SD Map Localization: A Deep Learning Approach

Using deep learning models to accurately localize vehicles on Standard Definition maps using GNSS and INS data

SAMEER JATHAVEDAN, SHEIK MEERAN RASHEED

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

SD Map Localization: A Deep Learning Approach

Using deep learning models to accurately localize vehicles on Standard Definition maps using real-time sensor data

Sameer Jathavedan, Sheik Meeran Rasheed

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2024

SD Map Localization: A Deep Learning Approach
Using deep learning models to accurately localize vehicles on Standard Definition maps using GNSS and INS data
Sameer Jathavedan, Sheik Meeran Rasheed

Supervisor: Mats Granath, Gothenburg University
Advisor: Axel Beauvisage, Zenseact
Advisor: Junsheng Fu, Zenseact
Examiner: Mats Granath, Gothenburg University

Master's Thesis 2024
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

SD Map Localization: A Deep Learning Approach
Using deep learning models to accurately localize vehicles on Standard Definition maps using real-time sensor data
SAMEER JATHAVEDAN, SHEIK MEERAN RASHEED
Department of Physics
Chalmers University of Technology

Abstract

Accurate localization is critical for the safe and efficient operation of autonomous vehicles, enabling precise navigation and real-time decision-making. This thesis focuses on improving Standard Definition (SD) map based localization, by leveraging deep learning techniques. The research addresses key questions, including how to optimally encode SD map data and sensor data, particularly the Global Navigation Satellite System and Inertial Navigation System sensor, for deep learning, and train models to perform accurate map matching and vehicle localization along the correct road segment.

The thesis develops a deep learning-based localization framework for autonomous vehicles, focusing on SD map data. It introduces three main components: a Polyline Encoder using either Graph Neural Networks (GNN) or Transformers, a Map Matching Network based on cross-attention, and a Point Prediction Network consisting of a simple Multi-Layer Perceptron. The model encodes ego trajectories and map links, matches the map data with the vehicle's trajectory, and predicts precise location. Our results show that the GNN consistently outperforms the Transformer on both map matching and point prediction. The model's performance varies based on the training and testing data used, with the last point of the trajectory often being sufficient for accurate localization. The study also compares the deep learning model with classical algorithms and finds that the GNN-based localization model significantly improves localization accuracy. Overall, our thesis demonstrates that leveraging deep learning techniques, particularly GNN-based architecture for encoding, along with cross-attention based architecture for map matching, has the potential to significantly enhance SD map localization for autonomous vehicles.

Keywords: Deep Learning, Map Localization, Transformer, Graph Neural Network.

Acknowledgements

We would like to thank and acknowledge Zenseact and our supervisors Axel Beauvisage, Junsheng Fu and Pontus Kielén for their support and guidance through all the stages of this project. We would also like to thank our academic supervisor, Mats Granath for his support throughout the thesis.

Sameer Jathavedan and Sheik Meeran Rasheed Abdul Rahuman

Gothenburg, 2024-09-26

List of Acronyms

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

AD	**A**utonomous **D**riving
ADAS	**A**dvanced **D**river **A**ssistance **S**ystems
GNN	**G**raph **N**eural **N**etwork
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
INS	**I**nertial **N**avigation **S**ystem
IMU	**I**nertial **M**easurement **U**nit
LSTM	**L**ong **S**hort-**T**erm **M**emory
MLP	**M**ulti-**L**ayer **P**erceptron
PP	**P**oint **P**redictor
RNN	**R**ecurrent **N**eural **N**etwork
UTM	**U**niversal **T**ransverse **M**ercator
WGS84	**W**orld **G**eodetic **S**ystem 1984
LiDAR	**L**ight **D**etection **a**nd **R**anging

Contents

List of Figures

List of Tables

1
Introduction

Autonomous vehicle systems consist of three essential components: perception, planning, and control. Within the perception system, one critical function is localization, which involves determining the vehicle's position within a global coordinate system [4]. Accurate localization is crucial for the safe and efficient operation of autonomous vehicles. A vehicle must have a precise understanding of its location to make informed decisions about navigation, obstacle avoidance, and interaction with elements in its environment. Improved localization enhances the vehicle's ability to make accurate predictions about its surroundings and respond to dynamic changes in real-time. This precision directly affects the performance of the planning and control systems, reducing the likelihood of collisions, improving route efficiency, and ensuring passenger safety. Moreover, better localization is vital in urban environments with complex road networks, where small errors in positioning can lead to significant deviations from the intended path.

Research in localization has given rise to various specialized fields, such as Simultaneous Localization and Mapping (SLAM), which operates without prior knowledge of the environment, and other approaches that rely on pre-existing maps for vehicle localization. This thesis examines map-based localization – and particularly – a subset of map-based localization known as Standard Definition (SD) map localization.

Maps used in autonomous vehicle localization vary significantly in detail, accuracy, and purpose. The most commonly used maps are High Definition (HD) maps and Standard Definition (SD) maps. High Definition Maps are extremely detailed and contain centimeter-level accuracy information, including precise lane boundaries, road curvature, traffic signs, and other essential data for controlling the vehicle within its lane. These maps are highly accurate because it is mainly used for controlling the vehicle in real-time. In contrast, Standard Definition Maps offer a lower level of detail, typically focusing on broader road features, such as road geometry, number of lanes, speed limit, traffic direction, and other road-level data. Because of the global level of information stored in the SD maps, they have large coverage which can be used for localization in the context of planning, which is done by retrieving road-level features of what is ahead. Thus HD and SD maps serve complementary roles in autonomous vehicle systems, with HD maps providing the precise, real-time control needed for accurate in-lane positioning, while SD maps offer essential road-level data that aids in broader planning and decision-making tasks. With the distinctions between HD and SD maps clarified, the focus of this thesis will be on SD maps. The objective is to achieve accurate vehicle localization by leveraging the

global-level information provided by SD maps, rather than HD maps. This will be accomplished by exploring the use of deep learning methods to enhance localization accuracy within the context of SD maps.

Deep learning offers significant potential in addressing the challenges associated with SD map-based localization. Traditional algorithms for localization often struggle with the variability and lower resolution inherent in SD maps, particularly in complex urban environments where the road geometry can be intricate. Additionally, classical models may face limitations when integrating data from multiple sensors, as they typically rely on predefined rules and heuristics, making it challenging to accommodate additional sensor inputs without extensive re-engineering. In contrast, deep learning models could be more adept at handling and fusing data from multiple sensors, as they are designed to learn complex correlations and patterns directly from the data. This adaptability suggests that deep learning models may be better suited to leveraging sensor data for improving localization accuracy.

By leveraging large datasets, deep learning has the potential to enhance the accuracy of map matching and location prediction within a link, even when detailed map information is sparse. Furthermore, deep learning approaches could adapt more readily to new environments and variations in road conditions, making them potentially more robust against the uncertainties that typically challenge SD map-based localization. As a result, deep learning could significantly enhance the overall effectiveness of localization in autonomous vehicle systems.

Map-based localization presents several challenges that must be addressed to ensure accurate vehicle positioning. The two primary problems in this domain are map-matching and predicting the location within a link.

Map matching is the process of associating the vehicle's current position with a specific location on the map, typically referred to as a "link." A link in SD maps represents a segment of the road or a specific part of the network. In the context of SD maps, links are fundamental building blocks that represent road segments or paths within the map. These links are crucial for navigation, as they define the possible routes that the vehicle can take. Correctly identifying which link the vehicle is on is vital for accurate localization, especially in environments with complex road networks or in situations where the GPS signal may be weak or unreliable.

Once the correct link is identified, the next challenge is determining the vehicle's exact location within that link. This involves calculating the precise position of the vehicle relative to the link's start point. Factors such as speed of the vehicle, road curvature, and the direction of traffic in the link can complicate this process. Accurate prediction of the vehicle's position within a link is essential for ensuring that the vehicle follows the correct path and makes appropriate navigational decisions.

1.1 Thesis Objectives

The research questions that this thesis attempts to answer are:

1. What is the optimal approach for encoding Standard Definition (SD) Map data to enhance its interpretability by deep learning models?
2. How can multi-modal sensor data from the vehicle be effectively integrated with the encoded SD Map data to facilitate comprehension by deep learning models?
3. What methodologies are effective for training the deep-learning model to accurately identify the most likely road segment currently navigated by the vehicle, based on this enhanced data representation?
4. Further, how to accurately predict the location of the vehicle along the correct road segment?

1.2 Limitations

To define the scope of the thesis, certain limitations and assumptions are taken into account. This research primarily focuses on using geo-positional data from GNSS sensors and inertial data from INS sensors to identify the correct link on the SD map and determine the vehicle's position along that link. While it is recognized that additional sensors, such as cameras, could improve localization performance, the study intentionally limits itself to these core data sources. This choice is based on the main goal of exploring whether deep learning models can be effectively used for localization, rather than expanding on existing research. By keeping this focused approach, the research aims to establish a basic understanding of how well deep learning can work for vehicle localization with essential sensor data, setting the stage for future, more detailed studies.

2

Background

2.1 Map Localization

While there exist numerous surveys on map localization in the literature, their contributions and insights are closely tied to the specific sensor configurations they investigate. For example, the authors in Kuutti et al. [19] categorize localization methods based on sensor classification without addressing whether prior maps are used. In contrast, Elhousni and Huang [9] is restricted to studies utilizing only LiDAR sensors. The work presented in Bresson et al. [2] extensively examines the subject from a SLAM perspective, focusing solely on techniques that construct and reuse long-term maps, including those that do not use prior maps. Surveys like Kubicka et al. [18] and Chao et al. [6] focus on the navigation and routing aspects of vehicular map-matching methods and propose a novel classification method, yet all surveyed approaches rely exclusively on the lane-level layer of prior maps. A more recent review, found in Laconte et al. [20], introduces a new taxonomy of localization methods for autonomous vehicles, though it is limited to highway scenarios.

HD Map localization is a widely researched area, particularly for lane detection and navigation in autonomous vehicles. Comprehensive surveys like Chalvatzaras et al. [5] highlight recent advancements in HD maps, sensors, datasets, and algorithmic solutions across various driving scenarios . Studies such as Ma et al. [23] introduce novel semantic localization algorithms using multiple sensors and sparse semantic HD maps to achieve high accuracy , while Cai et al. [3] present a cost-effective localization method combining a custom GPS, monocular camera, and HD maps to reduce errors . Additional research, such as Bauer et al. [1], demonstrates the significant impact of HD maps on localization accuracy, using a particle filter to combine Inertial Measurement Unit and GPS data. Furthermore, the study by Han et al. [13] introduces an ego-motion estimation technique that improves robustness and accuracy through a novel line segmentation matching model and geometric correction using inverse perspective mapping (IPM).

For SD maps, although the literature provides extensive information on map matching, no studies have been identified that address precise localization within SD maps. The topic of map matching will be explored further in the next section.

2.2 Map Matching

Today's automobile routing systems make use of positioning sensors such as the Global Navigation Satellite System (GNSS) in order to track the user's position on a map. For effective routing, it is important to know which road the user is currently on. The process of continually estimating a user's position on a road segment is known as map matching [14].

2.2.1 Classical Approaches

Map-matching was initially approached as a geometric problem, and most solutions used curve-matching techniques. The earliest, by Kim et al. [16], was based on the shape of road segments. It was simple but inaccurate at intersections. White et al. [35] and Taylor et al. [33] developed more advanced algorithms, incorporating direction, topological information. Some algorithms, like Srinivasan et al. [31], integrated GPS and dead reckoning outputs with a Kalman filter. However, these models were too simple for dense urban networks and had limitations like overlooking more congruent directions and being computationally intensive. The challenge remains to improve accuracy, especially in identifying the correct segment in dense urban areas.

Further advancements in map-matching were made with the use of more Kalman Filters, fuzzy logic and probabilistic approaches. Some of the developments include a multiple-hypothesis technique by Pyo, Shin, and Sung [25] and an algorithm based on Dempster–Shafer theory by Yang, Cai, and Yuan [7]. Both approaches have limitations, such as high computational intensity and lack of use of heading and direction information.

