Neural Augmented Kalman Filters for Road Network assisted GNSS positioning

Hans van Gorp [*1,2] **Davide Belli** [*2] **Amir Jalalirad** [2] **Bence Major** [2]

Abstract

The Global Navigation Satellite System (GNSS) provides critical positioning information globally, but its accuracy in dense urban environments is often compromised by multipath and non-line-of-sight errors. Road network data can be used to reduce the impact of these errors and enhance the accuracy of a positioning system. Previous works employing road network data are either limited to offline applications, or rely on Kalman Filter (KF) heuristics with little flexibility and robustness. We instead propose training a Temporal Graph Neural Network (TGNN) to integrate road network information into a KF. The TGNN is designed to predict the correct road segment and its associated uncertainty to be used in the measurement update step of the KF. We validate our approach with real-world GNSS data and open-source road networks, observing a 29% decrease in positioning error for challenging scenarios compared to a GNSS-only KF. To the best of our knowledge, ours is the first deep learning-based approach jointly employing road network data and GNSS measurements to determine the user position on Earth.

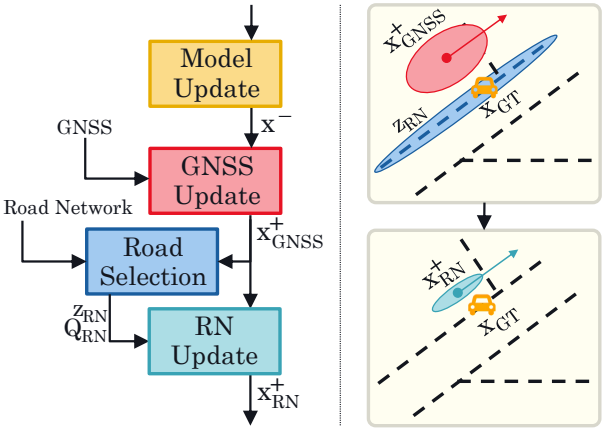


Figure 1. In order to leverage road network data in a GNSS-based KF, a Gaussian observation (mean z, covariance Q) needs to be constructed by selecting the correct road segment. In our proposed method, this road selection is performed by a Temporal Graph Convolutional Network which also predicts the appropriate covariance (left). The position and uncertainty predicted by the KF can be improved using this additional data source (right).

1. Introduction

The Global Navigation Satellite System (GNSS) provides accurate position and time information to billions of receivers worldwide. In 2023, an estimated 5.6 billion GNSS receivers have been produced, with this number expected to grow to 9 billion by 2033 (EUSPA, 2024). Of these, 10% were used in the road and automotive segment in 2023, growing to 15% by 2033. The coverage of GNSS is extensive, with around 120 satellites active in multiple constellations (GPS, Galileo, GLONASS, BeiDou), enabling localization systems anywhere on Earth.

However, even with this high number of satellites, accurate positioning in dense urban environments remains a challenge. Tall buildings can reflect the signal, leading to multipath errors, or block the direct view of the satellite, leading to non-line-of-sight errors. This is particularly problematic for applications such as lane detection and side-of-the-street detection for car navigation, delivery, and taxi services, which typically require positioning accuracies of 2 to 10 meters. However, positioning services relying solely on GNSS can result in errors up to 20 meters in urban canyons (Zhong & Groves, 2023). Even with complex 3D models, errors of no less than 10 meters are observed (Zhong & Groves, 2023). Leading companies use 3D Model Assisted solutions (van Diggelen, 2021) to improve positioning in urban environments, underscoring the limitations of GNSS-only localization in challenging scenarios. Addressing these challenges is crucial for enhancing the accuracy and reliability of positioning systems in urban settings.

[*]Equal contribution [1]Department of Electrical Engineering, Eindhoven University of Technology [2]Qualcomm AI Research. Work done during an internship at Qualcomm AI Research, Amsterdam. Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.. Correspondence to: Hans van Gorp <h.v.gorp@tue.nl>, Davide Belli <dbelli@qti.qualcomm.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

In automotive applications, positioning errors can be mitigated by introducing additional sources of information, such as vehicle motion sensors, cameras, radar and lidar (Skog & Handel, 2009). All of these require the integration of additional sensors besides the GNSS receiver, which are expensive or can be unavailable. In our work, we will instead focus on the integration of road network data, a cost-effective alternative which can be accessed globally from open-source platforms (OpenStreetMap contributors, 2017).

Kalman Filters (KFs) are the standard for processing GNSS measurements due to their efficiency, ability to fuse multiple sources of information, and the relatively affordable assumption of uni-modal Gaussian distributions for state and observations. While assuming normal distributions is reasonable for GNSS measurements, the same does not hold for road network observations, which makes their integration in a KF not straightforward. If one constructs a Gaussian observation with respect to each nearby road, the total observation becomes a multi-modal Gaussian. To address this, previous solutions first select a single "best" road segment and then construct a Gaussian observation around it. The selection can be based on belief theory (El Najjar & Bonnifait, 2005) or on Hidden Markov Models (HMM) combined with a Viterbi decoder (Atia et al., 2017), the latter of which has also been used in map-matching techniques (Hu et al., 2023; Song & Lee, 2023; Zhong et al., 2024).

Improving over previous approaches, we propose to augment the KF with a lightweight Temporal Graph Neural Network to select the road segment for the measurement update step as well as its uncertainty (see Figure 1). The main contributions in this paper are three-fold:

1. To the best of our knowledge, we are the first to propose a deep learning method to introduce road network inputs in a GNSS-based KF. We implement this as a Temporal Graph Neural Network predicting which road segments best match the GNSS trajectory from a moving vehicle. The predicted segments are used as additional measurements to update the KF state.

2. We introduce a standard deviation prediction head to regulate the impact of the road network observation in the KF update. We optimize this component end-to-end to minimize the positioning error.

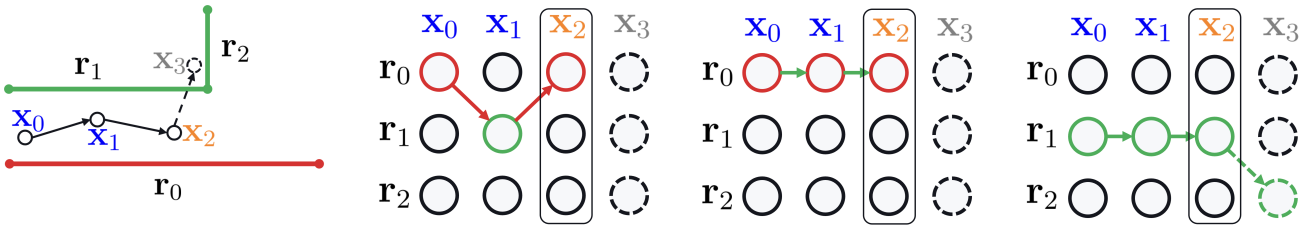
3. We evaluate the proposed method on challenging real-world GNSS measurements paired with open-source road network data, showing improvements over both a GNSS-only KF and a KF + Viterbi algorithm. We also include detailed ablations and qualitative analyses to assess the contribution from each component and the behavior in different scenarios.

2. Related Work

GNSS positioning with Neural Networks Recent work has extensively explored the use of Deep Learning methods to improve GNSS-based positioning systems. Some approaches process GNSS measurements with MLPs (Suzuki & Amano, 2021) or Neural Radiance Fields (Neamati et al., 2023) to predict which measurements are in line of sight. Other methods learn to estimate the pseudo-range error through Graph Neural Networks (Jalalirad et al., 2023) or Long Short-Term Memory Networks (Zhang et al., 2021). Neural Networks can also be trained to directly produce location corrections with respect to an input anchor point (Kanhere et al., 2022; Siemuri et al., 2021).

