SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I9P109

Volume 13 Issue 9, 147-159, September 2026

Original Article

Optimal Trajectory Prediction System for Automated Driving using Deep Learning

Brigitte Paola Mamani Concha[1], Enrique Aurelio Amu Jimenez[2], Jesús Talavera Suarez[3]

[1,2,3]*Universidad Nacional de San Agustin de Arequipa, Arequipa, Perú.*

[1]*Corresponding Author : bmamanico@unsa.edu.pe*

Received: 05 August 2026 Revised: 15 September 2026 Accepted: 18 September 2026 Published: 29 September 2026

Abstract - *Research in the field of autonomous driving has revealed that one of the main focuses is on predicting the future trajectories of vehicles. These accurate forecasts enable safe navigation and efficient route planning. A hybrid Convolutional Neural Network (CNN)-Long Short-Term Memory (LSTM) framework integrated with the Robot Operating System (ROS) is presented in this work. The aim of the framework is to estimate future vehicle trajectories in real time. The network was trained using vehicle motion data from the Argoverse Motion Forecasting dataset. CNN layers extract local temporal and kinematic patterns from input sequences, and LSTM layers model their temporal evolution. Once trained, the model is incorporated into an ROS-based pipeline where nodes dedicated to processing incoming trajectory information publish predicted paths for real-time visualisation in RViz. Performance of the model was evaluated using Mean Squared Error (MSE), Average Displacement Error (ADE) and Final Displacement Error (FDE), and was then compared to that of classical motion prediction methods. Research results are promising, as they enable the computational efficiency required for real-time operation alongside predictive accuracy. Additionally, robotic middleware, deep learning and a publicly available benchmark dataset were combined to develop a reproducible experimental framework that will facilitate the replication and evaluation of autonomous driving systems.*

Keywords - *Autonomous driving, Trajectory prediction, Deep Learning, CNN-LSTM, Simulation.*

1. Introduction

For accurate trajectory prediction and to avoid collisions and optimise vehicle behaviour in constantly changing, dynamic environments, it is essential that automated vehicle driving is safe and efficient. Although classical approaches based on kinematic or heuristic models work well in structured and fixed scenarios, they are not as robust when faced with complex interactions, future multimodality and the unstructured characteristics of urban traffic [1]. Over the last decade, Deep Learning (DL) has proven to be highly effective in modelling nonlinear relationships and complex temporal dependencies in motion prediction. Recurrent models, such as Long Short-Term Memory (LSTM) networks, have been widely used to capture temporal dynamics in human and vehicle trajectories. Meanwhile, variants with social clustering or generative architectures have improved the plausibility and diversity of predictions in multi-agent environments. Representative examples include Social LSTM for human trajectories, Social GAN for socially plausible predictions, and DESIRE for the multimodal representation of distant futures [2–4].

More recently, spatio-temporal architectures or those explicitly considering interactions through graphs have exhibited systematic gains on vehicular and pedestrian prediction benchmarks. A graph-structured model fusing agent dynamics and semantic maps was introduced in Trajectron++, and convolutional social pooling showed the importance of representing influence from neighbors locally in space to model it better. These advances support the validity of combining spatial (maps, occupancy) and temporal data (movement history) to obtain more realistic and accurate trajectories [5, 6]. The creation of models has likewise involved the use of truly realistic datasets and simulators. Notably, datasets such as Argoverse and the Waymo Open Dataset offer realistic vehicle trajectories and highly detailed semantic maps. Datasets can be useful for assessing model performance and generalization. On the other hand, simulation platforms such as CARLA and Gazebo allow researchers to define and test specific driving scenarios in a controlled environment. Nevertheless, several issues remain to be addressed. Predicting the future behaviour of multiple road users remains challenging, particularly when interactions become complex and uncertain. Furthermore, the generated trajectories should be accurate, dynamically feasible, and resilient to the unplanned manoeuvres of other agents within the environment. Another critical constraint is low computational latency, which is necessary for the predictions

to be used effectively in real-time applications. While approaches like MultiPath, MultiPath++ and prediction based on anchors have shown some success, no clear conclusion has been drawn on the best way to integrate rich spatial feature extraction methods, sequential learning models and vehicle dynamic constraints in a single solution [9, 10]. Accurate prediction of trajectory is very important for automated vehicle driving systems since it determines human decisions and route planning. This has led to the evolution of systems supporting the development and evaluation of such systems. A representative example is the Robot Operating System (ROS), which is being used as a middleware platform in robotics and is rapidly gaining traction because of its adaptability, modularity and the extensive library of tools provided by the community [11]. In this design, the perception, prediction and control modules can all be integrated into one single framework, which makes the construction of very complex and autonomous applications easy [11, 12]. Likewise, visualisation software such as RViz can be employed to access data from sensors, vehicle states and predicted trajectories using real-time data in an ideal manner.

To deal with these problems, it suggests a CNN-LSTM model combined with an autonomous driving environment based on ROS. This provides a way to predict trajectories. Furthermore, the model can leverage the Argoverse Motion Forecasting dataset, consisting of real-world vehicle observations, to learn motion behaviours from realistic data sources and accurately forecast future trajectories. Integrating deep learning with robotic middleware not only enables accurate trajectory estimation, but also real-time experimentation, visualisation and system evaluation within a unified environment. This is ideal for researching motion forecasting methods and their application in autonomous driving scenarios. The trained model enables real-time processing and visualisation of trajectories in RViz. Furthermore, this method integrates spatial feature extraction and time-sequence modelling. It outperforms traditional approaches and monolithic architectures in terms of metrics such as Mean Squared Error (MSE), Mean Displacement Error (ADE) and Final Displacement Error (FDE). The remainder of the paper is organised as follows: Section 2 presents related work in vehicle motion estimation, autonomous driving, and machine learning. Section 3 presents the methodology of this work. Section 4 explains the experimental development. Section 5 presents the results, which demonstrate the success of the proposed system. Section 6 presents the discussion that emerged from analysing the results. Finally, Section 7 presents the conclusions reached in this research.

2. Related Works

Alahi et al. [13] introduced a trajectory prediction model which explicitly considered pedestrian interactions. Temporal information and social relationships are skilfully incorporated in the methodology, as it has been designed for predicting mass human movement. Unlike conventional single-agent models, it considered the influence of nearby pedestrians when estimating future trajectories. Subsequent trajectory prediction methods that explicitly model interactions among multiple agents were based on this work.

Gupta et al. [14] proposed a generative adversarial framework called Social GAN for predicting multiple socially plausible trajectories. It incorporates an interaction-pooling system and diversity reduction to replicate the subsequent movement's multimodal character. Consequently, the model was not confined to a single, predetermined trajectory, but could generate various plausible future paths.

Deo and Trivedi [15] extended the concept of social interaction modeling to vehicle trajectory prediction using a convolutional architecture. They represented the local traffic environment as a spatial grid and processed this using convolutional layers. The model was used to simulate the interactions between vehicles in close proximity, in situations where traffic is dense and moving slowly. Compared with conventional LSTM-based approaches, this method achieved better prediction results on the NGSIM dataset.