The Hidden Markov Model (HMM) is introduced as a probabilistic model involving uncertain state transitions, with states being road segments and transitions governed by road network connectivity. Ren and Karimi [28] and Newson and Krumm [24] explored the use of HMM in map-matching, but their methods also had limitations, such as response delay and lack of use of heading or direction data.

2.2.2 Deep Learning Approaches

In order to leverage the data collected from trajectories, deep learning models were proposed.

2.2.2.1 Transformer-based Approaches

DeepMM [10] was the first study dedicated to the task of matching GPS trajectories to corresponding road segments, drawing inspiration from previous research on deep neural network models. It adopted an encoder-decoder system, akin to those used in neural machine translation with recurrent neural networks. Addressing the challenge of input trajectory and road matched trajectory having varying lengths, DeepMM incorporated attention mechanisms to complement the sequence-to-sequence model.

However, DeepMM faced limitations related to sensitivity to data quality and quantity. To overcome these challenges, the proposed Learning to Map Matching (L2MM) [15] introduces a robust deep learning-based model that specifically addresses issues associated with low-quality GPS trajectories and limited training data. L2MM further enhances trajectory representations, incorporates historical trajectory patterns, and employs pattern recognition to improve generalization, showcasing superior accuracy and efficiency in extensive experiments.

Transformer-based architectures [34] have been widely used for sequence modeling tasks, with models like BERT [8] applying transformer encoders to generate contextual embeddings for words in a sequence. Similarly, transformer encoders can be employed for encoding sequential data, such as polylines. Luo et. al [22] presents a framework using SD maps and a Transformer-based encoder to significantly enhance lane-topology prediction, offering a scalable alternative to traditional HD maps. This approach has informed the decision to incorporate a Transformer encoder for encoding SD map information in this thesis.

2.2.2.2 Graph Neural Network based Approaches

Graph Neural Networks (GNNs) were first introduced by Scarselli et al. in 2009 [29], which generalized neural networks to graph-structured data through a recursive, message-passing framework that iteratively propagates information across the graph until convergence. Later, Kipf and Welling [17] introduced Graph Convolutional Networks (GCNs) in 2017, improving efficiency by enabling a layer-wise, convolutional propagation mechanism over graphs. This advancement significantly boosted the applicability of GNNs to various domains, including tasks like node and graph classification.

GNN based models such as VectorNet [12] have been developed for vehicle trajectory prediction. These models adopt a graph-based representation of the environment, where each road segment, intersection, and other pertinent structures are depicted as nodes within a graph. Notably, the vehicle itself is also encoded as a node within this graph. By constructing such a graph, VectorNet effectively captures the intricate spatial relationships between the vehicle and its surrounding elements, enabling accurate trajectory forecasting. Following the development of VectorNet, additional trajectory prediction models have emerged, with one notable example being the TNT prediction model [36]. Leveraging VectorNet's capabilities, TNT utilizes a similar approach to encode scene context between the agent and its surrounding road elements within a graph-based representation. By assessing the likelihood of potential future trajectories based on the agent's position and combining this information with the context vector of the agent, TNT achieves accurate trajectory prediction.

Drawing inspiration from the VectorNet model, our thesis leverages a similar framework as the foundation for our model. Our approach involves representing our vehicle's multi-modal input data, including GNSS trajectory and INS data, as a node within the graph. Additionally, neighboring road segments are represented as

nodes in the same graph. By constructing this graph-based representation, we aim to exploit the spatial dependencies and contextual information encoded within the graph to predict the correct road segment on which the vehicle is currently positioned. In this way we can develop a generalized model capable of operating across various locations.

2.3 Localization Regulations and Standards

Safety standards for localization in autonomous vehicles require achieving an integrity level with a 10^{-8} probability of failure per hour, specifying lateral and longitudinal error bounds based on US road geometry standards, such as a lateral error of 0.57 meters (0.20 meters at 95% confidence) and a longitudinal error of 1.40 meters (0.48 meters at 95% confidence) [27]. Various standards, including those from SAE International, 5G PPP, and the European GSA, propose different accuracy and integrity levels for AVs, with accuracy requirements ranging from 0.1 to 0.3 meters and varying integrity levels, but there is no universal consensus on a standard, as highlighted by recent research [26]. Currently, no clear consensus exists for standards within SD maps; however, for this thesis, the minimum acceptable positioning error for SD map localization is set at 9 meters.

Evaluation methods for localization performance are categorized into two approaches: one comparing against a high-quality reference position trajectory, which may not be a true ground-truth, and another using static, map-based ground-truths such as pre-defined paths or HD maps, which introduce their own chart errors and challenges in accuracy assessment [26]. Despite these methods, no standardized approach exists for evaluating localization performance against a validated ground-truth dataset in real-world driving scenarios.

2.4 Terminology

In the context of this thesis, we adopt precise definitions to avoid ambiguities related to the term *map matching*. The following terms are defined as follows:

1. **Map Link:** A map link, denoted as $\mathcal{L}$, is a polyline representing a road segment, characterized by:
   - **Coordinates:** A set of geographic coordinates $\{(x_i, y_i)\}_{i=1}^{n}$ where each coordinate (x_i, y_i) corresponds to a point on the polyline.
   - **Direction of Traffic:** A directional attribute D indicating the permissible travel direction on the road segment.
   - **GeoID:** A unique identifier ID assigned to each map link, $\mathcal{L}$.
2. **SD Map:** An SD map, denoted as $\mathcal{M}$, is a graph comprising a set of connected map links $\{\mathcal{L}_i\}_{i=1}^{m}$, where each map link $\mathcal{L}_i$ is connected to others according to a connectivity matrix $\mathbf{C}$.
3. **Map Matching:** The map matching process, MM, involves selecting the appropriate map link $\mathcal{L}_j \in \mathcal{M}$ based on a sensor vector $\mathbf{S}$ that includes multiple

attributes of the vehicle's current state. The sensor vector $\mathbf{S}$ may include attributes such as position (x, y), speed v, heading ϕ, and other relevant metrics. Mathematically, the map matching function can be represented as:

$$\text{MM}(\mathbf{S}) = \mathcal{L}_j \quad \text{where} \quad \mathcal{L}_j \text{ is selected based on criteria } \mathcal{C}(\mathbf{S}, \mathcal{L}_j),$$

where $\mathcal{C}(\mathbf{S}, \mathcal{L}_j)$ is a composite function that evaluates the suitability of each map link $\mathcal{L}_j$ relative to the sensor vector $\mathbf{S}$. This function incorporates factors such as the vehicle's position, speed, heading, and possibly other dynamic attributes to determine the most likely road segment.

In this thesis, the SD map $\mathcal{M}$ include long roads that are divided into multiple road segments. Thus, the map matching process, MM, is specifically designed to identify the road segment $\mathcal{L}_j$ on which the vehicle is currently located, based on a comprehensive analysis of the sensor data.

2.5 Data

The dataset used consists of multiple samples with historical sensor information collected over various road types. Each sample consists of the following information:

1. **Sensor Input from the Ego Vehicle:**
   (a) **GNSS Coordinates**: We use the last 30 seconds of the vehicle's GNSS trajectory, converted to the UTM frame and rotated to align with the vehicle's heading for easier localization.
   (b) **INS Data for Each GNSS Coordinate**:
      i. Vehicle Velocity
      ii. Vehicle Heading
   (c) **Timestamp** for each GNSS coordinate.
2. **SD Map Links within the Given Frame**
   (a) We use a world map from OpenStreetMap to obtain the nearest map links to the last point in the historical GNSS trajectory, limited by the frame width to balance variation and data usability.
   (b) **Map Link Data**:
      i. **GeoID**: A unique identifier for each map link.
      ii. **Coordinates**: Geodetic coordinates of the points in the map link polyline.
      iii. **Attributes**:
         A. **GeoID**: Unique identifier for every map link in the world, assigned arbitrarily.
         B. **Direction**: Indicates the direction of travel along the map link. The possible values are: 0 for traffic open in both directions, 1 for traffic from the starting point to the ending point, 2 for traffic from the ending point to the starting point, and 3 for traffic closed in both directions.
3. **Ground Truth**
   (a) **Road Segment ID**: The GeoID for the correct map link.
   (b) **Offset**: This is a term used for the distance between the vehicle's current position and the start of the link.

Figure 2.1 is an illustration of the input which is given to the model.

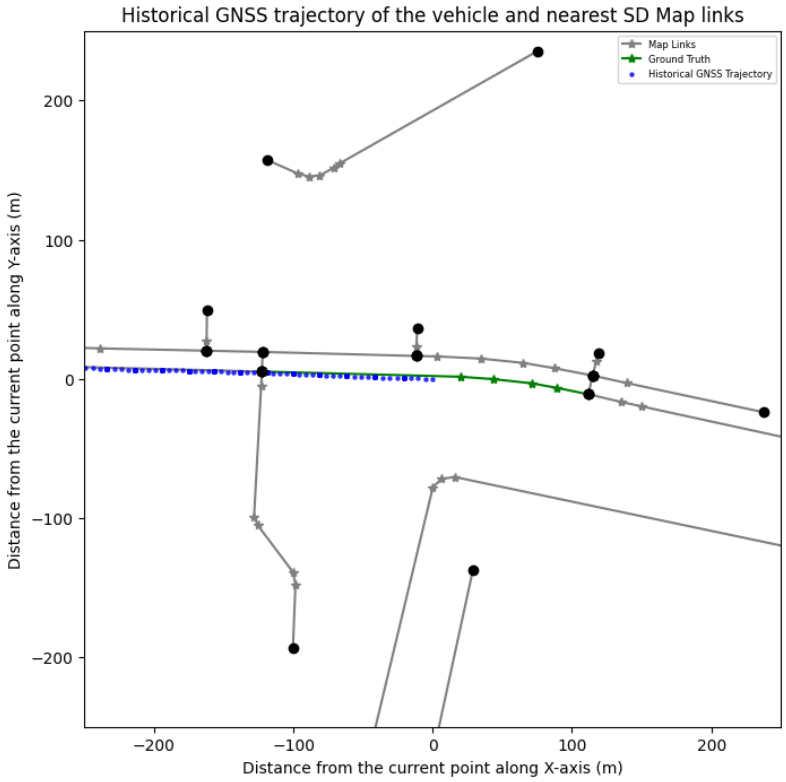

Figure 2.1: Example of an input sample. The grey polylines represent the map links. The black dots are the start and end of each map link and the grey points are the different points within the polyline. The blue points show the historical ego trajectory that is obtained from the vehicle. The green link is the ground truth map link.

The grey lines represent the different map links in the frame, where the black dots are delimitation's of the map links. The blue dotted line represents the historical GNSS ego trajectory. The green line represents the ground truth, i.e., the map link the vehicle is currently on. Other information such as INS data and direction ID are given to the model as additional parameters.

In addition to the provided information, it is crucial to consider the dataset size for the analysis. The original dataset, labeled as "v1," was further refined into two additional versions: v0 and v2. The difference between these versions are as follows:

- The v0 version serves as a subset used for preliminary experiments. It is identical to the v1 version with respect to features and labels, with no augmentation done.
- The v1 version is the original dataset, containing all the information given above.
- The v2 dataset incorporates extensive data augmentation to enhance model robustness.

Table 2.1 summarizes the distribution of training and testing samples across these

dataset versions, and the details of augmentation from v1 to v2 is elaborated upon in Section 3.6.

Dataset Version	**Number of Training Samples**	**Number of Testing Samples**
v0	2,504	300
v1	118,096	15,253
v2	934,901	120,105

Table 2.1: Summary of datasets used in the experiments, showing the number of training and testing samples for each dataset.

3
Methods

3.1 Polyline Encoding

The ego trajectory and map links are represented as sequences of geodetic coordinates with variable lengths. Prior to processing by the model, these sequences are transformed from the geodetic coordinate system (WGS84) to a local Cartesian coordinate system (UTM). This transformation removes geographic constraints, allowing the model to operate independently of location. The frame transformation is visualized in Fig. 3.1.