Neural Augmentation of Kalman Filters While previously mentioned works tackled the instantaneous positioning scenario, others investigated deep learning methods to track the receiver position over time. GNSS tracking solutions often consist of classical systems such as KFs, in which some of the components are augmented with data-driven Neural Networks (Shlezinger et al., 2024). Revach et al. (2022) proposes learning the optimal Kalman gain through the use of Recurrent Neural Networks (RNN), and Li et al. (2023) combines CNNs and LSTMs for similar purposes. Mohanty & Gao (2024) uses GNNs to estimate the state and state uncertainty matrices in the KFs, while Gao et al. (2020) and Han et al. (2021) apply Reinforcement Learning to learn the process noise covariance matrix. KFs can also be augmented with Neural Networks to process measurements from multiple sensors such as GNSS and Inertial Measuring Units (IMU) (Guo & Tu, 2021; Tang et al., 2022).

GNSS positioning with Road Networks The impact of multipath effects on GNSS signals can be mitigated by incorporating additional sources of information, such as road networks. Road networks can be integrated in GNSS positioning systems without the need for additional sensors. This data can be limited to a description of the road or lane topology, or include additional features like road segment categories, driving speed, and directionality. Quddus & Washington (2015) propose a genetic algorithm to select the most likely location among a set of candidate options based on local road network features. Other work jointly employ road network and GNSS data for the task of map matching, that is determining which sequence of road network segments best matches a GNSS measurement trajectory (Velaga et al., 2012; Feng et al., 2020; Hu et al., 2023). Map matching techniques only input information from the GNSS system to the map matching algorithm in order to select the most likely road. However, a feedback loop propagating road network information back into the GNSS positioning system could further improve its state and its ability to ac-



(a) Example scenario (b) Instant (c) Viterbi (d) Bidirectional Viterbi

Figure 2. Overview of different road selection algorithms. In (a) we show an example user trajectory estimated from noisy GNSS measurements, with past ($\mathbf{x}_0$, $\mathbf{x}_1$), current ($\mathbf{x}_2$), and the future ($\mathbf{x}_3$) locations. The road network has 3 segments including two parallel roads $\mathbf{r}_0$ and $\mathbf{r}_1$, with the true user trajectory following $\mathbf{r}_1$ and $\mathbf{r}_2$, marked in green. We compare three road selection methods, highlighting in green correct predictions for roads (nodes) and allowed transitions over the network (edges), while wrong ones are in red. An instantaneous solution (b) does not consider temporal relations, which might result in inconsistent predictions over time (jumping between similarly likely roads). Viterbi finds consistent solutions in terms of road connectivity, but can select the wrong path in case of noisy sequences. Bidirectional Viterbi (d) has access to future estimates and solves ambiguous cases by interpolating past and future trajectories.

curately process consecutive GNSS measurements. Thus, while map matching can be used to aid automated driving systems or model user driving behaviors, it does not provide a solution to the more general challenge of determining the exact user position on Earth at a given time. Other methods integrate road network information with KFs to improve the tracking performance over time. In this context, candidate locations on road segments are usually introduced to the KF as uni-modal Gaussian observations, to enable a direct update of the GNSS-based KF posterior. Features such as distance, heading, and speed alignment are used in heuristics or HMMs to determine the mean value of the distribution, while the standard deviation is typically estimated empirically and remains fixed during tracking (El Najjar & Bonnifait, 2005; Goh et al., 2012; Jagadeesh & Srikanthan, 2017; Li et al., 2024). In this paper, we demonstrate that the variance can be dynamically estimated for each timestep, thereby enabling more accurate localization when compared to a fixed variance. To the best of our knowledge, our method is also the first to propose a neural network to input road network data in a GNSS-based positioning system.

3. Learning Road Network Selection

In this section, we introduce a neural augmentation method for GNSS-based KF positioning to integrate the Road Network data in the KF measurement update. In Section 3.1, we first formalize how to update the KF estimate given the road segment on which the user is located. In section 3.2 we explain how the correct road segment can be selected using the Viterbi algorithm. In section 3.3 we introduce our main contribution: a Temporal Graph Neural Network trained to select the correct road segment. Lastly, in Section 3.4, we present an extension of our neural network to directly predict the road network covariance for the KF update.

3.1. Integrating Road Data into the Kalman Filter

To ensure our solution can be applied to road data from different providers, we make minimal assumptions on the available information. We only describe the roads by their center-lane segment, without relying on lane-level information, which is often incomplete, outdated or unavailable. For the same reasons, we treat road features such as driving speed and directionality as optionally available. We describe the dataset in more detail in Section 4.1.

For the purpose of road segment selection it is convenient to consider a graph representation in which the nodes are the road segments, and the edges are intersection or curvature points connecting two or more road segments in the network. We therefore define a road network graph as $\mathcal{G} = (\mathbf{R}, \mathbf{A})$ with $\mathbf{R} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ describing D-dimensional features for each of the N road segments, and $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ being a binary adjacency matrix representing the network structure.

Road network information can be used to post-process a KF estimate and generate an updated positioning prediction, for example by snapping the KF position on a selected road segment. Alternatively, the selected road segment could be directly integrated in the KF through an additional measurement update, as show in Figure 1. In this work, we consider a KF with mean $\mathbf{x} \in \mathbb{R}^8$ and covariance $\mathbf{P} \in \mathbb{R}^{8\times8}$ to track the user position $\mathbf{x}_{\text{pos}} \in \mathbb{R}^3$ and velocity $\mathbf{x}_{\text{vel}} \in \mathbb{R}^3$ on the local geodetic coordinate system as well as the receiver clock bias and drift $\mathbf{x}_{\text{clock}} \in \mathbb{R}^2$. The road network can be employed as a prior for the positioning task by updating the KF state based on the best matching road segment r^*. The road segment can be selected among a set of candidate segments as the one minimizing an arbitrary cost function $J(r_i)$ as:

$$r^* = \underset{r_i}{\operatorname{argmin}}\, J(r_i) \quad \text{with} \quad 0 \leq i < N, \tag{1}$$

A simple baseline for segment selection (see Figure 2b) can

be devised by defining the cost function in Equation 1 as the distance between the estimated user position $\mathbf{x}_{\text{pos}}$ and a road segment r_i:

$$J_{\text{pos}}(r_i) = \text{dist}(\mathbf{x}_{\text{pos}}, r_i), \tag{2}$$

where $\text{dist}(.,\ .)$ is the Euclidean distance between a point and a line segment in meters. We refer to this baseline as *Instant* in our experiments, as the selection only considers the current position estimate, and not the previous trajectory of the user. Given the KF posterior (denoted with $^+$) after the GNSS update step with mean $\mathbf{x}^+_{\text{GNSS}}$ and covariance $\mathbf{P}^+_{\text{GNSS}}$, the road network measurement update is:

$$\begin{aligned}
\mathbf{K} &= \mathbf{P}^+_{\text{GNSS}}\mathbf{H}^{\text{T}}\left(\mathbf{H}\mathbf{P}^+_{\text{GNSS}}\mathbf{H}^{\text{T}} + \mathbf{V}\right)^{-1}, \\
\mathbf{x}^+_{\text{RN}} &= \mathbf{x}^+_{\text{GNSS}} + \mathbf{K}\left(\mathbf{z} - \mathbf{H}\mathbf{x}^+_{\text{GNSS}}\right), \\
\mathbf{P}^+_{\text{RN}} &= \mathbf{P}^+_{\text{GNSS}} - \mathbf{K}\mathbf{H}\mathbf{P}^+_{\text{GNSS}},
\end{aligned} \tag{3}$$

where $\mathbf{z}$, $\mathbf{V}$ and $\mathbf{H}$ are respectively the mean, covariance and observation matrix for the road network measurement, $\mathbf{K}$ is the Kalman gain, $\mathbf{x}^+_{\text{RN}}$ and $\mathbf{P}^+_{\text{RN}}$ are the KF posterior mean and covariance after the road network update step, and $^{\text{T}}$ denotes the transpose operator. Appendix A includes the complete derivations showing how $\mathbf{z}$, $\mathbf{V}$ and $\mathbf{H}$ are derived from the road network position and heading.

