Exploring Cooperative Positioning and Dynamic Base Stations for Potential Vehicular Positioning Accuracy Improvement: A Comprehensive Approach

Tânia Guedes*, Fabricio Botelho*, Ivo Silva†, Helder Silva†, Cristiano Pendão†‡
*Bosch, Braga, Portugal
†Centro Algoritmi University of Minho, Guimarães, Portugal
‡Department of Engineering, University of Trás-os-Montes and Alto Douro, Vila Real, Portugal
email: {tania.guedes, fabricio.botelho}@pt.bosch.com, hdsilva@dei.uminho.pt, {ivo, cpendao}@dsi.uminho.pt

***Abstract*—Cooperative positioning has appeared as a promising strategy to improve the accuracy of vehicle positioning systems, particularly in urban environments where traditional static base stations are used. This paper proposes an approach for selecting vehicles to serve as dynamic reference stations to improve positioning using Artificial Intelligence, which enables the reduction of costs associated with correction services. By leveraging the presence of nearby vehicles, our method aims to improve the precision of positioning in challenging environments like urban canyons. To achieve this goal, we utilize simulation data to create a comprehensive dataset, capturing various environmental conditions and vehicle dynamics. We then employ machine learning techniques to use this dataset and identify optimal vehicles that can serve as reference stations for improving the positioning accuracy of other vehicles in real-time. By continuously learning and adapting to changing conditions, our approach offers a flexible and robust solution for cooperative positioning in dynamic urban settings.**

***Keywords*-Cooperative Positioning; Machine Learning; Localization; GNSS.**

I. Introduction

The navigation systems are an essential component of intelligent vehicles and are being used in Advanced Driver Assistance Systems (ADAS) applications [1]. In current era of developing autonomous systems, the combination of Cooperative Positioning (CP) and Machine Learning (ML) methodologies represents an excellent pathway towards attaining accurate vehicle positioning.

CP is identified as a potentially effective approach to enhance the precision of location estimation. In contrast to conventional positioning systems that depend exclusively on data from individual sensors, CP utilize the combined capabilities of numerous sensors to create a cooperative scenario where information from various sources is effortlessly merged and integrated [2].

Improving the position accuracy for each vehicle in a non-cooperative environment can be achieved using static base stations serving as references. Assuming these stations are positioned within the same area as the Global Positioning System (GPS) receiver, it can be assumed they are affected by the same error sources, considering similar atmospheric conditions. However, this may not bring benefits in terms of infrastructure cost and implementation, as it requires a large number of scattered static reference stations, for example, in a city [3]. Additionally, independent positioning using low-cost GPS receivers may result in low positioning accuracy, in some scenarios in the order of tens of meters, which is unacceptable for vehicles requiring high accuracy in their position while in motion. However, vehicles that rely just on dynamic reference stations, especially if they use only egocentric positioning, can not achieve high precision in positioning. The advantage of relative positioning lies in the use of distance measurements obtained through Global Navigation Satellite System (GNSS) signals, which offer higher precision. The purpose of CP is to mitigate errors associated with multipath and Non-Line-Of-Sight (NLOS). Thus, vehicles in the same area can contribute with measurements to enhance positioning, reducing the reliance on a large number of static reference base stations. Furthermore, it may enable even a vehicle using a low cost GPS system to increase the accuracy in the estimated position through measurements from others [4].

Assigning responsibility to vehicles to be used or viewed as dynamic reference stations requires consideration of several aspects to facilitate decision-making. Factors such as the vehicle's operating area, susceptibility to multipath effects, relevance of measurement errors (e.g., pseudorange errors), and shared data with other vehicles (e.g., common satellites) must be taken into account [5]. This is important as it enables the subsequent application of error correction services algorithms, such as differential positioning or Real-Time Kinematic (RTK) [6]. However, obtaining real-time errors for decision-making regarding whether a vehicle can be considered a reference station is challenging. Therefore, the use of simulators can facilitate the validation of this concept and the creation of a dataset incorporating measurements obtained by a receiver. This dataset can serve as a training basis for Artificial Intelligence (AI) algorithms to develop a model capable of assessing, in real-time, the likelihood of a vehicle being considered a dynamic reference station on a scale from 0 to 1, without the need to directly examine factors such as pseudorange errors.

Creating datasets to address the mentioned challenge can be difficult using real-world technology. To establish the CP topic, it is necessary to utilize multiple vehicles with reference systems that can serve as ground truth. However, these systems are often expensive, such as the iTrace [7], which can cost thousands. Therefore, switching to simulators enables the translation of real-world scenarios into a simulation environment, facilitating the acquisition of data in a convenient manner. The software utilized for data generation in this work includes the Car Learning to Act (CARLA) [8] and GPSsoft

Satellite Navigation toolbox from MATLAB [9].

