POLITECNICO DI TORINO
Repository ISTITUZIONALE

Hybrid Tree Based Models for GNSS Position Error Estimation

Original
Hybrid Tree Based Models for GNSS Position Error Estimation / Modi, S., Sulli, V., Valentini, F., D'Innocenzo, A.. - ELETTRONICO. - (2026), pp. 1-6. (ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC) Noordwijk (NL) 20-22 May 2026) [10.1109/navitec202669090.2026.11692521].

Availability:
This version is available at: 11583/3016395 since: 2026-10-07T10:50:31Z

Publisher:
IEEE

Published
DOI:10.1109/navitec202669090.2026.11692521

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

28 November 2026

Hybrid Tree Based Models for GNSS Position Error Estimation

Sikhar Modi*†‡, Vincenzo Sulli†‡, Francesco Valentini†‡, Alessandro D'Innocenzo†
*DAUIN, Politecnico di Torino, Torino, Italy;
†DISIM and Centre of Excellence Ex-EMERGE - University of L'Aquila, L'Aquila, Italy;
‡Radiolabs, Rome, Italy;
Email: sikhar.modi@outlook.com, {vincenzo.sulli; francesco.valentini}@radiolabs.it,
alessandro.dinnocenzo@univaq.it

***Abstract*—This paper tackles GNSS position error estimation in challenging environments where reliable error bounds are required for real-time applications. While regression trees and random forests can model nonlinear relations between GNSS observables and position error, their accuracy is limited by coarse input partitioning. We introduce two hybrid models, RT-LLR and RF-LLR, which combine tree-based partitioning with local linear regression on lagged GNSS observables to perform piecewise linear modeling of spatial–temporal error dependencies. The methods are evaluated over 20 runs using RMSE of the x, y, and z position error components. The hybrids tie or outperform the baseline trees in about 75% of x-axis cases, 95% of y-axis cases, and 85% of z-axis cases, achieving relative RMSE reductions up to roughly 22% horizontally and 17% vertically, with comparable runtime. These results show that hybrid tree–linear models provide more accurate and stable GNSS error estimates, making them suitable for deployment in low-cost receivers and cooperative positioning systems.**

***Index Terms*—Galileo, GNSS, GPS, Local Linear Regression, Machine Learning, Piecewise Affine Models, Position Error Estimation, Position Velocity Time, Random Forests, Regression Trees.**

I. Introduction

Global Navigation Satellite Systems (GNSS) form the backbone of modern positioning, navigation, and timing (PNT) infrastructure and serve as enablers in modern mobility to perform various critical functions. However, in environments such as urban areas, their reliability deteriorates significantly due to Non-Line-of-Sight (NLOS) conditions between the receiver and the satellite, with signal propagation further affected by multipath and atmospheric effects, including ionospheric and tropospheric delays [1], [2]. In this context, accurate and reliable **GNSS position error estimation** becomes crucial to ensure robust navigation performance in challenging environments. Traditional models, such as Least Squares and Kalman Filtering, are primarily used for estimating the receiver position and velocity by fusing pseudorange and Doppler measurements under linear and Gaussian noise assumptions - assumptions that rarely hold under dynamic, multipath-affected conditions [3], [4]. Recent works have explored data-driven methods for modeling GNSS error dynamics, yet the design of interpretable and computationally efficient solutions remains an open research challenge. This study addresses that challenge by proposing **hybrid machine learning methods** explicitly tailored for **GNSS position error estimation**, combining the adaptability of tree-based learning with the interpretability of system-theoretic formulations.

This publication is part of the project PNRR-NGEU which has received funding from the MUR – DM 630/2024

II. Literature Review

Learning-based models using only GNSS data have emerged to address the aforementioned challenges. Caparra et al. trained deep neural networks to estimate 3D GNSS errors from basic inputs like Dilution of Precision (DOP), Carrier-to-Noise ratio (C/N_0), and satellite count, achieving robust results even under poor visibility [5]. Lan et al. applied Long Short-Term Memory (LSTM) networks to capture temporal error dynamics, treating horizontal errors as scalars for improved performance in harsh conditions [6].

