We would like to express our heartfelt gratitude to our supervisor, Ahmed Ali-Eldin Hassan, for his invaluable guidance and unwavering support throughout this project. His expertise and insights have been instrumental in shaping our research and have inspired us to strive for excellence. We are also deeply thankful to our advisor, Pierre Louis Bullot from Alstom, whose encouragement and constructive feedback have significantly enhanced our work. We would also like to recognize the contributions of our colleagues who engaged in insightful discussions and provided valuable feedback during this research journey. Their academic support and collaborative spirit have enriched our thesis experience. Lastly, we extend our gratitude to our families for their understanding and support during our studies, as their encouragement has motivated us throughout this process.

BHARATH RAVICHANDRAN,
VARUNPRASAD RAVI
Gothenburg, 2025-09-16

Contents

List of Figures

List of Tables

1

Introduction

The global railway industry is entering a period of transformation driven by three converging forces: efficiency, safety, and automation. Rail freight remains one of the most sustainable modes of long-distance transportation, with trains capable of carrying thousands of tonnes of cargo across national and international corridors. In Europe alone, freight trains regularly extend over 400 to 750 meters, and in some cases exceed one kilometer in length. Each train can haul dozens of wagons containing mixed cargo types that must reach their destinations within tightly scheduled logistics chains. Unlike passenger trains, which usually maintain fixed formations, freight trains are highly dynamic. Wagons are attached, detached, or rearranged multiple times during a journey depending on cargo flows, terminal operations, and routing changes. This variability makes real-time visibility of train composition a critical requirement for safe and efficient operations.

At present, most freight operators continue to rely on static departure manifests, paper logs, and manual procedures to track wagon–locomotive assignments. Such methods were sufficient in the past when trains operated on less congested networks, turnaround times were slower, and safety margins were larger. However, in modern operations these approaches are increasingly inadequate. Freight flows have grown more complex, supply chains are time-sensitive, and networks are busier. Manual reporting introduces delays and errors that propagate throughout the logistics chain. When wagon compositions change in transit, updates are often slow or missing, leaving operators with outdated information about what a train is actually pulling.

Technological advancements such as GPS and wireless communication have made it possible to track rolling stock continuously, but location alone is not enough. A GPS signal may confirm where a wagon is, but it does not inherently specify which locomotive it belongs to. In yards with parallel tracks spaced just a few meters apart, two trains may appear intermingled on location plots. Noise in GPS signals, commonly fluctuating by 10–20 meters, can misrepresent positions and lead to incorrect assignments. Additionally, GPS reception is not always reliable: tunnels, deep cuttings, urban environments, and rural stretches with poor coverage frequently produce missing or intermittent signals. In such cases, operators are left to infer whether a wagon has been detached, delayed, or simply gone silent.

The situation becomes more urgent when viewed through the lens of safety. Accurate knowledge of train composition directly affects braking distance calculations, load distribution, and locomotive power requirements. A train pulling more wagons

than recorded may lack sufficient traction on gradients or safe braking capability on descents. A wagon that detaches unnoticed during transit poses risks of collisions or line blockages. For emergency responders, real-time digital manifests are critical: hazardous material wagons require specific handling protocols, and any mismatch between actual and reported composition can delay response during accidents.

Operationally, efficiency is compromised when wagon–locomotive assignments are not clear. Logistics planners may misroute wagons, leading to costly delays or missed deliveries. Shippers increasingly expect real-time visibility of cargo; without reliable tracking, railway freight risks falling behind competing transport modes such as road haulage. Maintenance scheduling also suffers, since accurate data on wagon utilization and mileage is difficult to maintain if compositions are uncertain.

Looking ahead, automation and digitalization place even higher demands on railway tracking systems. Trials of driverless or remotely operated freight trains are already underway in several countries. In such systems, there is no onboard crew to confirm wagon attachments visually or update manifests manually. Continuous, automated, and reliable digital tracking becomes the foundation for safe autonomous operations. Without it, autonomous freight runs would effectively be operating blind.

It is important to note that the work in this thesis approaches these challenges from a software simulation perspective only. The system does not use real GPS devices or physical sensors; instead, it simulates realistic train dynamics, generates GPS-like signals, and processes them through a complete software pipeline. This enables controlled testing of difficult scenarios such as tunnels, dropouts, and noisy signals, without the need for costly field hardware. The choice of software simulation makes it possible to focus on algorithm design, scalability, and system robustness before considering deployment in live networks.

This background illustrates why wagon–locomotive assignment, though seemingly a narrow technical issue, is in fact a cornerstone for modern freight railway safety, logistics, and future automation. Current methods fail to deliver the level of accuracy and continuity needed in today's operating environment. Bridging this gap requires a combination of predictive algorithms, deterministic assignment methods, and scalable system design is the central focus of this thesis.

1.1 Problem Statements

The challenges surrounding wagon–locomotive assignment can be grouped into three interconnected problem areas.

1. **The Dynamic Nature of Freight Operations**
   Freight trains are reconfigured continuously as wagons are coupled or uncoupled at intermediate yards. Departure manifests quickly become outdated, and in busy marshalling yards wagons on parallel tracks can be easily misidentified. A system capable of continuously updating wagon–locomotive linkages is therefore required [1], [2].
2. **The Limitations of Existing Tracking Technologies**

Conventional railway tracking methods, such as axle counters, RFID tags, and track circuits [3], have proven reliable for detecting presence but fall short in identifying wagon–locomotive relationships [4]. They confirm that a wagon is on a track section, but not which locomotive it is attached to. In yards with multiple parallel tracks, misidentifications are common. Furthermore, these technologies are largely location-bound, meaning they provide data only at fixed infrastructure points. Between those points, wagons remain "invisible." As trains may travel hundreds of kilometers between detection points, the lack of continuous tracking creates blind spots [3]. For modern rail logistics, such gaps are unacceptable.

3. **The Challenges of Real-World GPS Data**
   While GPS [5] enables continuous tracking, it introduces noise, outages, and asynchrony. Position errors of 10–20 meters are enough to confuse wagons on adjacent tracks, and signals are often lost in tunnels, forests, or urban environments. Locomotive and wagon data also arrive at different times, making reliable real-time assignment difficult [6], [7].

Consequences and Scope of the Thesis The consequences of these problems are wide-ranging. From a safety perspective, miscalculations in train length, weight, and braking capacity increase the risk of accidents. A single wagon left behind unnoticed can obstruct the network and cause collisions. From a logistics perspective, wagons assigned to the wrong locomotive may end up at incorrect destinations, disrupting cargo flows and eroding customer trust. From an operational perspective, manual checks and corrections are costly, slow, and inconsistent with the demands of digitalized and autonomous systems.

1.2 Purpose and Research Questions

The purpose of this thesis is to design, implement, and evaluate a fully software-simulated framework for continuous and accurate wagon–locomotive assignment in real time. The research explores how predictive filtering, deterministic grouping, and distributed data processing can be combined to overcome the limitations of noisy, asynchronous, and missing GPS-like data.

The key research questions are:

1. How can simulated noisy and incomplete GPS signals be processed to reliably infer wagon–locomotive assignments?

2. What deterministic rules are required to ensure robustness in simulated complex rail environments?

3. How can such a framework be scaled in software to model large railway networks?

1.3 What to Expect in This Thesis

This thesis contributes a validated software framework showing how predictive filters and deterministic assignment rules can achieve >90% accuracy in wagon–locomotive assignment under simulated real-world conditions. A simulator generates GPS-like signals of realistic train movements, including controlled errors such as signal noise, dropouts, and asynchronous reporting. These data streams are ingested using Kafka, corrected with predictive algorithms such as the Kalman filter, and processed through deterministic methods including tick synchronization, gating rules, and rigid-body drag modeling.

Results are stored in a PostgreSQL database and visualized with Flask and Leaflet.js. The evaluation demonstrates how these techniques provide accuracy, robustness, and scalability, bridging a critical gap in railway operations.

2

Background and Related Work

This chapter presents an overview of the technologies, methods, and algorithms relevant to modern railway asset tracking systems. It begins with the scale and operational complexity of freight trains, followed by a discussion of traditional infrastructure-based methods and their limitations. It then examines the challenges of GPS-based systems, the operational risks of not knowing exact train composition, and the necessity of predictive methods when data is missing. Finally, it outlines computational approaches, comparing machine learning with deterministic algorithms and explaining why the latter are significant in this thesis.

2.1 The Scale and Complexity of Freight Trains

Modern freight railways operate at a scale that makes real-time monitoring and management exceptionally complex. In Europe, for example, the railway network stretches across more than ten thousand kilometres, with several thousand kilometres of double track and widespread electrification on main lines. Trains running on this infrastructure often consist of dozens of wagons, reaching lengths between 400 and 750 metres, and in some cases exceeding a kilometre. These trains travels in highly variable operating environments like long-distance corridors where they move at high speed, local yards where they undergo shunting and reconfiguration, and industrial sidings where they are loaded and unloaded. Each of these operational contexts presents unique tracking challenges [8],[9].

In busy marshalling yards, wagons are frequently coupled or decoupled, with their assignments changing several times within a single journey. On the mainline, they may travel in parallel with other trains only a few metres apart, where even small location errors can cause ambiguity. Environmental factors complicate the scenario further. GPS signals are unreliable in tunnels, mountainous regions, and urban areas with tall buildings, while in rural stretches, cellular coverage is often too weak to provide reliable communication. For operators, these uncertainties make it difficult to know at any given moment the precise composition of a train or the exact position of its wagons [10].

This uncertainty is more than an inconvenience. Knowing which wagons belong to which locomotive underpins safety calculations such as braking distance, load management, and power requirements. It is also essential for efficient logistics, ensuring that cargo reaches the correct destination without misassignment or delay.

With trials in remote and autonomous operations already taking place, the need for continuous, digital situational awareness has become urgent. Autonomous trains cannot rely on human checks, they demand systems that provide minute-by-minute, reliable consistant data. The scale and complexity of modern freight railways thus set a high bar for tracking technologies, requiring approaches that combine accuracy, resilience, and scalability [11].

2.2 Why Traditional Methods Fall Short

For decades, railway operators have relied on infrastructure-based technologies such as track circuits, axle counters, and RFID tags to monitor trains . Track circuits are simple and effective at detecting whether a section of track is occupied. They work by using the train's wheels and axles to complete an electrical circuit, confirming presence. However, they cannot distinguish between locomotives and wagons or provide detailed consist information. Axle counters improve granularity by counting the number of axles crossing a point, which helps estimate train length, but they too fail to provide real-time positional tracking between sensors and can suffer from synchronization issues during shunting operations [3],[12].

RFID brought a step forward by enabling the unique identification of wagons through tagged transponders. When wagons pass by fixed readers, their IDs are logged automatically. Yet, this system still depends on infrastructure density and placement, meaning updates are only available at discrete checkpoints, such as depots or yard entrances. Continuous, network-wide visibility is not possible, and gaps between readers can span hundreds of kilometres. Furthermore, in yards where multiple tracks run side by side, RFID cannot resolve which locomotive a wagon is physically attached to [12].

The limitations of these traditional methods are particularly evident in modern freight operations. Trains are frequently reconfigured mid-journey, with wagons added or removed multiple times. Conventional systems cannot update compositions dynamically, leading to inconsistencies in databases and train manifests. This creates risks in dispatching, routing, and maintenance scheduling. From an operational perspective, the inability to maintain real-time visibility can result in misplaced wagons, inefficient load distribution, and slower responses to anomalies. From a safety point of view, wrong assumptions about train length or load can affect braking calculations and fuel consumption. Traditional systems, while reliable in their narrow function, fall short of delivering the continuous, integrated situational awareness demanded by today's railways.

2.3 The Challenge of GPS Data in the Real World

The emergence of GPS promised to overcome many limitations of infrastructure-based methods by providing continuous, location-independent tracking. In principle, GPS-equipped locomotives and wagons can be monitored across the entire network without relying on fixed sensors [13],[5]. This allows operators to know where assets

are at all times, even in remote or rural regions. However, practical deployments reveal significant shortcomings.

Firstly, GPS signals are inherently noisy. Positional errors of 10 to 20 metres are common, and in railway contexts, such deviations are large enough to cause misassignments. For example, a wagon positioned on one track may appear closer to a locomotive on a parallel track, leading the system to incorrectly group them together. Secondly, GPS data arrives asynchronously. Different wagons report at slightly different times, meaning locomotives and wagons may not have matching timestamps. Without correction, this temporal mismatch leads to unreliable consist determination [14].

Another major issue is the missing data. GPS devices frequently lose signal in tunnels, under dense foliage, or in built-up urban areas with reflective surfaces causing multipath interference. Hardware-level issues such as device reboots, antenna failures, or power-saving modes add further disruptions. Network connectivity compounds the problem: even if GPS data is collected, it must be transmitted back to the system, and in many rural areas, mobile coverage is poor or intermittent.

These challenges mean that raw GPS data cannot be trusted in isolation. Without corrective filtering, synchronization, and predictive methods, wagons appear to jump across tracks, disappear in tunnels, or drift from their true positions. In a system where safety and efficiency rely on precise consist information, these errors are unacceptable [15]. Addressing them requires robust computational methods that can smooth noisy signals, bridge gaps intelligently, and synchronize updates across vehicles.