Salzmann et al. [5] developed Trajectron++, a graph-based probabilistic model that represents dynamic agents together with semantic scene information. Information from heterogeneous agents and their surrounding environment is incorporated into the architecture so that dynamically feasible trajectories can be generated. Its graph-based formulation also enables the model to accommodate a variable number of interacting agents.

Chai et al. [16] proposed MultiPath, a trajectory prediction framework based on a discrete set of trajectory hypotheses combined with multimodal regression. Its contribution generates a prediction across several possible trajectories and also estimates their respective probabilities. An additional way to validate the model is that subsequent methods have been influenced by this way of representing uncertainty.

Gao et al. [17] introduced VectorNet, a neural architecture that represents maps and trajectories as vectorized elements rather than rasterized 2D images. The model uses topological relationships between these vectorised elements to encode map structure and agent trajectories. The use of this representation reduces the reliance on image-based scene encoding, while keeping important geometric and contextual information for predicting trajectories.

Phan-Minh et al. [18] proposed CoverNet, which formulates multimodal trajectory prediction as a classification problem over a predefined set of possible trajectories. The model selects candidate trajectory classes based on the

observed scene and the agent's state. Providing an alternative to direct trajectory regression, it evaluated the formulation using real-world driving data.

Manganlam et al. [19] proposed PECNet, which conditions the prediction on estimated endpoints. Capturing long-term multimodality is achieved by simplifying the generation of trajectories from distant targets, thereby improving the diversity and plausibility of the results.

Yuan et al. [20] developed AgentFormer, a Transformer-based model incorporating inter-agent and time-based attention. It was demonstrated by the work that recurrent models are not as effective as self-attention in scenarios where numerous agents are interacting with each other in a complex manner.

Chang et al. [1] presented Argoverse, a dataset with real vehicle trajectories and high-definition maps, designed for 3D tracking evaluation and prediction. Its open-access availability made it a key benchmark for measuring the impact of semantic information on the prediction accuracy of many proposed models.

Zhang [8] proposed SAC-LSTM, an algorithm that combines long- and short-term memory with a smooth, actor-critical framework for rapid path planning in mobile robots. Previous methods were outperformed by this one, as demonstrated by the results of the validation on the Robot Operating System (ROS) and Gazebo. The success rates were higher, and convergence was achieved more quickly. Shows the potential of hybrid approaches in dynamic environments.

Dosovitskiy et al. [7] developed CARLA, an open-source simulator that allows the generation of synthetic driving data with control over sensors, traffic, and scenarios. Tool has become the standard for training and validating models where real-world data is scarce.

The NGSIM dataset [21] is another widely used benchmark for vehicle trajectory prediction. It contains real-world traffic observations with a high temporal resolution, including driving manoeuvres such as lane changes and vehicle interactions. NGSIM is ideal for assessing the precision and resilience of motion prediction models in realistic traffic scenarios because of these features.

Cheng et al. [22] proposed GATraj, a trajectory prediction model incorporating graph attention mechanisms to represent the relationships between interacting agents.

The model's ability to allocate different weights to neighbouring agents based on their relevance to the target vehicle is known as the attention mechanism. The effectiveness of this approach lies in its ability to provide a realistic representation of diverse interactions in urban traffic scenarios.

3. Methodology

Researchers developed a trajectory prediction experiment using an approach that combines CNN-LSTM algorithms with a ROS-based autonomous driving research system. This novel approach uses trajectory data derived from the Argoverse motion prediction dataset. Data was publicly obtained under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license for non-profit academic research purposes only. Argoverse data is widely used by specialists to train neural network models that can predict future vehicle trajectories based on historical motion information. The trained model enables real-time processing and visualisation of trajectories in RViz. Furthermore, the approach integrates spatial feature extraction with time-sequence modelling, achieving enhancements in metrics such as Mean Squared Error (MSE), Mean Displacement Error (ADE) and Final Displacement Error (FDE), in monolithic architectures. Figure 1 illustrates the overall pipeline of the proposed system.

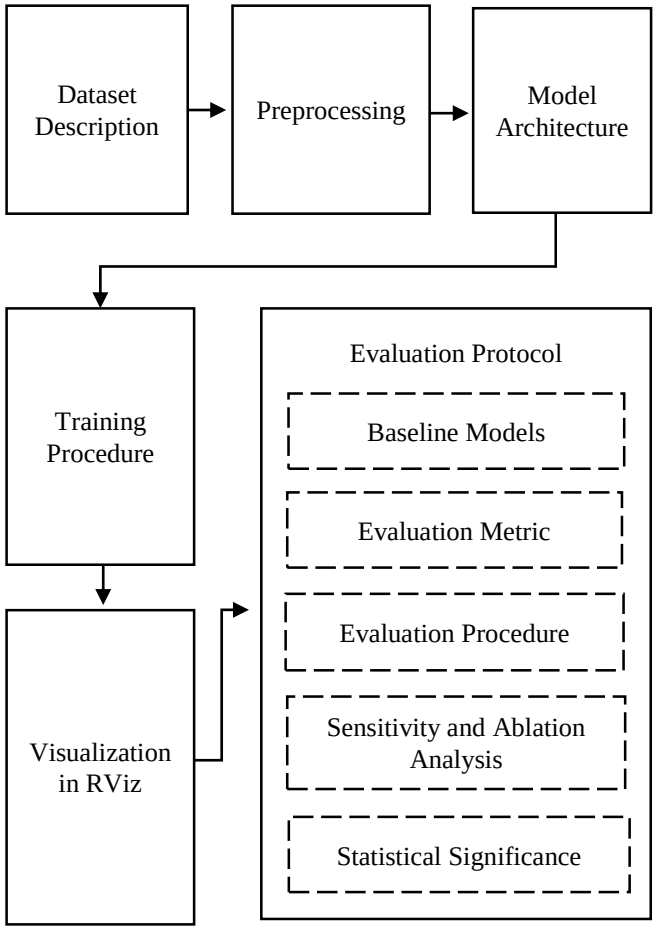


Fig. 1 Methodology of the proposed system

3.1. Dataset Description

The motion forecasting model was trained and evaluated using data derived from the Argoverse Motion Forecasting Dataset, a large-scale benchmark designed for motion

forecasting research in autonomous driving. Dataset contains annotated trajectories of multiple traffic participants collected in complex urban environments. Data includes time-series information describing the positions and motion of vehicles and other agents interacting in real-world traffic scenarios. Thanks to the sequential location coordinates (x, y) provided by the datasets recorded in various urban scenarios, including pedestrian crossings, lane changes, and vehicle interactions, it is possible to study Temporal's movement patterns. Dataset is frequently used to assess trajectory prediction algorithms because of its size and diversity of driving scenarios. The historical trajectory of a vehicle is used as the input to the prediction model. A fixed observation window is considered, containing the past motion states of the agent, while the model predicts the future motion over a predefined prediction horizon.