3.2. Road Selection with the Viterbi Algorithm

The instantaneous selection of road segments often results in unstable and inaccurate predictions, as it does not capture temporal patterns over consecutive time-steps. One solution posed in literature is the use of a Hidden Markov Model to process the vehicle trajectory over the graph. In such an HMM, the observation is the history of estimated user positions, and the latent variable is the road segment on which the user is currently located. The Viterbi algorithm can be used online to determine the maximum a posteriori as the most likely sequence of road segments (Figure 2c).

To create a non-learnable baseline for road selection with Viterbi, we consider a HMM in which each road segment is a state, and the transition and emission probability functions are defined considering the relation between the road network and the estimated user location from the KF. The emission probability for each road segment is computed based on a position and heading cost. The position cost J_{pos} is defined as in Equation 2, and the heading cost is calculated as:

$$J_\theta(r_i) = 1 - \left|\cos(\theta_{\mathbf{x}} - \theta_{r_i})\right|, \tag{4}$$

where $\theta_{\mathbf{x}}$ is the user heading and θ_{r_i} is the road heading. With the absolute value we allow matching both directions

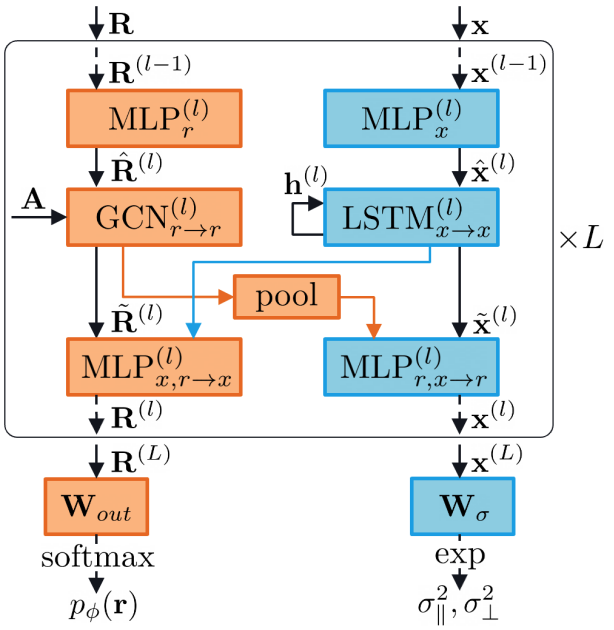


Figure 3. Architecture of the proposed TGNN.

along the road. The information about one-way road directionality, if available, will be encoded in the transition probabilities. Finally, the emission PDF is calculated using a weighted average of the heading and positions costs, as:

$$p(\mathbf{x}|r_i) = \max\left(1 - \frac{\beta J_{\text{pos}}(r_i) + J_\theta(r_i)}{2},\ \epsilon\right), \tag{5}$$

where β is a hyper-parameter to balance between the two cost functions, and $\epsilon = 0.01$ ensures that all road segments retain a non-zero emission probability. The transition probabilities between the different road segments are simply modeled with the PDF:

$$p(r_i|r_j) = \begin{cases} 1 & \text{if } r_i \in \text{k-Hop}(r_j; k) \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where the function k-Hop$(r_j; k)$ defines the k-hop graph neighborhood of r_j, which is a set including each road r_i that the user can reach within a maximum of k intersections starting from r_j, and taking one-way directionality into account. Setting $k > 1$ is useful to handle cases in which the vehicle is moving at high speed, traversing multiple road segments in a single time-step.

The Viterbi algorithm can efficiently run online through the recursive PDF formulation:

$$p(r_i, t) = \begin{cases} p(\mathbf{x}_{\text{pos}}|r_i) & \text{if } t = 0 \\ \max\limits_{r_j} p(r_j, t-1)\, p(r_i|r_j)\, p(\mathbf{x}_{\text{pos}}|r_i) & \text{if } t > 0 \end{cases}, \tag{7}$$

where at timestep $t = 0$ we assume a uniform prior over all roads. Therefore, the most likely road segment at the

current time-step t can be selected according to Equation 1 with the Viterbi cost function $J_{\text{Viterbi}}(r_i) = -p(r_i, t)$.

We refer to this method as *Viterbi* in our experiments.

3.3. Road Selection with a Trained TGNN

If future position estimates were available, a "bidirectional" Viterbi algorithm could be used to find a smoother trajectory on the road network, temporally consistent with both past and future observations (see Figure 2d). We refer to this bidirectional Viterbi algorithm as the *Oracle* method in our experiments, since it relies on future observations which are not available during online inference.

While future information is not available in practice, we propose training a neural network to learn to predict the optimal solutions found by the *Oracle*. We implement this as a novel Temporal Graph Neural Network architecture for road segment prediction. We choose Graph Neural Networks (GNNs) (Scarselli et al., 2008; Kipf & Welling, 2016; Zhou et al., 2020) as they can process structured graphs with varying number of nodes and are invariant to the nodes order. Temporal Graph Neural Networks (Zhao et al., 2019; Cao et al., 2020; Rossi et al., 2020) are a specific type of GNNs which can also process temporal relations in the graph. We show in Figure 3 the architecture of the proposed TGNN.

Our model ϕ takes as input the KF state $\mathbf{x}$, the road segment features $\mathbf{R}$, and the road network adjacency matrix $\mathbf{A}$, and outputs a probability for each road segment:

$$P_\phi(\mathbf{r}) = \phi\left(\mathbf{x}, \mathbf{R}, \mathbf{A}\right) \tag{8}$$

The neural network consists of a sequence of L repeated blocks which include feature transformation, local message passing and cross message passing layers. The user-level features $\mathbf{x}$ and road-level features $\mathbf{R}$ are processed on two distinct paths throughout the network, only merging in the cross message passing layers. In each block l, the feature transformation layers are implemented as two simple MLPs projecting the input features:

$$\hat{\mathbf{x}}^{(l)} = \text{MLP}_x^{(l)}\left(\mathbf{x}^{(l-1)}\right), \quad \hat{\mathbf{R}}^{(l)} = \text{MLP}_r^{(l)}\left(\mathbf{R}^{(l-1)}\right). \tag{9}$$

Local message passing layers consist of a Graph Convolutional Network (GCN) (Kipf & Welling, 2016) layer to propagate information across the road graph ($r \to r$), and a Long Short-Term Memory (LSTM) (Hochreiter, 1997) layer to capture temporal patterns ($x \to x$) in the history of KF states:

$$\tilde{\mathbf{x}}^{(l)}, \mathbf{h}_t^{(l)} = \text{LSTM}_{x\to x}^{(l)}\left(\hat{\mathbf{x}}^{(l)}, \mathbf{h}_{t-1}^{(l)}\right), \tag{10}$$

$$\tilde{\mathbf{R}}^{(l)} = \text{GCN}_{r\to r}^{(l)}\left(\hat{\mathbf{R}}^{(l)}, \mathbf{A}\right). \tag{11}$$

Finally, the cross message passing layers consist of MLPs which update the user-level features given the road-level features ($x, r \to x$), and vice-versa ($r, x \to r$):

$$\mathbf{x}^{(l)} = \text{MLP}_{x,r\to x}^{(l)}\left(\left[\tilde{\mathbf{x}}^{(l)}, \text{mean\_pool}\left(\tilde{\mathbf{R}}^{(l)}\right)\right]\right), \tag{12}$$

$$\mathbf{R}^{(l)} = \text{MLP}_{r,x\to r}^{(l)}\left(\left[\tilde{\mathbf{R}}^{(l)}, \tilde{\mathbf{x}}^{(l)}\right]\right). \tag{13}$$

Mean pooling is used to summarize the road-level feature vectors in a single feature vector of fixed dimensions. After L blocks, a linear layer $\mathbf{W}_{out}$ projects the road-level features into logits and a softmax function is used to convert these to probabilities: $P_\phi(\mathbf{r}) = \text{softmax}\left(\mathbf{W}_{out}\mathbf{R}^{(L)}\right)$.