2.4 Consequences of Not Knowing Train Composition

The consequences of not having accurate wagon–locomotive assignments are far-reaching. From a safety perspective, the risks are immediate. Freight trains are heavy, sometimes weighing thousands of tonnes, and their braking distances depend directly on total length and mass. If operators underestimate these parameters because of missing or incorrect data, they may not allocate enough stopping distance, especially in poor weather or on steep gradients. Locomotives may also be overloaded without operators realizing, stressing mechanical systems or even leading to breakdowns [16].

The logistical risks are equally serious. Misplaced or misassigned wagons disrupt supply chains, delay cargo deliveries, and cause significant financial losses. For industries relying on just-in-time logistics, such as automotive manufacturing or consumer goods distribution, even a single misrouted wagon can halt production lines or delay exports. Customers lose confidence when tracking data is inconsistent, undermining the competitiveness of rail transport compared to other modes.

There are also risks to network efficiency. A wagon that becomes detached without detection can obstruct tracks until discovered, delaying multiple trains and creating safety hazards. In emergency situations such as fires, derailments, or

chemical spills, responders rely on accurate consist data to know which wagons are involved and what cargo they carry. If the manifest is wrong, valuable response time is wasted, and hazards may go unaddressed [17] , [6].

With the emergence of autonomous freight operations, the stakes rise even further. In such systems, there is no crew onboard to visually check if wagons are correctly attached or if any have been left behind. On long remote stretches, failures may go unnoticed for hours. For this reason, continuous and accurate train composition data is not just useful but essential. Without it, both operational safety and logistics reliability are compromised.

2.5 Why Missing Data Must Be Predicted, Not Ignored

In the real world, missing data is not an exception, it is a routine occurrence. Freight trains travels vast regions where GPS and cellular coverage are poor or entirely absent. Devices may fail temporarily due to reboots, battery-saving modes, or antenna malfunctions. Network congestion can delay or drop transmissions. If a tracking system simply waits for data to reappear, operators are left blind during those intervals, unable to monitor train integrity or detect detachments.

Equally problematic is assuming that the last known position remains valid until new data arrive. A wagon detached in a yard may still appear linked to a train hours later if its GPS unit fails to report. Such errors create cascading risks, from safety miscalculations to misrouted cargo. For this reason, prediction is not optional, but necessary.

Predictive methods, such as Kalman filters [18], fill this gap by estimating future positions based on recent motion history and velocity. Even when updates are missing, the system projects a realistic trajectory, maintaining continuity of the consist. This ensures that wagons remain coherently attached in the digital record and that sudden detachments or anomalies are promptly flagged [19].

The prediction also addresses asynchronous data streams. By aligning vehicle positions with fixed time ticks, the system avoids inconsistencies caused by lagging or delayed messages. In effect, prediction transforms incomplete, noisy inputs into a continuous, coherent data stream. This resilience is particularly important in autonomous or remote operations, where no human is present to intervene when data gaps occur. By intelligently bridging missing intervals, the system provides operators with uninterrupted situational awareness, ensuring safe and efficient operations throughout the network [19],[20].

2.6 Computational Methods in Rail Systems

To overcome the challenges of noisy, missing, and asynchronous data, computational methods play a central role. Among these, the Kalman filter is particularly well-

suited for railway applications. It models train movements under a constant-velocity assumption and corrects estimates whenever new measurements arrive. This recursive approach provides smooth trajectories, bridges short data gaps, and produces reliable velocity estimates. Compared to more complex filters such as the Extended or Unscented Kalman filter, the standard formulation offers a balance of accuracy and computational efficiency, ideal for real-time systems [19],[20].

Clustering methods have also been studied, most notably DBSCAN, which groups vehicles based on spatial density without requiring the number of clusters in advance. DBSCAN [21] is attractive because it can handle variable train lengths and detect outliers such as detached wagons. However, it is highly sensitive to parameter choices and often unstable on curved tracks or in parallel-track situations, where wagons from different trains can be mistakenly grouped together [22].

For this reason, the implementation in this thesis adopts a graph-based approach using connected components. Vehicles are represented as nodes, and edges are drawn between those within a specified distance. This structure reflects the physical coupling of real trains, producing groupings that are both interpretable and reliable. When locomotives are present, the wagons are linked directly to them; when absent, the wagons remain in organized chains rather than being left unassigned [23].

This graph-based method integrates smoothly with other modules such as rigid-body drag modeling, staleness checks, and tick synchronization. Together, these components ensure that the train formations remain stable over time and resistant to GPS noise. Unlike machine learning models, which require training data and may act as black boxes, deterministic methods offer transparency and robustness. They are lightweight enough for deployment on edge devices and scalable enough to handle network-wide operations. In this way, computational methods bridge the gap between noisy GPS signals and the operational need for reliable real-time train composition data [24],[23],[25].

3

System Architecture and Design

The architectural design of the proposed railway asset tracking system reflects the need for real-time data handling, scalability, and modularity. The system integrates multiple components that collectively simulate GPS-based wagon tracking, process location data streams, apply signal recovery techniques, and present dynamic insights through a web-based interface. This chapter offers an extensive summary of the system architecture, component interactions, and the reasoning behind crucial design choices.

3.1 High-Level Architecture Overview

The system incorporates an event-driven modular architecture with the seamless data flow from simulated inputs and progressing to a visualization layer. The system is fundamentally composed of five core parts: a GPS data simulator, a message broker built on Kafka, a web-based Flask backend server, a PostgreSQL relational database, and a web interface for user interaction.

The GPS data simulator generates synthetic coordinates for locomotives and wagons, mimicking real train movements under varying speed, direction, and spacing. It models acceleration, deceleration, and parallel train movements, while also occasionally dropping wagon GPS signals to replicate real-world missing data. These data points are published to Kafka topics, forming a continuous real-time stream.

The Flask backend acts as a consumer that subscribes to the Kafka stream and processes each message. It applies a constant-velocity Kalman filter to smooth noisy GPS signals and to predict missing ones. It then aligns all vehicles to fixed five-second ticks to maintain temporal consistency. On each tick, the system groups vehicles into connected components based on proximity and determines valid assignments of wagons to locomotives. To ensure geometric stability across time, especially during curves or GPS fluctuations, a rigid-body drag mechanism is applied, which translates and rotates wagons according to locomotive displacement and heading. The processed results are stored in PostgreSQL, where they can be queried for both real-time and historical views.

The user interface retrieves this data to display locomotive and wagon positions, validate train compositions, and highlight anomalies such as stale, provisional, or unassigned wagons. The modular separation between simulator, broker, backend,

and visualization ensures high availability and extensibility, allowing individual components such as filtering, clustering, or assignment logic to be refined independently without affecting the rest of the pipeline.

3.2 Technology Stack Justification

The selection of technologies was driven by the project's requirements for real-time data streaming, lightweight server processing, scalable message handling, and relational data persistence.

Flask was selected as the web framework for this project due to its simplicity and minimalism. It offers all the functionality required to create RESTful APIs and web applications without additional complexity. Its lightweight design facilitates rapid prototyping and seamless integration with external systems, which makes it particularly well-suited for academic and experimental implementations. While alternatives such as Django exist, they tend to be more resource-intensive and overly complex for straightforward applications [26].

Apache Kafka serves a vital function as the message broker, facilitating decoupled interaction between the GPS data supplier and the client services. Kafka is ideal for distributed event streaming with high throughput and fault tolerance, that corresponds with the objective of continually managing GPS updates instantaneously. Its design enables effective interaction among components without tight coupling, thus improving scalability and maintainability. Other alternatives such as RabbitMQ or Redis were considered, but they may not match Kafka's efficiency in handling large-scale, high-throughput data streams [27].

PostgreSQL serves as the primary data store due to its robustness, strong support for relational modeling, and compliance with ACID (Atomicity, Consistency, Isolation, Durability) properties. In the context of this tracking system, ensuring reliable and consistent transactions is vital, as data accuracy and integrity are paramount. PostgreSQL also offers advanced features such as table inheritance, triggers, and indexing, along with support for spatial data types through PostGIS, which is essential for querying and managing location data. While MySQL and NoSQL databases were considered, they do not offer the same level of advanced relational and transactional support as PostgreSQL [28],[29].

On the front end, the interface is built using Leaflet.js and Bootstrap, providing a responsive and map-centric view of asset movements. Leaflet offers high-performance map rendering capabilities, while Bootstrap ensures that the user interface remains accessible, adaptable, and consistent across devices [30] ,[31].

3.3 Component Roles and Functional Design

Each component in the system fulfills a specific functional role within the end-to-end data processing pipeline.

- **GPS Simulator:**
  The simulator emulates the behavior of one or more moving locomotives and their associated wagons. It generates periodic location updates (every 5 seconds), assigns unique IDs to each asset, and mimics realistic train dynamics such as acceleration, deceleration, and inter-wagon spacing. It can simulate multiple trains running in parallel and occasionally introduces missing GPS data for wagons to evaluate robustness against signal loss.

- **Kafka Broker:**
  Kafka acts as the communication backbone, allowing the GPS data to be streamed asynchronously. It ensures that the producer (simulator) and consumer (Flask backend) remain decoupled and operate independently. Kafka also supports message replay and partitioning features that are advantageous for debugging and scaling [27].

- **Flask Backend:**
  The Flask backend forms the core of the processing logic. It ingests messages from Kafka, validates them, and stores them in a rolling buffer per vehicle. If the data is noisy or incomplete, the Kalman filter reconstructs missing positions by leveraging recent trajectories. Once filtered, all vehicles are synchronized to common tick timestamps to ensure temporal consistency [26].

  The system then applies a graph-based connected component approach to group vehicles within 20 meters of one another, forming the structural basis for consist detection. If locomotives are present, wagons are assigned to the nearest one, subject to additional gates such as maximum distance (60 m), heading angle, and cross-track offset. If no locomotive is close enough, wagons are provisionally grouped into chains. To maintain realistic consist geometry across ticks, a rigid-body drag mechanism adjusts wagon positions according to locomotive displacement and heading, preserving spacing and alignment even through curves or GPS noise.

- **Database (PostgreSQL):**
  PostgreSQL stores the processed results in normalized tables. The `assignments` table records wagon-level data, including positions, assigned locomotives, distances, validation states, and associated audit timestamps.

  The `locomotive_assignments` table captures locomotive positions and their wagon consist snapshots. These structures support efficient retrieval of both current status and historical trends [28],[29].

- **Frontend Interface:**
  The frontend connects to the Flask API and the PostgreSQL backend to provide users with a live dashboard. It displays the current position of each locomotive and wagon on a map, highlights anomalies such as stale or provisional assignments, and allows users to filter and inspect asset metadata. The UI supports both raw GPS views and processed assignment views for comparison.

3.4 Data Flow and Interaction Design

The data flow in the system is designed as a unidirectional, continuous streaming pipeline, ensuring that GPS data moves seamlessly from generation to visualization without unnecessary feedback loops. Each stage in the pipeline performs a distinct transformation, progressively enriching raw inputs into validated, structured, and actionable outputs.

The pipeline begins with the GPS simulator, which emits JSON-formatted messages at fixed intervals. Each message encodes the latitude, longitude, timestamp, and vehicle identifier for a locomotive or wagon. These messages are pushed into an Apache Kafka topic, where they are buffered and made available for downstream processing. Kafka allows multiple consumers to subscribe, which means that additional modules (e.g., anomaly detection or analytics) could be integrated in parallel without disturbing the core pipeline.

The next stage involves the Flask consumer, which subscribes to the Kafka topic and reads each incoming message in real time. Each message undergoes validation to check for missing, stale, or inconsistent data. For example, if a wagon reports coordinates with a timestamp misaligned from the current tick window, the entry is flagged in a dedicated validation field. When GPS coordinates are missing due to signal loss, the Kalman filter uses historical motion dynamics to estimate the most likely position. This ensures that vehicle trajectories remain smooth and continuous, even when real GPS data is incomplete or noisy.

After filtering and synchronization, the system applies connected component analysis to group vehicles based on spatial proximity. Each group is then analyzed to determine locomotive-wagon relationships. Wagons are assigned to locomotives using a combination of distance, angular orientation, and cross-track error criteria. In cases where no locomotive is present, the wagons are chained together as a "wagon-only consist." To improve temporal stability, the system incorporates a rigid-body drag mechanism that reduces fluctuations caused by GPS noise, ensuring that consist structures remain stable across successive ticks, even when trains traverse curves.

Once these assignments are finalized, the processed results are stored in a PostgreSQL database. The database schema includes tables for wagon-to-locomotive assignments, consist history, and locomotive snapshots, enabling both real-time monitoring and historical analytics. To support efficient visualization, the frontend queries this database periodically. The map view is refreshed at regular intervals, displaying locomotive and wagon positions, consist structures, and anomaly indicators such as unassigned wagons.

Finally, the architecture accounts for scalability. Kafka inherently supports horizontal scaling, allowing multiple consumer instances to be deployed across nodes. This enables parallel ingestion and processing of GPS data, making the design suitable for large-scale deployments involving hundreds of locomotives and thousands of wagons. Thus, while the current prototype focuses on proof-of-concept validation, the system can be extended to production scenario.

The complete workflow is illustrated in Figure 3.1, which shows the sequential flow of information from the GPS simulator through Kafka, Flask processing, Kalman filtering, assignment logic, database storage, and finally visualization.