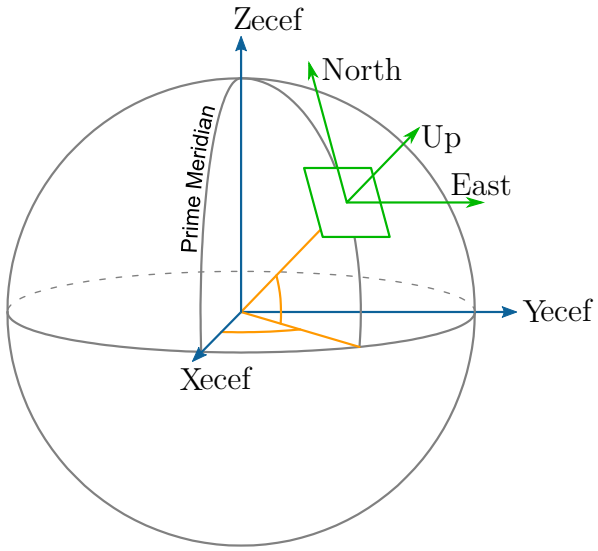

Figure 3.1: Conversion from Geodetic to Cartesian frame: The ego trajectory points and the map link coordinates, given in WGS84 reference frame (blue), are converted to a local Cartesian frame (green) using the UTM transformation.

Additionally, to make the input spatial features invariant to the locations of ego vehicle, we rotate the frame so that the heading of the ego vehicle is always zero - this means that in the top view frame of the map, the vehicle will always face towards the right. Further, we normalize the coordinates of all points to be centered around the location of ego vehicle at its last observed GNSS point. Hence, for the model, a vehicle moving along stationary map links would appear as if the vehicle is stationary and always at the origin, whereas all the map links keep moving, since their co-ordinates are changing with respect to the origin.

While a GNSS trajectory can be seen as a time series and map links are purely spatial, they can be both be geometrically represented by polylines. These variable-length sequences must be encoded into fixed-dimensional vectors. The polyline encoders are specifically designed to process the ego trajectory and map link polylines, regardless of their differing lengths, and transform them into embedded vectors with a consistent dimension. These embeddings are then utilized in subsequent modules for tasks such as map matching and point prediction.

The polyline encoder is engineered to process GNSS sequences from both the ego trajectory and the map links. Given that these sequences represent the same type of entity, a polyline, it was determined that employing a unified shared encoder would be more effective than utilizing separate encoders. [11]

We experimented with two different types of architectures to encode the polylines - a Graph Neural Network based encoder, and a Transformer based encoder.

3.1.1 Graph Neural Network based Encoder

Since the ego trajectory and map links are a sequence of GNSS coordinates, we take inspiration from the VectorNet model [12] and represent the sequences as polylines with multiple control points, thereby representing the whole frame as sets of vectors.

We use graph neural networks (GNNs) to incorporate these sets of vectors. we create a graph for each polyline and each vector as a node in the graph, and set the node features to be the start location and end location of each vector, along with other attributes such as Direction ID and INS/Map link data. We then aggregate the information of all the nodes in the graph to give out an embedded vector which encomposses the information of the whole polyline.

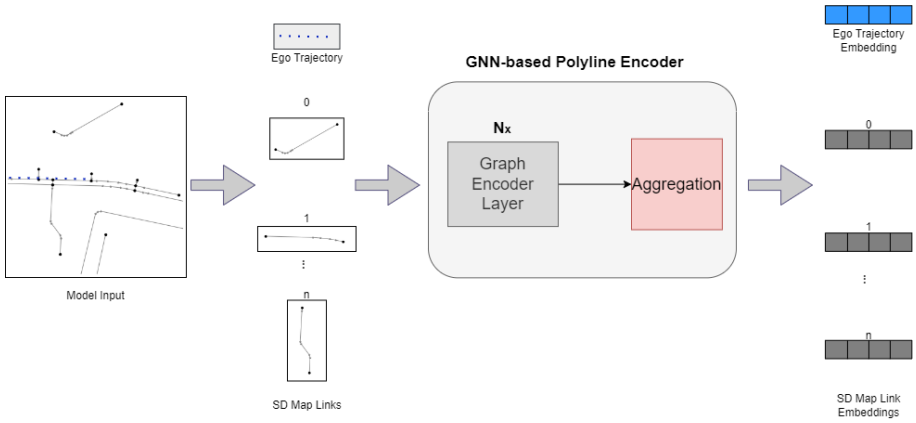

Figure 3.2: GNN-based Polyline Encoder: Each polyline is treated as a graph which passes through the same encoder

1. **Representing ego trajectory and map links as polylines:**
   Our vectorization process establishes a one-to-one mapping between continuous trajectories, and the vector set, even though the vector set is unordered.

This mapping allows us to construct a graph representation from the vector sets, which can then be encoded using graph neural networks. Specifically, we represent each vector $\mathbf{v}_i$ that belongs to a polyline $\mathbf{P}_j$ as a node in the graph. The node features are defined as

$$\mathbf{v}_i = [\mathbf{d}_i^s, \mathbf{d}_i^e, \mathbf{a}_i, j],$$

where $\mathbf{d}_i^s$ and $\mathbf{d}_i^e$ are the coordinates of the vector's start and end points. The coordinates $\mathbf{d}$ can be expressed as (x, y) for 2D coordinates. The attribute features $\mathbf{a}_i$ include information such as ego vehicle INS data, timestamps for trajectories, or road features like type or speed limit. The integer j is used for the direction ID which represents the traffic direction. Direction ID is particularly useful for map links, we represent them with unique integers for each traffic direction. They are as follows:

(a) **0**: Traffic is open both directions.
(b) **1**: Traffic From the starting node $\mathbf{d}_i^s$ towards the ending node $\mathbf{d}_i^e$.
(c) **2**: Traffic from the ending node $\mathbf{d}_i^e$ towards the starting node $\mathbf{d}_i^s$.
(d) **3**: Traffic is closed both directions.

2. **Polyline Graph Encoder Layer:**
To encapsulate the information inherent in a polyline, we construct a graph encoder at the vector level, interlinking all vector nodes belonging to the same polyline. Consider a polyline $\mathbf{P}$ represented by its nodes $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_P\}$. The propagation of information through a single layer of the graph encoder is expressed mathematically as:

$$\mathbf{v}_i^{(l+1)} = \phi_{\text{rel}}\left(g_{\text{enc}}(\mathbf{v}_i^{(l)}), \phi_{\text{agg}}\left(\{g_{\text{enc}}(\mathbf{v}_j^{(l)})\}\right)\right)$$

In this equation, $\mathbf{v}_i^{(l)}$ signifies the feature vector of node i at the l-th layer of the encoder, with $\mathbf{v}_i^{(0)}$ denoting the initial features of node i. The function $g_{\text{enc}}(\cdot)$ is responsible for encoding the features of individual nodes, while $\phi_{\text{agg}}(\cdot)$ aggregates the information from neighboring nodes, and $\phi_{\text{rel}}(\cdot)$ serves as the relational operator that connects node $\mathbf{v}_i$ with its neighbors.
Furthermore, the encoding function $g_{\text{enc}}(\cdot)$ is realized through a multi-layer perceptron (MLP) that employs shared weights across all nodes. This MLP consists of a single fully connected layer, followed by layer normalization and the application of a ReLU activation function. The aggregation function $\phi_{\text{agg}}(\cdot)$ utilizes max-pooling to gather information, while the relational function $\phi_{\text{rel}}(\cdot)$ operates through concatenation. This procedure is visually represented in Figure 3.3.
We can stack multiple layers of this graph encoder architecture, each featuring distinct weights for the encoding function $g_{\text{enc}}(\cdot)$. To derive features at the polyline level, we calculate:

$$\mathbf{p} = \phi_{\text{agg}}\left(\{\mathbf{v}_i^{(L_p)}\}\right)$$

where the aggregation function $\phi_{\text{agg}}(\cdot)$ again employs max-pooling to summarize the node features across the polyline.

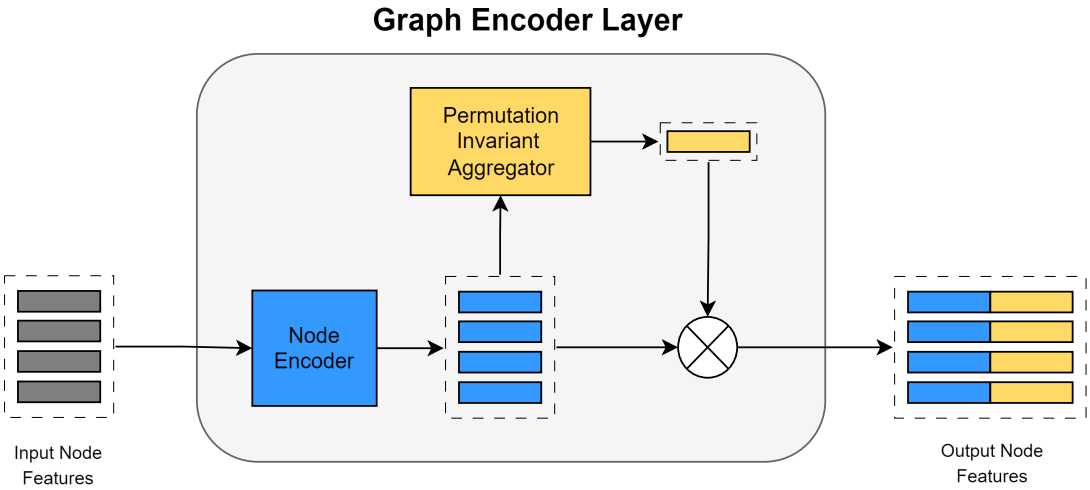

Figure 3.3: Information Aggregation within the same polyline done in one graph encoder layer as per the VectorNet approach. For the node encoder, a multi-layer perceptron is used and the permutation invariant aggregation is implemented using max-pooling. Finally we combine this aggregated information with the original node features by simple concatenation.

We adopt the VectorNet [12] graph convolution layer over the standard graph convolution layer [17] due to its optimization for encoding vector-based polyline representations. The relational and aggregation mechanisms in VectorNet have been shown to effectively capture the spatial relationships inherent in polyline data for trajectory prediction, making it an excellent choice for map localization.

Additionally, we employ concatenation as the aggregation strategy in the graph encoder layer instead of traditional methods such as mean or sum pooling. This choice facilitates the preservation of individual node feature information, thereby maintaining the distinct contributions of neighboring nodes. Such an approach enhances the representational capacity of the network, particularly in scenarios where unique node attributes are critical for the output.

3.1.2 Transformer based Encoder

As an alternative to the GNN based encoder, we explore the use of the transformer architecture to encode the sequence of geodetic coordinates. We choose the transformer because it is a valid approach for encoding sequential data. We choose the transformer architecture over other sequence modeling architectures such as RNNs and LSTMs and GRUs because of its ability to attend to long range sequences without loss of information.

We convert the ego trajectory and maplinks into polylines similar to the GNN based encoder. After this, the aggregation of the polyline information is done by a transformer encoder.

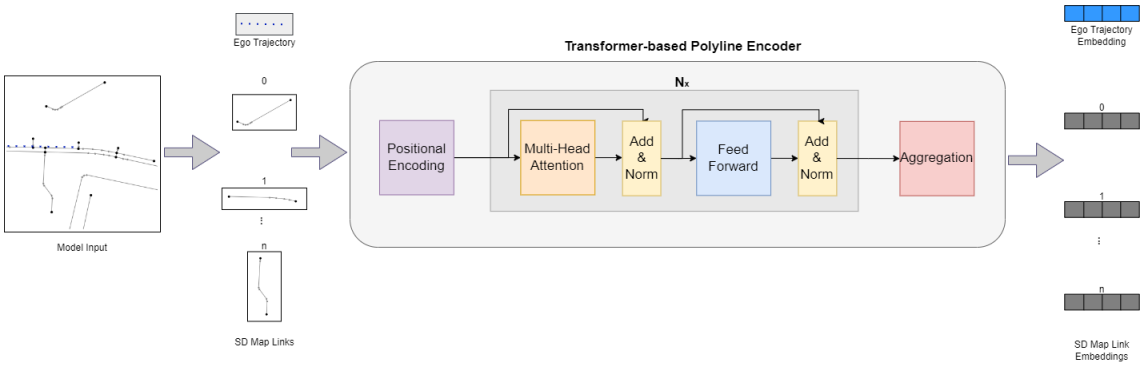

Figure 3.4: Transformer-based Polyline Encoder

The transformer-based polyline encoder is shown in the figure above. The sequence of edge vectors of the polylines are taken as input embeddings, on which we apply sinusoidal positional encoding, and then the attention mechanism. Finally we aggregate the mutually attended polyline edge embeddings using a pooling mechanism in order to aggregate the information of the sequence into a single fixed dimensional embedding vector. Mean pooling and Max pooling are explored in our experiments.