We train the model using a cross-entropy loss to match the predicted probabilities $p_\phi(\mathbf{r})$ with the road segments r^*_{oracle} selected by the bidirectional Viterbi *Oracle*:

$$L_{\text{CE}} = \text{CE}(r^*_{\text{oracle}}, P_\phi(\mathbf{r})). \tag{14}$$

Finally, the most likely road segment is selected as in Equation 1 based on the cost function $J_{\text{TGNN}}(r_i) = -P_\phi(r_i)$.

3.4. Learning to Predict the Road Uncertainty

The road network KF update described in Equation 3 requires not only the mean $\mathbf{z}$ of the selected road segment, but also its covariance $\mathbf{V}$. The covariance is a diagonal matrix with two non-zero values: the variance parallel to the road ($\sigma^2_\parallel$) and the variance perpendicular to the road ($\sigma^2_\perp$).

While these parameters can be tuned offline using a calibration set, we instead equip our neural network with an additional output head to predict them online. This allows the TGNN to dynamically adapt its confidence in the road network prediction depending on the current scenario, for example predicting higher uncertainty in case of cluttered or ambiguous road network structure, and lower uncertainty otherwise. The uncertainty prediction head is implemented as a linear projection head on top of the features at the last layer of TGNN:

$$\left[\sigma^2_\parallel, \sigma^2_\perp\right] = \exp\left(\mathbf{W}_\sigma \mathbf{x}^{(L)}\right). \tag{15}$$

To train the network for uncertainty prediction we use the (MSE) loss between the output of the KF and the ground truth user state. This is possible because the KF is fully differentiable (Shlezinger et al., 2024). The complete loss function used to train the TGNN is a combination of the road uncertainty and road selection losses, with a parameter λ to balance the two components:

$$L_{\text{MSE}} = \text{MSE}(\mathbf{x}_{\text{gt}}, \mathbf{x}^+_{\text{RN}}). \tag{16}$$

$$L = L_{\text{CE}} + \lambda L_{\text{MSE}}, \tag{17}$$

Table 1. Quantitative results comparing our method (in **bold**) to different baselines. The first two baselines do not use road network data, while the *Oracle* makes use of future information.

	Road Select.	Road Covariance	HE@50^{th} [m]	HE@95^{th} [m]
LS	-	-	20.43	115.97
KF	-	-	10.75	77.23
KF	Oracle	0	3.72	11.40
KF	Instant	Grid Search	7.96	68.86
KF	Viterbi	Grid Search	8.02	68.27
KF	**TGNN**	**TGNN**	8.74 $\pm$ 0.15	55.02 $\pm$ 2.21

Table 2. Ablating road selection or covariance prediction with non-learnable components (Viterbi or grid search, respectively).

	Road Select.	Road Covariance	HE@50^{th} [m]	HE@95^{th} [m]
KF	Viterbi	TGNN	8.69 $\pm$ 0.18	67.08 $\pm$ 3.49
KF	TGNN	Grid Search	8.36 $\pm$ 0.38	63.72 $\pm$ 9.00
KF	**TGNN**	**TGNN**	8.74 $\pm$ 0.15	55.02 $\pm$ 2.21

Table 3. Comparison between the *TGNN* architecture against the ablated variants *GNN* (without temporal LSTM) and *MLP* (without graph convolution and LSTM).

Neural Network	HE@50^{th} [m]	HE@95^{th} [m]
MLP	9.13 $\pm$ 0.31	59.68 $\pm$ 3.94
GNN	8.82 $\pm$ 0.17	56.90 $\pm$ 2.83
TGNN	8.74 $\pm$ 0.15	55.02 $\pm$ 2.21

Table 4. Effects of using subsets of the available input features with the proposed TGNN method.

Features	HE@50^{th} [m]	HE@95^{th} [m]
Viterbi Prior	10.16 $\pm$ 0.15	72.79 $\pm$ 1.21
Viterbi Prior + Distances	8.64 $\pm$ 0.17	61.43 $\pm$ 3.52
Viterbi Prior + Distances + Road Type + Max Speed	8.65 $\pm$ 0.21	56.29 $\pm$ 1.66
All	8.74 $\pm$ 0.15	55.02 $\pm$ 2.21

4. Results

4.1. Experimental Setup

Dataset In this manuscript we employ the real-world GNSS dataset introduced by Jalalirad et al. (2023). The data consists of multiple unconnected drives from four cities in different countries. For each drive, we have access to the ground-truth user location as well as GNSS pseudo-range measurements corrected for known error terms. We supplement the GNSS dataset with road network information using the open-source data provided by OpenStreetMap contributors (2017). The road network topology is described with an undirected road-level graph, where nodes represent either intersections or a curvature point in the road. Each road segment describes the center of the road, while the positions and offsets of multiple lanes in the road are not included. The following features are additionally paired to each road segment: coordinates of the two end points, segment length, number of lanes, maximum driving speed, roadway type (highway, primary, residential, etc.), and whether it is a one-way street. This information is not always present, in which case we set it to a default value when using as a neural network input. Other input features include the KF mean and uncertainty, and the probabilities from the Viterbi algorithm. The dataset is split in three folds with no regional overlap, and a leave-one-out cross-validation method is used in our experiments, averaging the scores across the holdout folds. Each fold includes both open-sky and challenging urban scenarios. We provide in Appendix B additional details on the processing of GNSS and Road Network data, as well as the complete list of features used as input to the *TGNN*.

Evaluation Metrics We evaluate the proposed methods and baselines using the cumulative distribution function (CDF) of the Horizontal Error (the distance between the predicted and target location). Specifically, we report results for the 50$^{\text{th}}$ percentile (median) and the 95$^{\text{th}}$ percentile of the CDF, with the latter being the primary metric. The horizontal error at higher percentiles reflects the performance in challenging scenarios where significant errors occur. These errors, in the order of $\sim 60m$, can severely impact downstream applications of GNSS positioning, such as lane estimation and side-of-the-street detection. Slight changes in HE@50, which is usually under ~ 10 meters, are relatively less important in this context. For each result from a learnable method, we report mean KPI and standard deviation over 10 seeds.

Methods and baselines The KF is initialized using the least squares method described in Jalalirad et al. (2023). We evaluate four different strategies to select the road segment for the KF update: *Instant* and *Viterbi* as baselines, the bidirectional Viterbi *Oracle*, and our proposed *TGNN* solution. Appendix C reports all implementation details and selected hyper-parameters to reproduce our experiments. Note that the existing methods discussed in Section 2 are either heuristics for KF snapping similar to our *Instant* baseline, or consider the task of map matching instead of user positioning, and are therefore not suitable baselines for our experiments.

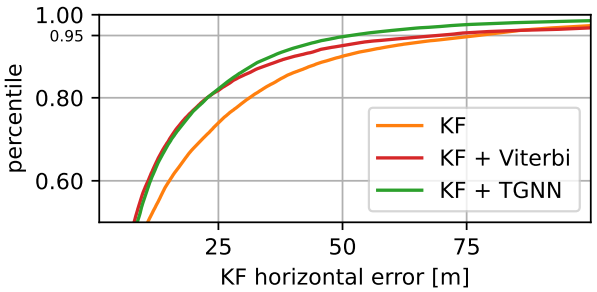


Figure 4. Horizontal Error CDF for different methods.