This paper contributes to improve individual positioning for vehicles in challenging scenarios, such as urban canyons, by exploring GNSS, other signals from infrastructure and vehicles in cooperation.

The paper is organized as follows. Section 2 describes fundamental knowledge regarding GNSS, CP, and ML, explicitly discussing their features. Sections 3 and 4 describes the metodology regarding the pipeline implemented to reach the results, which are described in Section 5. Finally, the respective conclusions and the work plan for the future are presented in Section 6.

II. Fundamentals

A. Cooperative Positioning

Intelligent Transport Systems (ITS) have potential to address challenges that still exist today, such as road accidents or incidents. The CP is part of these ITS because improving the individual positioning of each vehicle, it makes it possible to share this information with vehicle control system for decision making purposes [10].

CP methods rely on the exchange of position data between multiple stationary or mobile nodes to improve the accuracy of positioning [11]. Several traditional CP systems have been created with the intention of improving the precision of GPS positioning. Nonetheless, for these systems to work autonomously the GPS signal coverage must be adequate. Furthermore, CP methods have directed their attention towards enhancing positioning in regions with limited signal visibility. As a result, it was decided to utilize multi source sensor fusion to mitigate the limitations of each individual source. In regions characterized by suboptimal GPS accuracy, the utilization of Inertial Measurement Unit (IMU) sensor, Radio Detection And Ranging (Radar) or Light Detection and Ranging (LiDAR) data may facilitate a more precise car position [12].

Nevertheless, CP is not devoid of challenges and limitations. One significant challenge lies in ensuring seamless communication between vehicles, particularly in scenarios with high traffic or intermittent network connectivity. Additionally, the effectiveness of CP may be hindered by factors such as obstructions, signal interference, and varying environmental conditions, which can impact the reliability of positional data exchange [13].

B. Factors contributing to GNSS errors

Pseudoranges are utilized to determine positions on the Earth's surface, necessitating a minimum of four satellites for calculation. A fourth satellite is required to adjust for receiver clock errors [15]. Pseudorange, considered a pseudo-distance between the satellite and the receiver, is obtained through the ”time of flight” of radio signals. Measurement values for pseudo-range can be affected by various error sources during signal transmission from the satellite to the receiver. Minor timing errors can result in several meters of deviation in the measured pseudo-range. For instance, an error of ten nanoseconds in satellite time measurement introduces three meters of error in the pseudorange measurement. Several sources of GNSS errors are:

- Ionospheric and tropospheric errors
- Satellite clock errors
- Receiver clock errors
- Ephemeris data errors
- Receiver noise
- Multipath error
- Dilution of Precision (DOP)

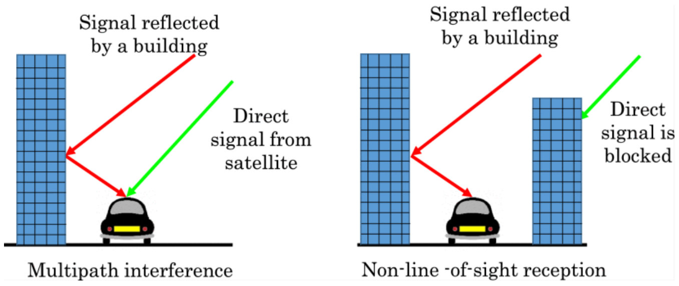
Signal reflected by a building
Direct signal from satellite
Multipath interference
Signal reflected by a building
Direct signal is blocked
Non-line -of-sight reception

Figure 1. Multipath interference and NLOS reception situation [30].

Multipath errors represent the most common source of GNSS error [14]. They arise from interference and the reception of satellite signals beyond direct line of sight. Multipath is a phenomenon of propagation of radio signals. The reception of multipath signals occurs when there is more than one replica of the signal reaching the receiver, for example, the line-of-sight signal plus a reflected copy. The reception of only a reflected signal is known as NLOS, which is more problematic than multipath. Figure 1 shows how the buildings can cause distortion in data transmitted by a satellite.

C. Correction Services

In the preceding subsection, we explored various sources of error that can impact GNSS positioning accuracy. Mitigating these errors is essential for improving performance, achieved through corrections applied to raw GNSS observations.

In literature, two primary approaches to error handling are identified: Observation Space Representation (OSR) and State Space Representation (SSR) [16]. OSR-based systems utilize GNSS observations (pseudo-range and/or carrier-phase) from a reference station, optionally incorporating corrections from a network of Continuously Operating Reference Stations (CORS). Errors in OSR are typically aggregated as a single sum, varying in accuracy depending on available measurements and infrastructure [17]. In contrast, SSR-based systems adopt a distinct strategy for error management. Here, physical errors affecting GNSS observations are decorrelated, modeled, and represented flexibly. SSR enables users to correct their positions using observations from a single receiver and network corrections.