The prediction tags are derived directly from the future development of the ego state, with the time index moved from the first moment of each observation period. It constructs a sequence of future positions (x, y) in the egocentric coordinate frame defined at time t_0. Adopting such a strategy eliminates the need for external expert controllers, thereby ensuring consistency between the input trajectories and the prediction targets.

Adoption of the strategy eliminates the need for external expert controllers, thereby ensuring consistency between the input trajectories and the prediction targets. To prevent information leakage during training and evaluation, the dataset has been divided into complete scenarios. Data partitioning in this manner guarantees that the model is tested in circumstances unknown to it, with no temporal continuity between the splits. Different initialisation seeds were also used to improve the robustness of the experiments. Finally, several automatic integrity checks were implemented to ensure the dataset's reliability. As a result of this process, a stable, reproducible and representative dataset was created for training, validating and evaluating the proposed future path estimation system.

3.2. Preprocessing

Data preprocessing aimed to ensure the temporal and spatial coherence of the training sequences. All trajectories were expressed in an egocentric frame defined at $t0$, where the x-axis is oriented in the vehicle's forward direction, the y-axis is to the left, and the initial orientation is normalized with yaw = 0. For each time window, a rigid transformation was applied from the global simulator frame to the egocentric frame at $t0$, which eliminated the accumulated drift associated with navigation and guaranteed invariance with respect to absolute position and orientation. The resampling of state and sensor streams was conducted at 10 Hz over a common timeline. In cases of desynchronisation or data loss, the application of linear interpolation in position and orientation was implemented, while the reconstruction of longitudinal velocity was achieved by means of smoothed differencing. A uniform cadence is ensured by this procedure, and aliasing in the time series is reduced.

The prediction scheme adopted a short horizon with sliding windows; the input was set to T_{past} = 2.0 s (20 steps), and the output to T_{future} = 3.0 s (30 steps). The egocentric frame was anchored at the most recent point in time. Principal variables were x, y, yaw and v, which could be complemented by numerically derived and filtered longitudinal acceleration and yaw rate if this improved validation and avoided overparameterisation. The z-score method was used to normalise all variables. Statistics were obtained only from the training set, which was then frozen for validation, testing and online inference. Samples were generated using sliding windows with 50% overlap to ensure coherence and prevent information leakage. Any segments dominated by prolonged stops or kinematic anomalies that could have biased the sample set were discarded. Furthermore, the windows were configured so as not to cross scenario boundaries or simulation reset points. The process consequently yielded a balanced, temporally coherent and artefact-free dataset intended for system testing and training.

3.3. Model Architecture

Resulting model combines one-dimensional temporal convolutions (1D CNNs) and LSTM networks in order to model both short-term movement patterns and longer-term temporal dependencies. While the convolutional layers process short segments of the input time series and extract local kinematic features, such as changes in acceleration and steering behaviour, the LSTM layers process these features. LSTM layers are able to retain temporal information across successive observations and model the evolution of the vehicle's state over the prediction horizon. Using such a representation, future trajectories can be estimated while maintaining temporal coherence among the predicted positions.

Three functional blocks comprise the model. First, the application of the time extraction module is the use of two to three Conv1D layers with kernel sizes of 3 to 5 and 32 to 64 filters per layer, combined with Rectified Linear Unit (ReLU) activation, batch normalization, and moderate dropout (p = 0.1–0.2) to stabilize activation statistics and prevent overfitting. Consequently, a rich feature map is produced that encodes local changes and short-range nonlinear relationships. Next, depending on the best validation performance, a sequential aggregation module that deploys either a many-to-many or a many-to-one configuration of 1–2 128-unit LSTM layers, capturing longer temporal dependencies than those covered by the convolutional kernels. Finally, the temporal decoder projects the LSTM output onto the 30-step target sequence (T_{future} = 3.0 s) using time-distributed fully connected layers that generate the coordinates $\{x, y\}$ in the egocentric frame. Initializations were used for ReLU layers

and Xavier for linear projections, resulting in a final linear output without activation, maintaining the physical scale of the trajectories.

A variety of regulation techniques are employed to circumvent excessive adjustment and to maintain the integrity of the training regimen. These included the use of dropout in convolutional blocks and between LSTM layers, early stopping with a 10–15-epoch patience, monitoring the validation MSE, global gradient clipping with a 1.0-norm threshold to prevent bursts, and adaptive learning rate tuning with plateau reduction (factor 0.5–0.7). Training was carried out using the Adam optimiser with an initial learning rate of 1×10^{-3}, with each sequence preserved in temporal order and batches of 64–128 samples used. To ensure geometric interpretability and consistency, the predicted trajectories were kept within the egocentric framework, and inverse roto-translation was applied only when metrics were required within the global simulator framework. Light smoothing is permitted during inference, provided that it does not cause any delays or disruptions to kinematic consistency. Real-time inference at 10 Hz in ROS is enabled, even on modest hardware, making it ideal for predicting the motion of a minimum viable vehicle.

3.4. Training Procedure

While the model was being trained, it used the Adam optimiser with an initial learning rate of 1×10^{-3}. Its value was dynamically adjusted using a *ReduceLROnPlateau* scheduler, which reduced it by a factor of 0.5 when the validation MSE did not improve for five consecutive epochs. The minimum threshold was set to 1×10^{-6}. The training regime involved batches of 128 samples and a maximum of 100 epochs, with early stopping applied with a patience of 10 epochs to select the final model based on the lowest validation MSE (with the network and optimiser state being checked at this point). To mitigate biases derived from the sampling order, the batches were shuffled at each epoch, while the partition into training, validation and testing was performed at the level of complete scenarios to avoid spatial or temporal overlaps.

Deterministic seeds were established in Python, NumPy and PyTorch, and backend indicators were enabled to ensure repeatability between runs. Weight initialisation used a normal distribution in the convolutional layers and a Xavier uniform distribution in the recurrent and linear layers. Moderate dropout (p = 0.1–0.2) in convolutions and Long Short-Term Memory (LSTM) units, L2 weight decay of 1 × 10^(−5), and global gradient clipping with a maximum norm of 1.0 were incorporated to provide regularisation and ensure numerical stability, thereby avoiding gradient explosion in long sequences. Data preprocessing involved per-variable normalisation within the egocentric framework using statistics from the training set, which was applied consistently during the validation and testing stages. The data loader constructed sliding windows with 50% overlap and discarded segments with prolonged dwells or severe anomalies. During training, it monitored global and per-step Mean Squared Error (MSE), as well as trajectory accuracy in the validation set, to detect overfitting. Finally, the main hyperparameters (the number of filters, the number of LSTM units, the dropout rate and the window length) were tuned using a performance-guided restricted grid search in the validation set, with the minimisation of the MSE at the 3 s horizon taken as the primary criterion.

3.5. Visualization in RViz

Real-time visualization was incorporated to support the evaluation of the proposed vehicle motion estimation. During the experiments, the current vehicle state and the estimated future path were displayed in RViz, where the ego vehicle (represented as a white pickup truck) was shown together with the surrounding traffic information.