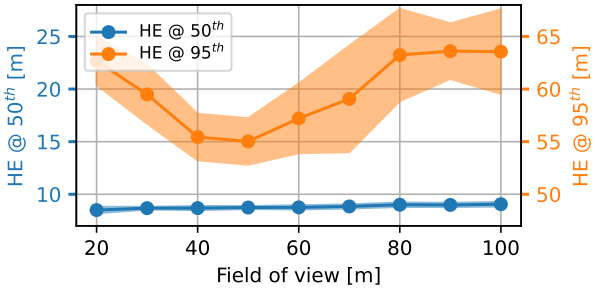


Figure 5. Effect of changing the field of view of the proposed TGNN model on both the 50^{th} and 95^{th} horizontal error percentile. While the error does not change in benign scenarios, the performance in challenging environment significantly decays when choosing too small or large fields of view.

4.2. Quantitative Results

We report the end-to-end positioning error averaged over folds for different positioning algorithms in Table 1, where LS refers to the Least Squares algorithm for instantaneous positioning. We find that even simple techniques that use road network data (*Instant* and *Viterbi*) improve against the GNN-only KF, resulting in a 10 meter decrease in localization error at the 95^{th} percentile. Our proposed *TGNN* approach decreases the positioning error in challenging scenarios by an additional 13 meters, with less than 1 m error increase at the 50^{th} percentile. This constitutes a meaningful increase in performance for the challenging scenario, where differences of 13 meters can make a significant difference in downstream applications. The low standard deviation over 10 random weight initializations displays the robustness of the proposed approach.

We also report the performance for the *Oracle* approach, in which we assume a perfect road network classifier with covariance set to zero (i.e. completely trusting the road network). Despite the use of the *Oracle*, the positioning error does not go to zero because of three reasons. First, there are mismatches between map information and the actual road network structure. Second, we do not take road width into account and snap to the center of the road, because lane level information is often unavailable. Third, while generally robust, there are occasional failures of the bidirectional Viterbi *Oracle*. This highlights that performance could be improved by increasing the fidelity and number of features available for the road network data.

In Figure 4 we include a visualization of the horizontal error CDFs. We observe that both methods using road network measurements improve over the GNSS-only KF at the median as well as high percentiles, The proposed *TGNN* approach further improves over the Viterbi baseline on the challenging environments above the 90^{th} percentiles.

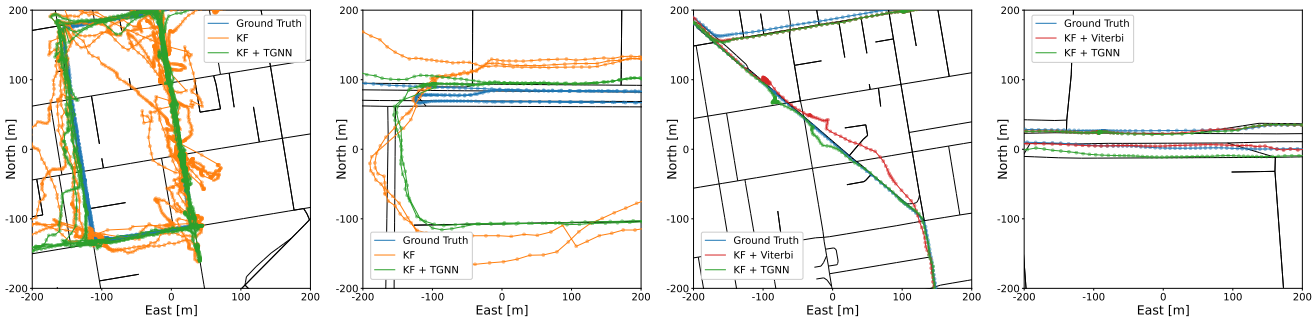
4.3. Ablations

To evaluate the impact of out implementation choices, we conduct four different ablation studies.

Predictor type First, we investigate whether both the road selection or covariance prediction components benefit from being modeled through a neural network, by replacing either of them with their non-learned counterparts: *Viterbi* and grid search. As shown in Table 2, the two ablations result in degraded performance, suggesting that both road selection and covariance estimation tasks can be better tackled with a learned model. In particular, differently from grid search, the learned covariance estimation module can dynamically tune the predicted uncertainty at test time, and better adapt to changing test scenarios.

Architecture Second, we assess the impact of different modules in the *TGNN* architecture by ablating them one by one. The results of these ablations are shown in Table 3. The *GNN* model is obtained by removing the temporal LSTM component. This model cannot capture dynamics that span across multiple time-steps such as changes in vehicle speed and heading, resulting in a 2 meters increase in the positioning error in challenging scenarios. We name *MLP* a simpler architecture variation where the KF state values are simply concatenated to the road features and processed by a sequence of MLPs. Without graph convolutions, information cannot be propagated across road segments in the graph, and the neural network must predict the probability for each segment independently of the others. This results in a further increase in positioning error by 3 meters. We therefore confirm that the neural network must be equipped with both temporal (LSTM) and spatial (GCN) reasoning capabilities to improve its performance. We refer to Appendix D for implementation details on the ablated architectures.

Input features Third, we test the effect of ablating several road input features, as shown in Table 4. The costs assigned to each road using Equations 2 and 4 are the most impactful, with additional features such as road type and maximum driving speed being also important. However, the best results are obtained when using all the available features (listed in Appendix B). Interestingly, a model with



(a) TGNN outperforms GNSS-only (b) TGNN underperforms GNSS-only (c) TGNN outperforms Viterbi (d) TGNN underperforms Viterbi

Figure 6. Qualitative results for the proposed TGNN model compared against the Viterbi baseline, zoomed in to interesting parts of the drive. Full images can be found in supplementary material Appendix E .

only the *Viterbi* prior input features, which describe the probability of the roads assigned in the previous time step, can also improve over the standard KF. This could be the case in situations where the GNSS measurements are noisy, while the road selection task is trivial (only one road is present in the receptive field) and can therefore improve the KF state.

Field of view Finally, we experiment with changing the field of view for our method, which defines the set of road segment candidates for selection based on a radius around the user position estimate. We visualize in Figure 5 how changing this parameter impacts the positioning error. The performance at the 95th percentile degrades quickly when the field of view is too large or too small. In the former case, this is likely because the correct road to select shifts out of focus more frequently, while in the latter case, the number of candidate road segments increases significantly, which might make the classification task more challenging for the neural network. The field of view is fixed to 50 meters in all other experiments.

4.4. Qualitative Results

We include a qualitative analysis to visualize and discuss patterns in the model behavior. Figure 6 zooms in on interesting scenarios, while in Appendix E we include the qualitative results for the full drives. We first compare the proposed KF + *TGNN* against a simple KF using only GNSS measurements. In Figure 6a we show a drive where *TGNN* significantly outperforms the GNSS-only baseline. In this challenging urban scenario the GNSS signal is very noisy, which results in unstable and incorrect predictions for the simple KF baseline. By using the road network measurements, the *TGNN* can compare the noisy GNSS measurements with the prior knowledge about the road structure, and significantly reduce the localization error. In Figure 6b we show *the only drive* in which the *TGNN* underperforms the simple KF baseline. We notice how the large GNSS noise causes both solutions to drift away from the ground-truth trajectory. However, the *TGNN* solution confidently snaps to a road segment well-aligned to the noisy GNSS measurements but disconnected to the correct segment, which prevents the KF from recovering its state over time. We hypothesize that this performance drift could be significantly limited by incorporating an IMU component in the KF, as measuring the vehicle heading could prevent selecting road segments with incorrect heading.