Tree-based methods offer interpretability and efficiency. Qi et al. (2024) used Random Forest regression to estimate multipath parameters and mitigate effects in urban GNSS navigation, reducing 2D and 3D positioning errors by 8.5% over traditional techniques [7]. Zhang et al. employed Random Forests for classifying GNSS conditions (e.g., open sky vs. blockage) to inform error mitigation [8]. Niu et al. applied bagged decision trees to detect error types, enhancing Kalman filter outputs [9].

III. Proposed Hybrid Modeling Framework

To broaden the scope of applicability and impact of data-driven techniques, this work introduces a novel hybrid modeling framework that merges machine learning adaptability with the interpretability of system-theoretic formulations. The proposed models — **Piecewise Affine Regression Trees and Random Forests with Local Linear Models on Lagged Exogenous Inputs (RT-LLR and RF-LLR)** — are designed specifically for **GNSS position error estimation**. The core principle derives from a formulation referred to as **Tree-Guided Piecewise Linear Regression over Lagged Exogenous Inputs**, which extends the classical ARX (AutoRegressive with Exogenous inputs) paradigm as explained in **Section IV** by removing autoregressive feedback dependencies. Unlike conventional ARX systems that rely on past output values, this framework estimates position errors solely

from lagged exogenous inputs, since the regressive terms of the position error are unknown.

To enhance the modeling capabilities beyond the traditional ARX framework, the proposed hybrid approaches introduce targeted modifications to the local regression methodology. In the original Learning ARX formulation, regression trees are combined with affine models obtained via quadratic programming in each leaf. To minimize the computational cost, the Tree-Guided Local Linear Regression (RT-LLR) method replaces this optimization step with a regularized ridge regression procedure for estimating leaf-specific local linear models. This introduces a checking mechanism for feature selection by focusing only on the relevant non-constant input variables per leaf, which enhances model sparsity and interpretability. Additionally, the proposed methods include a fallback mechanism to default to the original prediction in cases where no local model is available for a leaf. Building on this, the Random Forest-Guided Local Linear Regression (RF-LLR) framework generalizes the concept by estimating independent local linear models within each leaf of every tree in the forest, aggregating their predictions to improve robustness and accuracy. This ensemble approach combines the adaptability of Random Forests with the fine granularity of local linear modeling, yielding a powerful and interpretable hybrid tool tailored for robust GNSS position error estimation.

IV. Learning ARX models via regression trees

This section describes the Classification and Regression Tree (CART) algorithm [10], which forms the foundation of the hybrid identification procedure adopted in this work. Understanding CART is essential because it provides the partitioning mechanism that is proposed in [11], to identify from a dataset a model of a dynamical system that combines the CART algorithm and the ARX identification. We used the CART algorithm to generate the piecewise local models in the proposed models.

For the sake of simplicity, we only briefly recall the partitioning algorithm of CART, and refer the reader to [10] for more details. In a supervised learning framework, we consider an input dataset $\mathcal{X} = \{x_i\}_{i=1}^N$ and an output dataset $\mathcal{Y} = \{y_i\}_{i=1}^N$ of N samples each, where $y_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^n$. The final goal of CART is to identify a function $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}$ to estimate $\hat{y} = \mathcal{T}(x)$. To this aim, the dataset is iteratively partitioned according to a tree graph structure grown during the algorithm, by a set of hyper-rectangles $R_1, \dots, R_{|\mathcal{T}|}$ that induce a partition of the input space $\mathbb{R}^n$: to each hyper-rectangle corresponds one (and only one) leaf ℓ_j of the grown tree graph. Then, $\hat{y}$ is estimated in each leaf ℓ_j using a constant $c_j = \text{ave}(y_i | x_i \in \ell_j)$ given by the average of the samples in the partition.