Actually, several correction services are available for both OSR and SSR approaches. Notable OSR services include Differential GNSS (DGNSS), Real-Time Kinematic (RTK), and Network-RTK. For SSR, available services include Satellite-

Based Augmentation Systems (SBAS), Precise Point Positioning (PPP), and International GNSS Service (IGS) products.

The primary function of reference stations is to enhance positioning. These stations are commonly positioned at well-known reference points, elevated locations with minimal signal obstruction, or high vantage points. In this paper, like in [3] we propose the concept of dynamic reference stations, where receivers installed on certain vehicles can serve as references for others needing to enhance their positioning accuracy. By sharing raw measurements, such as pseudoranges in an OSR-based approach like Differential GNSS (DGNSS), it becomes feasible to improve positioning accuracy. This is because errors associated with atmospheric conditions are consistent within the same geographical area, and additionally, errors associated with satellite orbits and clocks are cancelled. These methods are advantageous when used with phase measurements, which have higher precision than using only pseudoranges.

D. Machine Learning

A comprehensive overview on ML applications using GNSS data may be found in [18]. These investigations highlight the use of machine learning in GNSS. The three most common algorithms used are neural networks, decision trees, and Support Vector Machines (SVM). We decided use three different approaches for this study: Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and Random Forest (RF).

1) LSTM: Neural Networks are part of a larger category that includes Recurrent Neural Networks (RNNs). RNNs are specialized for dealing with sequential or time series data and their distinguishing feature is the ability to retain memory from previous inputs to influence current inputs as well as outputs.

For addressing the issue of vanishing gradients and managing long-term dependencies effectively, LSTM was introduced. Cells in the hidden layers of LSTM networks have three gates, an input gate, an output gate and a forget gate. These gates open or close based on input and the last hidden state encouraging selective retention or deletion of information respectively by an LSTM. LSTMs have been proven effective in capturing long-term dependencies as shown by [19] [20] and [21] among others due to this selectiveness in retaining information over a long period of time.

2) CNN: CNNs are known as Convnets and are specialized in analyzing visual data by detecting and recognizing patterns through convolutions. They can be adapted to process various signal data types although originally used for image analysis [22].

Every layer of a CNN is an improvement over the one before it and finds intricate designs. The process begins with convolution operations that use filters to detect desired features in the inputs [23].

CNN architecture usually include three kinds of layers: convolutional, pooling and fully connected [24]. Convolution layers are responsible for most computations as they traverse the input images using filters to detect features. After convolution layers, there are pooling layers which decrease spatial dimensions while retaining important information.

3) Random Forest: The RF algorithms are a versatile ensemble learning method applicable in both classification and regression. It holds the view that by aggregating the forecasts of many decision trees created while training, more accurate and consistent results can be observed than with any single tree alone [25].

During Training, RF builds an ensemble of decision trees. A random selection from the features plus a fraction of the training data are used to construct each decision tree. In a decision-tree framework, which is binary comprising nodes and branches, each internal node represents a value derived as a result of applying a feature whereas class labels or numeric values are contained in leaf nodes signifying classification or regression respectively. This will help mitigate overfitting problems and improve generalization.

As for RF voting system classification uses it to sum its predictions while averaging works for regression at the end of every training process for individual decision trees. For classification purposes, this is seen as majority voting rule in which if there is an odd number of votes within class labels then they should be considered during final outcome determination. However, ties imply no preference toward any label but only balanced outcome distribution between involved classes. Although this is the mean squared error criterion, the mean predictor outperforms individual regressors as it provides a more stable prediction [26].

III. Metodology: Creating the Dataset and Input data

The simulated data is obtained from the CARLA simulator in conjunction with Matlab's SatNav Toolbox framework. Figure 2 illustrates the organization of data from the simulator side. CARLA is responsible for simulating the environment and vehicles' motion, thus allowing to generate reference data (exact position of the vehicles) with other sensors, such as the IMU and odometer (travelled distance). CARLA also has a GNSS virtual sensor that can be configured to output latitude, longitude and altitude whose configurable parameters are the sensor bias and standard deviation. Since the CARLA's GNSS sensor model does not account for error sources such as thermal noise, multipath, atmospheric delays, we used the the SatNav Toolbox for Matlab instead. This tool takes into account the error sources, mentioned previously, to generate raw GNSS measurements (pseudoranges and carrier phases). In order to represent the same 3D space in SatNav and CARLA the satellites' positions generated by SatNav had to be converted from SatNav's coordinate system to CARLA's coordinate system, such that they correspond to the same position in both simulators.