Deviations in vehicle motion under various driving conditions could be identified thanks to this setup, which allowed a direct visual comparison between the anticipated and ground-truth trajectories. In addition to qualitative evaluation, the visualization environment was a useful tool for tracking data flow via ROS nodes and confirming the consistency of the prediction process while it was being carried out. In this way, RViz successfully complemented the numerical evaluations of performance in the driving tests.

3.6. Evaluation Protocol

3.6.1. Baseline Models

In order to establish a solid baseline and quantify the improvement achieved by the proposed model, two widely recognized kinematic approaches were selected as baselines for prediction pipeline. The first one is the Constant Velocity (CV) model, which assumes that the speed and direction of the vehicle don't change. It uses a method to calculate the future position of the vehicle.

The second is the Constant Turn–Velocity (CTRV) model, which assumes constant values for both *v* and the turn rate (yaw_{rate}), and integrates the pose under a simplified bicycle model that preserves local curvature. Both benchmark methods were evaluated in the same scenarios and time windows as the CNN-LSTM scheme, producing future position sequences {x, y} in the egocentric frame. This allows a direct and rigorous comparison of the performance of kinematic approximations and the deep learning-based approach.

3.6.2. Evaluation Metric

The performance of the models was evaluated using two quantitative metrics. First metric is the Mean Squared Error (MSE) of the predicted positions, expressed in Equation (1).

$$MSE = \frac{1}{N} \sum_{i=1}^{N} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \qquad (1)$$

Where (x_i, y_i) corresponds to the ground-truth position and $(\hat{x}_i, \hat{y}_i)$ to the predicted position. The second metric is trajectory accuracy, defined as the percentage of predicted points that fall within a radius r from the ground-truth trajectory. Two tolerance thresholds were considered in Equation (2).

$$r \in \{0.5m, 1.0m\} \quad (2)$$

These metrics allow the evaluation of both the absolute positional error and the spatial reliability of the predicted trajectories. The loss function was defined as the MSE averaged over all future steps and both spatial axes, see Equation (3). Optionally, an increasing temporal weighting $w(t)$ was applied to give greater relevance to distant horizons, normalizing the weights so that the total sum equaled the number of steps to preserve the scale of the error.

$$L_{MSE} = \frac{1}{N \, x \, T_f} \sum_{i=1}^{N} \sum_{T=1}^{T_f} w(t) \left\| \hat{p}_{i,t} - p_{i,t} \right\|^2 \quad (3)$$

Optionally, an increasing temporal weighting $w(t)$ was applied to give greater relevance to distant horizons, normalizing the weights so that the total sum equaled the number of steps to preserve the scale of the error.

3.6.3. Evaluation Procedure

All metrics were computed using the test set, which was kept completely separate from the training and validation datasets to ensure an unbiased estimation of the model's generalisation capability. The MSE was calculated over the entire prediction horizon and also analyzed step-by-step to obtain temporal error curves. Additionally, trajectory accuracy was reported both as an average over the entire prediction horizon and at each prediction step to evaluate how prediction quality degrades over time. Analysis of error accumulation was enabled by the reporting of performance metrics at specific time horizons of 1 s, 2 s and 3 s, allowing for the evaluation of prediction stability as the forecast horizon increases.

3.6.4. Sensitivity and Ablation Analysis

Sensitivity analysis was used to examine the effects of the design on prediction performance. First, the impact of varying the length of the input window was evaluated using the combinations defined in Equation (4).

$$T_{past} \in \{1.0, 2.0, 3.0\} \, s \quad (4)$$

And in Equation (5).

$$T_{future} \in \{2.0, 3.0, 4.0\} \, s \quad (5)$$

Analysis allows the characterization of the trade-off between the amount of historical context available to the model and the prediction error at increasing horizons. Second, robustness to sensor noise was evaluated by introducing additive Gaussian perturbations to position, heading, and velocity measurements using realistic ranges in Equation (6).

$$\sigma_{pos} \in [0.03, 0.10]\text{m}$$
$$\sigma_{yaw} \in [0.2 \circ, 0.5 \circ] \quad (6)$$
$$\sigma_{\text{v}} \in [0.05, 0.15] \, \text{m/s}$$

The deterioration in performance when faced with noisy circumstances was juxtaposed with the CV and CTRV benchmarks. Third, the impact of architectural components was analyzed by comparing three variants: The proposed CNN–LSTM hybrid model, a standalone LSTM network with a comparable number of parameters, and a temporal CNN model with a recurrent decoder. The contribution of convolutional layers in extracting local motion patterns and the role of LSTM memory in modelling long-term temporal dependencies can be identified through this comparison. Finally, the effect of the sampling frequency was investigated by contrasting runs at 5 Hz and 10 Hz with a constant time horizon, to determine the impact of undersampling on prediction stability, error accumulation and the computational implications for real-time deployment.

3.6.5. Statistical Significance

Differences are reported as scenario and overall means and standard deviations, where applicable, and relative percentage improvements in CV and CTRV are calculated. For comparisons between model variants, paired-samples t-tests on scenario metrics are included, with a significance threshold in Equation (7).

$$\alpha = 0.05 \quad (7)$$

The results are presented using: MSE versus prediction horizon curves, and summary tables reporting metrics at 1s, 2s, and 3s. This presentation ensures a clear and reproducible interpretation of the experimental results.

4. Experimental Development

4.1. Simulation and Middleware Platform

Experimentation was performed on a simulation platform with Gazebo Classic 11 version, an ODE physics engine and ROS Noetic with *gazebo_ros_pkgs*. This infrastructure enabled stable reproduction of the Ackermann dynamics of the ego vehicle, creation of standardized urban and rural scenarios, and ground-truth measurements available for training and ground-truth vehicle states available for motion forecasting system validation.

The simulation environment was designed to be in a controlled traffic setting with no adverse weather conditions

or extreme traffic densities, since the goal of this study is not on global path planning or high-level vehicle control, but on vehicle motion prediction.

The implementation of the experimental stack was done using Ubuntu 20.04 LTS and involved various ROS packages for vehicle control, sensor simulation, and scenario definition. The packages *ros_control* and *gazebo_ros_control* were used to provide vehicle actuation, and the *vector_gazebo_plugins* package was used to simulate inertial sensors (IMU). Simulation environments were set up using Simulation Description Format (SDF) world files with basic road signage, intersections, and traffic lights. Moderate traffic levels were simulated by creating optional dynamic actors, vehicles and pedestrians using ActorPlugin. Experimental data were collected with deterministic ROSBAG configurations (*use_sim_time* and */clock*) to simulate time synchronization and repeatability of simulations. Table 1 summarizes the main components of the simulation platform and middleware used in this study. Figures 2 and 3 show the 3D designs of the scenario and robot, respectively.