We now briefly illustrate the approach proposed in [11], which combines the CART algorithm and ARX identification to build a model of a dynamical system starting from a dataset of trajectories. Consider a set of N sample trajectories $\{z_i(0), \dots, z_i(r), d_i(0), \dots, d_i(r-1)\}_{i=1}^N$, with a measurement variable $z_i(t) \in \mathbb{R}, t = 0, \dots, r$ and an exogenous variable $d_i(t) \in \mathbb{R}^\nu, t = 0, \dots, r-1$. Let us define an input dataset $\mathcal{X} = \{x_i\}_{i=1}^N$ with $x_i := (z_i(0), \dots, z_i(r-1), d_i(0), \dots, d_i(r-1)) \in \mathbb{R}^n, n = r(\nu+1)$, and an output dataset $\mathcal{Y} = \{y_i\}_{i=1}^N$ with $y_i := z_i(r) \in \mathbb{R}$. In [11] we derived Algorithm 1: we first applied the standard CART algorithm to a training dataset $\mathcal{X}$ and $\mathcal{Y}$, thus obtaining a regression tree $\mathcal{T}$ with leaves $\ell_1, \dots, \ell_{|\mathcal{T}|}$ and corresponding hyper-rectangles $R_1, \dots, R_{|\mathcal{T}|}$ that partition the input space; then, instead of using the constant predictor c_j in each leaf, we proposed to assign an affine predictor $m_j \in \mathbb{R}^n, m_{j,0} \in \mathbb{R}$ to each leaf, thus obtaining $\mathcal{T}(x)$ the prediction of the regression tree given by $\mathcal{T}(x) = \sum_{j=1}^{|\mathcal{T}|} (m_j^T x + m_{j,0}) \cdot I\{x \in R_j\}$.

Algorithm 1 [11]

1: **Input:** dataset $(\mathcal{X}, \mathcal{Y})$
2: Apply CART algorithm to $(\mathcal{X}, \mathcal{Y})$, obtaining as output a tree structure $\mathcal{T}$ with leaves $\ell_1, \dots, \ell_{\mathcal{T}}$
3: **for all** $j = 1, \dots, |\mathcal{T}|$ **do**
4: Compute in leaf ℓ_j an affine predictor $m_j \in \mathbb{R}^n, m_{j,0} \in \mathbb{R}$ by solving the quadratic program

$$\min_{m_j, m_{j,0}} \sum_{x_i \in \ell_j} (y_i - m_j^T x_i - m_{j,0})^2 \tag{1}$$

5: **end for**

V. GNSS Data Generation Method

The GNSS dataset collection involved a simulative Hardware-in-the-Loop (HiL) approach, in which signals from the satellite infrastructure are emulated at Radio Frequency (RF signals) and used to feed the GNSS receiver. The setup was achieved by using some of the capabilities of the P-CAR (PNT Center for Automated Road-Transport) laboratory [12]. Namely, the P-CAR's primary objective is to support testing and validation of Connected and Automated Vehicle (CAV) functionalities exploiting PNT data. Using an Everything-in-the-Loop (XiL) methodology within a multi-sensor framework (e.g., GNSS, IMU, cameras, radar, communication systems), P-CAR enables the execution of realistic, repeatable test scenarios, including fault and degradation injection. In this work, P-CAR was specifically used for GNSS infrastructure emulation.

A. HiL Setup and Workflow

The data collection process involves an HiL simulation composed of a configuration and pre-processing phase, followed by real-time test execution. **Figure 1** illustrates the setup and overall workflow. The main tools used in this process are:

- **Vehicle simulator**: An instance of CarMaker simulates both the road environment and vehicle dynamics, providing the required ground truth data.
- **Multipath generator**: The custom tool GNSSTE computes GNSS multipath components based on the environment, ground truth, and satellite ephemerides. It uses a ray-tracing approach to assess both Line-of-Sight

(LOS) and NLOS conditions and corresponding multipath effects.