A 20 minute simulation was run to generate synthetic data from 6 vehicles moving in Town03 from the CARLA simulator. Data was generated at a sample rate of 20 Hz. The following data were obtained: vehicle's true position; vehicle's received pseudoranges and carrier phases; ground

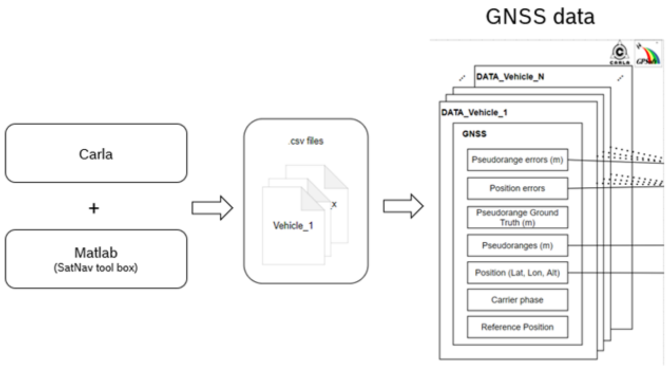

Figure 2. Organization of GNSS data.

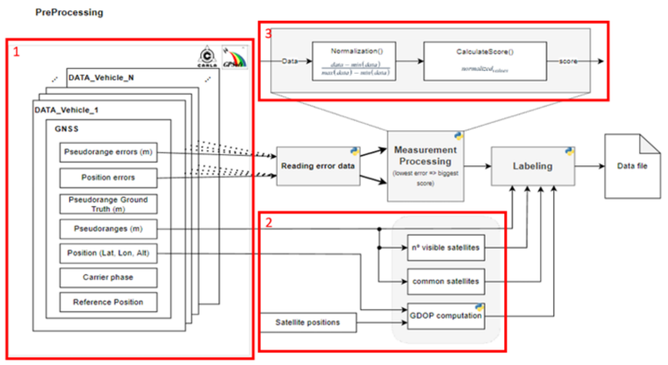

Figure 3. Data Pre-processing architecture, 1-Initial data, 2-selected features, 3-normalization and score calculation.

truth pseudoranges (for the GPS constellation, 24 satellites) for each vehicle; and satellites' positions. Other sensors' data were also generated for vehicles, e.g., IMU and odometer, but were not used for the purpose of this paper.

Figure 3 is a schematic which represents the steps that were taken to get to the dataset. To make it clear how we handled the data, we added red boxes 1, 2, and 3. First, the dataset's features were chosen, and they can be seen in box number 2. These are the pseudoranges, number of visible satellites, the number of common satellites and the Geometric Dilution Of Precision (GDOP).

A module called "Reading error data" was added to the labelling process to read data related to pseudorange and position errors. The MinMaxScaler method, Equation 1, is used to normalise the error values in Box 3. GDOP could be used, but since most of the vehicles are in the same area, the geometry of the satellites is almost the same, so it wasn't thought to be very useful. Instead, it was decided to add it as a feature and not use it during the labelling process. After normalisation, the inverse of the value that was found is used as a score, and it is saved in the dataset file with the other features. The Signal to Noise Ratio (SNR) could be used if possible. But its simulation complexity indicates that it needs to be handled carefully and cannot be obtained directly. Besides that, SNR is not generated by SatNav toolbox.

$$\text{normalizedValue} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

The score is a number between 0 and 1 which indicates how well the vehicle is positioned and can be used as a dynamic reference station. The value should be close to 1 if the vehicle is to be used as a reference and vice-versa. Equation 2 is used to compute the score.

$$\text{Score} = 1 - \sum (\text{normPseudorangeEr} \times 0.5 + \text{normPositionEr} \times 0.5) \tag{2}$$

The developed labeling algorithm starts reading the csv files that contains all information regarding GNSS and vehicle data. The data is divided into two parts: error data (pseudo range errors, position errors) and features (n° of visible satellites, common satellites between vehicles, GDOP). All error values are normalized and a score for each vehicle can be obtained. The final step is related to dataset format where a group of features corresponds to a score value.

Table I presents a data distribution that demonstrates the amount of data associated with scores. It is clear that there is a difference between the lowest and highest scores. This could present a challenge for the final results as the dataset is not completely balanced. It can be said that a model's performance and reliability get better when the output provides a high score.

TABLE I
DATASET DISTRIBUTION, COUNTING THE NUMBER OF DATA RELATED TO SCORES.

Label(target)	Count
score <0.6	12847
$0.6 \geq$ score < 0.8	62508
score >0.8	68603

IV. MODEL TRAINING: EXPLORING LSTM, CNN AND RF

The dataset from the previous phase was trained using LSTM, CNN and RF algorithms. The dataset is not directly applicable to an AI model. Both data and neural network models must be prepared. Python and PyTorch framework library were utilised to create the models. Missing values are difficult to handle because they can affect the final result. The 24 satellites in the GPS constellation are included in created dataset, but they are not all visible at once. Missing values were replaced by -1 in order to solve the issue. The dataset had been divided using 60%, 20% and 20% of the data for the training, validation and test phases, respectively. This type of division is frequently applied to training data and is essential to prevent overfitting or even imbalanced data related issues in the future.