3.2 Map Matching

With the ego trajectory and map links now encoded into fixed-dimensional vectors, we approached the first part of our problem: determining whether a neural network can be trained to identify the specific road on which a vehicle is currently travelling, given the encoded ego trajectory and the various roads present on the map.

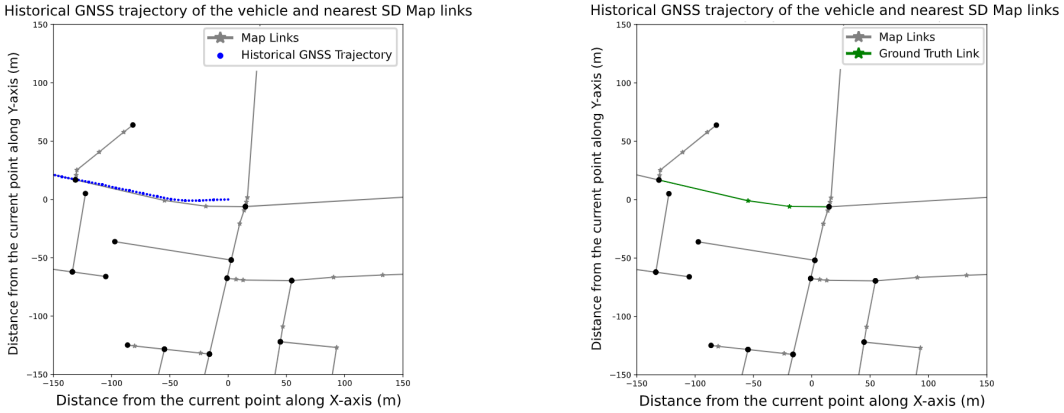

Figure 3.5: Map Matching: Inputs to the model (left) and Ground Truth (right). The blue sequence (left) is the historical vehicle trajectory, the vectors of which consist of GNSS and INS sensor data. The green highlighted link (right) indicates the ground truth map link. This is what the map matching model should predict.

3.2.1 Map Matching Based on a Fixed Number of Map Links

Initially, we focused on a map matching task where the input to the network consisted of the encoded ego trajectory along with five map links. We trained a Multi-Layer Perceptron (MLP) to see if it could accurately classify the map link corresponding to the road the vehicle is currently on. Through this experiment, we gained valuable insights into the network's ability to comprehend the map's spatial information and make informed decisions accordingly. The results (available in section 4.2.1) indicated that the MLP was indeed capable of processing the encoded data and distinguishing the correct map link.

3.2.2 Map Matching with a Variable Number of Map Links

Although the previous method demonstrated success, it is inherently limited by the fixed number of inputs required by the MLP. In real-world scenarios, the number of roads within a given ego frame can vary significantly. Therefore, it became necessary to design a network that can accommodate a variable number of input map links. To address this challenge, we explored and tested two different solutions: Cosine Similarity, and Cross Attention.

3.2.2.1 Cosine Similarity

In the first approach, we utilized cosine similarity to measure the similarity between the embedded ego trajectory and the various map links. Cosine similarity was chosen as the metric for this approach because it effectively measures the directional alignment between two vectors, making it well-suited for comparing the ego trajectory with various map links. Unlike other distance-based metrics, cosine similarity is scale-invariant, meaning it focuses solely on the angle between vectors rather than their magnitude, which is particularly useful when the trajectory and map link embeddings may have different scales. Additionally, this method allows for the independent comparison of each map link to the ego trajectory, accommodating scenarios with a variable number of map links without requiring a fixed input size. The process involved calculating cosine similarity scores between the ego trajectory embedding and each map link embedding, after which the map link with the highest similarity score was selected as the most likely candidate for the road the vehicle was on.

While this method is straightforward, it has a notable limitation: cosine similarity itself is not a trainable metric. For this approach to work effectively, the polyline encoder would need to be trained to produce embeddings such that the ego trajectory is encoded in a way that it closely matches the correct map link. However, this becomes particularly challenging because the ego trajectory typically represents a time span of around 30 seconds, during which the vehicle could traverse multiple roads. This variability makes it difficult to ensure that the ego trajectory is encoded in close proximity to any single map link.

3.2.2.2 Cross Attention Network

To overcome the limitations of the cosine similarity approach, we explored the use of a Cross Attention Network. The idea for employing cross attention in our map matching task is inspired by its application in feature matching, where cross attention is used to infer latent alignments between different data modalities, such as image regions and text descriptions, enabling fine-grained matching [21]. Similarly, in our approach, cross attention allows us to compute attention scores between the ego trajectory and map links, facilitating a more dynamic and context-sensitive matching process.

One significant advantage of this method over cosine similarity is that it is trainable. The network can be optimized during training to improve its ability to focus on the most relevant map link given the ego trajectory. This trainability allows the network to adapt its attention mechanism to better handle the complexities of the input data.

The detailed workings and advantages of the Cross Attention Network are elaborated upon in the following sections.

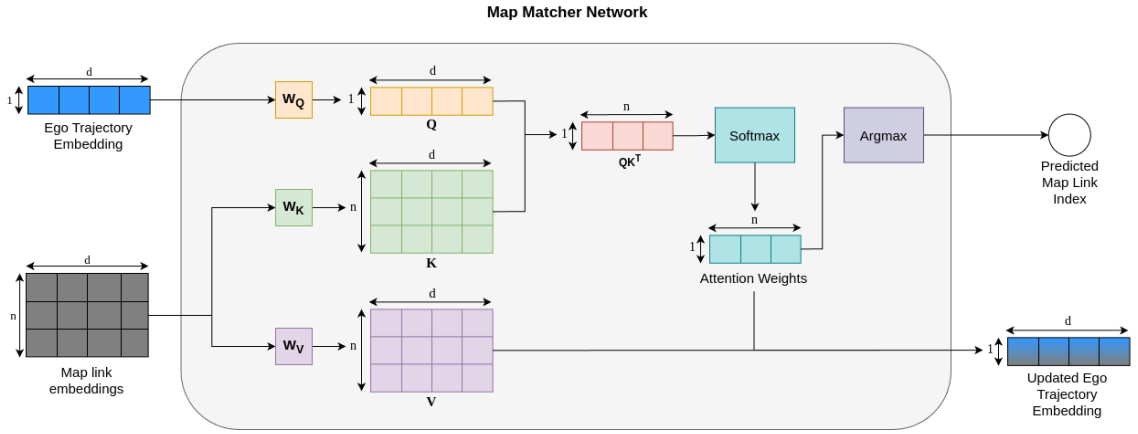

Figure 3.6: Cross attention for map matching

The cross attention is used for matching the ego trajectory with relevant map links, allowing for precise determination of the vehicle's current position on the map. This process involves several key steps, as illustrated in the accompanying diagram (Figure 3.6).

3.2.2.3 Input: Ego Trajectory and Map Links

The process begins with the input of the embedded ego trajectory and map links. The **ego trajectory** represents the vehicle's path and is embedded into a high-dimensional feature space, capturing dynamic attributes such as position, velocity, and heading. Simultaneously, **map links**, representing various segments of the road network, are also embedded into a compatible high-dimensional space, encapsulating static features like geometry, connectivity, and traffic rules.

3.2.2.4 Cross Attention Mechanism

At the core of this process is the cross attention mechanism, which facilitates the dynamic alignment of the ego trajectory with the most relevant map links:

- **Query (Q)**: The embedded ego trajectory acts as the query, seeking relevant information from the map links.
- **Keys (K) and Values (V)**: The embedded map links serve as both the keys and values. Keys help in determining the relevance of each map link, while values provide the actual information to be incorporated with the ego trajectory.

3.2.2.5 Attention Calculation

The cross attention mechanism computes attention scores by evaluating the dot product of the query ($\mathbf{Q}$) with each key ($\mathbf{K}$). This operation measures the similarity between the ego trajectory and each map link. The computed scores are then normalized using a softmax function, converting them into probabilities that represent the attention each map link receives. This can be mathematically represented as:

$$\text{Attention Scores} = \text{softmax}\left(\frac{\mathbf{Q} \cdot \mathbf{K}^\top}{\sqrt{d_k}}\right)$$

where $\mathbf{Q}$ is the query matrix (ego trajectory), $\mathbf{K}$ is the key matrix (map links), and d_k is the dimensionality of the keys. The softmax function ensures that the attention scores sum to 1, highlighting the relevance of each map link to the ego trajectory. This step is essential for determining the focus on specific map links based on the ego trajectory's context (illustrated by the interaction arrows and scoring in Figure 3.6).

3.2.2.6 Map Link Selection

The map link with the highest attention score is identified as the associated map link. This step involves selecting the map link that best aligns with the ego trajectory, reflecting the network's determination of the vehicle's position relative to the road network (as indicated by the highest attention path selection in Figure 3.6).

3.3 Offset Prediction

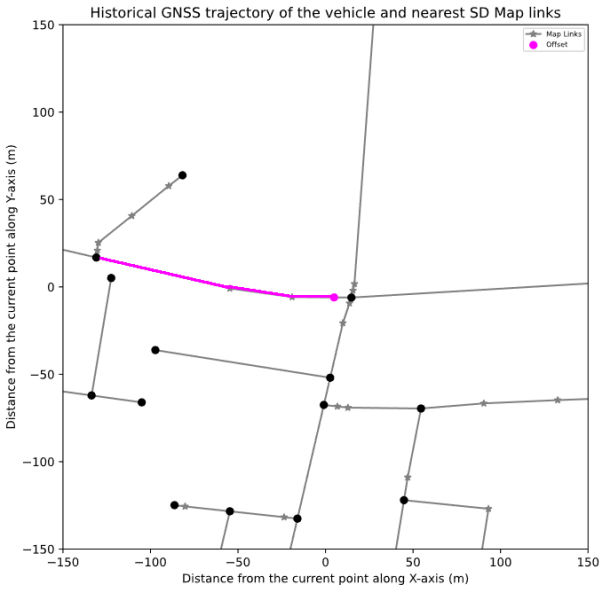

Figure 3.7: Offset Prediction: The pink line is the offset distance of the vehicle from start of the map link

In the previous section, we demonstrated the ability to develop a network capable of classifying the correct map link based on the map information and ego trajectory, specifically using the Cross Attention Network. Moving forward, the next challenge was to predict the vehicle's position along the identified map link—specifically, how far along the vehicle is from the start of the ground truth map link.

To address this, we trained a model combining a GNN based encoder and a MLP. This model was designed to take the ego trajectory and the correct map link as inputs and predict the precise offset distance, or how far the vehicle is from the start of the map link along its length. The rationale for pursuing this prediction approach was rooted in the nature of our dataset, which included ground truth labels in the form of offset distances.

However, upon testing this method, we observed that the network struggled to accurately predict the offset distance, often producing results that deviated significantly from the correct value.

These challenges highlight the complexities of accurately predicting the offset distance along a map link and suggest areas where further refinement and integration of the networks may be necessary to improve performance.