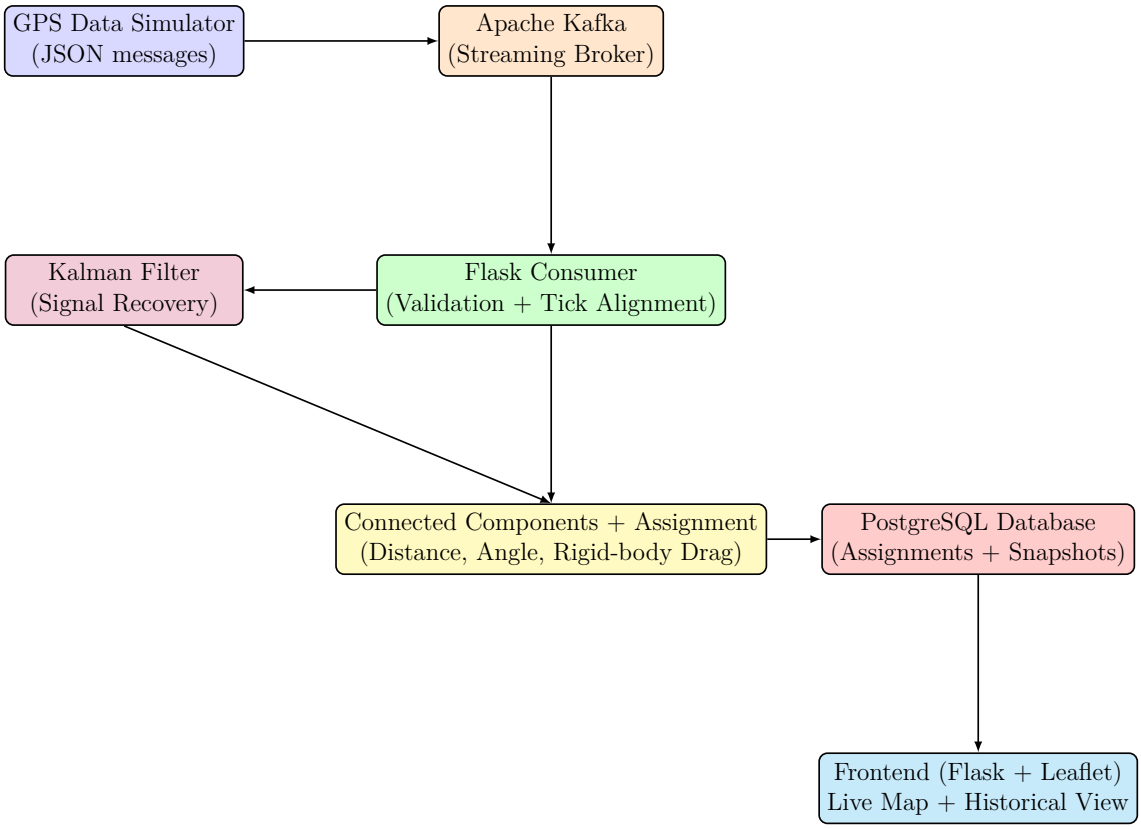


Figure 3.1: System data flow

4

Implementation

This section outlines the detailed implementation strategy used in designing the Real-time Asset Tracking and Management Information System (RATMIS). The design focuses on ensuring accurate, real-time visibility of locomotives and wagons by combining signal prediction, time synchronization, deterministic assignment logic, and a visualization layer. The following subsections describe the backend modules and their interactions, beginning from data simulation, ingestion and subscribing Kafka topics, and progressing through Kalman filtering, tick alignment, wagon-to-locomotive assignment, database persistence, and user interface integration.

4.1 GPS Data Simulation

In order to test and validate the asset tracking platform without requiring real GPS data from physical trains, we developed a synthetic GPS simulation module using Python. The primary objective was to model realistic movements of multiple trains with multiple wagons across predefined coordinates, introducing variability in speed, direction, spacing, and signal integrity.

Each simulated train (represented as a locomotive) was associated with three wagons. The simulation logic mimicked the real-world behavior of a moving train where wagons trail behind the locomotive with fixed longitudinal offsets of 10 m, 20 m, and 30 m. To create more complex scenarios, the simulator can also generate two trains running in parallel, offset laterally by 25 m, testing how the assignment system separates them.

Trains accelerate smoothly from rest, maintain cruise speed, and decelerate toward the end of the journey. GPS coordinates are generated by incrementally shifting the locomotive position using geospatial calculations, with each wagon's location derived by applying its fixed offset relative to the locomotive. To simulate signal loss scenarios, wagons occasionally emit missing (NaN) latitude or longitude values. These noisy readings are essential to test the robustness of the filtering and assignment algorithms.

The simulated messages are serialized in JSON format and published to a Kafka topic named `gps_train_location topic` to mimic live streaming from physical assets. A CSV log of the generated data is also maintained to serve as ground truth for evaluation.

4.2 Kafka Producer and Consumer Workflow

Real-time data streaming in the system is powered by Apache Kafka. The GPS data simulator, described previously, acts as a Kafka producer, continuously publishing JSON-formatted messages to the `gps_train_location topic`. Each message contains metadata such as vehicle ID, object type, timestamp, and geographic coordinates, simulating the live signals one would expect from locomotives and wagons.

On the backend, a Kafka consumer is embedded within a Flask application. This consumer subscribes to the GPS topic and sequentially processes incoming messages. Upon receiving a message, the system deserializes and validates its contents before appending it to a rolling buffer that holds the most recent observations for each vehicle. This sliding window allows the backend to access short-term history for prediction and consistency checks. To avoid processing outdated or irrelevant messages, a cleanup routine periodically removes entries older than forty seconds, ensuring that all subsequent operations work on fresh and timely data.

4.3 Kalman Filter Prediction Logic

Railway environments often present challenges for continuous GPS tracking because of tunnels, dense infrastructure, and other factors that lead to noisy or missing signals. To address this, the system applies a Kalman filter, a mathematical framework for estimating unknown variables from noisy data. The filter is particularly suitable here because train movements can be described using relatively simple linear dynamics, and the uncertainty in GPS readings can be modeled as Gaussian noise. This makes the Kalman filter both efficient and accurate for real-time railway tracking.

The system represents each vehicle with a state vector that includes both its position and velocity in a local planar coordinate system. By converting latitude and longitude into projected coordinates in meters, the state becomes:

$$x_k = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$$

where p_x, p_y are the vehicle's position components and v_x, v_y are the velocity components. This formulation allows the filter not only to track where the vehicle is but also to predict how it is moving, which is essential when GPS updates are unavailable.

The **prediction step** advances the state forward in time using a constant-velocity model. Assuming a time step Δt, the state evolves as:

$$x_{k|k-1} = F\, x_{k-1|k-1} + w_k$$

with transition matrix

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here, w_k is process noise, capturing small unmodeled accelerations or disturbances. The covariance of the state estimate is also updated in this step to reflect growing uncertainty as time progresses.

Whenever a new GPS reading is available, the **update step** corrects the predicted state. Observations are expressed as:

$$z_k = Hx_k + v_k,$$

where z_k contains the measured position, H is the observation matrix selecting position components, and v_k is the measurement noise. The Kalman gain is defined as:

$$K_k = P_{k|k-1}H^T \left(HP_{k|k-1}H^T + R\right)^{-1},$$

which determines how much the filter should trust the new GPS reading compared to its own prediction. The corrected state is then computed as:

$$x_{k|k} = x_{k|k-1} + K_k \left(z_k - Hx_{k|k-1}\right),$$

and the covariance is reduced to reflect improved certainty.

The filter operates seamlessly even when data is incomplete. If a GPS reading is missing or corrupted, the system simply skips the update step and continues with prediction alone. This allows trajectories to remain smooth and uninterrupted, filling in gaps where GPS would otherwise fail. Beyond providing estimated latitude and longitude, the filter also yields an estimated speed, computed as:

$$\hat{v} = \sqrt{v_x^2 + v_y^2}.$$

This velocity estimate is not only useful for consistency checks but also serves as input to later modules such as tick synchronization and rigid-body drag modeling.

In summary, the Kalman filter enables the system to overcome one of the biggest limitations of GPS tracking in railways: unreliability in harsh environments. By combining predictions from a motion model with noisy GPS readings, it produces reliable, continuous, and smooth position and velocity estimates. These estimates ensure that locomotives and wagons remain trackable even when signals drop out, which is crucial for building a robust, real-time railway asset tracking system.

4.4 Tick Synchronization and Staleness Handling

A major improvement in the system is the introduction of time synchronization through fixed ticks. Every vehicle update is aligned to a five-second tick, providing a consistent temporal framework across the system. Since not all GPS messages arrive exactly on tick boundaries, the system projects positions forward or backward to the nearest tick using velocity and heading estimates. The result is a set of tick-aligned coordinates that represent all vehicles at the same instant in time.

Alongside tick alignment, the system introduces staleness checks. Each vehicle's synchronized state records the original timestamp, the tick timestamp, and the age difference between the two. If this age exceeds a tolerance (around three seconds), the data is marked as stale. This ensures that outdated wagon readings are not paired with fresh locomotive data, preventing false assignments. By anchoring all logic to synchronized ticks and actively flagging stale information, the system guarantees that spatial comparisons are both fair and temporally consistent.

Figure 4.1 illustrates the overall flow of tick synchronization and staleness handling. Incoming GPS messages are first mapped to the nearest five-second tick. If the projection is within the allowed tolerance, the adjusted coordinates are accepted; otherwise, they are flagged as stale. The clean, synchronized data is then used for subsequent assignment logic.

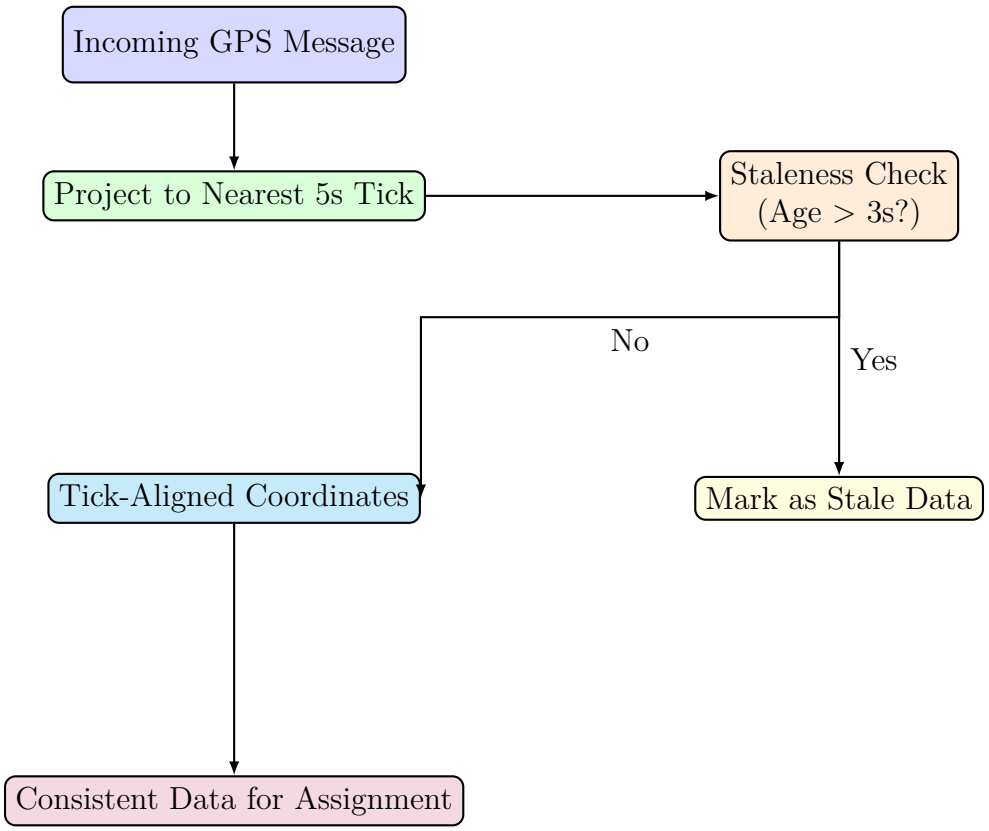


Figure 4.1: Flow diagram of tick synchronization and staleness handling.

4.5 Consist Formation with Connected Components and Rigid-Body Drag

Once vehicles are synchronized, the system proceeds to form train consists by grouping wagons and locomotives. Vehicles are treated as nodes in a graph, and an edge is drawn between two vehicles if they are within 20 m of one another. The connected component containing the current vehicle is then extracted and used to infer consist structure.

If a locomotive is present in the component, wagons are assigned to the nearest locomotive subject to multiple gates:

- Maximum distance of 60 m
- Heading alignment within an angular threshold
- Cross-track offset not exceeding 12 m

If no locomotive is present, the wagons are grouped together into a provisional chain. To enhance stability, the system incorporates rigid-body drag, where wagons inherit the locomotive's displacement between ticks. This means that when the locomotive moves forward, its wagons are carried along by the same translation, and once a stable heading is available, wagons are also rotated around the locomotive's trajectory. The effect is that wagons follow smoothly through curves and maintain realistic spacing, avoiding the jitter that can occur with raw GPS or clustering-only methods.