Table 1. Setting up the simulation environment

Component	Version / Tool	Description / Main use
Operating system	Ubuntu 20.04 LTS	Base platform for ROS and Gazebo
Robotic middleware	ROS Noetic Ninjemys	Orchestration, messaging, and data logging
Physical simulator	Gazebo Classic 11 (ODE)	Vehicle dynamics and sensor simulation; fixed-step and real-time factor control
Main packages	gazebo_ros_pkgs, gazebo_plugins	Gazebo–ROS integration, basic simulation plugins
Virtual sensors	vector_gazebo_plugins (IMU)	GPS/IMU Simulation for Vehicle States
Vehicle control	ros_control, gazebo_ros_control	Actuator interfaces and simplified control (Ackermann)
Definition of worlds	SDF maps	Urban and rural scenarios with intersections, signs and traffic lights
Dynamic actors	ActorPlugin	Optional pedestrian and vehicle simulation for moderate traffic
Truth- Ground	/gazebo/model_states, /gazebo/link_states, /tf	Reference states and transformations for training and validation
Temporality	/clock, use_sim_time	Time synchronization and deterministic reproducibility
Data logging	rosbag (MCAP/ROS1)	Capture and reproduction of experimental data

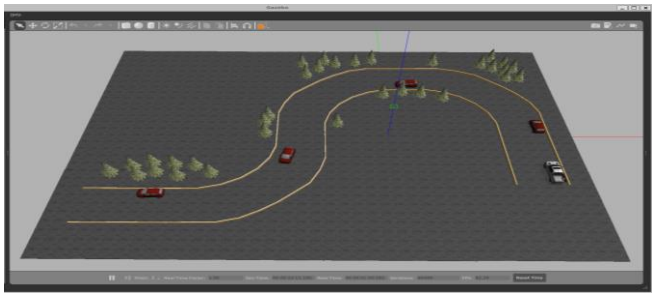


Fig. 2 Test scenario

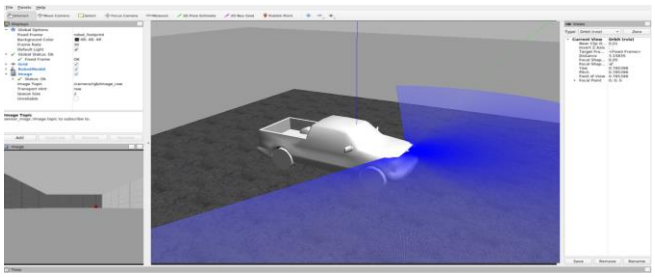


Fig. 3 3D design of the vehicle

4.2. Deployment in ROS

The ROS Noetic architecture orchestrates the data flow from simulation to inference and evaluation of the trajectory prediction model. This pipeline includes a node for acquiring and synchronizing the ego vehicle state history from the simulation topics, a node for creating the sliding time windows for the offline training, and a node for the online inference that loads the pre-trained CNN-LSTM model and publishes the estimated future trajectory. Moreover, an evaluation node is used throughout the simulated time to compare the predictions and the ground truth given by Gazebo, and to compute performance measures such as MSE and trajectory accuracy.

All orchestration is done via roslaunch, and both rosbag registration and parameters are stored on the ROS server for traceability and reproducibility. Message interfaces adhere to ROS standards: vehicle state (x, y, yaw, v, a) is represented by a *nav_msgs/Odometry* or a compact message; output of prediction (sequences of *geometry_msgs/PoseStamped* or *nav_msgs/Path*) is published with horizon H and resolution Δt; minimal ego control

(*ackermann_msgs/AckermannDriveStamped* or *geometry_msgs/Twist*). Some optional services are available (*std_srvs/Trigger*, PredictTrajectory) for reset or to take a direct trajectory query. Preprocessing includes egocentric transformations (tf/tf2), data normalization and time windowing at 10-20 Hz with a stride 1. Training is executed offline, outside ROS, with data exported from rosbag to CSV / Parquet and scenario validation, and inference is executed in real-time within *predictor_node*, at 10 Hz with the target of 50 ms or less latency. Synchronization is done with *use_sim_time=true* and with message filters (ApproximateTime mode) and by controlling the Real-Time Factor (RTF) in Gazebo such that no message queueing occurs. To guarantee that the reproduction process can be repeated, every relevant topic (states, commands, predictions) is recorded systematically on the ROS server together with persistent parameters (model, horizons T, H, frequency Δt, normalizations). The architecture was designed in the catkin structure as a workspace with modular packages and corresponding launch files and YAML configuration files for controllers and model parameters: *trajectory_prediction_bringup*, *trajectory_prediction_nodes* and *trajectory_prediction_msgs*. The ROS computational graph and the interaction between the main nodes and topics are shown in Figure 4.