- **GNSS emulator**: The StellaNGC tool by M3 Systems emulates the GNSS signal environment, using satellite ephemerides and vehicle ground truth. It can simulate ideal conditions or incorporate global/local hazards. Global hazards cover i) satellite clock and ephemeris errors, and ii) atmospheric effects (ionosphere and troposphere delays), while local hazards concern LOS/NLOS and multipath conditions. Satellite-Based Augmentation System (SBAS) corrections can also be included to mitigate global hazards.

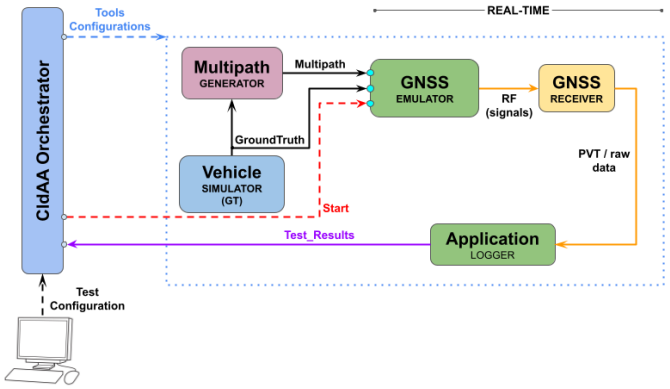

Fig. 1. Workflow for the dataset collection

The testing workflow is automated through the Cloud-based Architecture and Test Automation (CldAA) Orchestrator, which handles configuration distribution across potentially geo-distributed tools, coordinates test execution, and manages output data collection for reporting.
In the first step, the tool suite is properly configured for the desired test-case (e.g. in nominal conditions or with faults/degradations). Then, the vehicle simulation is executed to generate the ground truth data needed by GNSSTE to compute multipath components. Ground truth and multipath data are used in the GNSS emulator to generate realistic RF signals. In the real-time phase, the RF (signals) output of the emulator feeds the GNSS receiver, and the produced raw data (raw data/PVT) are logged (by the "Application logger" block) and stored in a dedicated database.

B. Dataset Configurations

The GNSS emulator was configured with a 10 Hz internal update rate. The receiver used is a customized Raspberry Pi equipped with a U-blox ZED-F9P-05B module, set to receive both L1C/A (GPS) and E1B/C (Galileo) signals, and to provide PVT fixes at 1 Hz (one order of magnitude below the emulator update rate).
The dataset consists of 20 test runs, grouped into four distinct scenarios, with each scenario executed five times (each run lasts one hour). Global hazards have been derived from real-world, RINEX and SP3, files corresponding to five different dates in 2021 (Jan 20, Mar 7, Jun 15, Sep 26, and Nov 13, between 3:00 p.m. and 4:00 p.m.). The virtual proving ground used in testing is located at 41.9525° N, 13.6210° E and, for scenarios involving multipath, includes buildings and vegetation along the track (see **Figure 2**).

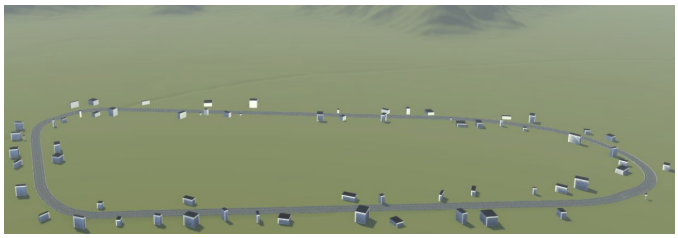

Fig. 2. Virtual proving ground

Table I summarizes the used satellite segment configurations. The "00" scenario refers to the L1 frequency (GPS) alone, in presence of multipath conditions. The "01" scenario is a repetition of "00", but with the inclusion of SBAS signals. The other scenarios refer to both L1 (GPS) and E1 (Galileo) and do not take into account multipath effects. Scenario "03" includes SBAS signals, which are not provided in "02".