Each assignment is annotated with a validation label that reflects its reliability. Wagons with fresh, synchronized data are marked as "YES," while those assigned during early heading estimation are marked as "PROVISIONAL." Stale readings result in "STALE_WAGON" or "STALE_LOCO," and wagon-only groups are identified as "CHAIN" or "NO." This classification gives operators a transparent view of assignment quality at each tick.

Figure 4.2 illustrates the flow of consist formation and assignment. Vehicles are first mapped into connected components, then validated against gating rules. Depending on locomotive presence, wagons either attach to locomotives or form provisional chains. Finally, rigid-body drag ensures smooth motion and reliability labels classify the result.

4.6 PostgreSQL Schema and Data Persistence

To ensure durability and analytical capability, all processed results are stored in a PostgreSQL database. Two main tables are maintained. The `assignments` table records every wagon update with its filtered coordinates, assigned locomotive, distance, validation state, and associated audit timestamps, including which locomotive and wagon readings were used in the calculation. The `locomotive_assignments` table stores the latest locomotive position along with a JSON list of its assigned

wagons, supporting efficient visualization queries.

PostgreSQL indexing is applied to frequently queried fields such as vehicle_id and timestamp, ensuring that the system can efficiently retrieve both live and historical data. The schema design balances relational consistency with flexibility, supporting advanced queries such as anomaly detection and historical performance analysis.

4.7 User Interface and Visualization Layer

The processed assignments are visualized through a web interface developed with Flask and Leaflet.js. This interface provides operators with real-time situational awareness of train movements across the network.

The All Train view displays all locomotives and wagons on a live map, updating automatically every ten seconds. Locomotives are shown as triangles and wagons as circles or squares, with color codes representing assignment. Trails depict recent movements, and interactive popups provide metadata including vehicle ID, timestamp, assigned locomotive, and validation status.

For deeper analysis, a per-wagon view allows users to track the trajectory of a specific wagon. This view highlights anomalies, such as unassigned positions, and provides historical logs of movements and assignments in tabular form. These insights are delivered through dedicated Flask routes that query PostgreSQL for both raw and processed data.

By combining tick-aligned backend logic with a map-centric visualization layer, the system ensures that complex prediction and assignment decisions are communicated in an intuitive and accessible manner. Operators can thus monitor, validate, and analyze train compositions in real time without needing to engage with the underlying algorithms.

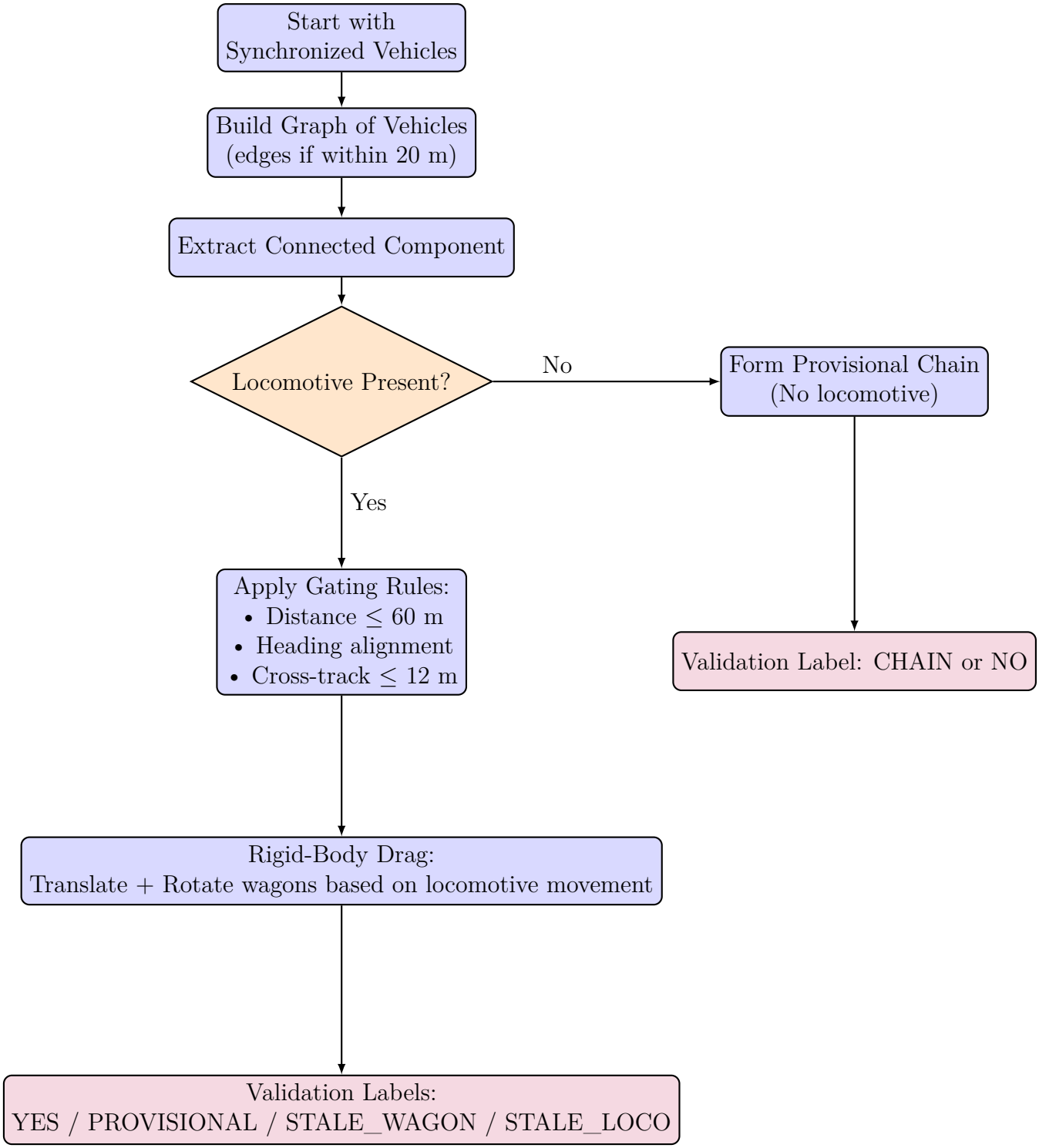


Figure 4.2: Detailed flow of consist formation with connected components, gating rules, rigid-body drag, and assignment validation.

5

Results

5.1 Evaluation of the System

The evaluation compares predicted wagon–locomotive assignments against ground truth over a recent one-hour window. Predictions come from the `assignments` table and include the wagon identifier, the predicted locomotive, the predicted wagon position, a predicted distance to the locomotive, and a timestamp. Ground truth locomotive labels are reconstructed deterministically from the wagon identifier prefix (e.g., "1000-series → Locomotive_1"), which gives a true locomotive for each $(\texttt{wagon\_id}, ts)$ pair. When a simulator CSV is present, true wagon positions are also loaded, and timestamps on both sides are parsed and normalized to UTC with a robust datetime parser to avoid time-zone or suffix issues. This ensures that time alignment and downstream error calculations are fair and repeatable.

Predictions and ground truth positions are matched by a nearest-neighbor time join that permits a tolerance of ± 10 seconds, with an option to snap both streams to a five-second grid before the merge. The join returns a per-row variable `match_age_s`, so we know exactly how far each matched pair is apart in time. Position errors are computed only for tight matches to avoid penalizing the model for clock drift. This matching logic is implemented in the function `nearest_time_join`, which also sorts and type-aligns inputs to make the as-of merge reliable.

Two distance-aware confidence scores are computed for each prediction. The first is a simple exponential score, $\exp(-d/\text{scale})$, which falls off smoothly with distance. The second is a multi-wagon spacing score that becomes large when the measured wagon–locomotive distance is close to an integer multiple of the effective wagon spacing ($\approx 10\,\text{m} \times \sqrt{2} \approx 14.14\,\text{m}$ in this simulator). The multi-spacing score penalizes the residual distance from the nearest multiple, capturing the physical layout of a train. It is particularly effective here because correct assignments line up with that geometry. A threshold τ is selected by sweeping τ from 0.05 to 0.95 and choosing the value that maximizes the F1 score on the evaluation set, which yields a principled operating point that balances precision and recall.

Once a score and threshold are fixed, predictions are treated as "correct" or "incorrect," and standard classification metrics are computed. Accuracy is the overall fraction of correct decisions. Precision measures what fraction of predictions labeled "correct" truly are correct. Recall measures what fraction of all truly correct cases

the system manages to capture. The F1 score is the harmonic mean of precision and recall. Balanced accuracy is also reported to account for class imbalance. A confusion matrix provides the raw counts of true negatives, false positives, false negatives, and true positives. In addition, a precision–recall (PR) curve is traced from the continuous scores to compute AUPRC, a threshold-free summary of ranking quality. All of these calculations are performed with the same inputs used for the confusion matrix, ensuring consistency.

Temporal behavior is assessed per wagon. The flip rate measures how often the predicted locomotive changes between consecutive timestamps. The ID-switch rate measures how often such a flip happens while the true locomotive stayed the same, i.e., a spurious oscillation. These stability statistics are summarized across wagons to reveal jitter that may be visually distracting in live dashboards even when accuracy is high. Coverage is also reported when simulator truth is available, indicating the fraction of rows that could be matched closely in time for spatial error calculations.

Table 5.1: Evaluation metrics.

Metric	**Value**
Assignment Accuracy	88%
Classification Accuracy	94%
Precision	1.00
Recall	0.93
F1 Score	0.97
AUPRC	0.997

Table 5.1 shows the headline results for this run. Assignment accuracy is 88% when measured as a direct label match without thresholding. After converting the distance-based score to a decision, the system attains 94% classification accuracy. Precision is 1.00, meaning the system made no false-positive acceptances—whenever it said an assignment was correct, it was correct. Recall is 0.93, reflecting a small number of misses, and the resulting F1 score is 0.97, confirming an excellent balance between precision and recall. The AUPRC is 0.997, which indicates that the score cleanly separates correct from incorrect assignments across thresholds, not only at the chosen operating point. These numbers are consistent with a conservative system that almost never accepts a wrong match, at the cost of occasionally rejecting a true one.

The distance plots in Figures 5.1 and 5.2 are especially diagnostic. Correct assignments cluster neatly around expected wagon spacing multiples, while incorrect assignments fall at scattered, irregular distances. This geometric separation explains why the distance-based scoring and thresholding produce such strong classification results.

Figure 5.3 confirms this numerically. The absence of false positives explains the perfect precision of 1.00, while the 11 false negatives account for the recall of 0.93. If desired, the threshold τ could be tuned to slightly increase recall at the cost of marginally reducing precision.

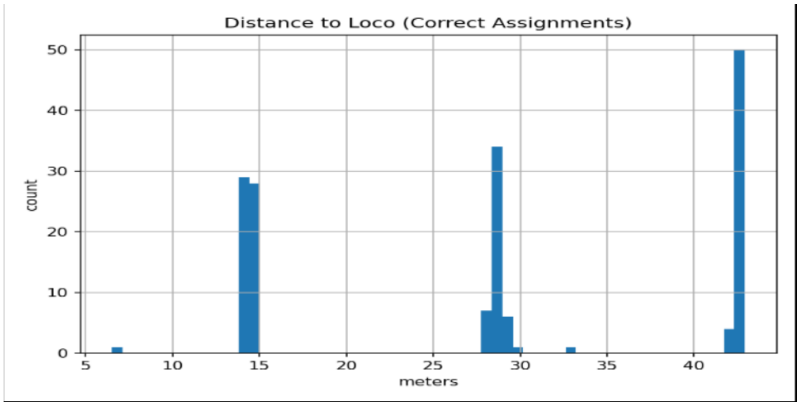


Figure 5.1: Histogram of distances to locomotive for correct assignments.

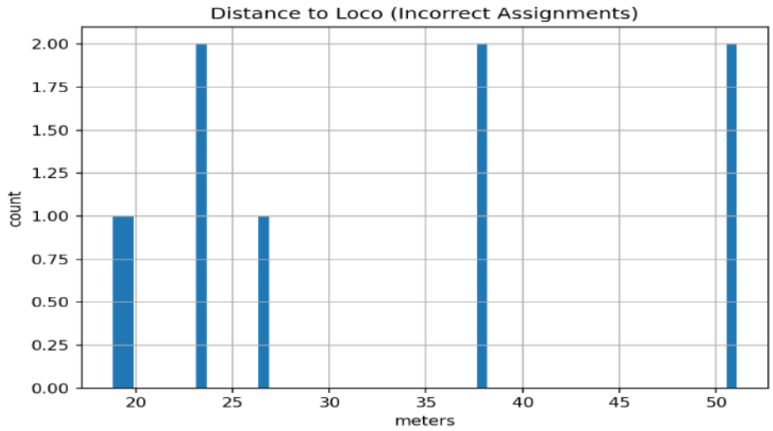


Figure 5.2: Histogram of distances to locomotive for incorrect assignments.

Finally, Figure 5.4 shows the PR curve, where the AUPRC of 0.997 highlights the robustness of the scoring method across thresholds, not just at the chosen operating point. This curve indicates that the system can achieve different trade-offs between precision and recall with very little degradation in overall decision quality.

Table 5.2: Stability results.