3.4 Point Prediction

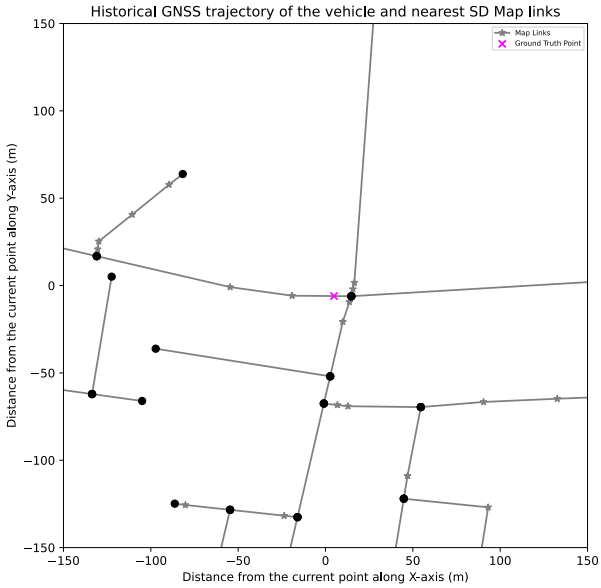

Figure 3.8: Point Prediction: The pink point is the exact position of the vehicle on the map link

Given the challenges associated with offset prediction, we reformulated the problem into a point prediction task. To simplify the model's learning process, the `offset` labels were transformed into `ground_truth_point` labels, which represent a 2D vector indicating the vehicle's true position relative to the last point of the ego trajectory. This reformulation allows the model to focus on directly predicting the ground truth point rather than calculating the distance from the start of the map link. Moreover, by learning to predict a point on the SD map, it would be fairly straightforward recover the distance along the matched link by projecting the point onto the matched link.

With this new formulation in place, we developed a network aimed at predicting the vehicle's position as a point. The point prediction network consists of a simple Multi-Layer Perceptron (MLP) with an output layer containing two neurons. The network is designed to take as input a vector formed by concatenating the ego trajectory with the combined features of all map links. Using this input, the network is trained to output a 2D point (x, y), representing the vehicle's localized position, with the last point of the ego trajectory serving as the origin. During training, the loss function incorporates two components: the distance between the predicted point and the ground truth point, and the distance between the predicted point and its nearest point on the ground truth map link. In this way, we can constrain the predicted point to the map link.

3.5 Combined Localization

Having developed networks for matching the ego trajectory to the correct map link and predicting the position on the map link, we now integrate these networks into a unified localization system. This combined localization network simultaneously produces both outputs, allowing the network to learn and adapt from the entire scenario. The architecture of the combined localization network is illustrated in Figure 3.9.

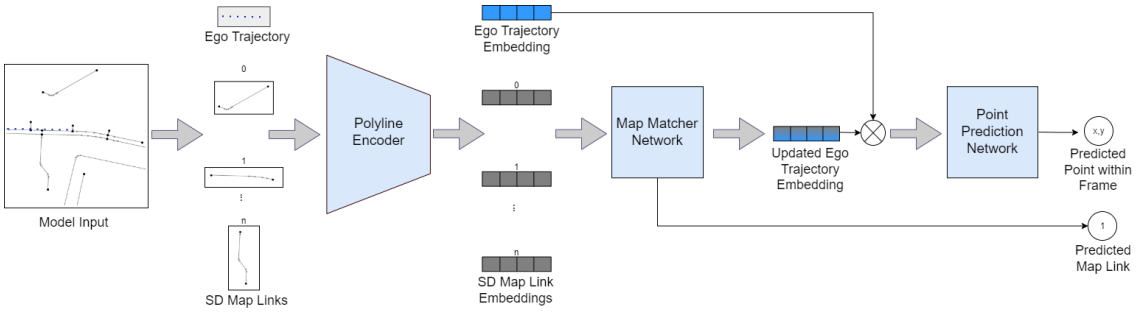

Figure 3.9: Deep Learning Map Localization Architecture: It consists of three modules - the polyline encoder, the map matching network, and the position prediction network.

As depicted in Figure 3.9, the Localizer model is composed of three fundamental modules: a polyline encoder, a map matching network, and a point prediction network.

3.5.1 Polyline Encoder

The polyline encoder is responsible for encoding the map link and ego trajectory into a fixed-dimensional embedding vector. This embedding serves as the input to the point prediction network. The details of this process are elaborated in the following subsection.

3.5.2 Map Matcher Network

The map matcher network employs the Cross Attention Network to classify the correct map link based on the available map information and to refine the ego trajectory using context vectors.

3.5.2.1 Context Vector Creation

Using attention scores, the embeddings of the map links are weighted to create a **context vector**. This vector represents a weighted sum of the map link embeddings, prioritizing information from the most relevant segments. The context vector effectively captures the relevant map data in relation to the ego trajectory, as illustrated by the combination of vectors leading to the context vector in Figure 3.6.

3.5.2.2 Ego Trajectory Update

The context vector updates the embedded ego trajectory by integrating the map link information. This update enriches the ego trajectory with pertinent details from the map link, preparing it for the subsequent stage, which is the point prediction network. This enhancement is shown by the trajectory improvement and output flow in Figure 3.6.

3.5.2.3 Output to Point Prediction Network

The updated ego trajectory, now enhanced with map link data, is passed to the point prediction network. This integration facilitates more accurate predictions and navigation decisions based on the refined alignment of the ego trajectory with the current map segment.

3.5.3 Point Prediction Network

The point prediction network takes the updated ego trajectory embedding as input and predicts the vehicle's position as a point.

3.6 Dataset Improvements

This section outlines the key enhancements made to the dataset. The three versions we used were labelled v0, v1 and v2 as mentioned in section 2.1. The augmentations done from v1 to v2 are explained in detail as follows:

1. **Dropped Duplicate GNSS Points**: In the ego trajectory, GNSS measurements were initially recorded at a frequency of 1 Hz, while timestamps were recorded at a higher frequency of 50 Hz. This led to approximately 50 duplicate GNSS measurements between updates. To avoid redundancy and ensure compatibility with different sampling rates during inference, these duplicate GNSS points were removed in the v2 dataset.
2. **Included Delta Timestamps**: After removing duplicate GNSS points, it was important to retain temporal information. For this, delta timestamps were introduced. The delta timestamp represents the time difference between the current iteration and the timestamp when the corresponding GNSS point was first recorded. This allows the model to understand the temporal context during inference, where the timestamps are updated with each iteration.
3. **Included Longitudinal Velocities**: To enhance the dataset, the longitudinal velocities of the vehicle were included for each GNSS point when it was recorded. This provides additional information that the model can use to better infer the vehicle's movement.
4. **Subsampling to Simulate Time Differences**: The original dataset always aligned the current iteration with the arrival of a new GNSS point, resulting in no time difference between the current iteration and the last recorded GNSS point. To simulate scenarios where there is a time gap, subsampling was applied. Specifically, the last 5% of the ego trajectory's GNSS points were

removed and then gradually regenerated. This process creates scenarios where the current iteration might not coincide with the latest GNSS point, helping the model to handle such cases during inference.

5. **Enhanced Edge Vectors with Delta Timestamps and Velocities**: In each edge vector for the ego trajectory, the delta timestamp and velocity are consistently taken from the second GNSS point of that edge vector. This ensures that the model has accurate velocity data, especially for the last edge in the trajectory, where the second point represents the last recorded GNSS point.For the non-moving map links, the delta timestamp and the velocity attributes is set to 0.

These improvements ensure that the dataset provides more accurate temporal and kinematic data, which in turn allows the model to perform better during inference, especially in scenarios involving varying sampling rates and time differences. The v2 version was used for training and evaluating the performance of the combined localizer and the baseline.

3.7 Ablation Studies

In this section, we perform a series of ablation studies to evaluate the performance of our localization model under different conditions and compare it against baseline and alternative methods.

3.7.1 Baselines

3.7.1.1 Naive Approaches

To establish baseline performance on our dataset, we implement two naive approaches:

1. **Last-Point Localization:** This approach assumes the absence of a localization solution. We use the last recorded point of the ego vehicle's trajectory as the estimated location. For the map-matching task, the nearest map link to this point is selected. This method serves as a fundamental naive baseline, which other solutions should surpass. We apply this solution on both, v1 and v2, versions of our dataset.
2. **Dead Reckoning:** For samples recorded between two GNSS updates (i.e., the subsampled data in the v2 dataset), we apply dead reckoning using the time intervals and longitudinal velocity to estimate the ego vehicle's position. This provides a baseline for the model's expected performance when leveraging inertial and temporal data in the v2 dataset. Note that this approach is not applicable to the v1 dataset due to the lack of difference between the time of the last ego vehicle position and the current time, which makes dead reckoning ineffective.

The naive approaches are applied only to the testing sets of the different versions of the dataset since they are not data-driven and do not require any training.

3.7.1.2 Map Matching

In order to compare the map matching performance, we implement the Hidden Markov Model (HMM) [24], which is the current state-of-the-art classical algorithm for map matching.

The HMM for map matching is implemented in three main stages: Initially, the road network is represented by map links, which are predefined in the SD map. Each map link corresponds to a specific segment of the road network and serves as a state in the model. Then, probabilities are calculated for each GNSS observation relative to these map links. This involves assessing the likelihood of a GNSS point being on a particular map link, typically based on the distance between the GNSS point and the nearest map link segment. Finally, the Viterbi algorithm is applied to determine the most probable sequence of map links. This step involves calculating the sequence of road segments that maximizes the probability of the observed GNSS sequence, while also considering the transition probabilities between map links. The outcome is a path aligned with the road network, accurately reflecting the vehicle's route despite GNSS errors and noise.

The HMM is applied to all testing samples from version v1 of the dataset to ensure a fair comparison with the deep learning alternative.

3.7.2 Analysis of Usage of Historical Ego Trajectory

To assess the model's reliance on historical trajectory information, we conduct experiments across four distinct scenarios:

1. Trained using full ego trajectory, tested using full ego trajectory.
2. Trained using full ego trajectory, tested using only the last point.
3. Trained using only the last point, tested using only the last point.
4. Trained using only the last point, tested using full ego trajectory.

3.7.3 Analysis of Usage of Inertial and Temporal data

One of the primary goals of our localization solution is to be able to localize at a frequency higher than the GNSS sensor update rate. To do this, and to handle short-term GNSS outages, we incorporate temporal and inertial data into the model. The following three cases are examined to test the usage of these sensors:

1. Ego Trajectory with direction of travel of the road - This scenario utilizes the ego trajectory data along with the direction of travel as input. It allows us to evaluate the model's performance with a minimal set of sensor data, focusing on how well it can infer localization information using basic trajectory and directional cues.
2. Ego Trajectory with direction of travel and delta timestamp - Here, we extend the input data to include both the direction of travel and temporal information, represented by delta timestamps. This experiment aims to determine whether the addition of temporal data enhances the model's ability to make more

frequent and accurate point predictions, especially in cases where predictions are required at a higher frequency than GNSS sensor updates.

3. Ego Trajectory with direction of travel and delta timestamp as well as longitudinal velocity - In this scenario, the model is provided with the ego trajectory, direction of travel, temporal data, and additional longitudinal velocity information. This setup evaluates the potential improvements in localization performance by combining temporal data with longitudinal velocity, assessing whether this comprehensive input enhances the model's accuracy and robustness in predicting vehicle position.

3.8 Training and Evaluation Metrics

3.8.1 Training

The model was trained using a joint optimization approach that combines both map matching and regression tasks. For the map matching component, the model's outputs are processed through a softmax activation function, and the resulting probabilities are then used to compute the Negative Log Likelihood Loss (NLLLoss). This loss measures the model's performance in predicting the correct map link by comparing the predicted probabilities against the ground truth.

For the regression component, two distinct losses are computed to guide the model's learning process. The first loss measures the Euclidean distance between the predicted point and the ground truth point. To calculate this, the squared differences between the predicted and ground truth coordinates are summed, and the square root of this sum gives the Euclidean distance.

Additionally, the second regression loss evaluates the distance between the predicted point and its closest point on the ground truth map link. This metric ensures that the predicted point is not only close to the ground truth point but also appropriately aligned with the correct map link. By using both losses, we aim to constrain the predicted point to be near the ground truth point while also ensuring it is well-positioned relative to the correct map link.