A. LSTM

LSTM algorithm requires a diferent data organization. Once, this algorithm is based on series data it's necessary to organize the entire dataset in samples. Each sample contains a certain number of time steps which represents a sequence of data. This characteristic allows to identify not only vehicles that can be utilized as dynamic stations, but also detect an environment scenario like entering or leaving urban canyons or other challenging environments. This can provide a more complex solution to the challenge that this paper is trying to solve.

Considering 6 timesteps per sample, a total of 3427 samples were obtained. The data shape array is represented by: (number of samples, timesteps, number of features, number of vehicles) and numerically by: (3427, 6, 4, 6). However, LSTM does not support 4D arrays as input. To handle this problem, it was necessary to reshape the array to 3D throught multiplying the number of features by number of vehicles resulting in a shape array of (3427, 6, 24).

In model implementation, it was decided to create two simple models. The difference between them is related to the number of internal layers, and the idea is to check the results considering the model complexity.

Model 1 is initialized with an LSTM layer, two linear layers and a Retified Linear Unit (ReLU) activation function. The LSTM layer takes as input the size of the input data and the size of the hidden layer. The first linear layer transforms the output of the LSTM layer to an intermediate output and the second linear layer transforms this intermediate output to the final output size. ReLU activation function introduces non linearity into the model [27].

During forward propagation the model initializes two tensors with zeros, representing the initial hidden state and cell state of the LSTM layer. These tensors are moved to the GPU for faster computation. The model's output is six values, which represents the score for each vehicle.

The model is trained using the Adam optimizer with a learning rate of 0.001. The training process is set to run for 1000 epochs and the batch size is set to 32. The size of the hidden layer is set to 50.

B. CNN and Random Forest

We implemented a CNN as a regression model to predict the score (target). The CNN model was chosen to prove the effectiveness in capturing spatial patterns in data, which was expected to be beneficial for our purpose. We tried to use different configurations of the model by changing the number of convolutional layers. The idea of this was to investigate how the complexity of the model affects the performance. A more complex model with more layers can potentially learn more patterns in the data, but it also runs the risk of overfitting to the training data. On the other hand, a simple model might not capture all the relevant patterns, but it is less likely to take overfitting.

The model was trained using the Adam optimizer with a learning rate of 0.001. The Mean Squared Error (MSE) was utilized in loss function during training, which measures the average squared difference between the predicted and actual values.

Regarding our Random Forest Regressor, we initialized it with 100. Following the training phase, we used the trained model to predict scores for our test data.

V. Results and analyses

This section provides the results obtained using different AI models. The performance of these models was assessed through the training phases described in the previous section. The algorithms were trained under the following computational conditions:

- **Processor:** AMD Ryzen Threadripper PRO 5995WX 64-Cores
- **GPU:** NVIDIA RTX 4090

Processor capabilities like parallel processing, memory and cache efficiency accelerates the training tasks. Utilizing PyTorch enhances performance, underscoring the importance of processor choice in efficient tensor processing.

The results are presented using metrics commonly utilized in regression problems [28], such as MSE and R square (R2). Some plots are presented to compare the output of the model with the respective true value. These plots were obtained through test dataset.

From the perspective of the main topic associated with this paper, we are evaluating how well we can develop a model that allows us to assign a score to each vehicle in a given area in real-time. This score will determine whether or not the vehicle can be seen as a dynamic reference station to help another vehicle improve its position.

A. LSTM

As previously mentioned, two models were developed. The features utilized in these models are represented as follows: the number of visible satellites is denoted as f1, the common satellites as f2, the pseudoranges as f3 and the GDOP as f4.

TABLE II
LSTM RESULTS

Model ID	Features	Parameters	MSE	R2 Score
LSTM1.1	f1	12488	0.0002	0.9615
LSTM1.2	f1, f2, f3, f4	12488	0.0109	0.9011
LSTM2.1	f1	17078	0.0037	0.9120
LSTM2.2	f1, f3	17078	0.0021	0.9610

Table II presents the performance of two models. Among the models evaluated, the first model, LSTM1.1, exhibits the lowest MSE and the highest R2 Score. This suggests that LSTM1.1 outperforms the other models. The feature "number of visible satellites," employed in this model, appears to be more effective in predicting the target variable compared to the additional features utilized in other models. LSTM1.1 compared to LSTM2.1, only uses a single internal LSTM layer. This could indicate that, given the data at hand, simpler models may yield better results.