Statistic	**Value**
Flip rate (per wagon)	$\approx 10\%$
ID switch rate (per wagon)	$\approx 10\%$

In summary, the evaluation pipeline loads predictions, reconstructs ground truth, and aligns both streams in time. It then scores each row using geometry-aware distance functions and selects an operating threshold to balance precision and recall. With these settings, the system achieves 88% assignment accuracy, 94% classification accuracy, perfect precision, 0.93 recall, 0.97 F1, and a near-perfect AUPRC of 0.997. The figures corroborate the story: correct cases conform to physical spacing, incorrect cases do not, and the PR curve indicates robust separability. The principal improvement area is temporal steadiness, which can be addressed with simple smoothing while preserving the model's strong decision quality.

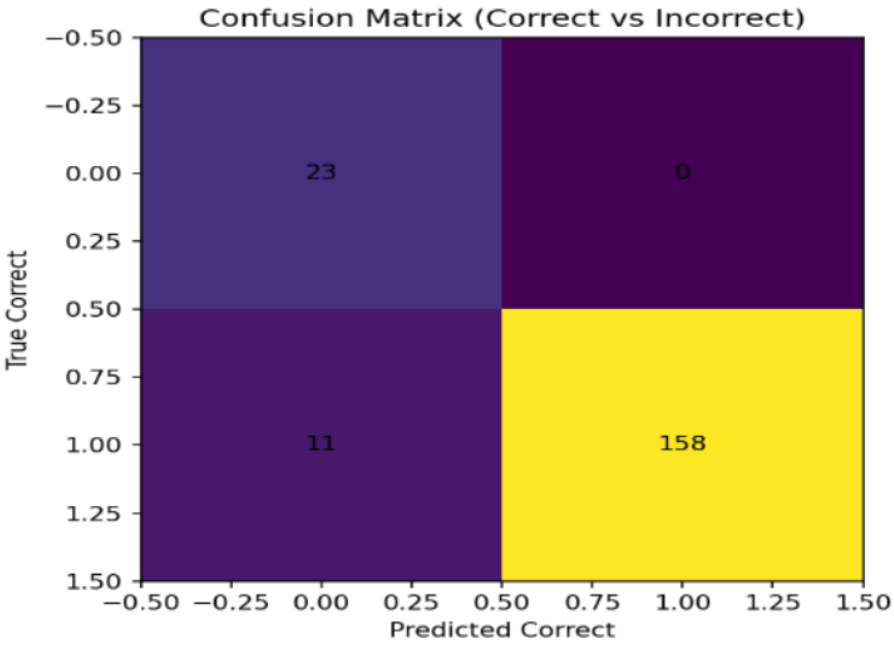


Figure 5.3: Confusion matrix of predicted vs. true assignments.

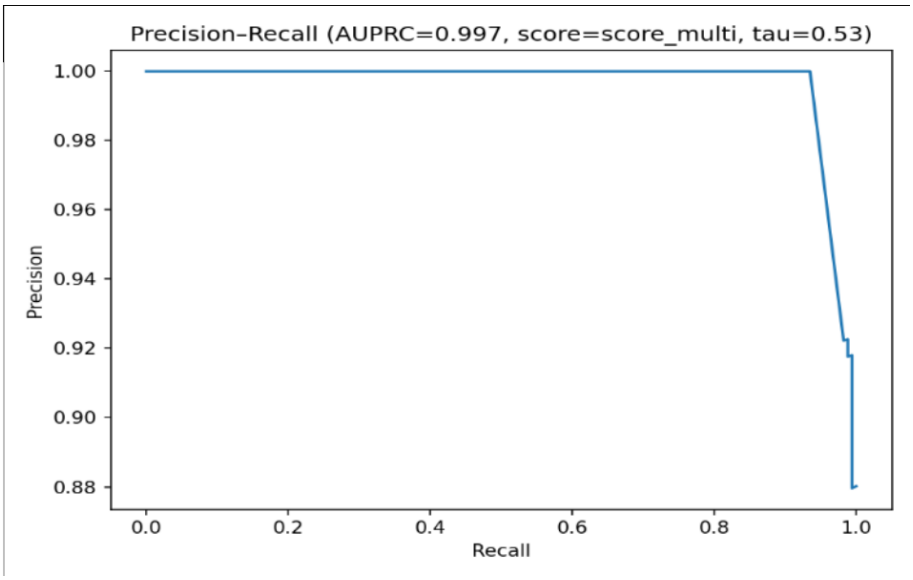


Figure 5.4: Precision–Recall (PR) curve for the assignment system.

6

Conclusion

6.1 Future Work

Although this thesis has demonstrated a software-simulated framework for wagon–locomotive assignment, several important directions remain open for further investigation. A primary next step is validation with real-world GPS data collected from locomotives and wagons operating under true conditions. Such data would expose the algorithms to multipath interference in dense cities, long GPS outages in tunnels, and the device-specific behaviors of railway telematics units such as reporting delays or power-saving modes. Testing against these operational realities would determine how well the high accuracy achieved in simulation translates into practice.

Another avenue for future work is the integration of additional data sources beyond GPS. While technologies such as axle counters, RFID tags, or track circuits are insufficient as standalone solutions because they only provide point detections, they can serve as valuable redundant inputs when fused with continuous GPS-based tracking. For example, Digital Automatic Coupling telemetry could verify the physical connections between wagons, RFID or axle counters at strategic nodes could serve as cross-checks to increase reliability, and inertial measurement units could offer dead-reckoning capability during GPS gaps. Combining these heterogeneous sources would increase robustness in environments where GPS alone is ambiguous, particularly in dense yards with many parallel tracks or during prolonged signal loss.

The current implementation has been demonstrated at the level of a few locomotives and wagons, but railway operations involve hundreds of trains and thousands of wagons moving simultaneously. Scaling the system to this level requires distributed processing across Kafka partitions, optimized database schemas for long-term storage, and careful latency management to guarantee that operators and dispatchers receive timely results. Stress-testing at network scale would show whether the framework is capable of supporting the demands of infrastructure managers.

It is also important to situate the framework within the broader landscape of European railway digitalisation. Projects such as the European Rail Traffic Management System and Digital Automatic Coupling will require live digital composition data to manage traffic, automate yard operations, and optimise capacity. Future research should examine how the methods developed in this thesis can be integrated with these initiatives to contribute to the wider transformation of railways.

Finally, an especially promising direction is the application of this framework to autonomous or remotely operated freight trains. In such cases there is no crew onboard to confirm wagon attachments or update manifests manually, so continuous and automated assignment becomes the foundation of safe operations. A validated assignment pipeline that maintains accuracy and stability under GPS noise, dropouts, and asynchronous updates would serve as the digital backbone for these emerging autonomous systems.

6.2 Conclusion

This thesis has addressed the problem of real-time wagon–locomotive assignment under the constraints of noisy, missing, and asynchronous GPS-like data. While the railway industry has long relied on static departure manifests and infrastructure-based detection technologies, these methods are no longer sufficient in modern freight operations where train compositions change dynamically, signals are unreliable, and supply chains require constant visibility.

The work presented here demonstrates that a software-only pipeline can overcome these limitations. Using Kafka for streaming, Kalman filtering for predictive smoothing, tick alignment for temporal synchronization, and graph-based deterministic grouping for composition inference, the system was able to transform fragmented GPS data into a coherent and continuous digital record of train composition. Stability was reinforced by distance and angle gating, stickiness rules, and rigid-body drag modeling, all of which ensured that wagons remained consistently linked to the correct locomotive even under noisy or incomplete conditions.

Evaluation results confirm the effectiveness of this approach. The system achieved an assignment accuracy of nearly ninety percent in its baseline form, with refined thresholds producing an F1 score of 0.97. Precision was perfect at 1.0, meaning the system did not generate false assignments, while recall of 0.93 indicated that the majority of true assignments were successfully recovered. Stability analysis showed that assignments did not fluctuate excessively, with flip and ID-switch rates around ten percent, and the precision–recall curve demonstrated robustness with an area under the curve close to 0.997. Together, these results provide strong evidence that the framework can deliver reliable real-time train composition even when GPS data is incomplete or misaligned.

Although limited to a simulated environment, the thesis contributes a validated proof of concept that predictive filtering combined with deterministic rules can address a problem that existing GPS-only or infrastructure-only approaches have not solved. The key contributions are the development of a simulation framework for realistic GPS-like data generation, the integration of predictive and deterministic methods into a complete processing pipeline, and the demonstration of high accuracy, precision, and stability under controlled but challenging conditions.

In conclusion, the research shows that real-time digital train composition is both achievable and necessary. By ensuring accurate wagon–locomotive assignments, the framework supports safer braking distance calculations, more reliable logistics opera-

tions, and greater readiness for the next generation of autonomous and digitalised rail freight systems. While future validation on real-world data remains essential, this work lays the foundation for reliable, scalable, and operationally meaningful solutions to one of the central challenges of modern rail freight.

Bibliography

[1] A. Brezulianu et al., "Active control parameters monitoring for freight trains, using wireless sensor network platform and internet of things," *Processes*, vol. 8, no. 6, p. 639, 2020.

[2] I. Moya et al., "Freight wagon digitalization for condition monitoring and advanced operation," *Sensors*, vol. 23, no. 17, p. 7448, 2023.

[3] M. Rahimi, H. Liu, I. D. Cardenas, A. Starr, A. Hall, and R. Anderson, "A review on technologies for localisation and navigation in autonomous railway maintenance systems," *Sensors*, vol. 22, no. 11, p. 4185, 2022.

[4] H. Winter, V. Willert, and J. Adamy, "Train-borne localization exploiting track-geometry constraints–a practical evaluation," *arXiv preprint arXiv:1906.07569*, 2019.

[5] W. Löffler and M. Bengtsson, "Train localization during gnss outages: A minimalist approach using track geometry and imu sensor data," in *2024 27th International Conference on Information Fusion (FUSION)*, IEEE, 2024, pp. 1–8.

[6] A. Neri et al., "A method for multipath detection and mitigation in railway control applications," in *Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016)*, 2016, pp. 1843–1855.

[7] Z. Yan, L. Ruotsalainen, X. Chen, and X. Tang, "An ins-assisted vector tracking receiver with multipath error estimation for dense urban canyons," *GPS Solutions*, vol. 27, no. 2, p. 88, 2023.

[8] A. V. Tulupov, A. V. Beloshitsky, E. A. Shitov, and Y. A. Shitova, "Innovative, scientific and technological priorities of railway freight transport," *WORLD*, vol. 19, no. 5, pp. 186–195, 2021.

[9] M. Balog, H. Sokhatska, and A. Iakovets, "Intelligent systems in the railway freight management," in *International scientific-technical conference manufacturing*, Springer, 2019, pp. 390–405.

[10] Y. Wang et al., "Raillomer: Rail vehicle localization and mapping with lidar-imu-odometer-gnss data fusion," *arXiv preprint arXiv:2111.15043*, 2021.

[11] J. Beugin, C. Legrand, J. Marais, M. Berbineau, and E.-M. El-Koursi, "Safety appraisal of gnss-based localization systems used in train spacing control," *IEEE Access*, vol. 6, pp. 9898–9916, 2018.

[12] O. Olaby, M. Hamadache, D. Soper, P. Winship, and R. Dixon, "Development of a novel railway positioning system using rfid technology," *Sensors*, vol. 22, no. 6, p. 2401, 2022.

[13] S. Spinsante and C. Stallo, "Hybridized-gnss approaches to train positioning: Challenges and open issues on uncertainty," *Sensors*, vol. 20, no. 7, p. 1885, 2020.

[14] H. Noureddine, D. Castelain, and R. Pyndiah, "Train tracking and shadowing estimation based on received signal strength," in *International workshop on communication technologies for vehicles*, Springer, 2011, pp. 23–33.

[15] Y. Wang, P. Wang, X. Wang, and X. Liu, "Position synchronization for track geometry inspection data via big-data fusion and incremental learning," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 544–565, 2018.

[16] Y. Lai and C. P. Barkan, "Train braking distance ratio: A parameter for railway signal system design," in *83rd Annual Meeting of the Transportation Research Board, Washington, DC*, 2004.

[17] A. Pernestål, A. Engholm, M. Bemler, and G. Gidofalvi, "How will digitalization change road freight transport? scenarios tested in sweden," *Sustainability*, vol. 13, no. 1, p. 304, 2020.

[18] L. Zhao and W. Y. Ochieng, "An extended kalman filter algorithm for integrating gps and low-cost dead reckoning system data for vehicle performance and emissions monitoring," Loughborough University, Tech. Rep.

[19] A. O. Agbailu, A. Seno, and O. O. Clement, "Kalman filter algorithm versus other methods of estimating missing values: Time series evidence," *Studies*, vol. 4, no. 2, pp. 1–9, 2020.

[20] D. M. Kreindler and C. J. Lumsden, "The effects of the irregular sample and missing data in time series analysis," in *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*, CRC Press, 2016, pp. 149–172.

[21] T. Luo, X. Zheng, G. Xu, K. Fu, and W. Ren, "An improved dbscan algorithm to detect stops in individual trajectories," *ISPRS International Journal of Geo-Information*, vol. 6, no. 3, p. 63, Feb. 2017.

[22] L. Gong, T. Yamamoto, and T. Morikawa, "Identification of activity stop locations in gps trajectories by dbscan-te method combined with support vector machines," *Transportation research procedia*, vol. 32, pp. 146–154, 2018.