TABLE I
DATASET: HERE F: FREQUENCY BANDS; M: MULTIPATH (Y/N); G/S: MISMATCH BETWEEN GROUND AND SPACE SECTIONS (Y/N)

Scenario	Run	GPS			Galileo			SBAS
		F	M	G/S	F	M	G/S	
5x00	00-04	L1	y	y	n	n	n	n
5x01	05-09	L1	y	y	n	n	n	y
5x02	10-14	L1	n	y	E1	n	n	n
5x03	15-19	L1	n	y	E1	n	n	y

All runs are impacted by ionosphere and troposphere effects. The "Klobuchar" model is used for the ionosphere (model parameters extracted from navigational RINEX files), while the "MOPS" model (i.e., Minimum Operational Performance Standards) is used for the troposphere. When applied (i.e., "G/S" set to "y"), additional global hazards (i.e., clock and ephemeris errors) are handled by introducing a mismatch between the ground and space segments in the emulator configuration (i.e., internal combined use of RINEX and SP3 files).

VI. DATA PRE-PROCESSING AND FEATURE SELECTION

The raw data obtained from the GNSS receiver (via UBX protocols) was decoded into a .csv format using a custom MATLAB–Python automation pipeline. We then filtered out samples without a valid GNSS fix, removed invalid or constant observation fields, and replaced missing numerical values with zeros to avoid perturbing the estimations yielding a high-quality dataset for modeling. After processing, the dataset was partitioned into three variables: input features (X), target errors (Y), and temporal indices (t), maintaining strict temporal order to prevent data leakage. This pre-processing pipeline

established a reproducible data foundation from which the regression models can generalize under controlled conditions. The target errors are x_{error}, y_{error} and z_{error} and they are obtained using the below equation:

$$A_{error} = A_{GT} - A_{GNSS} \tag{2}$$

where A represents the three spatial dimensions, namely X, Y, and Z-axis, A_{GT} is the ground truth set in the simulator and A_{GNSS} is the position value estimated by the receiver for a given direction.

In the RT-LLR and RF-LLR approaches, a feature importance analysis using Decision Tree and Random Forest regressors, respectively, was conducted on over 300 GNSS features. This analysis identified the ten most influential features (either in their real-time or lagged forms), as presented in **Table II**, while ensuring the shortlisting of independent feature with reasonable weight and preventing information leakage from implicit ground-truth correlations.

TABLE II
GNSS FEATURES THAT HAVE THE MOST WEIGHT IN POSITION ERROR ESTIMATION

Feature	Unit, Scale	Description
codePhase	ms, 2^{-21}	Code phase
cpMes	cycles	Carrier phase measurement
doMes	Hz	Doppler measurement (positive sign for approaching satellites)
dopplerHz	Hz, 0.2	Doppler measurement
dopplerMS	m/s, 0.04	Doppler measurement
fracChips	(0 - 1023)	fractional value of the code phase measurement (0..1023)
prMes	m	Pseudorange measurement
wholeChips	(0 - 1022)	whole value of the code phase measurement (0..1022 for GPS)
pDOP	-	Position Dilution of Precision
intCodePhase	ms	Integer part of the code phase

This data-driven selection ensures that learning resources concentrate on the GNSS attributes that are most relevant for position error prediction. Consequently, the models capture temporal dependencies of GNSS error dynamics using only lagged exogenous variables, deliberately avoiding recursive inclusion of past output errors, which in our application would introduce data leakage and lead to unrealistic model behavior and hinder real-time estimations. This design choice enhances model consistency and interpretability under non-stationary noise and signal variability.

VII. MODEL TESTING METHODOLOGY

We performed model training, validation, and testing in Python (v3.9.6) running on Macbook Pro M3 Pro equipped with an Apple M3 Pro SoC (12-core CPU: 6 performance + 6 efficiency cores, 18-core GPU with 18GB unified memory, using an 80:10:10 dataset split. Given the large dataset size, we employed randomized hyperparameter search in combination with ten-fold cross-validation to ensure robust statistical evaluation. Four models were bench marked: a standard Regression Tree (RT), a Random Forest (RF) [9], and the proposed hybrid RT-LLR and RF-LLR architectures. Each model estimated the three-dimensional position-error vector and was evaluated using RMSE across all test runs.