B. CNN

Table III presents different models changing the number of convolutional layers: model c1 has two layers, model c2 has three layers, model c3 has 4 layers and model c4 has 10 convolutional layers.

TABLE III
CNN RESULTS

Model	Features	MSE	R2 Score
c1	f1, f2, f3, f4	0.000557	0.984
c2	f1, f2, f3, f4	0.000397	0.988
c3	f1, f2, f3, f4	0.000351	0.989
c4	f1, f2, f3, f4	0.000272	0.992

The features utilized remained consistent across all tests. The results obtained were generally good, however, the model c4 achieved the highest results. This model, in comparison to the others, incorporates a higher number of convolutional layers.

The superior performance of model c4 may suggest that the additional convolutional layers helped the model to better learn from the data and make more accurate predictions.

C. Random Forest

TABLE IV
RANDOM FOREST MODEL PERFORMANCE METRICS

Model	Features	Parameters	MSE	R2 Score
RF1	f1, f4	4515040	0.0017	0.9730
RF2	f1, f3, f4	20731525	0.0001	0.9993
RF3	f1, f2, f3, f4	19715917	0.0001	0.9992
RF4	f2, f3, f4	19716091	0.0001	0.9992
RF5	f1, f2, f3	19716415	0.0001	0.9992
RF6	f1, f3, f4	19859980	0.0001	0.9956

Table IV demonstrates that the Random Forest algorithm achieved satisfactory results. Given its ease of implementation and speed in producing results, it can be considered a viable option for this dataset. The performance of the Random Forest model across all evaluation metrics strongly suggests it as a good algorithm to use with GNSS data.

D. Model results vs Groundtruth

Figure 4 presents the outcomes for the three distinct algorithms utilized: LSTM, CNN, and Random Forest. The vertical axis represents predicted values, while the horizontal axis corresponds to ground truth values. These results are derived from 20% of the total dataset, previously designated as the test dataset.

Most values fit what would be expected, with notable performance observed in the results generated by the Random Forest algorithm. However, the LSTM achieved some anomalous results regarding certain score values. This may be attributed to the dataset nature and its application in time series based algorithm.

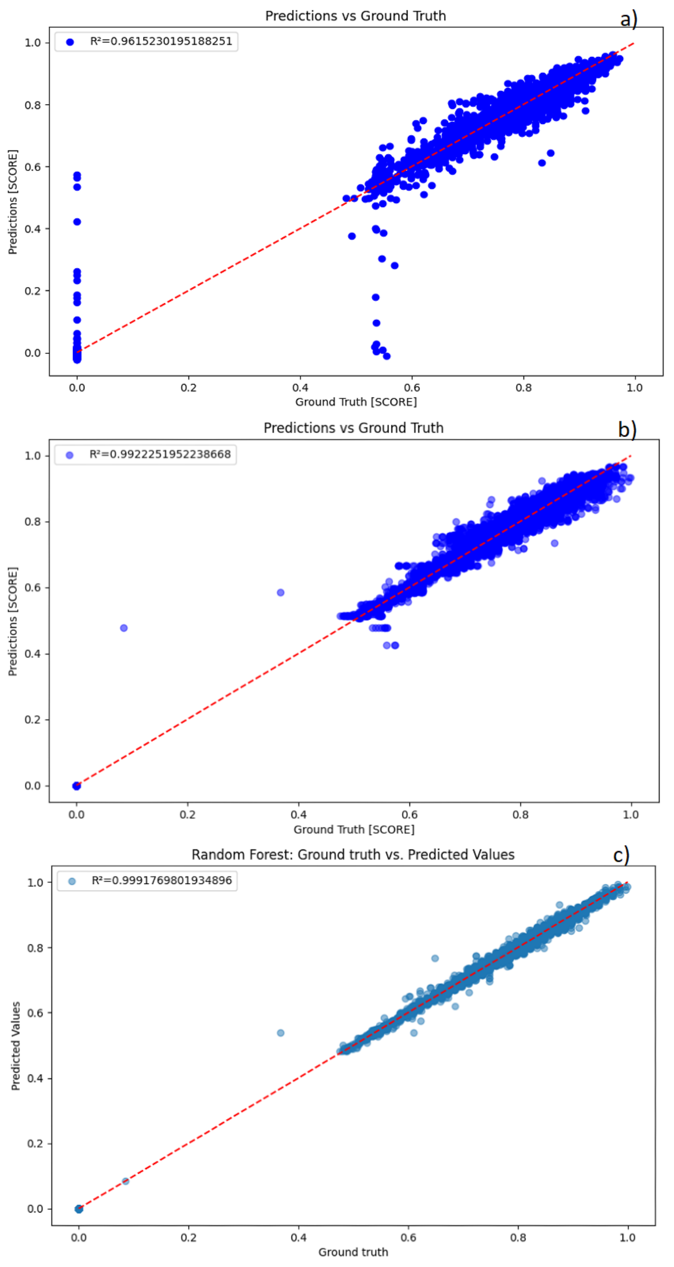

Figure 4. Inference results. a) LSTM, b) CNN regressor, c) Random Forest.