[23] Y. Weng et al., "An efficient algorithm for extracting railway tracks based on spatial-channel graph convolutional network and deep neural residual network," *ISPRS International Journal of Geo-Information*, vol. 13, no. 9, p. 309, 2024.

[24] D. P. Singh and M. Yadav, "Deep learning-based semantic segmentation of three-dimensional point cloud: A comprehensive review," *International Journal of Remote Sensing*, vol. 45, no. 2, pp. 532–586, 2024.

[25] Y. Dong et al., "Heterogeneous graph attention network for rail fastener looseness detection using distributed acoustic sensing and accelerometer data fusion," *Automation in Construction*, vol. 172, p. 106 051, 2025.

[26] M. Grinberg, *Flask web development.* " O'Reilly Media, Inc.", 2018.

[27] J. Kreps, N. Narkhede, J. Rao, et al., "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, Athens, Greece, vol. 11, 2011, pp. 1–7.

[28] M. Stonebraker and L. A. Rowe, "The design of postgres," *ACM Sigmod Record*, vol. 15, no. 2, pp. 340–355, 1986.

[29] A. Kucuk, S. M. Hamdi, B. Aydin, M. A. Schuh, and R. A. Angryk, "Pg-trajectory: A postgresql/postgis based data model for spatiotemporal trajectories," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, IEEE, 2016, pp. 81–88.

[30] V. Agafonkin, "Leaflet: An open-source javascript library for mobilefriendly interactive maps, 2015," *URL http://leafletjs. com*, 2020.

[31] B. Jakobus, *Mastering Bootstrap 4: Master the latest version of Bootstrap 4 to build highly customized responsive web apps*. Packt Publishing Ltd, 2018.

A

Appendix

A.1 Algorithms and Mathematical Models

A.1.1 Kalman Filter Function

The Kalman filter is implemented in Python as:

```
def apply_kalman_filter(df_in: pd.DataFrame) -> pd.DataFrame:
    if df_in is None or len(df_in) == 0:
        return df_in

    df = df_in.copy()
    # Normalize and sort time
    df['timestamp'] = pd.to_datetime(df['timestamp'], utc=True, errors='coerce')
    df = df.sort_values('timestamp').reset_index(drop=True)

    # Helpers: lat/lon <-> local meters (simple, accurate enough for city-scale)
    R_EARTH = 6371000.0  # meters

    # choose reference as first non-null lat/lon
    if df['latitude'].notna().any() and df['longitude'].notna().any():
        lat0 = float(df.loc[df['latitude'].notna().idxmax(), 'latitude'])
        lon0 = float(df.loc[df['longitude'].notna().idxmax(), 'longitude'])
    else:
        # nothing valid, just return placeholders
        df['latitude_pred'] = df['latitude']
        df['longitude_pred'] = df['longitude']
        df['speed_pred_mps'] = np.nan
        return df

    clat = math.cos(math.radians(lat0))

    def ll_to_xy(lat, lon):
        if pd.isna(lat) or pd.isna(lon):
            return np.nan, np.nan
        x = math.radians(lon - lon0) * clat * R_EARTH
```

```
        y = math.radians(lat - lat0) * R_EARTH
        return x, y

    def xy_to_ll(x, y):
        lon = lon0 + math.degrees(x / (R_EARTH * clat))
        lat = lat0 + math.degrees(y / R_EARTH)
        return lat, lon

    # Build observation arrays in meters
    obs_x = []
    obs_y = []
    for lat, lon in zip(df['latitude'].tolist(), df['longitude'].tolist()):
        x, y = ll_to_xy(lat, lon)
        obs_x.append(x)
        obs_y.append(y)
    obs_x = np.array(obs_x, dtype=float)
    obs_y = np.array(obs_y, dtype=float)

    # time deltas
    ts = df['timestamp'].values
    tsec = (ts.astype('datetime64[ns]').astype(np.int64) / 1e9).astype(float)

    dt_arr = np.maximum(1e-3, np.diff(tsec, prepend=tsec[0]))

    # State: [x, y, vx, vy]^T
    x = np.zeros((4, 1), dtype=float)

    # Initial position from first available observation
    if np.isnan(obs_x[0]) or np.isnan(obs_y[0]):
        # if first is missing, try to find first non-nan
        first_idx = int(np.where(~(np.isnan(obs_x) | np.isnan(obs_y)))[0][0])
        x[0, 0] = obs_x[first_idx]
        x[1, 0] = obs_y[first_idx]
    else:
        x[0, 0] = obs_x[0]
        x[1, 0] = obs_y[0]

    # Initial velocity from first displacement if possible (else 0)
    if len(obs_x) >= 2:
        j = 1
        # find the next non-nan obs to estimate initial velocity
        while j < len(obs_x) and (np.isnan(obs_x[j]) or np.isnan(obs_y[j])):
            j += 1
        if j < len(obs_x):
            dt0 = max(1e-3, tsec[j] - tsec[0])
            x[2, 0] = (obs_x[j] - x[0, 0]) / dt0
```

```
        x[3, 0] = (obs_y[j] - x[1, 0]) / dt0

# Covariance
P = np.eye(4, dtype=float) * 100.0

# Process/measurement models
H = np.array([[1, 0, 0, 0],
              [0, 1, 0, 0]], dtype=float)

# Measurement noise (meters). Smartphone GNSS ~ 5-10 m typically.
sigma_meas = 6.0
Rm = np.array([[sigma_meas**2, 0],
               [0, sigma_meas**2]], dtype=float)

# Acceleration (process) noise (m/s^2). Tuned conservative.
sigma_a = 1.5

lat_pred_list = []
lon_pred_list = []
spd_pred_list = []

for i in range(len(df)):
    dt = float(dt_arr[i])
    # State transition
    F = np.array([[1, 0, dt, 0],
                  [0, 1, 0, dt],
                  [0, 0, 1,  0],
                  [0, 0, 0,  1]], dtype=float)

    # Process noise Q for constant-acceleration white noise
    dt2 = dt * dt
    dt3 = dt2 * dt
    q = sigma_a**2
    Q = np.array([[dt3/3,    0,    dt2/2, 0],
                  [0,    dt3/3,    0,   dt2/2],
                  [dt2/2,   0,     dt,   0],
                  [0,    dt2/2,    0,    dt]], dtype=float) * q

    # Predict
    x = F @ x
    P = F @ P @ F.T + Q

    # Predicted speed (for gating + output)
    spd_pred = float(math.hypot(x[2, 0], x[3, 0]))
    # Adaptive gate: allow at least 100 m or ~4 seconds of motion
    gate_m = max(100.0, 4.0 * spd_pred * dt)
```

```
        # Measurement update (if available & within gate)
        zx = obs_x[i]
        zy = obs_y[i]
        z_valid = not (np.isnan(zx) or np.isnan(zy))
        if z_valid:
            # innovation
            y_res = np.array([[zx - x[0, 0]],
                              [zy - x[1, 0]]], dtype=float)
            # gate on Euclidean distance
            if (y_res[0, 0]**2 + y_res[1, 0]**2) <= (gate_m * gate_m):
                S = H @ P @ H.T + Rm
                K = P @ H.T @ np.linalg.inv(S)
                x = x + K @ y_res
                I = np.eye(4)
                P = (I - K @ H) @ P
            else:
                # reject as outlier, keep prediction only
                pass

        # Store outputs (predicted lat/lon always available)
        lat_p, lon_p = xy_to_ll(x[0, 0], x[1, 0])
        lat_pred_list.append(lat_p)
        lon_pred_list.append(lon_p)
        spd_pred_list.append(spd_pred)

    df['latitude_pred'] = lat_pred_list
    df['longitude_pred'] = lon_pred_list
    df['speed_pred_mps'] = spd_pred_list
    return df
```

A.1.2 Rigid-Body Drag Logic

The rigid-body drag logic is implemented as:

```
def _rigid_drag(prev_anchor, loco_lat_now, loco_lon_now):
    lat_ref = float(prev_anchor['llat'])
    lon_ref = float(prev_anchor['llon'])
    wlat_prev = float(prev_anchor['wlat'])
    wlon_prev = float(prev_anchor['wlon'])

    m_per_deg_lat, m_per_deg_lon = _meters_per_deg(lat_ref)

    # previous offset (meters) from loco->wagon in ENU
    off_x = (wlon_prev - lon_ref) * m_per_deg_lon
    off_y = (wlat_prev - lat_ref) * m_per_deg_lat
```

```
# loco heading now (prev loco -> current loco)
hdg_now = _bearing_deg(lat_ref, lon_ref, loco_lat_now, loco_lon_now)
hdg_prev = prev_anchor.get('hdg_deg', None)

# loco translation in ENU (meters) relative to previous loco
dx = (loco_lon_now - lon_ref) * m_per_deg_lon
dy = (loco_lat_now - lat_ref) * m_per_deg_lat

if hdg_prev is None:
    # first step: no rotation estimate yet; do pure translation
    x_new = off_x + dx
    y_new = off_y + dy
else:

    dpsi_deg = ((hdg_now - hdg_prev + 540.0) % 360.0) - 180.0
    # wrap to [-180,180]
    dpsi = math.radians(dpsi_deg)
    cosd, sind = math.cos(dpsi), math.sin(dpsi)
    x_rot = cosd * off_x - sind * off_y
    y_rot = sind * off_x + cosd * off_y
    x_new = x_rot + dx
    y_new = y_rot + dy

# back to lat/lon
wlat_new = lat_ref + y_new / m_per_deg_lat
wlon_new = lon_ref + x_new / m_per_deg_lon
return float(wlat_new), float(wlon_new), float(hdg_now)
```

This applies a rotation + translation to the wagon's previous offset relative to the locomotive, maintaining consist alignment.

A.2 Experimental Setup

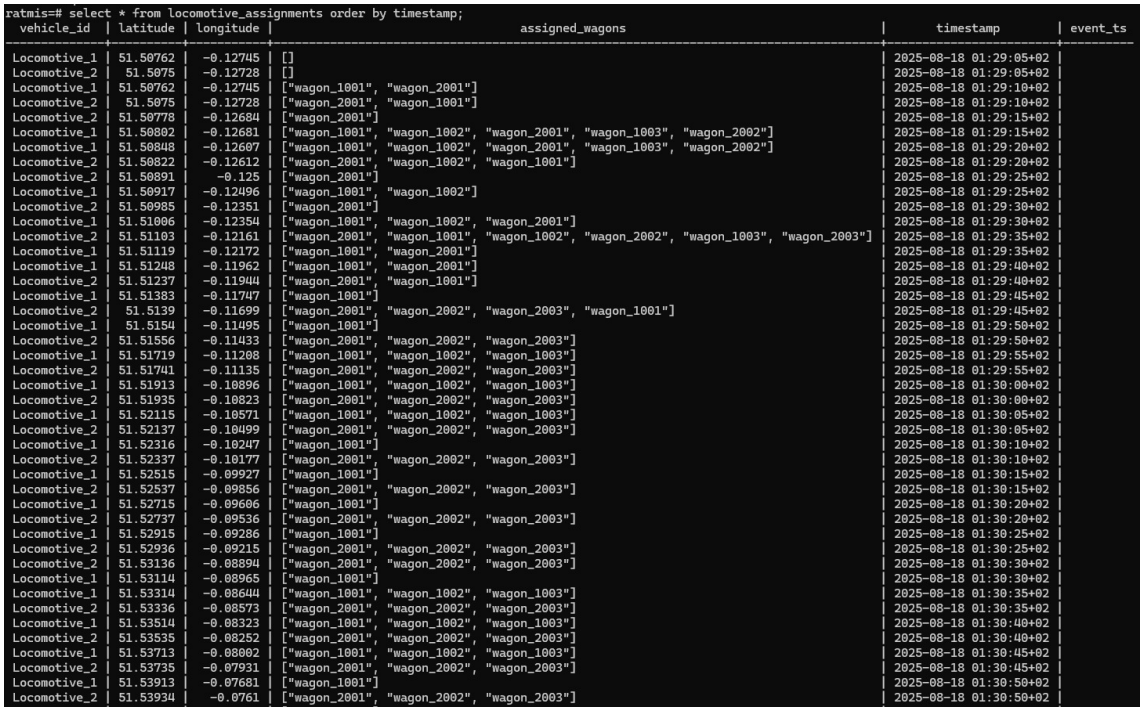
- **Simulator:** Python-based, generating wagon/locomotive traces with configurable offsets and noise.
- **Data Transport:** Kafka (v3.5.0) used for distributed streaming.
- **Backend:** Flask (v2.2) consumer handling filtering and assignment.
- **Database:** PostgreSQL (v14) for persistence.
- **Hardware:** Intel r5 laptop with 16 GB RAM, Ubuntu 22.04.