Next, we compare the *TGNN* against the *Viterbi* baseline to process road network measurements. In Figure 6c we show an example in which the proposed method outperforms the baseline by confidently following the correct road segment. Figure 6d describes a road network with a few parallel roads with multiple lanes each. In this case, the *TGNN* incorrectly selects a road segment with a consistent offset with respect to the ground truth trajectory. In addition to using IMU measurements, we believe that a lane-level representation of the network might reduce these errors, as the Kalman Filter would not be forced to follow one of the two road center lanes, but could instead select an intermediate lane segment better matching the current observations.

5. Conclusions

In conclusion, this paper presents a novel method for enhancing GNSS-based vehicle positioning by integrating road network information into a KF using a Temporal Graph Neural Network. To the best of our knowledge, this is the first approach employing a neural network to process the road network data in a positioning system. Our approach significantly reduces horizontal positioning errors in real-world challenging scenarios, demonstrating the effectiveness of our data-driven solution compared to other methods.

Future work could explore using a more fine-grained representation of the road network, such as including lane-level information, or using a visual representation of the map. Another novel research direction would be learning to filter or correct the GNSS measurements based on the road network structure before using them in the KF update step.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning for GNSS positioning. Positioning systems can be embedded in a variety of real-world applications. Positioning systems deployed for critical applications such as autonomous driving should be thoroughly tested and integrated with strong safety measures, both software and hardware, to ensure the highest standard of safety for the consumer.

There may be other potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Atia, M. M., Hilal, A. R., Stellings, C., Hartwell, E., Toonstra, J., Miners, W. B., and Basir, O. A. A low-cost lane-determination system using gnss/imu fusion and hmm-based multistage map matching. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3027–3037, 2017.

Bevis, M., Businger, S., Chiswell, S., Herring, T. A., Anthes, R. A., Rocken, C., and Ware, R. H. Gps meteorology: Mapping zenith wet delays onto precipitable water. *Journal of Applied Meteorology (1988-2005)*, pp. 379–386, 1994.

Bidikar, B., Rao, G. S., and Ganesh, L. Sagnac effect and set error based pseudorange modeling for gps applications. *Procedia Computer Science*, 87:172–177, 2016.

Boeing, G. Osmnx: A python package to work with graph-theoretic openstreetmap street networks. *Journal of Open Source Software*, 2(12):215, 2017. doi: 10.21105/joss.00215. URL https://doi.org/10.21105/joss.00215.

Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems*, 33:17766–17778, 2020.

El Najjar, M. E. and Bonnifait, P. A road-matching method for precise vehicle localization using belief theory and kalman filtering. *Autonomous Robots*, 19:173–191, 2005.

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

EUSPA. *EO and GNSS Market Report*. Publications Office of the European Union, 2024. doi: 10.2878/73092.

Feng, J., Li, Y., Zhao, K., Xu, Z., Xia, T., Zhang, J., and Jin, D. Deepmm: Deep learning based map matching with data augmentation. *IEEE Transactions on Mobile Computing*, 21(7):2372–2384, 2020.

Gao, X., Luo, H., Ning, B., Zhao, F., Bao, L., Gong, Y., Xiao, Y., and Jiang, J. Rl-akf: An adaptive kalman filter navigation algorithm based on reinforcement learning for ground vehicles. *Remote Sensing*, 12(11):1704, 2020.

Goh, C. Y., Dauwels, J., Mitrovic, N., Asif, M. T., Oran, A., and Jaillet, P. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 776–781. IEEE, 2012.

Guo, C. and Tu, W. A novel self-learning gnss/ins integrated navigation method. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pp. 168–179, 2021.

Han, K., Lee, S., Song, Y.-J., Lee, H.-B., Park, D.-H., and Won, J.-H. Precise positioning with machine learning based kalman filter using gnss/imu measurements from android smartphone. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pp. 3094–3102, 2021.

Hochreiter, S. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Hu, H., Qian, S., Ouyang, J., Cao, J., Han, H., Wang, J., and Chen, Y. Amm: an adaptive online map matching algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):5039–5051, 2023.

Ioffe, S. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Jagadeesh, G. R. and Srikanthan, T. Online map-matching of noisy and sparse location data with hidden markov and route choice models. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2423–2434, 2017.

Jalalirad, A., Belli, D., Major, B., Jee, S., Shah, H., and Morrison, W. Gnss positioning using cost function regulated multilateration and graph neural networks. In *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*, pp. 3049–3061, 2023.

Kanhere, A. V., Gupta, S., Shetty, A., and Gao, G. Improving gnss positioning using neural-network-based corrections. *NAVIGATION: Journal of the Institute of Navigation*, 69(4), 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Klobuchar, J. Ionospheric time-delay algorithm for single-frequency gps users. *IEEE transactions on aerospace and electronic systems*, 23(3):325–331, 1987.

Li, S., Mikhaylov, M., Mikhaylov, N., and Pany, T. Deep learning based kalman filter for gnss/ins integration: Neural network architecture and feature selection. In *2023 International Conference on Localization and GNSS (ICL-GNSS)*, pp. 1–7. IEEE, 2023.

Li, W., Chen, Y., Wang, S., Li, H., and Fan, Q. A novel map matching method based on improved hidden markov and conditional random fields model. *International Journal of Digital Earth*, 17(1):2328366, 2024.

Mohanty, A. and Gao, G. Tightly coupled graph neural network and kalman filter for smartphone positioning. *NAVIGATION: Journal of the Institute of Navigation*, 71 (4), 2024.

Neamati, D., Gupta, S., Partha, M., and Gao, G. Neural city maps for gnss nlos prediction. In *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*, pp. 2073–2087, 2023.

OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

Quddus, M. and Washington, S. Shortest path and vehicle trajectory aided map-matching for low frequency gps data. *Transportation Research Part C: Emerging Technologies*, 55:328–339, 2015.

Revach, G., Shlezinger, N., Ni, X., Escoriza, A. L., Van Sloun, R. J., and Eldar, Y. C. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70: 1532–1547, 2022.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Shlezinger, N., Revach, G., Ghosh, A., Chatterjee, S., Tang, S., Imbiriba, T., Dunik, J., Straka, O., Closas, P., and Eldar, Y. C. Ai-aided kalman filters. *arXiv preprint arXiv:2410.12289*, 2024.

Siemuri, A., Selvan, K., Kuusniemi, H., Välisuo, P., and Elmusrati, M. S. Improving precision gnss positioning and navigation accuracy on smartphones using machine learning. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pp. 3081–3093, 2021.

Skog, I. and Handel, P. In-car positioning and navigation technologies—a survey. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):4–21, 2009.

Song, H. Y. and Lee, J. H. A map matching algorithm based on modified hidden markov model considering time series dependency over larger time span. *Heliyon*, 9(11), 2023.

Suzuki, T. and Amano, Y. Nlos multipath classification of gnss signal correlation output using machine learning. *Sensors*, 21(7):2503, 2021.

Tang, Y., Jiang, J., Liu, J., Yan, P., Tao, Y., and Liu, J. A gru and akf-based hybrid algorithm for improving ins/gnss navigation accuracy during gnss outage. *Remote Sensing*, 14(3):752, 2022.

van Diggelen, F. End game for urban gnss: Google's use of 3d building models. *Inside GNSS*, 2021.

Velaga, N. R., Quddus, M. A., Bristow, A. L., and Zheng, Y. Map-aided integrity monitoring of a land vehicle navigation system. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):848–858, 2012.

Zhang, G., Xu, P., Xu, H., and Hsu, L.-T. Prediction on the urban gnss measurement uncertainty based on deep learning networks with long short-term memory. *IEEE Sensors Journal*, 21(18):20563–20577, 2021.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., and Li, H. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9):3848–3858, 2019.