VI. CONCLUSION AND FUTURE WORK

The main focus of this work is to determine possible vehicles to be dynamic reference station and as well as to improve an estimated position with precision using AI algorithms such as LSTM, CNN and RF.

The algorithms utilized have demonstrated the potential to achive promising results using GNSS data. However, this approach still has some limitations and there is space to improve the methodology. The main limitations are related to dataset requirements for AI model applications, because using incomplete data to train algorithms can result in inaccurate results. GNSS data and missing data for non visible satellites presents a significant challenge to be addressed. Additionally, the number of vehicles may change over time.

Solutions to these challenges are related to obtaining more data using a longer simulation, acquiring data for all 24 satellites and alternative algorithms capable of handling varying input data sizes. These algorithms could be transformers and Graph Convolutional Networks (GCNs) [29].

The labeling method applied in this study was a direct approach considering only evaluation of the provided data. However, it is intended that in the future an alternative labeling method will be implemented, that may involve, for example, the application of a method based on GNSS error corrections. By leveraging data from simulations, it is possible to compare the performance of error correction methods against the available ground truth and create a dataset where labeling is obtained using a similar scoring idea, but inversely reflecting the positioning error after applying GNSS corrections. This proposed approach, although more complex, aims to enhance the robustness of the labeling process.

ACKNOWLEDGMENT

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 179491; Funding Reference: SIFN-01-9999-FN-179491].

REFERENCES

[1] F. de Ponte Müller, E. M. Diaz, and I. Rashdan, “Cooperative positioning and radar sensor fusion for relative localization of vehicles,” in 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, pp. 1060-1065, 2016. https://doi.org/10.1109/IVS.2016.7535520.

[2] J. Gabela et al., “Experimental Evaluation of a UWB-Based Cooperative Positioning System for Pedestrians in GNSS-Denied Environment,” Sensors, vol. 19, no. 23, pp. 5274, Nov. 2019. https://doi.org/10.3390/s19235274.

[3] M. Rohani, D. Gingras, and D. Gruyer, “A Novel Approach for Improved Vehicular Positioning Using Cooperative Map Matching and Dynamic Base Station DGPS Concept,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 1, pp. 230-239, Jan. 2016. https://doi.org/10.1109/TITS.2015.2465141.

[4] H. J. Kim, Y. H. Kim, J. H. Lee, S. J. Park, B. S. Ko, and J. W. Song, “Improving the Accuracy of Vehicle Position in an Urban Environment Using the Outlier Mitigation Algorithm Based on GNSS Multi-Position Clustering,” Remote Sens., vol. 15, pp. 3791, 2023. https://doi.org/10.3390/rs15153791.

[5] N. Zhou, W. Chen, C. Li, L. Du, D. Zhang, and M. Zhang, “An Application-Oriented Method Based on Cooperative Map Matching for Improving Vehicular Positioning Accuracy,” Electronics, vol. 11, pp. 3258, 2022. https://doi.org/10.3390/electronics11193258.

[6] X. Li, J. Huang, and X. Li, “Review of PPP–RTK: achievements, challenges, and opportunities,” Satell Navig 3, 28, 2022, Accessed: May. 28, 2024. https://doi.org/10.1186/s43020-022-00089-9.

[7] “Systems and Solutions for Inertial Navigation, Stabilization, Guidance and Control, Made in Germany,” Accessed: May. 28, 2024. Available: https://www.imar-navigation.de/.

[8] “CARLA Simulator,” Accessed: May. 28, 2024. Available: https://carla.org/.

[9] “MathWorks, Navigation Toolbox - MATLAB, 2024,” Accessed: May. 28, 2024. Available: https://www.mathworks.com/products/navigation.html.

[10] E. C. M. Adrian, S. L. Chandra, S. C. M. Etienne, E. C. M. Christian, and S. P. Kulwant, “Intelligent transport systems in multimodal logistics: A case of role and contribution through wireless vehicular networks in a sea port location,” International Journal of Production Economics, vol. 137, no. 1, pp. 165-175, 2012, Accessed: May. 28, 2024. https://doi.org/10.1016/j.ijpe.2011.11.006.

[11] J. Yao, A. T. Balaei, M. Hassan, N. Alam and A. G. Dempster, “Improving Cooperative Positioning for Vehicular Networks,” IEEE Transactions on Vehicular Technology, vol. 60, pp. 2810-2823, 2011, Accessed: May. 28, 2024. https://doi.org/10.1109/TVT.2011.2158616.