VIII. RESULTS AND PERFORMANCE ANALYSIS

In this section we present the quantitative evaluation of the hybrid RT-LLR and RF-LLR models compared with the baseline Regression Tree (RT) and Random Forest (RF) architectures and we will showcase the RMSE of position error estimates for the testing dataset along the x, y, and z axes for assessment.

Table III provides a comparative evaluation of prediction errors between two modeling approaches: standard Regression Trees and Random Forests (labeled as "a") and their hybrids with local linear regression on lagged exogenous inputs (labeled as "b"), measured across 20 experimental runs for the x, y, and z directions. The relative performance change is calculated using the formula:

$$2 \times \frac{b-a}{b+a} \times 100 \tag{3}$$

TABLE III
COMPARISON OF RMSE BETWEEN THE BEST PREDICTIONS MADE BY RT & RF AND THE BEST PREDICTIONS MADE BY RT-LLR & RF-LLR.

Run	RT & RF (a) (meters, m)			RT-LLR & RF-LLR (b) (m)		
	x_{error}	y_{error}	z_{error}	x_{error}	y_{error}	z_{error}
00	0.6484	0.5432	1.0430	0.6144	0.5773	0.9966
01	1.9373	0.4218	1.1379	1.8686	0.4218	0.9593
02	0.4827	0.1231	0.3150	0.4827	0.1231	0.3150
03	0.4191	0.2240	1.0079	0.4024	0.1674	0.8652
04	1.7758	0.9562	0.7623	1.8199	0.9562	0.7778
05	1.0877	0.4968	0.6175	1.0923	0.4963	0.5679
06	0.3958	0.1072	0.2911	0.4728	0.1077	0.2924
07	0.8328	0.3347	0.6553	0.7477	0.3347	0.6553
08	1.7356	0.1967	0.8352	1.3878	0.1967	0.7252
09	4.7923	0.5112	2.2183	4.7085	0.5109	2.1993
10	0.0290	0.0356	0.0301	0.0289	0.0356	0.0298
11	0.0407	0.0475	0.0634	0.0504	0.0475	0.0605
12	0.0283	0.0443	0.0372	0.0281	0.0443	0.0372
13	0.0294	0.0481	0.0373	0.0295	0.0481	0.0379
14	0.0429	0.0667	0.0759	0.0402	0.0667	0.0759
15	0.0486	0.0380	0.0819	0.0486	0.0380	0.0777
16	0.0533	0.0591	0.0395	0.0449	0.0591	0.0378
17	0.0312	0.0454	0.0366	0.0315	0.0454	0.0367
18	0.0359	0.0448	0.0369	0.0354	0.0448	0.0374
19	0.0891	0.0939	0.1249	0.0863	0.0939	0.1380

The key takeaways are:

- **X-axis:** RT-LLR & RF-LLR reduce RMSE in 8 runs, tie in 7, and worsen in 5. Largest gain: Run 08 at –22.27%. Largest loss: Run 11 at +21.30%.
- **Y-axis:** Improvement occurs in 1 run, ties in 18, and degradation in 1 run. Largest gain: Run 03 at –28.92%. Largest loss: Run 00 at +6.09%.
- **Z-axis:** RMSE improves in 9 runs, ties in 8, and worsens in 3. Largest gain: Run 01 at –17.03%. Largest loss: Run 19 at +9.97%.

Overall, **Figure 3** shows that the hybrid models tie or outperform the base trees in approximately 75% of x-axis cases, 95%

of y-axis cases, and 85% of z-axis cases, confirming robust, scenario-adaptive improvements.