The loss function can be formulated mathematically as follows:

Complete Loss Function:
The model was trained using a joint optimization approach that combines map matching and point prediction tasks. The complete loss function L integrates the following components:

1. **Map Matching Loss:** This component uses the Negative Log Likelihood Loss (NLLLoss). The model's outputs, processed through a softmax activation function, generate probabilities for each map link. The NLLLoss measures the model's performance in predicting the correct map link by comparing the predicted probabilities against the ground truth.
$$L_{\text{map}} = -\log(p_{\text{gt}})$$
where p_{gt} is the probability assigned by the model to the ground truth map link.
2. **Point Prediction Loss:** This consists of two sub-components:
   (a) **Point Loss:** Measures the squared Euclidean distance between the predicted point and the ground truth point:
   $$L_{\text{point}} = \|\mathbf{p}_{\text{pred}} - \mathbf{p}_{\text{gt}}\|^2$$
   where $\mathbf{p}_{\text{pred}}$ is the predicted 2D point and $\mathbf{p}_{\text{gt}}$ is the ground truth 2D point.
   (b) **Link Loss:** Evaluates the Euclidean distance from the predicted point to its nearest point on the ground truth map link. The nearest point $\mathbf{p}_{\text{link}}$ is defined as the point on the link that minimizes the Euclidean distance to $\mathbf{p}_{\text{pred}}$:
   $$L_{\text{link}} = \min_t \|\mathbf{p}_{\text{pred}} - \mathbf{p}_{\text{link}}(t)\|^2$$
   where $\mathbf{p}_{\text{link}}(t)$ represents a point on the ground truth map link parameterized by t. The value of t is chosen to minimize the squared Euclidean distance between $\mathbf{p}_{\text{pred}}$ and the link.

The combined loss function L is given by:

$$L = \alpha \cdot L_{\text{map}} + \beta \cdot L_{\text{point}} + \gamma \cdot L_{\text{link}}$$

where α, β, and γ are weighting factors that balance the importance of each component.

3.8.2 Evaluation Metrics

The following metrics were used to assess the performance of the model:

1. **Map Matching Accuracy**: This metric measures the proportion of correct map link map matchings by the model. It is computed as the ratio of the number of correctly matched map links to the total number of matches attempted. Higher accuracy indicates better performance in predicting the correct map links.
2. **Route Accuracy**: In scenarios where the ego trajectory terminates near a

point where map links switch, it is important to account for the temporal proximity of the trajectory to these transitions. Route Accuracy addresses this by considering a broader context. Specifically, it evaluates whether the model correctly classifies any of the map links that the ego vehicle was within a 3-second window before or after the current point. This metric helps accommodate cases where exact point-to-link alignment is less critical. This metric was introduced starting from the v2 dataset.

3. **Point Predictor Point Loss (PP Point Loss)**: This metric assesses the regression performance by calculating the Euclidean distance between the ground truth point and the predicted point. It provides a measure of how close the predicted point is to the actual ground truth location.
4. **Point Predictor Link Loss (PP Link Loss)**: To further refine the regression performance, the Euclidean distance between the predicted point and the nearest point on the ground truth map link is computed. This loss helps ensure that the predicted point is not only close to the ground truth location but also appropriately aligned with the correct map link.

4
Results

The experiments conducted in this study aim to evaluate the performance of various models across multiple versions of the dataset. As summarized in Table 2.1 in Section 2.5, three versions of the dataset, labeled v0, v1, and v2, were utilized, each varying significantly in the number of training and testing samples. The progression from v0 to v2 represents an increase in the scale of the data, allowing for a comprehensive assessment of model performance across different data sizes.

Dataset version v0, the smallest with 2,504 training samples and 300 testing samples, serves as a baseline for evaluating model effectiveness on a limited data set. Version v1 introduces a more substantial dataset, with over 118,000 training samples, enabling the testing of models under more realistic scenarios. Finally, version v2, with nearly one million training samples, challenges the models with a large-scale dataset, assessing their scalability and robustness.

It is important to note that version v2 is derived from the same dataset as v1 but includes a subsampling process, as mentioned in Section 3.6. Specifically, up to 5% of the ego trajectory data points were removed from each sample, resulting in the removal of up to 8 points per sample. This subsampling resulted in an expanded dataset, with approximately 934,000 samples from the 118,000 original samples.

In the following sections, we will present the results obtained from these experiments, highlighting how each model performed across the different dataset versions. The analysis will focus on comparing map matcher accuracy, assessing the impact of dataset size, and evaluating the suitability of various model architectures for the tasks at hand.

4.1 Baselines

4.1.1 Naive Approaches

The performance metrics of the baseline approaches, including Last Point Localization and Dead Reckoning, are summarized in Table 4.1. For the v1 dataset, the Dead Reckoning approach is not applicable due to the absence of timestamp differences. In contrast, in the v2 dataset, Dead Reckoning outperforms Last Point Localization in terms of both Naive Projection Accuracy and Average Point Loss. Naive Projection Accuracy refers to the metric where, for the map-matching task,

Dataset	Approach	Naive Projection Accuracy ↑	Average Point Loss (m) ↓	Average Link Loss (m) ↓
v1	Last Point Localization	0.9263	3.8935	2.1092
v1	Dead Reckoning	N/A	N/A	N/A
v2	Last Point Localization	0.8666	13.5532	2.4501
v2	Dead Reckoning	0.9069	6.9779	2.1850

Table 4.1: Baseline performance metrics for Last Point Localization and Dead Reckoning approaches on v1 and v2 datasets.

the nearest map link to the estimated location point is naively selected.

These baselines provide a reference point against which more sophisticated models can be compared.

4.1.2 Map Matching using the Hidden Markov Model

To establish a baseline for the map-matching module, we evaluate the performance of the Hidden Markov Model on the v1 dataset. This analysis serves as a benchmark for comparing the performance of the deep learning-based map-matching model against a state-of-the-art classical algorithm.

Model	Dataset	Map Matching Accuracy	Route Accuracy
Hidden Markov Model	v1	0.9463	0.9681

Table 4.2: Baseline performance: Hidden Markov Model based map matcher on the v1 dataset.

The values in table 4.2 represent the performance of the Hidden Markov Model (HMM) on the v1 dataset. For the map matching accuracy, the HMM is run on the sequence of GNSS points in the ego trajectory, and a match is considered correct if the predicted map link is the same as the ground truth link at the time of prediction. For route accuracy, a match is considered correct if the predicted map link is one of any of the ground truth links within a 6-second time window. The accuracy is computed as:

$$\text{Accuracy} = \frac{\text{Number of correct matches}}{\text{Total number of matches done}}$$

where correct matches are determined based on the aforementioned criteria for map matching and route accuracy.

4.2 Map Matching

In this section of the results chapter, we present and compare the outcomes of two approaches: one using a fixed number of map links and another using a variable number of map links.

4.2.1 Map Matching based on a Fixed Number of Map Links

Encoding Model	Map Matching Model	Dataset	Map Matcher Accuracy ↑
GNN	MLP	v0	0.9105

Table 4.3: Performance of the GNN model with an MLP map matcher on the v0 dataset, showing the map matcher's accuracy when using only the 50 nearest map links.

After training a GNN which encodes both the ego trajectory and map links and then an MLP which takes these embeddings as the input, the model had 91% map matching accuracy. This shows us that a GNN can successfully encode the polylines so that a downstream model such as a simple MLP can select the correct map link out of the available fixed set of map links.

4.2.2 Map Matching based a Variable Number of Map Links

Encoding Model	Map Matching Model	Dataset	Map Matcher Accuracy ↑
Transformer	Cosine Similarity	v0	0.4355
GNN	Cosine Similarity	v0	0.8781
Transformer	Cross Attention	v0	0.9211
GNN	Cross Attention	v0	0.9552

Table 4.4: Comparison of different encoding models and map matching models on the v0 dataset, with corresponding accuracy when using all map links within a 300x300m frame. This could vary across samples since some samples could be urban scenarios with multiple map links whereas others could be highway scenarios with very few map links to select from.

Table 4.4 presents a comparison of the accuracy between Cosine Similarity and Cross Attention map matching models for the GNN and transformer encoders. From the table, it is evident that the Cross Attention map matcher consistently outperforms Cosine Similarity, regardless of the encoding model. The GNN with Cross Attention achieves the highest accuracy at 95.52%, indicating that this combination is

the most effective for selecting map links from a set of a variable number of links.

In contrast, the Transformer encoding model with Cosine Similarity map matcher shows the poorest performance, with only 43.55% accuracy. This suggests that the Transformer does not effectively encode ego trajectories and map links into a shared embedding space when using Cosine Similarity. The significant gap in performance between Cosine Similarity and Cross Attention for the Transformer model also highlights the importance of selecting an appropriate map matcher to leverage the encoding model's capabilities.

The GNN's relatively strong performance with both map matchers further demonstrates its robustness in handling different cases, making it a versatile option for scenarios with varying link densities.

Encoding Model	**Map Matching Model**	**Dataset**	**Map Matcher Accuracy ↑**
Transformer	Cross Attention	v1	0.9190
GNN	Cross Attention	v1	0.9454

Table 4.5: Comparison between the GNN encoder and the Transformer encoder for map matching on the v1 dataset, which is significantly larger than the v0 dataset.

Table 4.5 presents a comparison of the accuracy between GNN and Transformer encoders using the cross-attention map matching model on the v1 dataset. From the table, we can see that even on the significantly larger v1 dataset, which is approximately 50 times bigger than the v0 dataset, the Transformer model does not perform as well as the GNN. Despite the expectation that a larger dataset would typically enhance a Transformer's performance, the GNN continues to outperform it, achieving a map matcher accuracy of 94.54% compared to the Transformer's 91.9%. This result highlights the GNN's superior ability to manage and learn from larger, more complex datasets, further underscoring its robustness and suitability for scenarios with varying link densities.

4.3 Offset and Point Prediction

Initial attempts at offset prediction using an MLP which takes the ego trajectory and the current map link as input yielded poor results with the offset distance being predicted with an average localization error of 45.16m from the ground truth point. Since this error is significantly greater than the decided error limit of 9m, the approach to directly predict the offset was discontinued, and further experiments were directed towards predicting the localized position of the ego vehicle.

Encoding Model	Map Matching Model	Point Prediction Model	Dataset	Map Matcher Accuracy ↑	Point Predictor Point Loss (m) ↓	Point Predictor Link Loss (m) ↓
Transformer	Cross Attention	MLP	v1	0.9260	3.5183	1.2457
GNN	Cross Attention	MLP	v1	0.9471	2.8566	0.5843

Table 4.6: Comparison of the Transformer vs the GNN on the complete localization task on the v1 dataset.

4.4 Combined Localization

Table 4.6 provides an overview of the complete localization task using both Transformer and GNN-based encoders. The results indicate that while both encoders are capable of performing the localization task, the GNN consistently outperforms the Transformer across all evaluated metrics (Refer section 3.8.2). This suggests that although localization can be achieved with either approach, the GNN-based encoder offers a 2% improved accuracy in the map-matching task and a 0.7m reduction in both the point loss and the link loss, thereby making it the more effective choice for complete localization.

Model trained on full trajectory:

Delta Timestamp	Long. Velocity	Direction ID	Test Set Ego Trajectory Length	Map Matcher Accuracy ↑	Route Accuracy ↑	PP Point Loss (m) ↓	PP Link Loss (m) ↓
False	False	True	Full	0.9231	0.9691	3.1848	0.7693
True	False	True	Full	0.9212	0.9683	2.9080	0.7911
True	**True**	**True**	**Full**	**0.9229**	**0.9686**	**2.7509**	**0.6905**
True	**True**	**True**	**Only Last Point**	**0.9001**	**0.9501**	**15.3273**	**1.5009**

Model trained on last ego point:

Delta Timestamp	Long. Velocity	Direction ID	Test Set Ego Trajectory Length	Map Matcher Accuracy ↑	Route Accuracy ↑	PP Point Loss (m) ↓	PP Link Loss (m) ↓
False	False	True	Only Last Point	0.9189	0.9693	3.1762	0.7243
True	False	True	Only Last Point	0.9243	0.9675	3.0238	0.7044
True	**True**	**True**	**Only Last Point**	**0.9334**	**0.9704**	**2.9334**	**0.6205**
True	**True**	**True**	**Full**	**0.9299**	**0.9661**	**4.4636**	**0.8158**

Table 4.7: Performance comparison of the GNN Localizer model on the v2 dataset with different feature configurations.