[12] D.J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, “Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review,” Sensors, vol. 21, pp. 2140, 2021, Accessed: May. 28, 2024. https://doi.org/10.3390/s21062140.

[13] R. Shrestha, S.Y. Nam, R. Bajracharya, and S. Kim, “Evolution of V2X Communication and Integration of Blockchain for Security Enhancements,” Electronics, vol. 9, pp. 1338, 2020, Accessed: May. 28, 2024. https://doi.org/10.3390/electronics9091338.

[14] Q. Zhang, L. Zhang, A. Sun, X. Meng, D. Zhao, and C. Hancock, “GNSS Carrier-Phase Multipath Modeling and Correction: A Review and Prospect of Data Processing Methods,” Remote Sens., vol. 16, pp. 189, 2024, Accessed: May. 28, 2024. https://doi.org/10.3390/rs16010189

[15] E. Kaplan and C. Hegarty, *Understanding GPS/GNSS: Principles and Applications, Third Edition*, 2017, Accessed: May. 28, 2024.

[16] Y. Zhu, G. Feng, B. Qiu, and X. Zheng, ”State Space Representation (SSR) for Real-Time PPP-RTK,” in 2018 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS), 2018, pp. 1-5. DOI: 10.1109/MARSS.2018.8481158.

[17] “SSR Vs. OSR - Geo++ — GNSS technology,” Accessed: May. 28, 2024. Available: https://www.geopp.de/.

[18] A. Siemuri, K. Selvan, H. Kuusniemi, P. Valisuo, and M. S. Elmusrati, “A Systematic Review of Machine Learning Techniques for GNSS Use Cases,” IEEE Transactions on Aerospace and Electronic Systems, vol. 58, no. 6, pp. 5043-5077, Dec. 2022. https://doi.org/10.1109/TAES.2022.3219366.

[19] “How to apply LSTM using PyTorch,” Accessed: May. 28, 2024. Available: https://cnvrg.io/pytorch-lstm/.

[20] “MathWorks, Navigation Toolbox - MATLAB, 2024,” Accessed: May. 28, 2024. Available: https://de.mathworks.com/discovery/lstm.html.

[21] “exploring-potential-long short-term-memory-lstm-networks-pandey,” Accessed: May. 28, 2024. Available: https://www.linkedin.com/pulse/exploring-potential-longshort-term-memory-lstm-networks-pandey.

[22] B. Yildiz et al., “CNN based sensor fusion method for real-time autonomous robotics systems,” Electrical Electronics Engineering, Karamanoglu Mehmetbey University, Karaman, Turkey, Electrical Electronics Engineering, Konya Technical University, Konya, Turkey, R&D Robotics Software Engineer at Elfatek Elektronik Ltd. Şti., Konya, Turkey, Robotic Automation Control Laboratory (RAC-LAB), Konya Technical University, Konya, Turkey, 2022, Accessed: May. 28, 2024.

[23] L. Alzubaidi et al., “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” J Big Data, vol. 8, pp. 53, 2021, Accessed: May. 28, 2024. https://doi.org/10.1186/s40537-021-00444-8.

[24] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, “A review of convolutional neural networks in computer vision,” Artif. Intell. Rev., vol. 57, no. 4, pp. 99, Accessed: May. 28, 2024. https://doi.org/10.1007/s10462-024-10721-6.

[25] A. Loe, S. Murray, and Z. Wu, “Random Forest for Dynamic Risk Prediction of Recurrent Events: A Pseudo-Observation Approach,” arXiv, Accessed: May. 28, 2024. https://ar5iv.org/abs/2312.00770.

[26] H. Zhang, D. Nettleton, and Z. Zhu, ”Regression-Enhanced Random Forests,” *arXiv*, vol. 19, pp. 1-26, Accessed: May. 28, 2024. https://arxiv.org/abs/1904.10416.

[27] J. Ansel et al., “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation,” *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Accessed: May. 28, 2024. https://www.jokeren.tech/publication/jason-2024-pytorch/.

[28] “3 Best metrics to evaluate Regression Model? — by Songhao Wu — Towards Data Science,” Accessed: May. 28, 2024. Available: https://towardsdatascience.com/.

[29] A. Mohanty and G. Gao, “Learning GNSS Positioning Corrections for Smartphones Using Graph Convolution Neural Networks,” Navigation: Journal of the Institute of Navigation, vol. 70, no. 4, pp. e622, 2023. https://doi.org/10.33012/navi.622.

[30] Y. Gu, L. -T. Hsu and S. Kamijo, “GNSS/Onboard Inertial Sensor Integration With the Aid of 3-D Building Map for Lane-Level Vehicle Self-Localization in Urban Canyon,” Accessed: Jun. 25, 2024. Available: https://ieeexplore.ieee.org/document/7314948.