Zhong, Q. and Groves, P. Optimizing los/nlos modeling and solution determination for 3d-mapping-aided gnss positioning. In *ION GNSS+ 2023*, 2023.

Zhong, Y., Hu, R., Bai, X., Li, X., Hsu, L.-T., and Wen, W. Enhancing gnss positioning accuracy for road monitoring systems: A factor graph optimization approach aided by geospatial information. *IEEE Transactions on Instrumentation and Measurement*, 2024.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

A. Deriving z, H and V

For Equation 2, we need to define the measurement matrix $\mathbf{H}$, the observation vector $\mathbf{z}$ and the covariance matrix $\mathbf{V}$. The measurement matrix consists of two parts. First we select from the KF state-space vector $\mathbf{x}^+_{\text{GNSS}}$ only the current user position $\mathbf{x}_{\text{pos}}$ in terms of x and y components, which will be updated with the road network measurement. This is obtained through the selection matrix $\mathbf{H}_{\text{select}}$. Secondly, we rotate $\mathbf{x}_{\text{pos}}$ into a perpendicular and parallel component with respect to the selected road using the clockwise rotation matrix $\mathbf{H}_{\text{rot}}$:

$$\mathbf{H} = \mathbf{H}_{\text{rot}}\mathbf{H}_{\text{select}} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \end{bmatrix} \tag{18}$$

where θ is the heading of the selected road r^*. Multiplying $\mathbf{H}$ with $\mathbf{x}^+_{\text{GNSS}}$ thus results in:

$$\mathbf{H}\mathbf{x}^+_{\text{GNSS}} = \mathbf{H}_{\text{rot}}\mathbf{x}_{\text{pos}} = \begin{bmatrix} x_\parallel \\ x_\perp \end{bmatrix}, \tag{19}$$

where $\mathbf{x}_\parallel$ is the parallel and $\mathbf{x}_\perp$ the perpendicular component of the user location projected on the selected road.

The mean of the observation is a vector of size two, also consisting of a perpendicular and parallel component. The mean can be found by rotating the center point of the road $\mathbf{r}^*_{\text{pos}}$ in the clockwise direction as:

$$\mathbf{z} = \mathbf{H}_{\text{rot}}\mathbf{r}^*_{\text{pos}} = \begin{bmatrix} z_\parallel \\ z_\perp \end{bmatrix}. \tag{20}$$

Finally, we soft-threshold $z_\parallel$ in order to project the vehicle onto the closest point on the road, for cases where the orthogonal projection would fall outside the road segment extreme points. This can be done by soft-thresholding $z_\parallel$ using the length of the road segment r^*_l:

$$z^*_\parallel = \text{sign}\left(z_\parallel\right) \max\left(|z_\parallel| - \frac{1}{2}r^*_l, 0\right). \tag{21}$$

The covariance of the road observation can then be simply expressed with two variables along the diagonal as:

$$\mathbf{V} = \begin{bmatrix} \sigma^2_\parallel & 0 \\ 0 & \sigma^2_\perp \end{bmatrix}, \tag{22}$$

where $\sigma^2_\parallel$ is the variance of the road observation parallel to the road and $\sigma^2_\perp$ is the variance perpendicular to the road. These parameters can be tuned on some calibration data or predicted with a neural network, as we will discuss later in this section.

B. Dataset

We provide in this section additional details regarding the processing of GNSS and Road Network data for our experiments.

GNSS data User location and time was obtained from geodetic-survey grade receiver with a mobile phone form factor antenna and an Inertial Measurement Unit. The pseudo-ranges were measured at a 1Hz frequency using the position data and the satellite constellation orbital data, with the receiver's clock bias measured by a stable external reference clock source. Ionospheric and tropospheric delays were corrected using Klobuchar (Klobuchar, 1987) and Saastamoinen (Bevis et al., 1994) models, respectively. Additionally, the Sagnac effect due to the Earth's rotation was corrected for (Bidikar et al., 2016). The time biases between the different GNSS constellations were estimated using a preliminary weighted least squares (WLS) localization and removed, allowing the model to use pseudorange measurements from all constellations. To attain this preliminary WLS, we can make use of a localization acquired by a cellular network or an externally injected position available on the device.

Road network data We use the OSMnx package (Boeing, 2017) to extract the road network for a bounding box around each GNSS drive route. We only retain the driving road network by specifying the OSM tag "highway"[1], which removes

[1] https://wiki.openstreetmap.org/wiki/Key:highway

other structures such as cycleways and footways. The extracted road segments are of varying lengths, with some spanning over 500 meters long while others being just 1 meter. To make the segments length more consistent and easier to process by the various methods, we divide all roads longer than 25 meters into equal segments up to 25 meters in length. As a last step, we convert the graph to its dual, such that all road segments become nodes, and all intersections become edges for the purpose of the graph convolution operations (see Figure 2).

B.1. Input features

We here describe the most useful road and vehicle input features, denotes by $\mathbf{R}$ and $\mathbf{x}$, respectively.

Road Features The most useful road features were found to be: *Distances*, *Road Type*, *Max Speed*, *Road Heading*, *Oneway*, and *Viterbi Prior*. *Distances* consists of the Euclidean distance calculated using Equation 3 and the angular distance calculated using Equation 4. The *Road Type* feature is a one-hot encoding of the road type as provided by OpenStreetMap. While the "highway" tag covers 27 different types of roads, not all of them are relevant to our applications (e.g., via_ferrata). Therefore, we only one-hot encoded the following types: 'motorway', 'motorway_link', 'trunk', 'trunk_link', 'primary', 'primary_link', 'secondary', 'secondary_link', 'tertiary', 'tertiary_link', 'unclassified'[2], 'residential', 'living_street', 'service'. All other types defaulted to a separate value in the one-hot vector. We provide *Max Speed* as the maximum driving speed in meters per second. To the above input features, we add an additional Viterbi prior feature. *Road Heading* describes the road heading in polar coordinates. To that end, we provide the sine and cosine of the azimuth of the road, which automatically encodes the modulo nature of the azimuth. *Oneway* is a simple boolean feature that encodes whether the road is oneway (1) or bidirectional (0). Lastly, to mimic the Viterbi algorithm's capability to track state probabilities over time, we append the probabilities assigned to each road to the road features $\mathbf{R}$ of the next time step. Furthermore, we propagate these probabilities to the k-hop neighbors (following Equation 6) and take the maximum. For each $k \in 1, \dots, K$, this maximum neighboring probability is added as a feature to $\mathbf{R}$ for the next time step.

Vehicle Features The most useful vehicle features were found to be: *Vehicle Heading* and *Position Uncertainty*. *Vehicle Heading* desribes the azimuth of the vehicle's heading in the same way as the *Road Heading* feature, i.e., in terms of the sine and cosine of the azimuth. Furthermore, we also provide the magnitude of the estimated speed in meters per second. This allows TGNN and to compare the *Vehicle Heading* with the *Road Heading* and *Max Speed* features of each road in a straight-forward manner. The *Position Uncertainty* feature encodes the covariance matrix of the KF in terms of the uncertainty about East and North. That is, we provide $\mathbf{V}_{\text{pos}}$ in terms of the three scalars $\sigma^2_{xx}, \sigma^2_{xy}, \sigma^2_{yy}$. Note that providing the other covariance σ^2_{yx} would be redundant with σ^2_{xy}.

C. Hyper-parameters

We report in this section all implementation details and hyper-parameter choices to facilitate reproducing our experimental results. The K-hop distance considered in the Viterbi algorithm is set to $K = 2$ to increase robustness in high-speed scenarios. The cost weighting for the emission probability is set to $\beta = 0.01$, such that a position difference of 100 meters is weighted equally as a difference in heading of 90 degrees. We found this trade-off to show the best results in preliminary experiments with the *Viterbi* baseline, and fixed the hyper-parameter for all our experiments.