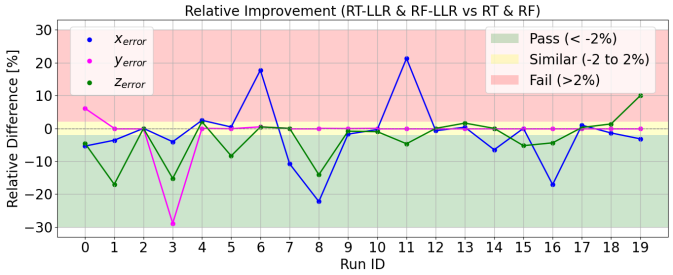

Fig. 3. Relative difference (%) between Standard (a) and Hybrid (b) ML models

Moreover, runtime analysis showed that the computational cost of the hybrid methods remains comparable to that of the baseline RT and RF models, confirming their suitability for real-time embedded GNSS applications.

Across the 20 runs, the LLR-enhanced models consistently match or improve the baseline RT & RF performance, with the most notable gains appearing in the z-axis—for example, reductions from 1.0430 m to 0.9966 m in Run 00, 1.1379 m to 0.9593 m in Run 01, and 1.0079 m to 0.8652 m in Run 03. Improvements in the x-axis are also common, such as 0.8328 m to 0.7477 m in Run 07 and 1.7356 m to 1.3878 m in Run 08, while y-axis values remain largely unchanged. Run 09 stands out as an outlier, with unusually large errors (x = 4.7923 m, z = 2.2183 m) due to the weakness of its underlying dataset, though even here the LLR variant yields slight reductions. Averaged across all runs, the overall RMSE values for RT & RF are x = 0.6921 m, y = 0.2477 m, and z = 0.5407 m, while the LLR-enhanced models achieve x = 0.6620 m, y = 0.2475 m, and z = 0.5207 m. Because each run represents a distinct scenario with different data quality and environmental conditions, these averages reflect performance across heterogeneous settings rather than a single controlled environment. Despite this variability, the aggregated RMSE remains below 1 m on all axes, demonstrating consistent sub-meter accuracy across diverse scenarios.

The empirical results validate our hypothesis that **piecewise linear modeling of lagged GNSS observables** effectively captures the spatial and temporal dependencies in position errors without depending on recursive feedback or extensive sensor fusion. By dividing the input space into understandable, locally linear regions through tree-based learning, the approach decomposes complex, nonlinear error dynamics into manageable sub-models suitable for real-time processing. To balance the interpretability, accuracy, and computational efficiency, these proposed hybrid models emerge as promising candidates for deployment in low-cost GNSS receivers and cooperative positioning systems.

IX. Conclusion, Limitations and Future Work

Ultimately, the proposed hybrid methods bridge classical dynamic system identification and modern machine learning, delivering interpretable, lightweight, and reproducible estimators optimized for intelligent transportation systems and onboard GNSS processing.

By uniting theoretical formulation, controlled experimentation, and practical implementation, this work advances the broader objective of developing reliable and explainable positioning technologies for the next generation of connected and automated mobility.

The main contribution of this work is the design of interpretable hybrid models (RT-LLR and RF-LLR) that extend classical system identification toward data-driven GNSS error estimation. Unlike conventional ARX structures, the proposed approach operates solely on lagged exogenous inputs, avoiding feedback loops and thus preserving model causality for real-time applications. The results demonstrate substantial improvement in position error estimation accuracy without sacrificing computational efficiency or interpretability. These characteristics make the framework a strong candidate for embedded GNSS receivers and certified PNT validation environments.

Despite strong performance under emulated conditions, several limitations are acknowledged. The current training dataset originates exclusively from HiL experiments, limiting generalization to open-world driving conditions. Additionally, each local linear model presumes homogeneity within its partition, which may be disrupted by abrupt environmental changes. These issues motivate **adaptive partition refinement** and **online retraining** strategies. Moreover, while feature reduction optimizes response time and prevents overfitting, it may obscure subtle dependencies observable only under complex multipath scenarios. Future research will therefore pursue data generation in a real-world scenario for correlating the simulated scenario, investigating a hybrid learning-and-fusion pipeline integrating the present GNSS-only approach with inertial sensor.