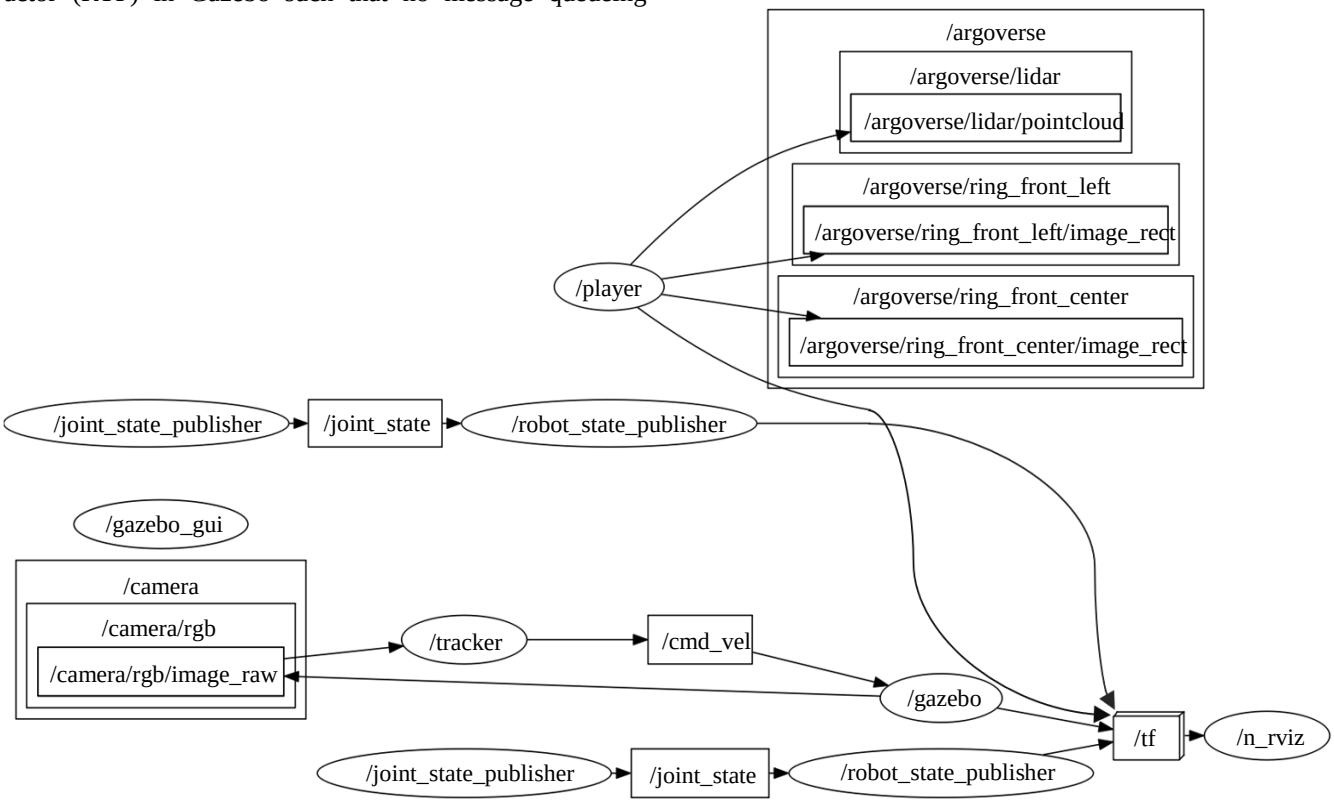


Fig. 4 ROS computational graph

4.3. Experimental Configuration

For testing and simulations, it was decided to test the system in two environments, urban and rural. The experimental scenarios included intersections, straight road segments, and curved lanes, allowing the model to be tested across a representative range of traffic situations. 15 hours of continuous simulation data were generated, and about 540,000 trajectory samples were collected with a sampling frequency of 10 Hz (0.1 s). The data set was divided at the scenario level, with 70% for training, 15% for validation and 15% for testing. The labels of the trajectories were obtained from the real vehicle state provided by the Gazebo topic /gazebo/model_states. Consequently, this approach produces reference trajectories free of control noise or collision artifacts. Further, the input-output structure is in a sliding window formulation in which each input sequence contains 2.0 s of historical movement (20-time steps), while the model predicts 3.0 s of future trajectory (30-time steps).Input variables are: Position (x, y), Orientation yaw, Longitudinal velocity v, and acceleration a (when available). All variables are expressed in the egocentric coordinate frame. Four prediction approaches were compared: the proposed CNN–LSTM hybrid model, a pure LSTM model used as an ablation baseline, the CV kinematic model, and the CTRV model. Inference was executed within a ROS node that loads the trained model exported in ONNX format. End-to-end latency was measured using ros::Time and the /clock simulation signal. The system targets inference latency below 10 ms per prediction on GPU hardware and below 50 ms on CPU, allowing stable operation at a publishing rate of 10 Hz.

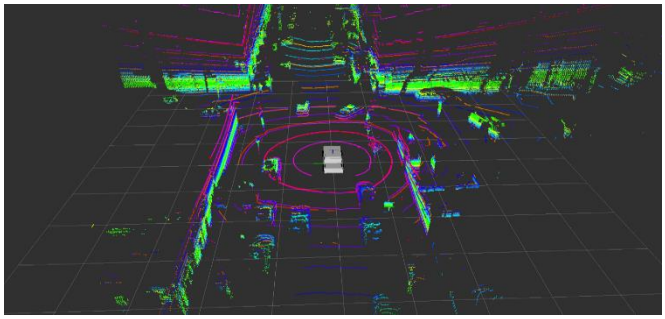
5. Results

5.1. Overall System Performance

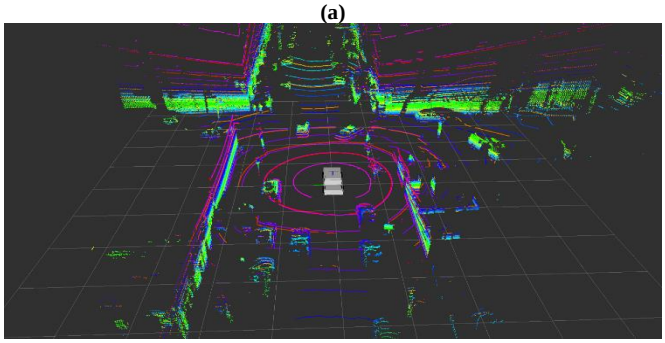
The proposed model using a CNN–LSTM architecture, evaluated on the test set, showed consistently better performance than the reference kinematic models, with a 3-second future time horizon. The model achieved an MSE average of 0.035 m², an ADE of 0.17 m (95% confidence interval: 0.16–0.18), and a FDE of 0.41 m (95% CI: 0.39–0.44). Spatial reliability of the predictions: 92.1% of the predicted points were within 0.5m of the ground truth trajectory, and 99.1% within 1.0m.

The MSE of the CTRV was 0.084 m2, the ADE was 0.29 m, and the FDE was 0.73 m, with 78.3% and 95.0% accuracy for the ADE and FDE, respectively. The CV model showed an MSE of 0.112 m², ADE of 0.38 m, and FDE of 0.95 m, with accuracies of 70.2% and 93.1%. The paired Wilcoxon test on trajectories ($p < 0.001$) supports the significance of the improvement provided by the CNN–LSTM approach over the baselines.

The results indicate that the temporal convolutions and sequential memory are able to accurately represent the temporal dependencies and local curvatures of the trajectory, which is not possible with traditional kinematic models. Examples of qualitative trajectories from the Argoverse Motion Forecasting Dataset are shown in Figures 5(a) and 5(b). These demonstrate how the model behaves in real urban driving scenarios. Figures 6(a) to 6(f) show a predicted robot trajectory generated by the proposed system.

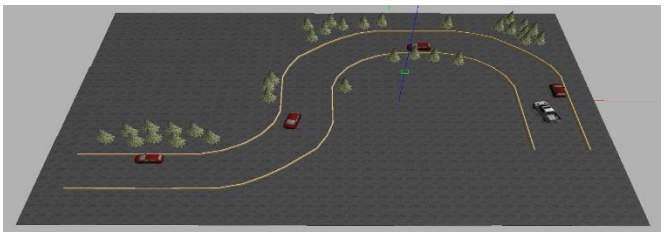


(a)

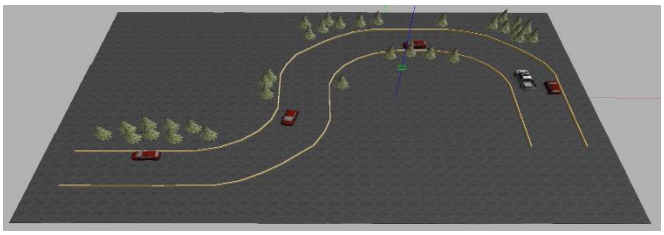


(b)

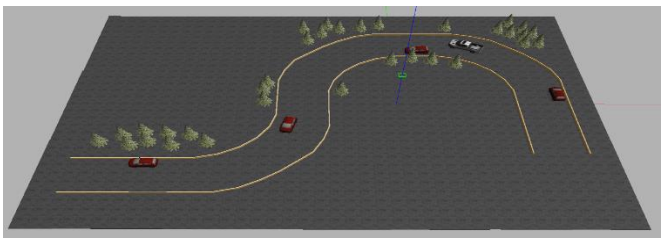
Fig. 5 Argoverse dataset scenarios, (a) Initial simulation stage, and (b) Final simulation stage.