A.3 User Interface Snapshots

This appendix provides screenshots of the developed web frontend and system views. The figures illustrate different components, including assignment tables, user interface pages, and both raw and processed views.

```
ratmis=# select * from assignments order by timestamp;
 vehicle_id | object_type | latitude | longitude | assigned_to_locomotive | distance_to_locomotive |      timestamp      | validated | wagon_chain |     wagon_ts_used     |     loco_ts_used      | event
_ts
------------+-------------+----------+-----------+------------------------+------------------------+---------------------+-----------+-------------+-----------------------+-----------------------+------
---
 wagon_1003 | wagon       | 51.50735 |  -0.12789 | UNASSIGNED             |                        | 2025-08-18 01:29:05+02 | NO        | wagon_1003  | 2025-08-18 01:29:08+02 |                       |
 wagon_1002 | wagon       | 51.50744 |  -0.12774 | UNASSIGNED             |                        | 2025-08-18 01:29:05+02 | NO        | wagon_1002  | 2025-08-18 01:29:08+02 |                       |
 wagon_1001 | wagon       | 51.50753 |   -0.1276 | UNASSIGNED             |                        | 2025-08-18 01:29:05+02 | NO        | wagon_1001  | 2025-08-18 01:29:08+02 |                       |
 wagon_2003 | wagon       | 51.50723 | -0.12771 | UNASSIGNED             |                        | 2025-08-18 01:29:05+02 | NO        | wagon_2003  | 2025-08-18 01:29:08+02 |                       |
 wagon_2002 | wagon       | 51.50732 | -0.12756 | UNASSIGNED             |                        | 2025-08-18 01:29:05+02 | NO        | wagon_2002  | 2025-08-18 01:29:08+02 |                       |
 wagon_2001 | wagon       | 51.50741 | -0.12742 | UNASSIGNED             |                        | 2025-08-18 01:29:05+02 | NO        | wagon_2001  | 2025-08-18 01:29:08+02 |                       |
 wagon_2001 | wagon       | 51.50741 | -0.12742 | Locomotive_1           |     23.4568335635104775 | 2025-08-18 01:29:10+02 | YES       |             | 2025-08-18 01:29:08+02 | 2025-08-18 01:29:08+02 |
 wagon_2003 | wagon       | 51.50723 | -0.12771 | Locomotive_2           |                        | 2025-08-18 01:29:10+02 | PROVISIONAL | wagon_2003 | 2025-08-18 01:29:08+02 |                       |
 wagon_1003 | wagon       | 51.50735 | -0.12789 | Locomotive_1           |                        | 2025-08-18 01:29:10+02 | PROVISIONAL | wagon_1003 | 2025-08-18 01:29:08+02 |                       |
 wagon_1002 | wagon       | 51.50744 | -0.12774 | Locomotive_1           |                        | 2025-08-18 01:29:10+02 | PROVISIONAL | wagon_1002 | 2025-08-18 01:29:08+02 |                       |
 wagon_2002 | wagon       | 51.50732 | -0.12756 | Locomotive_2           |                        | 2025-08-18 01:29:10+02 | PROVISIONAL | wagon_2002 | 2025-08-18 01:29:08+02 |                       |
 wagon_1001 | wagon       | 51.50753 |   -0.1276 | Locomotive_1           |     14.4472570123688857 | 2025-08-18 01:29:10+02 | YES       |             | 2025-08-18 01:29:08+02 | 2025-08-18 01:29:08+02 |
 wagon_1003 | wagon       | 51.50778 | -0.12725 | Locomotive_1           |      42.843847907162261 | 2025-08-18 01:29:15+02 | YES       |             | 2025-08-18 01:29:14+02 | 2025-08-18 01:29:14+02 |
 wagon_1001 | wagon       | 51.50793 | -0.12696 | Locomotive_1           |     14.4471922637165515 | 2025-08-18 01:29:15+02 | YES       |             | 2025-08-18 01:29:14+02 | 2025-08-18 01:29:14+02 |
 wagon_2001 | wagon       | 51.50769 | -0.12698 | Locomotive_2           |      13.955051303859999 | 2025-08-18 01:29:15+02 | YES       |             | 2025-08-18 01:29:14+02 | 2025-08-18 01:29:14+02 |
 wagon_2002 | wagon       |  51.5076 | -0.12712 | Locomotive_1           |      51.062893853301177 | 2025-08-18 01:29:15+02 | YES       |             | 2025-08-18 01:29:14+02 | 2025-08-18 01:29:14+02 |
 wagon_1002 | wagon       | 51.50784 |  -0.1271 | Locomotive_1           |     28.3980810273833852 | 2025-08-18 01:29:15+02 | YES       |             | 2025-08-18 01:29:14+02 | 2025-08-18 01:29:14+02 |
 wagon_2003 | wagon       | 51.50751 | -0.12727 | Locomotive_2           |     42.3517130187421146 | 2025-08-18 01:29:15+02 | YES       |             | 2025-08-18 01:29:14+02 | 2025-08-18 01:29:14+02 |
 wagon_1001 | wagon       | 51.50839 | -0.12621 | Locomotive_2           |     19.4983557303647736 | 2025-08-18 01:29:20+02 | YES       |             | 2025-08-18 01:29:19+02 | 2025-08-18 01:29:19+02 |
 wagon_2001 | wagon       | 51.50813 | -0.12626 | Locomotive_2           |      13.9549866258880563 | 2025-08-18 01:29:20+02 | YES       |             | 2025-08-18 01:29:19+02 | 2025-08-18 01:29:19+02 |
 wagon_1003 | wagon       | 51.50821 | -0.12651 | Locomotive_1           |      42.843629691889954 | 2025-08-18 01:29:20+02 | YES       |             | 2025-08-18 01:29:19+02 | 2025-08-18 01:29:19+02 |
 wagon_1002 | wagon       |  51.5083 | -0.12636 | Locomotive_2           |     18.8104097001413855 | 2025-08-18 01:29:20+02 | YES       |             | 2025-08-18 01:29:19+02 | 2025-08-18 01:29:19+02 |
 wagon_2003 | wagon       | 51.50795 | -0.12655 | Locomotive_2           |     42.3515118995797755 | 2025-08-18 01:29:20+02 | YES       |             | 2025-08-18 01:29:19+02 | 2025-08-18 01:29:19+02 |
 wagon_2002 | wagon       | 51.50806 | -0.12638 | Locomotive_1           |      51.062806942264118 | 2025-08-18 01:29:20+02 | YES       |             | 2025-08-18 01:29:19+02 | 2025-08-18 01:29:19+02 |
 wagon_2003 | wagon       | 51.50864 | -0.12543 | Locomotive_2           |      42.3513231028017 | 2025-08-18 01:29:25+02 | YES       |             | 2025-08-18 01:29:24+02 | 2025-08-18 01:29:24+02 |
 wagon_1001 | wagon       | 51.50908 | -0.12511 | Locomotive_1           |     14.5475339401373689 | 2025-08-18 01:29:25+02 | YES       |             | 2025-08-18 01:29:24+02 | 2025-08-18 01:29:24+02 |
 wagon_1002 | wagon       | 51.50899 | -0.12525 | Locomotive_1           |     28.3976633176328273 | 2025-08-18 01:29:25+02 | YES       |             | 2025-08-18 01:29:24+02 | 2025-08-18 01:29:24+02 |
 wagon_2001 | wagon       | 51.50882 | -0.12514 | Locomotive_2           |      13.9548854626328284 | 2025-08-18 01:29:25+02 | YES       |             | 2025-08-18 01:29:24+02 | 2025-08-18 01:29:24+02 |
 wagon_1003 | wagon       |  51.5089 |  -0.1254 | Locomotive_1           |      42.8434279571103565 | 2025-08-18 01:29:25+02 | YES       |             | 2025-08-18 01:29:24+02 | 2025-08-18 01:29:24+02 |
 wagon_2002 | wagon       | 51.50873 | -0.12529 | Locomotive_2           |     28.4964683632752227 | 2025-08-18 01:29:25+02 | YES       |             | 2025-08-18 01:29:24+02 | 2025-08-18 01:29:24+02 |
 wagon_1003 | wagon       | 51.50979 | -0.12398 | Locomotive_1           |      42.842984399057443 | 2025-08-18 01:29:30+02 | YES       |             | 2025-08-18 01:29:29+02 | 2025-08-18 01:29:29+02 |
 wagon_2003 | wagon       | 51.50958 | -0.12394 | Locomotive_2           |      42.3520338990446646 | 2025-08-18 01:29:30+02 | YES       |             | 2025-08-18 01:29:29+02 | 2025-08-18 01:29:29+02 |
 wagon_2002 | wagon       | 51.50967 |  -0.1238 | Locomotive_2           |     28.4961738293811377 | 2025-08-18 01:29:30+02 | YES       |             | 2025-08-18 01:29:29+02 | 2025-08-18 01:29:29+02 |
 wagon_1001 | wagon       | 51.50997 | -0.12369 | Locomotive_1           |     14.4468588306432226 | 2025-08-18 01:29:30+02 | YES       |             | 2025-08-18 01:29:29+02 | 2025-08-18 01:29:29+02 |
 wagon_1002 | wagon       | 51.50988 | -0.12383 | Locomotive_1           |     28.3973673003210477 | 2025-08-18 01:29:30+02 | YES       |             | 2025-08-18 01:29:29+02 | 2025-08-18 01:29:29+02 |
 wagon_2001 | wagon       | 51.50976 | -0.12365 | Locomotive_2           |      13.9547472776609487 | 2025-08-18 01:29:30+02 | YES       |             | 2025-08-18 01:29:29+02 | 2025-08-18 01:29:29+02 |
 wagon_2001 | wagon       | 51.51094 | -0.12175 | Locomotive_2           |      13.9545741110842239 | 2025-08-18 01:29:35+02 | YES       |             | 2025-08-18 01:29:34+02 | 2025-08-18 01:29:34+02 |
 wagon_2002 | wagon       | 51.51085 |  -0.1219 | Locomotive_2           |      28.4958056742 4395 | 2025-08-18 01:29:35+02 | YES       |             | 2025-08-18 01:29:34+02 | 2025-08-18 01:29:34+02 |
 wagon_2003 | wagon       | 51.51076 | -0.12204 | Locomotive_2           |     42.3507511033888866 | 2025-08-18 01:29:35+02 | YES       |             | 2025-08-18 01:29:34+02 | 2025-08-18 01:29:34+02 |
 wagon_1002 | wagon       | 51.51101 | -0.12201 | Locomotive_1           |      28.3970180945904 9 | 2025-08-18 01:29:35+02 | YES       |             | 2025-08-18 01:29:34+02 | 2025-08-18 01:29:34+02 |
 wagon_1001 | wagon       |  51.5111 | -0.12186 | Locomotive_1           |     13.8580691831640 91 | 2025-08-18 01:29:35+02 | YES       |             | 2025-08-18 01:29:34+02 | 2025-08-18 01:29:34+02 |
 wagon_1003 | wagon       | 51.51092 | -0.12216 | Locomotive_1           |     42.8420109390401375 | 2025-08-18 01:29:35+02 | YES       |             | 2025-08-18 01:29:34+02 | 2025-08-18 01:29:34+02 |
 wagon_2001 | wagon       | 51.51228 | -0.11959 | Locomotive_2           |     14.5470079579236 29 | 2025-08-18 01:29:40+02 | YES       |             | 2025-08-18 01:29:39+02 | 2025-08-18 01:29:39+02 |
 wagon_2003 | wagon       |  51.5121 | -0.11987 | Locomotive_2           |     42.3502040925458 45 | 2025-08-18 01:29:40+02 | YES       |             | 2025-08-18 01:29:39+02 | 2025-08-18 01:29:39+02 |
 wagon_1001 | wagon       | 51.51239 | -0.11977 | Locomotive_1           |     14.5469895912716 57 | 2025-08-18 01:29:40+02 | YES       |             | 2025-08-18 01:29:39+02 | 2025-08-18 01:29:39+02 |
 wagon_1003 | wagon       | 51.51221 | -0.12006 | Locomotive_1           |     42.8411419455656 595 | 2025-08-18 01:29:40+02 | YES       |             | 2025-08-18 01:29:39+02 | 2025-08-18 01:29:39+02 |
```