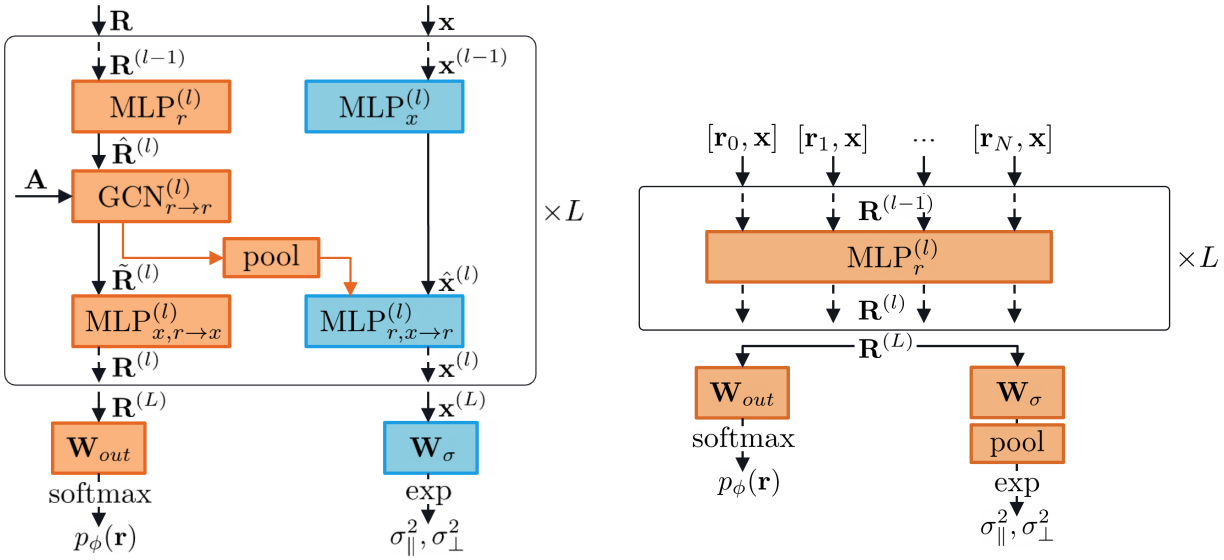
To choose the road measurement variances in methods without uncertainty prediction, we perform a grid search over the values: $\sigma^2_\perp \in \{0, 1, 2, 3, \dots, 10\}$ and $\sigma^2_\parallel \in \{0, 1, 2, 3, \dots, 10, 100, 200, 300, \dots, 1000, \infty\}$. We calculate the horizontal error at the 95th error percentile for each combination on the training set. The combination with the lowest horizontal error at the 95th percentile is subsequently selected to be used on the hold-out test set. This strategy has the limitation that for all fixes under consideration, the same variance terms are used.

The TGNN architecture consists of $L = 4$ blocks and all hidden sizes are set to 32. We use SiLU activation (Elfwing et al., 2018) after each layer paired with batch normalization (Ioffe, 2015). When learning the road variances with TGNN, the cost weighting for the CE + MSE loss is set to $\lambda = 0.01$ such that 100 meters of position distance becomes unit cost. We train TGNN with a batch size of 8 over 5000 iterations. We use the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.001 and a weight decay of 0.001.

[2]From OpenStreetMap: "The word 'unclassified' is a historical artefact of the UK road system and does not mean that the classification is unknown"

Computational Costs The TGNN model has less than 50k parameters and its compute cost amounts to 1.7 MFLOPs, which is low enough to achieve real-time inference performance even on efficient embedded processors.

D. Neural Network Ablations



(a) *GNN* ablation.

(b) *MLP* ablation.

Figure 7. Architecture for the *GNN* and *MLP* ablations.

We provide diagrams describing the neural network for the ablations of the *TGNN* architecture. The *GNN* model (Figure 7) is obtained by removing the temporal LSTM component. This model cannot capture dynamics that span across multiple time-steps such as changes in vehicle speed and heading. We name *MLP* (Figure 8) a simpler architecture variation where the KF state values are simply concatenated to the road features and processed by a sequence of MLPs. Without graph convolutions, information cannot be propagated across road segments in the graph, and the neural network must predict the probability for each segment independently of the others.

E. Full Qualitative Figures

We include in Figures 9 to 12 full-size qualitative visualizations of model performance in different drives. Figures 9 and 10 compare the proposed KF + TGNN approach against a simple KF using only GNSS measurements. Figures 11 and 12 compare KF + TGNN against a the KF + Viterbi baseline to process road network measurements.

Figure 8. Full drive where TGNN outperforms GNSS-only.

Figure 9. Full drive where TGNN underperforms GNSS-only.

Figure 10. Full drive where TGNN outperforms Viterbi.

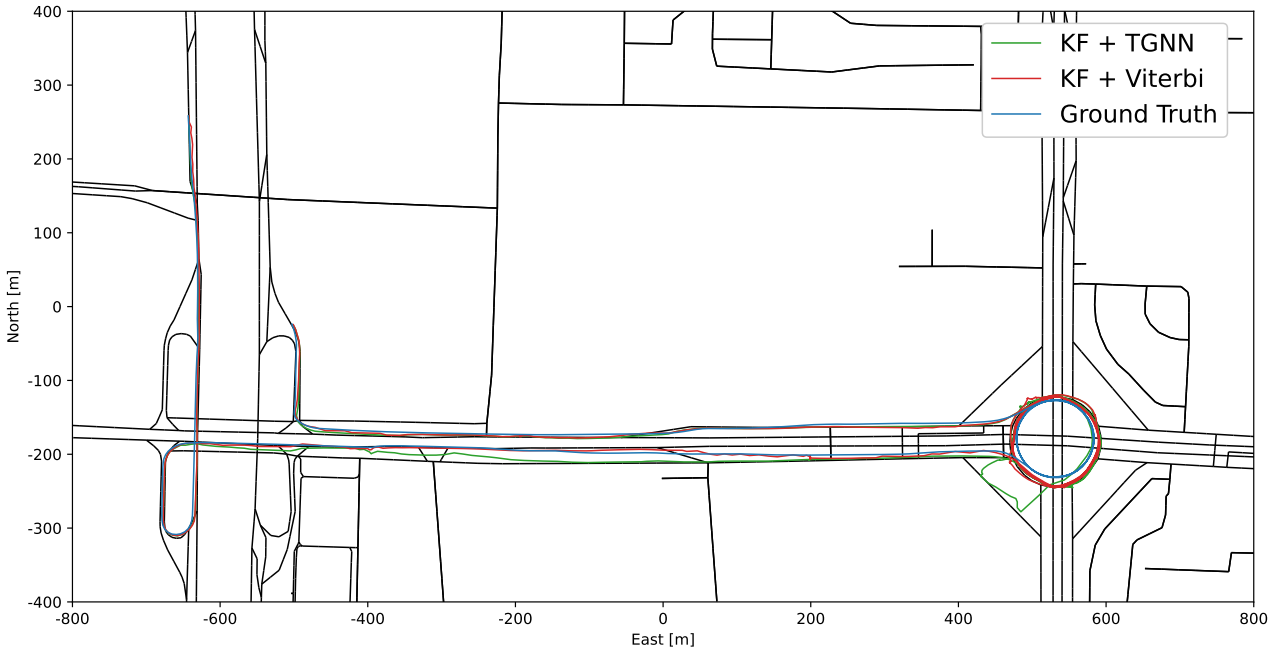


Figure 11. Full drive where TGNN underperforms Viterbi.