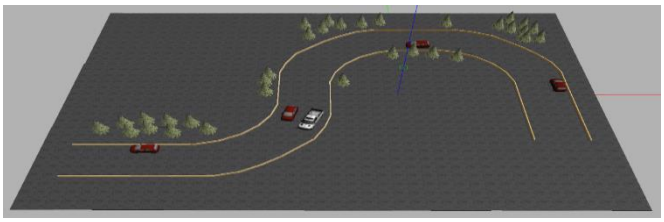
(a)



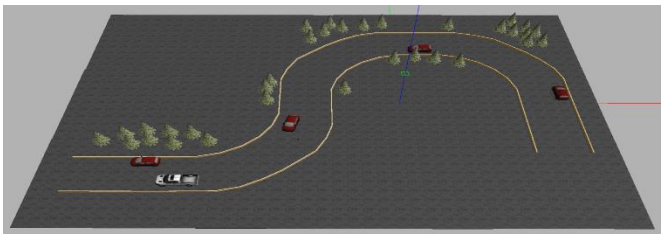
(b)



(c)



(d)



(e)

Fig. 6 Predicted robot trajectory at different stages is as follows, (a) Initial stage, (b) Early trajectory development, (c) Intermediate trajectory evolution, (d) Mid-to-late trajectory stage, (e) Final segment approaching the target path.

In most cases, the model-predicted trajectories visually overlap the actual reference along straight sections, reflecting high prediction fidelity. The model tends to slightly underestimate the radius of curvature in tight curves and roundabouts; however, the error converges rapidly. This occurs during the first 0.7 seconds of the prediction horizon. The dynamics suggest that the model can correct small initial discrepancies early on, which helps maintain the stability and coherence of the prediction sequence.

5.2. Ablation and Sensitivity

Ablation and sensitivity experiments helped to elucidate the contribution of important model parts and configurations. With the removal of the temporal convolution block (CNN) and only pure LSTM, the ADE went up to 0.21 m, the FDE to 0.52 m and the accuracy under a 0.5 m threshold dropped to 87.4%. This finding validates the use of temporal convolutions as a means to capture important information about local structure, such as micromaneuvers and implicit derivatives, to augment the representation prior to sequential aggregation. Further, the ADE was increased by 0.20 m when the acceleration was not provided as an input variable, which corresponds to an increase of 0.018 m with respect to the full model, indicating that the use of acceleration as an input variable aids in predicting changes in velocity and hence in the overall consistency of the prediction.

As for the historical window length, it was noticed that a shorter window length (1 s or 10 steps) results in a higher ADE (0.22 m) and a longer window length (3 s or 30 steps) results in a smaller ADE (0.16 m). The latter has a marginal gain over the 2 s configuration, but the cost of computation is 18% higher; thus, the 2 s window was considered to be a good compromise between accuracy and latency. Finally, it was discovered that normalization by scenario (rather than by map) reduces the ADE by about 6%, indicating that local statistics help to offset such biases between maps and enhance generalization of the model within scenarios. The quantitative values were obtained for each condition and are summarized in Table 2. The results of this study enable one to unambiguously identify the contribution of each component and each design choice; hence, it offers valuable guidance for the development of efficient and powerful trajectory predictors.

5.3. Robustness Analysis

A robustness analysis was conducted by introducing controlled disturbances into the input signals and operating conditions. The model demonstrated stable performance in the presence of sensor noise, sample loss and moderate latency. Increasing the noise level in the IMU and GPS signals by a factor of 1.5 resulted in an increase in the ADE to 0.18 m, with 89.7% of the predictions remaining below the 0.5 m error threshold. When 10% of the samples were randomly removed, the ADE increased to 0.19 metres, and the percentage of predictions below the threshold decreased to 88.9%. Finally, introducing an artificial latency of 50 ms into the odometry data resulted in an ADE of 0.20 m, with 88.5% of predictions remaining below the 0.5 m threshold. Linear temporal interpolation was then applied to compensate for the introduced delay (see Figure 7). Overall, these results suggest that the sliding-window scheme and the implemented preprocessing mechanisms help to maintain stable system performance under the evaluated disturbance conditions.

Table 2. Ablation and sensitivity analysis

Configu-ration evaluated	ADE [m]	FDE [m]	Precision <0.5 m [%]	Main observations
Complete model	0.17	0.41	92.1	Reference
Pure LSTM	0.21	0.52	87.4	Lose local patterns
No acceleration	0.20	0.49	89.3	Less anticipation of speed changes
Window 1 s (10 steps)	0.22	0.55	86.7	Insufficient context
Window 3 s (30 steps)	0.16	0.39	93.0	High precision, higher computational cost
Normali-zation by scenario	0.16	0.40	93.4	≈6% improve-ment in ADE compared to global normalization

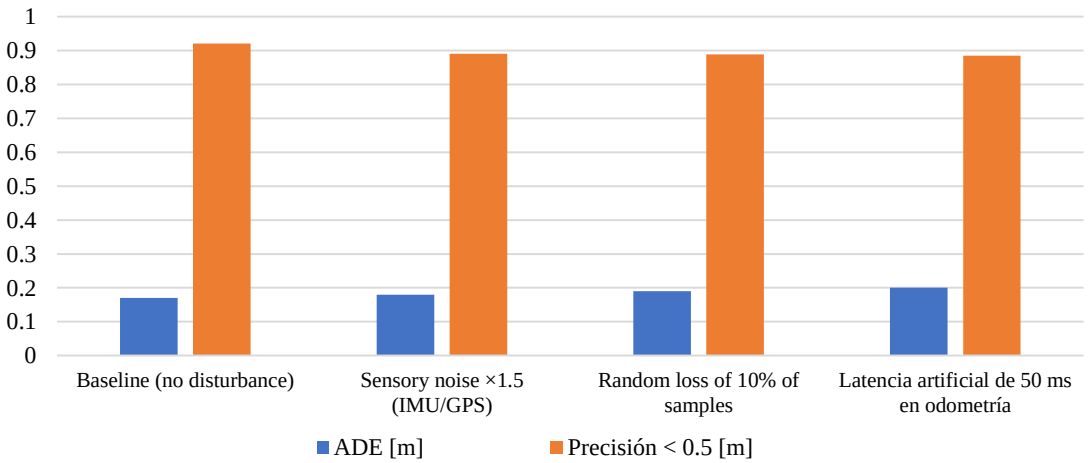


Fig. 7 Impact of perturbations on CNN–LSTM model performance

5.4. Real-Time Performance

Another test undertaken involved real-time operation within the ROS environment, for which the computational performance of the inference system was evaluated. The learned model was exported to ONNX format and performed inside a specialized ROS prediction node. Stable operation at a publication frequency of 10 Hz was made possible by the average inference delay on GPU hardware remaining below 10 ms per prediction and the CPU execution-maintained latency below 50 ms. These results confirm that the proposed system satisfies real-time constraints for integration in autonomous driving simulation pipelines. Table 3 summarizes the computational performance of the inference system.