Figure A.1: Wagon Assignments Table



```
ratmis=# select * from locomotive_assignments order by timestamp;
  vehicle_id  | latitude | longitude |                                  assigned_wagons                                  |       timestamp        | event_ts
--------------+----------+-----------+-----------------------------------------------------------------------------------+------------------------+----------
 Locomotive_1 | 51.50762 |  -0.12745 | []                                                                                | 2025-08-18 01:29:05+02 |
 Locomotive_2 |  51.5075 |  -0.12728 | []                                                                                | 2025-08-18 01:29:05+02 |
 Locomotive_1 | 51.50762 |  -0.12745 | ["wagon_1001", "wagon_2001"]                                                      | 2025-08-18 01:29:10+02 |
 Locomotive_2 |  51.5075 |  -0.12728 | ["wagon_2001", "wagon_1001"]                                                      | 2025-08-18 01:29:10+02 |
 Locomotive_2 | 51.50778 |  -0.12684 | ["wagon_2001"]                                                                    | 2025-08-18 01:29:15+02 |
 Locomotive_1 | 51.50802 |  -0.12681 | ["wagon_1001", "wagon_1002", "wagon_2001", "wagon_1003", "wagon_2002"]            | 2025-08-18 01:29:15+02 |
 Locomotive_1 | 51.50848 |  -0.12607 | ["wagon_1001", "wagon_1002", "wagon_2001", "wagon_1003", "wagon_2002"]            | 2025-08-18 01:29:20+02 |
 Locomotive_2 | 51.50822 |  -0.12612 | ["wagon_2001", "wagon_1002", "wagon_1001"]                                        | 2025-08-18 01:29:20+02 |
 Locomotive_2 | 51.50891 |    -0.125 | ["wagon_2001"]                                                                    | 2025-08-18 01:29:25+02 |
 Locomotive_1 | 51.50917 |  -0.12496 | ["wagon_1001", "wagon_1002"]                                                      | 2025-08-18 01:29:25+02 |
 Locomotive_2 | 51.50985 |  -0.12351 | ["wagon_2001"]                                                                    | 2025-08-18 01:29:30+02 |
 Locomotive_1 | 51.51006 |  -0.12354 | ["wagon_1001", "wagon_1002", "wagon_2001"]                                        | 2025-08-18 01:29:30+02 |
 Locomotive_2 | 51.51103 |  -0.12161 | ["wagon_2001", "wagon_1001", "wagon_1002", "wagon_2002", "wagon_1003", "wagon_2003"] | 2025-08-18 01:29:35+02 |
 Locomotive_1 | 51.51119 |  -0.12172 | ["wagon_1001", "wagon_2001"]                                                      | 2025-08-18 01:29:35+02 |
 Locomotive_1 | 51.51248 |  -0.11962 | ["wagon_1001", "wagon_2001"]                                                      | 2025-08-18 01:29:40+02 |
 Locomotive_2 | 51.51237 |  -0.11944 | ["wagon_2001", "wagon_1001"]                                                      | 2025-08-18 01:29:40+02 |
 Locomotive_1 | 51.51383 |  -0.11747 | ["wagon_1001"]                                                                    | 2025-08-18 01:29:45+02 |
 Locomotive_2 |  51.5139 |  -0.11699 | ["wagon_2001", "wagon_2002", "wagon_2003", "wagon_1001"]                          | 2025-08-18 01:29:45+02 |
 Locomotive_1 |  51.5154 |  -0.11495 | ["wagon_1001"]                                                                    | 2025-08-18 01:29:50+02 |
 Locomotive_2 | 51.51556 |  -0.11433 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:29:50+02 |
 Locomotive_1 | 51.51719 |  -0.11208 | ["wagon_1001", "wagon_1002", "wagon_1003"]                                        | 2025-08-18 01:29:55+02 |
 Locomotive_2 | 51.51741 |  -0.11135 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:29:55+02 |
 Locomotive_1 | 51.51913 |  -0.10896 | ["wagon_1001", "wagon_1002", "wagon_1003"]                                        | 2025-08-18 01:30:00+02 |
 Locomotive_2 | 51.51935 |  -0.10823 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:00+02 |
 Locomotive_1 | 51.52115 |  -0.10571 | ["wagon_1001", "wagon_1002", "wagon_1003"]                                        | 2025-08-18 01:30:05+02 |
 Locomotive_2 | 51.52137 |  -0.10499 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:05+02 |
 Locomotive_1 | 51.52316 |  -0.10247 | ["wagon_1001"]                                                                    | 2025-08-18 01:30:10+02 |
 Locomotive_2 | 51.52337 |  -0.10177 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:10+02 |
 Locomotive_1 | 51.52515 |  -0.09927 | ["wagon_1001"]                                                                    | 2025-08-18 01:30:15+02 |
 Locomotive_2 | 51.52537 |  -0.09856 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:15+02 |
 Locomotive_1 | 51.52715 |  -0.09606 | ["wagon_1001"]                                                                    | 2025-08-18 01:30:20+02 |
 Locomotive_2 | 51.52737 |  -0.09536 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:20+02 |
 Locomotive_1 | 51.52915 |  -0.09286 | ["wagon_1001"]                                                                    | 2025-08-18 01:30:25+02 |
 Locomotive_2 | 51.52936 |  -0.09215 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:25+02 |
 Locomotive_2 | 51.53136 |  -0.08894 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:30+02 |
 Locomotive_1 | 51.53114 |  -0.08965 | ["wagon_1001"]                                                                    | 2025-08-18 01:30:30+02 |
 Locomotive_1 | 51.53314 |  -0.08644 | ["wagon_1001", "wagon_1002", "wagon_1003"]                                        | 2025-08-18 01:30:35+02 |
 Locomotive_2 | 51.53336 |  -0.08573 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:35+02 |
 Locomotive_1 | 51.53514 |  -0.08323 | ["wagon_1001", "wagon_1002", "wagon_1003"]                                        | 2025-08-18 01:30:40+02 |
 Locomotive_2 | 51.53535 |  -0.08252 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:40+02 |
 Locomotive_1 | 51.53713 |  -0.08002 | ["wagon_1001", "wagon_1002", "wagon_1003"]                                        | 2025-08-18 01:30:45+02 |
 Locomotive_2 | 51.53735 |  -0.07931 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:45+02 |
 Locomotive_1 | 51.53913 |  -0.07681 | ["wagon_1001"]                                                                    | 2025-08-18 01:30:50+02 |
 Locomotive_2 | 51.53934 |   -0.0761 | ["wagon_2001", "wagon_2002", "wagon_2003"]                                        | 2025-08-18 01:30:50+02 |
```

Figure A.2: Locomotive Assignments Table

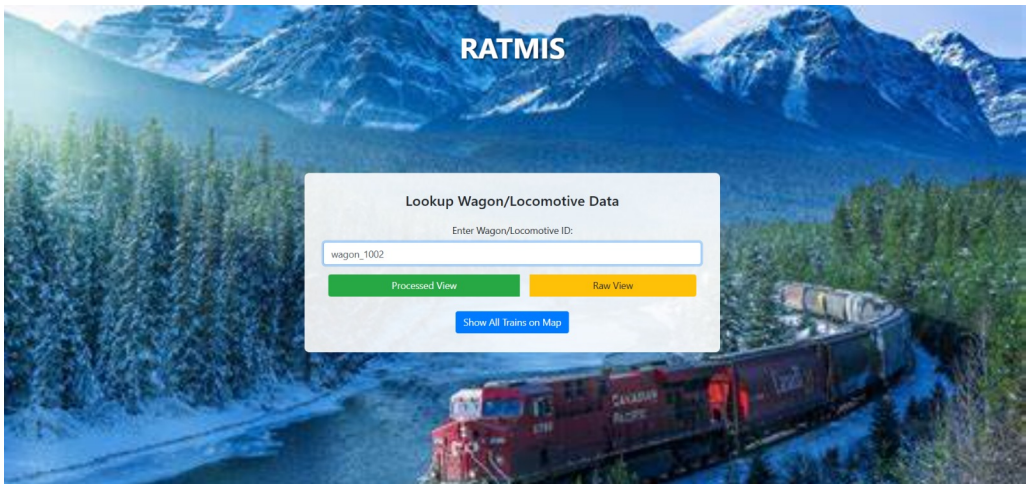


Figure A.3: User Interface Webpage

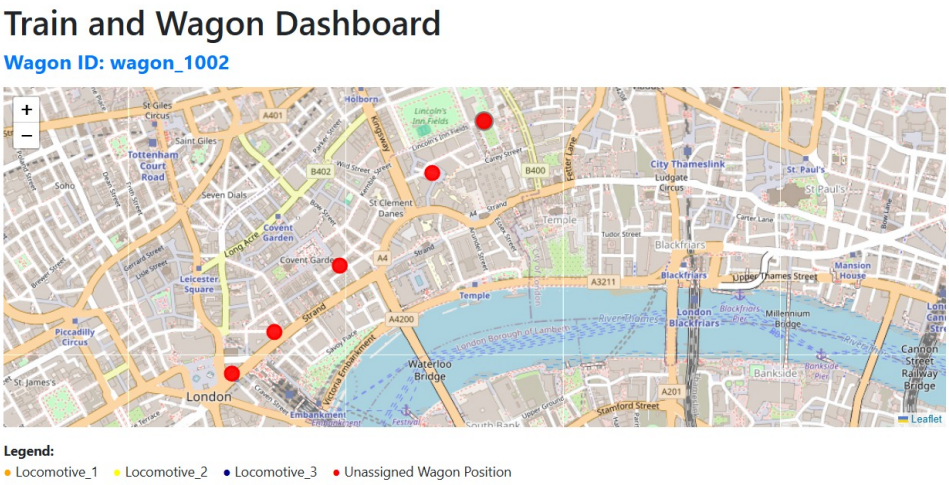


Figure A.4: Raw View Webpage

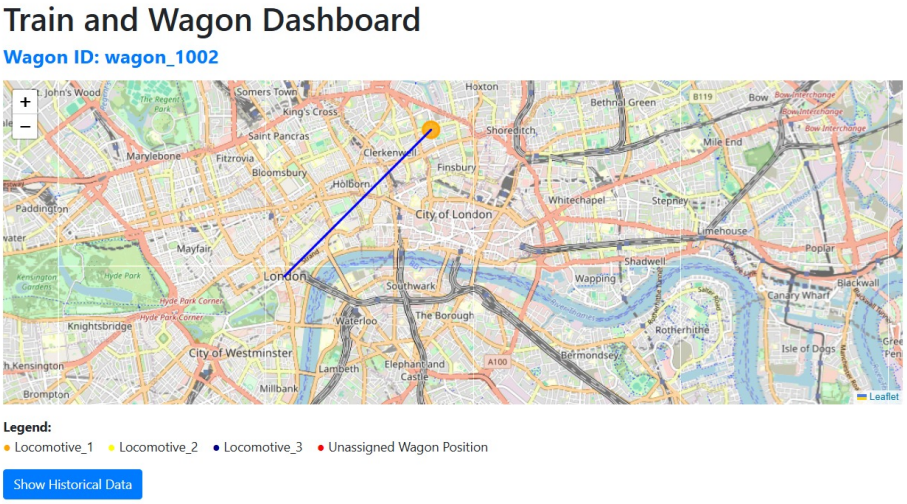


Figure A.5: Processed Individual Wagon View

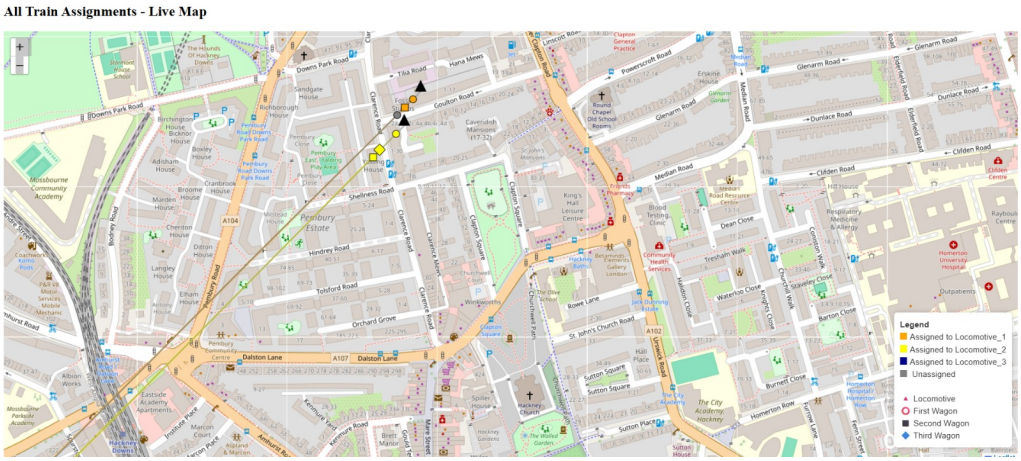


Figure A.6: All Train View

A.4 Acronyms

Acronym	Description
ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
AUPRC	Area Under the Precision–Recall Curve
CSV	Comma-Separated Values
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
GPS	Global Positioning System
ID	Identifier
JSON	JavaScript Object Notation
PR Curve	Precision–Recall Curve
RFID	Radio-Frequency Identification
RATMIS	Real-time Asset Tracking and Management Information System
UI	User Interface
UTC	Coordinated Universal Time

Table A.1: List of Acronyms used in this thesis.

A.5 Glossary

Term	Definition
Assignment Accuracy	Percentage of correct wagon–locomotive assignments
Axle Counters	Trackside devices counting wheel axles
Connected Components	Graph-based grouping of vehicles by distance
Confusion Matrix	Evaluation table comparing predictions vs. truth
Event-Driven Architecture	Asynchronous design pattern used in the system
Evaluation Metrics	Accuracy, precision, recall, F1, etc.
F1 Score	Harmonic mean of precision and recall
Flask	Lightweight Python web framework
Kalman Filter	Estimation method for smoothing and prediction
Kafka	Apache Kafka message broker for event streaming
Leaflet.js	JavaScript library for interactive maps
Locomotive Assignments Table	Database table for locomotive–wagon data
Multipath Interference	GPS error from reflected signals
Noise (GPS)	Random fluctuations in GPS accuracy
Parallel Tracks Problem	Misassignments from GPS noise across nearby tracks
PostGIS	Spatial extension for PostgreSQL
PostgreSQL	Relational database used in the system
Precision	Fraction of correct positive predictions
Recall	Fraction of true positives captured
Rigid-Body Drag	Consist modeling method that propagates locomotive movement
Simulator (GPS Data Simulator)	Module generating synthetic GPS signals
Staleness Handling	Labeling outdated GPS data
Tick Synchronization	Aligning all vehicles to fixed time steps
Wagon–Locomotive Assignment	Core process of linking wagons to locomotives
Wagon Chain	Group of wagons provisionally linked without locomotive
Flip Rate	Metric showing how often predicted locomotive changes over time
ID-Switch Rate	Metric showing spurious assignment flips when the true locomotive remains unchanged

Table A.2: Glossary of key terms used in this thesis.