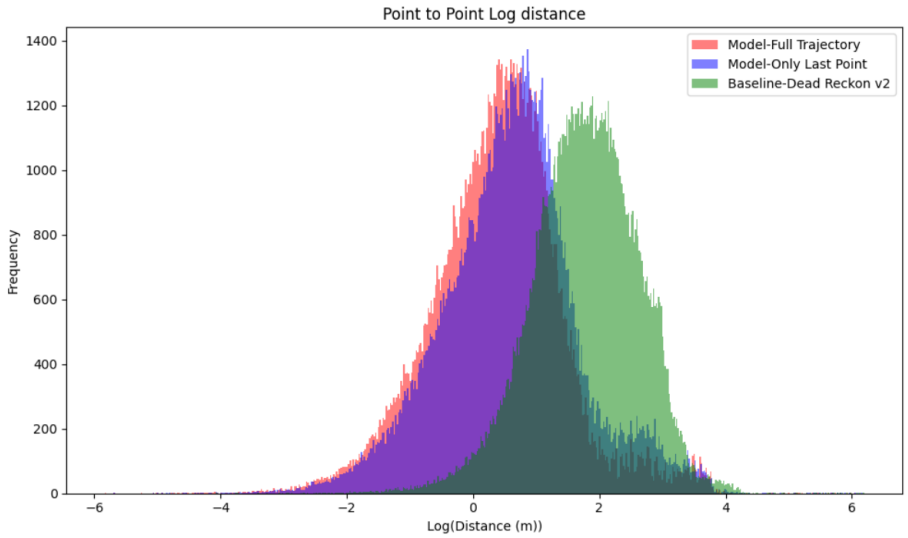

Figure 4.1: Distribution of log-transformed point losses for the GNN-encoder-based localizer. The red distribution represents the model trained and tested on the full ego trajectory, while the blue shows the model trained and tested on only the last point. The green distribution corresponds to the dead reckoning baseline approach. The full trajectory-based model achieves better point loss performance.

Fig 4.1 shows the distribution of log-transformed point losses of the GNN-based combined localizer. The distributions represent the point losses in the point prediction task. In red, we train and test the model on the full ego trajectory, whereas the blue distribution represents the model trained and tested on only the last point of the ego trajectory. The green distribution, on the other hand, represents the naive localization approach where dead reckoning is performed on the v2 dataset. We can visually observe that using the full trajectory for the model yields better performance compared to both the last-point-only approach and the baseline dead reckoning method.

4.5 Ablation Studies

Table 4.7 presents a performance comparison of the GNN Localizer model with different feature configurations and training/testing setups. The analysis of the results is as follows:

- **Full Trajectory with All Additional Features:** When the model is trained on the full ego trajectory using all three additional features—delta timestamp, longitudinal velocity, and direction ID—it achieves among the highest map matcher accuracy (0.9229) and route accuracy (0.9686) among the tested configurations. This setup also results in the lowest regression losses, with a point loss of 2.7509 m and a link loss of 0.6905 m. These results indicate that incorporating the complete trajectory and all available features leads to superior performance in both accuracy and regression tasks, effectively capturing the spatial-temporal details of the data.

- **Full Trajectory with Partial Features:** When trained on the full trajectory but using only partial features (excluding longitudinal velocity), the model shows slightly lower map matcher accuracy (0.9212) and route accuracy (0.9683). The regression losses are moderately higher, with a point loss of 2.9080 m and a link loss of 0.7911 m. These results highlight that while the model can still perform well with a reduced feature set, the exclusion of some features slightly impacts both accuracy and regression performance.
- **Full Trajectory with All Additional Features, Tested on Last Point:** Surprisingly, when the model is trained on the full trajectory with all features—delta timestamp, longitudinal velocity, and direction ID—but tested only on the last point of the trajectory, it performs very poorly. The regression metrics show a significant drop in performance, with a high point loss of 15.3273 m and a link loss of 1.5009 m. Despite decent map matcher accuracy (0.9001) and route accuracy (0.9501), the substantial increase in regression errors suggests that the model, when trained on comprehensive trajectory data, struggles to generalize effectively when evaluated under the limited scope of a single point. This underscores the challenge of applying a model trained on detailed, multi-point data to a scenario where only minimal input is provided during testing.
- **Last Point with All Additional Features:** When the model is trained on only the last point of the ego trajectory with all three features included, it achieves the highest map matcher accuracy (0.9334) and the best route accuracy (0.9704) when tested on the same last-point condition. The regression losses are also competitive, with a point loss of 2.9334 m and a link loss of 0.6205 m. This suggests that even when the training is limited to the last point, the inclusion of all features allows the model to maintain high accuracy and effective regression performance.
- **Last Point with Partial Features:** When trained on the last point of the ego trajectory but using only partial features—excluding longitudinal velocity—the model shows respectable performance despite the reduced feature set. It achieves a map matcher accuracy of 0.9243 and a route accuracy of 0.9675 when tested on the same last-point condition. The regression losses are also fairly competitive, with a point loss of 3.0238 m and a link loss of 0.7044 m. This indicates that even without the longitudinal velocity, the model can effectively perform map matching and route prediction tasks when trained on just the last point. The relatively small increase in regression errors compared to configurations with all features suggests that the direction ID and delta timestamp provide sufficient information to maintain robust performance in the absence of the full feature set.
- **Last Point with All Features, Tested on Full Trajectory:** When the model is trained on the last point of the ego trajectory with all features and then tested on the full trajectory, its performance remains relatively strong in terms of map matcher accuracy (0.9299) and route accuracy (0.9661). However, the regression performance declines, with a point loss of 4.4636 m and a link loss of 0.8158 m. This outcome suggests that while the model can maintain good accuracy metrics, it struggles to generalize in regression tasks when

the test data includes the full trajectory. The model, likely optimized for the specific context of the last point, finds it challenging to adapt to the more complex, extended input of the full trajectory during testing.

Overall, the results indicate that the model's performance is highly dependent on both the feature set and the temporal scope (full trajectory vs. last point) used during training and testing. While the results are similar for almost all cases, it is evident that the model trained on the full trajectory performs very poorly when tested on only the last point. Similarly, the localizer model trained on the last point and tested on the entire ego trajectory does not perform as well, however not as bad as the opposite case. It is also interesting to see that for two models trained and tested on the full feature set - one on the full trajectory and one on only the last point - the model based on the last point gives slightly better overall results. The possible reasons for the better performance of the last point model are discussed in Section 5.1.3.

4.6 Failure Cases

In this section, we analyze the performance of the best performing localizer model by examining scenarios where the model incorrectly predicts map links. Our aim is to conduct a qualitative analysis of the situations where the model fails to correctly identify the correct map link. The following are some observed failure cases:

4.6.1 Failure Case: Uncertain Between Two Links

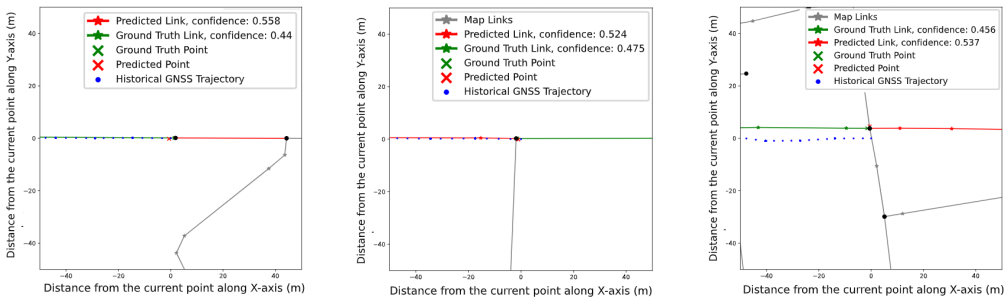

Figure 4.2: Failure Cases: Uncertain between Two Links

Figure 4.2 illustrates instances where the model is uncertain about choosing between two possible map links. This often occurs when the ego vehicle's trajectory ends near a point where two links change or converge, such as at an intersection or a junction. In these scenarios, the model finds it difficult to distinguish between the two links as they exhibit similar probabilities. The proximity of the trajectory to both links increases the model's uncertainty, leading to confusion in selecting the correct link. These cases are eliminated when we consider the route accuracy of the model.

4.6.2 Failure Case: High Confidence in Clearly Incorrect Link

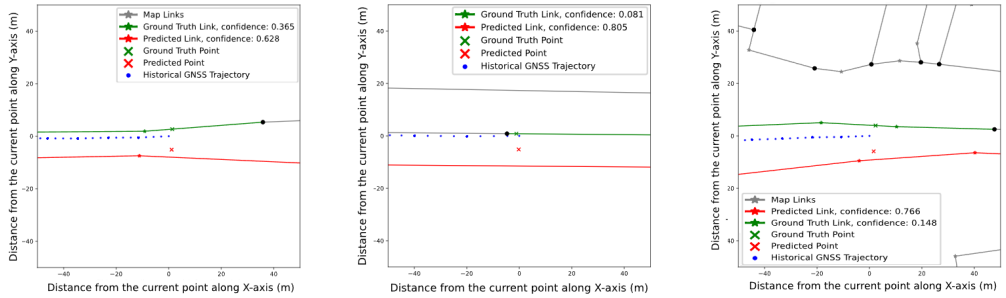

Figure 4.3: Failure Cases: High Confidence in Clearly Incorrect Links

Figure 4.3 presents cases where the model demonstrates high confidence in predicting a map link that is clearly incorrect. In these instances, the ego vehicle's trajectory is closer to the ground truth map link, yet the model selects a different link with high certainty. This phenomenon is particularly common when two map links run parallel to each other. It can also be observed that the model's point prediction is farther from the ground truth than the last point of the ego trajectory, further indicating that the model's choice of map link affects the accuracy of its point prediction.

4.6.3 Failure Case: Split/Merge Scenarios

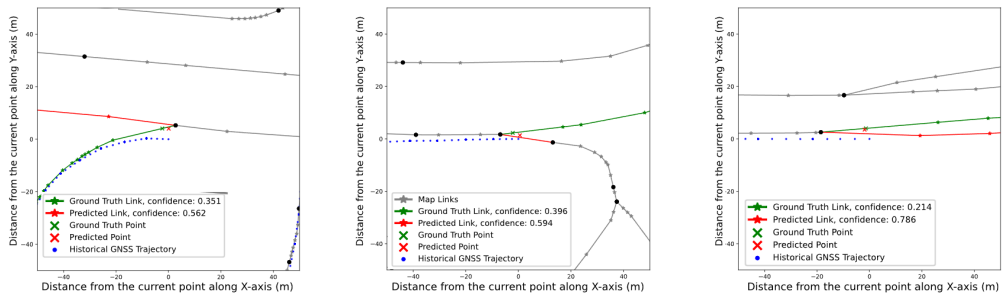

Figure 4.4: Failure Cases: Split/Merging Map Links

Figure 4.4 illustrates failure cases where the model struggles to handle road segments that involve splitting or merging of map links. In split scenarios, a single road divides into multiple branches, each represented by a different map link. Conversely, in merge scenarios, multiple road segments converge into a single link. These transitions can be challenging for the model, as there is often insufficient information at the point of prediction to confidently make a correct guess.

5
Conclusion

5.1 Discussions

5.1.1 Map Matching

In Table 4.4, we observe that map matching using cross attention yields better results compared to cosine similarity. This improvement can be attributed to the way embeddings capture features from polylines. While cosine similarity only measures the distance between embeddings in higher-dimensional space, it seems insufficient for distinguishing between different map links. On the other hand, cross attention between the ego trajectory query vector and the key vectors of the map links appears to enhance the similarity detection between the ego trajectory and map links. This may also be due to the learnable weights in the query and key vectors during cross attention, allowing the model to learn the similarities between embeddings in high-dimensional space.

5.1.2 Offset Prediction

Offset prediction, as mentioned in section 4.3, proves to be a challenging task. Several factors contribute to this difficulty:

1. **Inconsistent Starting Points and Arbitrary Offset Distances:** The dataset's inconsistent starting points for map links, where either end can be considered the start, lead to arbitrary offset distances. This variability complicates the prediction task, as the network must account for the differing starting points and the associated complexity of summing distances along various segments of the map link.
2. **Independence from the Classification Network:** The offset prediction network was developed independently from the Cross Attention Network used for classification. Integrating the two networks would be challenging, as the classification network would need to consistently and accurately identify the correct map link for its output to serve as a reliable input for the offset prediction network.