X. Acknowledgment

This work has been developed in the framework of the Navisp Element 3 “P–CAR – PNT Center for Automated Road-Transport” project, funded by ASI through the NAVISP-3 program of ESA to realize a PNT laboratory for testing and validation of CAD - Connected Autonomous Driving functions. The authors acknowledge financial support from Centre of Excellence on Connected, Geo-Localized and Cybersecure Vehicle, funded by Italian Government under CIPE resolution n. 70/2017.

References

[1] B. Guermah, H. E. Ghazi, T. Sadiki, Y. Ben Maissa and E. Ahouzi, “A comparative performance analysis of position estimation algorithms for GNSS localization in urban areas,” 2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS), Marrakesh, Morocco, 2016, pp. 1-7, doi: 10.1109/ACOSIS.2016.7843950.

[2] Wen, Weisong, Hsu, Li-Ta, “3D LiDAR Aided GNSS Real-time Kinematic Positioning,” Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), St. Louis, Missouri, September 2021, pp. 2212-2220. https://doi.org/10.33012/2021.18072.

[3] M. Alfakih, M. Keche, and H. Benoudnine, “A Kalman-filter-based fusion method for accurate urban localisation,” IET Communications, vol. 15, no. 5, pp. 653–663, 2021. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cmu2.12096.

[4] Li, Qianxia, Xue Hou, Yuanbin Ye, Wenfeng Zhang, Qingsong Li, and Yuezhen Cai. 2025. “Multipath Identification and Mitigation for Enhanced GNSS Positioning in Urban Environments,” Sensors 25, no. 19: 6061. https://doi.org/10.3390/s25196061.

[5] Caparra, Gianluca, Zoccarato, Paolo, Melman, Floor, “Machine Learning Correction for Improved PVT Accuracy,” Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), St. Louis, Missouri, September 2021, pp. 3392-3401. https://doi.org/10.33012/2021.17974.

[6] T. Lan, A. Dodinoiu, A. Geffert, and U. Becker, ”Machine-learning-based position error estimation for satellite-based localization systems,” in Proc. Eur. Test Telemetry Conf. (ettc2020), pp. 89–97, Braunschweig, Germany, 2020.

[7] Qi, X., Xu, B., Wang, Z., Hsu L. ”Random forest-based multipath parameter estimation,” GPS Solut 28, 126 (2024), https://doi.org/10.1007/s10291-024-01667-x.

[8] Zhang, G., & Hsu, L.-T. (2017). ”Adaptive GNSS/INS integration based on supervised machine learning approach,” PolyU Institutional Research Archive. https://ira.lib.polyu.edu.hk/handle/10397/77104.

[9] Niu, X., Dai, Y., Liu, T. et al. ”Feature-based GNSS positioning error consistency optimization for GNSS/INS integrated system,” GPS Solut 27, 89 (2023). https://doi.org/10.1007/s10291-023-01421-9.

[10] Breiman, L., Friedman, J., Olshen, R.A., & Stone, C.J. (1984). ”Classification and Regression Trees (1st ed.). Chapman and Hall/CRC,” https://doi.org/10.1201/9781315139470.

[11] Francesco Smarra, Giovanni Domenico Di Girolamo, Vittorio De Iuliis, Achin Jain, Rahul Mangharam, Alessandro D’Innocenzo, ”Data-driven switching modeling for MPC using Regression Trees and Random Forests,” Nonlinear Analysis: Hybrid Systems, Volume 36, 2020, 100882, ISSN 1751-570X, https://doi.org/10.1016/j.nahs.2020.100882.

[12] F. Valentini, E. Cinque, V. Sulli, F. Rispoli, M. Pratesi, F. Santucci, and A. Neri, “On the validation of multi-sensor high integrity positioning solutions for the connected car: The p-car infrastructure,” in Proceedings of the 2022 International Technical Meeting of The Institute of Navigation, Long Beach, California, January 2022, pp. 1358–1370.