Table 3. Computational performance of the inference system

Configuration	Average latency per inference [ms]	Equivalent frequency [Hz]	Observations
CPU (Intel i7-9700, sin GPU)	3.8	~260	Power consumption ≈12% per node; supports 10–20 Hz publishing.
GPU (RTX 2060, ONNX Runtime CUDA)	1.1	~900	Permite horizontes mayores o mayor frecuencia de publicación.
Rosbag reproduction (1× y 2×)	-	-	No missed deadlines; maintains RTF = 1.0 in Gazebo with active sensors; computational headroom.

6. Discussions

The hybrid CNN-LSTM network has proven effective in automated driving applications, particularly in the prediction pipeline, where it has demonstrated substantial improvements in error measures (ADE and FDE) over both baseline kinematic models and monolithic LSTM-based models. Ablation experiments verified that augmenting temporal convolutions achieves a relative improvement of around 19% in ADE, demonstrating the importance of considering local patterns before sequential integration with recurrent memories.

From a practical point of view, integrating the model into ROS enabled it to be deployed as a near-real-time inference node with less than 5 ms latency on a standard CPU, operating at frequencies of 50–100 Hz, without the need for specialised hardware. This validates the feasibility of the approach in perception and prediction pipelines in simulated Gazebo settings at least, and confirms its viability as part of autonomous navigation architectures.

Scenario analysis showed greater errors when the system was trained in urban situations involving many abrupt manoeuvres (yields, turns and stops), and irregular trajectories. In rural conditions, where these are less frequent, the system generalises better. This indicates potential for making the system more robust in complex environments by using semantic context cues, such as traffic light state or lane geometry. Furthermore, the operational interpretation of the metrics used (MSE, ADE, FDE and accuracy below thresholds) is clear: exceeding 90% accuracy at a 0.5 m threshold and a 3 s horizon is consistent with the requirements for short-term prediction and local planning, including safety margins in the planner.

Regarding durability, the model could tolerate sensory noise and sampling irregularities - situations that are typical of ROS when high sensor loads are applied or during data replay. The temporal interpolation and scenario normalization approaches worked well to keep the performance stable. However, threats to validity exist: (i) the results are restricted to standardized conditions, in Gazebo, where there is no adverse weather or heavy traffic; (ii) the model was trained only with ego states, not explicitly with interaction with nearby agents; and (iii) although confidence intervals and nonparametric tests have been used, the number of scenarios could be increased for greater statistical diversity.

7. Conclusion

A proposed system for predicting trajectories in automated driving is presented in this document, using convolutional neural networks (LSTM) for trajectory prediction and spatial context analysis, combined by late fusion and recurrent decoding. This design was evaluated in simulated environments using Mean Squared Error (MSE) and accuracy metrics. This provides future researchers with reproducible, efficient information and techniques for anticipating short- and medium-term vehicle movement. Furthermore, ablation studies enabled the contribution of each component to be attributed, demonstrating that the proposed system outperforms other kinematic methods and LSTM architectures that do not utilise spatial context. Based on the results obtained, it can be concluded that further steps towards more complex and dynamic scenarios of automatic handling are feasible.

However, the group admits that some things still need to be improved. Before applying the model to real cars, further work is needed to increase its accuracy due to the limitations

of the simulator, as noted in this initial research, which included a lack of adverse circumstances and a lack of closed-loop validation. Therefore, the following work will focus on improving the model by incorporating spatio-temporal attention processes, random uncertainty, and validating it in simulated dynamic environments.

Acknowledgments

The authors wish to express their sincere gratitude to the National University of San Agustín de Arequipa for providing the knowledge that made this work possible.

References

[1] Ming-Fang Chang et al., "Argoverse: 3D Tracking and Forecasting with Rich Maps," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, pp. 8748-8757, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Matteo Lisotto, Pasquale Coscia, and Lamberto Ballan, "Social and Scene-Aware Trajectory Prediction in Crowded Spaces," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, Korea (South), pp. 2567-2574, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jing-Tai Chang, and Feng-Cheng Lin, "An Enhanced Time-Attention-Based Social GAN for Pedestrian Trajectory Prediction and Risk Alert System Utilizing Edge Computing Device," *2025 Seventh International Symposium on Computer, Consumer and Control (IS3C)*, Taichung, Taiwan, pp. 1-4, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jianwei Tang et al., "Stochastic Human Motion Prediction with Memory of Action Transition and Action Characteristic," *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, pp. 1883-1893, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Tim Salzmann et al., "Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data," *16th European Conference Computer Vision*, Glasgow, UK, pp. 683-700, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[6] Sidra Rashid et al., "Optimizing Multi-Layer LSTM Based on Non-Local Spatiotemporal Interactions for Vehicle Trajectory Prediction," *IEEE Access*, vol. 13, pp. 147690-147702, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[7] Alexey Dosovitskiy et al., "CARLA: An Open Urban Driving Simulator," *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78, pp. 1-16, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yongchao Zhang, and Pengzhan Chen, "Path Planning of a Mobile Robot for a Dynamic Indoor Environment Based on an SAC-LSTM Algorithm," *Sensors*, vol. 23, no. 24, pp. 1-28, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale, "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst," *arXiv preprint*, pp. 1-20, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[10] Balakrishnan Varadarajan et al., "MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction," *2022 International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA, pp. 7814-7821, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Anis Koubaa, *Robot Operating System (ROS): The Complete Reference*, Springer International Publishing, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[12] Morgan Quigley, Brian Gerkey, and William D. Smart, *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*, O'Reilly Media, pp. 1-448, 2015. [Google Scholar] [Publisher Link]

[13] Alexandre Alahi et al., "Social LSTM: Human Trajectory Prediction in Crowded Spaces," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 961-971, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[14] Agrim Gupta et al., "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 2255-2264, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[15] Nachiket Deo, and Mohan M. Trivedi, "Convolutional Social Pooling for Vehicle Trajectory Prediction," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, USA, pp. 1549-15498, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[16] Yuning Chai et al., "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction," *arXiv preprint*, pp. 1-14, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] Jiyang Gao et al., "VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 11522-11530, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Tung Phan-Minh et al., "CoverNet: Multimodal Behavior Prediction Using Trajectory Sets," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle*, WA, USA, pp. 14062-14071, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[19] Karttikeya Mangalam et al., "It Is Not the Journey but the Destination: Endpoint Conditioned Trajectory Prediction," *16th European Conference Computer Vision*, Glasgow, UK, pp. 759-776, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[20] Ye Yuan et al., "AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada, pp. 9793-9803, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Benjamin Coifman, and Lizhe Li, "A Critical Evaluation of the Next Generation Simulation (NGSIM) Vehicle Trajectory Dataset," *Transportation Research Part B: Methodological*, vol. 105, pp. 362-377, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[22] Hao Cheng et al., "GATraj: A Graph- and Attention-Based Multi-Agent Trajectory Prediction Model," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 205, pp. 163-175, 2023. [CrossRef] [Google Scholar] [Publisher Link]