5.1.3 Combined Localizer

1. **Performance of the GNN Localizer:** As shown in Table 4.7, the GNN localizer generally performs well. The model with all features trained on the last ego point achieves the highest map matching accuracy. On the other hand,

the model trained on the entire trajectory with all three features shows the lowest PP point loss and PP link loss. However, the overall performance of both models is similar, with only minor differences.

2. **Effectiveness of Last Point Training:** The model's strong performance when trained solely on the last ego point suggests that the last point often provides sufficient information for accurate map matching and point prediction. This implies that the rest of the trajectory is not being leveraged for improved accuracy. One possible explanation is the absence of map link connectivity information, which prevents the model from fully utilizing the entire trajectory. Since the map links are treated as disconnected entities, the model is unable to infer which links the vehicle has previously traversed. Historical trajectory data would only be valuable if the model could identify the past map links to predict the current one. Therefore, incorporating map link connectivity offers significant potential for enhancing the model's performance.
3. **Role of Direction ID and Coordinates:** Another noteworthy observation is that the model trained with only the direction ID and last point still performs competitively. This implies that temporal or speed information may not be crucial for the model's predictions. Instead, the model primarily relies on the coordinates of the trajectory and map links, indicating a geometric approach to matching the ego trajectory with the map link.

5.1.4 Failure Cases

One plausible explanation for the failure case mentioned in section 4.6.2, where the model exhibits high confidence in selecting an incorrect link is that during data collection, the ego vehicle may have traveled in a different lane or even on the opposite side of the road, possibly due to temporary diversions. Since the data was collected several years ago, this route may not align with the current SD Map, leading to inaccuracies in the ground truth labeling. Another reason could be the model's difficulty in distinguishing between closely spaced parallel roads. As highlighted in section 2, the model does not perform better than simply using the last point from the ego trajectory. Connecting earlier parts of the trajectory with the correct links might help address these issues.These findings suggest that further research is needed to enhance the model's performance in such scenarios, particularly in refining its handling of parallel road segments and improving its understanding of road configurations. One potential solution is to provide the model with map link connectivity information, as it currently lacks knowledge of which links are connected. By incorporating this information, the model could more effectively associate earlier parts of the trajectory with the correct map links.

Additionally, the model's difficulty in cases where there are splits and merges, as discussed in section 4.6.3, can be attributed to abrupt changes in the road structure that may not be immediately reflected in the trajectory data. As a result, the model may initially select the incorrect link, as it has not yet fully adjusted to the new configuration. This challenge is further compounded by the fact that the model relies on consistent and confident information from the ego trajectory to make accurate

predictions. In split and merge scenarios, the trajectory may initially be ambiguous, leading to a delay in the model's adaptation to the correct link.
This issue is common in map matching when based solely on geometry. Including additional input sources that provide more semantic meaning, such as lane markings, road type, and traffic signs, could help resolve these cases by providing more context for accurate predictions.

These issues highlight a shortcoming in the model's expected behavior. Ideally, the model should leverage velocity information to distinguish between links on either side of a split or merge. However, the observed disregard for velocity data in such scenarios exacerbates the problem, indicating a need for further investigation. Future work should focus on addressing this limitation by enhancing the model's integration of velocity information to improve its ability to correctly differentiate between road segments during split and merge events.

5.1.5 Polyline encoder: Transformer v/s. GNN

While the Transformer architecture proves to be a viable solution for encoding polylines and demonstrates potential for tasks within this framework, its performance lags behind that of the GNN. One possible explanation is that Transformers are inherently more data-hungry compared to GNNs. However, even when evaluated on a significantly larger dataset, as shown in the comparison table for the v1 dataset, the Transformer still underperforms relative to the GNN.

Another factor could be related to how information is retained in the architectures. In the GNN architecture, inspired by VectorNet, nodes retain information after each message-passing layer. This is achieved by increasing the dimension size from d_{in} to d_{out} where $d_{out} = d_{in} + d_{in}$ after each graph encoder layer, due to the concatenation of the aggregated vector, as depicted in the Fig. 3.3. In contrast, Transformers may lose some information during the self-attention mechanism, as the original vectors are transformed into encoded vectors of the same dimension. A potential improvement for the Transformer encoder architecture could involve concatenating the self-attended vectors with the original vectors, an approach that could be explored in future work.

5.2 Future Work

- Information regarding connectivity between the map links is currently not being given to the model. Some methods to incorporate this information into the model could help improve performance, especially for the cases such as where the model is certain about wrong links on parallel roads.
- Perform a quantitative analysis of the different road situations that the model fails to localize on.
- Expand the model by adding more input modalities - lane marking detections, traffic signs or even raw images.

- Implement dead reckoning to enhance model robustness during brief GNSS signal losses.

5.3 Conclusion

This work addresses the research questions raised in the section 1.1 for the encoding and integration of Standard Definition (SD) map data and multi-modal sensor data to enhance interpretability by deep learning models. Both Graph Neural Networks (GNNs) and Transformers emerge as viable options for encoding SD map data, with GNN showing superior performance across various downstream tasks. A unified polyline encoder is developed to effectively integrate multi-modal sensor data, including inertial and historical temporal information, with the encoded map data, enabling a comprehensive understanding of the vehicle's environment. Additionally, the cross-attention mechanism proves to be highly effective in determining the most likely road segment the vehicle is currently navigating. By extracting relevant information from the surrounding map area and utilizing an MLP head, the approach achieves an impressive 92.29% accuracy in map matching, with a point prediction error of less than 2.75 meters for the vehicle's position along the correct road segment. These findings advance autonomous vehicle localization by offering data-driven methodologies that are robust and reliable.

Bibliography

[1] Sven Bauer, Yasamin Alkhorshid, and Gerd Wanielik. Using high-definition maps for precise urban vehicle localization. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 492–497, 2016.

[2] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[3] Hao Cai, Zhaozheng Hu, Gang Huang, Dunyao Zhu, and Xiaocong Su. Integration of gps, monocular vision, and high definition (hd) map for accurate vehicle localization. *Sensors*, 18(10), 2018.

[4] Athanasios Chalvatzaras, Ioannis Pratikakis, and Angelos A. Amanatiadis. A survey on map-based localization techniques for autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 8(2):1574–1596, 2023.

[5] Athanasios Chalvatzaras, Ioannis Pratikakis, and Angelos A. Amanatiadis. A survey on map-based localization techniques for autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 8(2):1574–1596, 2023.

[6] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. A survey on map-matching algorithms, 2019.

[7] Yang Dakai, Cai Baigen, and Yuan Yifang. An improved map-matching algorithm used in vehicle navigation system. In *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, volume 2, pages 1246–1250, 2003.

[8] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Mahdi Elhousni and Xinming Huang. A survey on 3d lidar localization for autonomous vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1879–1884, 2020.

[10] Jie Feng, Yong Li, Kai Zhao, Zhao Xu, Tong Xia, Jinglin Zhang, and Depeng Jin. Deepmm: Deep learning based map matching with data augmentation. *IEEE Transactions on Mobile Computing*, 21:2372–2384, 7 2022.

[11] Christopher Fifty. Deciding which tasks should train together in multi-task neural networks, 2021. Posted by Christopher Fifty, Research Engineer, Google Research, Brain Team.

[12] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. 5 2020.

[13] Seung-Jun Han, Jungyu Kang, Yongwoo Jo, Dongjin Lee, and Jeongdan Choi. Robust ego-motion estimation and map matching technique for autonomous

vehicle localization with high definition digital map. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 630–635, 2018.

[14] Mahdi Hashemi and Hassan A. Karimi. A critical review of real-time map-matching algorithms: Current issues and future directions, 2014.

[15] Linli Jiang, Chao-Xiong Chen, and Chao Chen. L2mm: Learning to map matching with deep models for low-quality gps trajectory data. *ACM Transactions on Knowledge Discovery from Data*, 17:1–25, 6 2023.

[16] JS Kim. Node based map matching algorithm for car navigation system. In *International symposium on automotive technology & automation (29th: 1996: Florence, Italy). Global deployment of advanced transportation telematics/ITS*, 1996.

[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[18] Matej Kubicka, Arben Cela, Hugues Mounier, and Silviu-Iulian Niculescu. Comparative study and application-oriented classification of vehicular map-matching methods. *IEEE Intelligent Transportation Systems Magazine*, 10(2):150–166, 2018.

[19] Sampo Kuutti, Saber Fallah, Konstantinos Katsaros, Mehrdad Dianati, Francis Mccullough, and Alexandros Mouzakitis. A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet of Things Journal*, 5(2):829–846, 2018.

[20] Johann Laconte, Abderrahim Kasmi, Romuald Aufrère, Maxime Vaidis, and Roland Chapuis. A survey of localization methods for autonomous vehicles in highway scenarios. *Sensors*, 22(1), 2022.

[21] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[22] Katie Z Luo, Xinshuo Weng, Yan Wang, Shuang Wu, Jie Li, Kilian Q Weinberger, Yue Wang, and Marco Pavone. Augmenting lane perception and topology understanding with standard definition navigation maps. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4029–4035. IEEE, 2024.

[23] Wei-Chiu Ma, Ignacio Tartavull, Ioan Andrei Bârsan, Shenlong Wang, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, and Raquel Urtasun. Exploiting sparse semantic hd maps for self-driving vehicle localization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5304–5311, 2019.

[24] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343, 2009.

[25] Jong-Sun Pyo, Dong-Ho Shin, and Tae-Kyung Sung. Development of a map matching method using the multiple hypothesis technique. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*, pages 23–27. IEEE, 2001.

[26] Karl Rehrl and Simon Gröchenig. Evaluating localization accuracy of automated driving systems. *Sensors*, 21(17), 2021.

[27] Tyler G.R. Reid, Sarah E. Houts, Robert Cammarata, Graham Mills, Siddharth Agarwal, Ankit Vora, and Gaurav Pandey. Localization requirements for autonomous vehicles. *SAE International Journal of Connected and Automated Vehicles*, 2(3), September 2019.

[28] Ming Ren and Hassan A Karimi. A hidden markov model-based map-matching algorithm for wheelchair navigation. *The Journal of Navigation*, 62(3):383–395, 2009.

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[30] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer Handbook of Robotics. Springer Berlin Heidelberg, 2008.

[31] D Srinivasan, Ruey Long Cheu, and Chuan Wei Tan. Development of an improved erp system using gps and ai techniques. In *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, volume 1, pages 554–559. IEEE, 2003.

[32] Global GPS Systems. The difference between gnss and gps explained. `https://globalgpssystems.com/gnss/the-difference-between-gnss-and-gps-explained/`, 2024. Accessed: August 28, 2024.

[33] George Taylor, Geoffrey Blewitt, Doerte Steup, Simon Corbett, and Adrijana Car. Road reduction filtering for gps-gis navigation. *Transactions in GIS*, 5(3):193–207, 2001.

[34] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[35] Christopher E White, David Bernstein, and Alain L Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation research part c: emerging technologies*, 8(1-6):91–108, 2000.

[36] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. Tnt: Target-driven trajectory prediction.

A

Appendix

A.1 Sensor Descriptions

A.1.1 GNSS

Global Navigation Satellite Systems (GNSS) use satellites to provide accurate positioning data anywhere on Earth. It integrates signals from multiple satellite constellations such as GPS, GLONASS, Galileo, Beidou and other regional systems to determine location with high precision. [32]

A.1.2 INS

Inertial Navigation Systems (INS) use accelerometers and gyroscopes to calculate position, velocity, and orientation without external references. It provides continuous tracking by integrating measurements over time using algorithms like Kalman Filtering. [30]

A.1.3 OxTS

OxTS produces high-precision GNSS and INS data, recording both types at a frequency of 40 Hz. In contrast, the production-ready GNSS receiver in the vehicle captures GNSS data only, at a frequency of 1 Hz.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY