Machine Learning-based Trajectory Prediction for VRU Collision Avoidance in V2X Environments

Raúl Parada[1], Anton Aguilar[1], Jesus Alonso-Zarate[2] and Francisco Vázquez-Gallego[1]
[1]Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Av. Carl Friedrich Gauss 7, 08860, Castelldefels, Barcelona, Spain
[2]i2CAT Foundation, Barcelona, Spain
{raul.parada, anton.aguilar, francisco.vazquez}@cttc.es, jesus.alonso@i2cat.net

***Abstract*—The fifth generation (5G) of communication networks aims to accelerate the adoption of incipient vertical industries which will leverage innovative smart applications and services such as Cooperative, Connected and Automated Mobility. One of the objectives globally within that area is reducing to zero the number of fatal vehicle accidents. Unfortunately, human errors are the main cause of them, where vulnerable road users (VRUs) are involved in half of the cases. A possible approach to reduce accidents is estimating the probability of collision between two vehicles based on their estimated trajectories. These trajectories are usually tracked on-board using sophisticated devices such as cameras and LiDAR. However, VRUs are generally not equipped with such equipment and, ideally, VRUs carry smartphones with active geolocation capabilities based on satellite-based positioning systems. In this paper, we propose a novel vehicular service based on a regression algorithm to predict trajectories by uniquely using Cartesian coordinates. We compare different types of regression techniques in terms of prediction time window, position accuracy and processing time using Weka. Results show that the Alternating Model Tree (AMT) technique can predict the next position with an error of less than 3.2 centimeters, increasing up to 1 meter when predicting the next 5 positions with a period of 1 second between consecutive positions. In this case, a prediction time window of 5 s is processed within 1.25 milliseconds. AMT resulted as the lowest complex and most accurate algorithm in a multiple-step prediction position.**

***Index Terms*—ITS, Collision Avoidance, Trajectory Prediction, VRU**

I. Introduction

5G mobile communication networks, and beyond, will provide ultra-low latency and ultra-high reliability in high mobility and densely connected scenarios. The automotive sector will benefit from a communications network capable of providing such features. The Cooperative, Connected and Automated Mobility (CCAM) paradigm envisions connected vehicles with enhanced context-awareness capabilities thanks to the integration of hundreds of sensors and the exchange of information among mobility actors. The use of vehicle-to-everything (V2X) communication allows vehicles to exchange information with other vehicles (V2V), with the roadside infrastructure (V2I), with pedestrians (V2P), and with communication networks (V2N).

The implementation of advanced connected vehicular services will place stringent requirements on the underlying communication system, which must ensure that information is delivered securely and on time. In this view, Mobile Edge Computing (MEC) becomes a critical component of V2X communication [1]. This is because MEC places processing, storage, and networking capacity closer to the network edge (i.e., closer to the end user), thus making low-latency and high-bandwidth requirements easier to meet. Furthermore, MEC servers can be easily installed and integrated within the radio access network, deployed between the core network and cellular base stations.

The envisioned CCAM services which can benefit from 5G connectivity and MEC technology aim to reduce the environmental impact of transportation by increasing traffic efficiency, improve passenger experience, and reduce the number of fatalities on the road. According to [2], more than 1.35 % of people die yearly in urban areas due to on-road accidents; half of them involve vulnerable road users (VRUs) such as pedestrians, cyclists, and motorcyclists. Most of these traffic accidents are due to a late collision detection by the driver [3]. In this scenario, the worldwide growing adoption of personal mobility vehicles (PMVs) in urban environments, such as electric standing scooters, electric skateboards, and self-balancing unicycles, might further increase the number of accidents where VRUs are involved. For these reasons, recent research works have investigated the potential of using geolocation-based applications on smartphones to protect VRUs using trajectory prediction techniques for collision avoidance. In [4], the authors propose a dynamic time warping approach to process hundreds of geospatial positions in a few milliseconds to estimate only a time window of 1 second. The work in [5] analyses the influence of positioning and prediction inaccuracies of VRUs in collision avoidance systems. Results show that for vehicle velocities of 50 km/h, the predicted VRU position should have a standard deviation of less than 55 cm to minimize the risk of accident. Another key factor for collision avoidance is the time window in which future positions can be predicted: in general, the longer the duration of the time window, the sooner a possible collision can be predicted, and the higher the probability of avoiding it. In [6], a smartphone application based on neural network techniques was able to predict pedestrian trajectories in a time window of 2 seconds with an accuracy of 1.5 m in average, which is worse than the 55 cm requirement defined in [5]. In addition, the average reaction time of a human driver is 2.5 seconds [7] and, we should add the breaking time, which depends on multiple factors such as speed, road condition, wheels condition, etc. Hence, we expect a prediction time window of 5 seconds, for secure VRU environment.

Motivated by this context, this work aims to improve the performance results presented in the literature by using multiple state-of-the-art regression techniques for trajectory prediction. Within these techniques, recurrent neural networks have been explored in the literature [8] for trajectory prediction with position error accuracy of 3.34 meters in a time window prediction of 5 seconds. The main contributions of this paper are:

- We propose an innovative vehicular service based on MEC and V2X communications to predict trajectories and estimate the probability of collision between vehicles and VRUs.
- We evaluate and compare the performance of multiple regression algorithms for trajectory prediction in terms of the prediction time window, position accuracy and processing time.
- We implement the algorithm with the best performance into a low-cost embedded computer and measure the processing time needed to predict trajectories.

The remainder of this paper is organized as follows. Section II introduces the problem motivation and related work. The system architecture of the proposed vehicular service is described in Section III. The selected regression algorithms are explained in Section IV. Section V is devoted to evaluating and comparing the performance of the algorithms considered. Once the most efficient one is identified, its prediction processing time is measured on a commercial off-the-shelf embedded computer. Finally, we conclude the paper in Section VI.

II. Related Work

To date, several authors have worked on collision avoidance techniques for the protection of VRUs by using context-aware devices, such as cameras, LiDAR, and radio-frequency sensors [9], combined with deep learning techniques. For example, Pham et al. [10] proposed a camera-based aid system for PMVs to reduce the invasion of pedestrian spaces as well as the crossing between PMVs and pedestrian flows. Muhammad et al. [11] proposed a vision-based rear-end collision detection system in which a camera is mounted on a motorcycle to avoid collisions by recognizing objects. Although the proposed system can provide reliable results on rear-end vehicle detections, this solution requires high-definition cameras to classify road elements. Kusakari et al. [12] proposed a deep neural network model and computer vision on-board a mobility scooter for pedestrian detection with collision danger estimation. All these works proposed different techniques for the protection and safety of VRUs, however, none of them exploited trajectory prediction capabilities.

In the context of pedestrian trajectory prediction based on computer vision, Raman et al. [13] used a camera to predict the movement of pedestrians based on analysis of dimensional changes in video frames. A Hidden Markov Model was used to classify up to 8 possible directions of movement. Matsunaga et al. [14] proposed a virtual reality assistant to ease collision avoidance using hololens with an integrated 3D-map of the environment. By 3D mapping the environment, the system can detect static obstacles and walls, and it displays on the hololens a safe trajectory where the driver can move avoiding

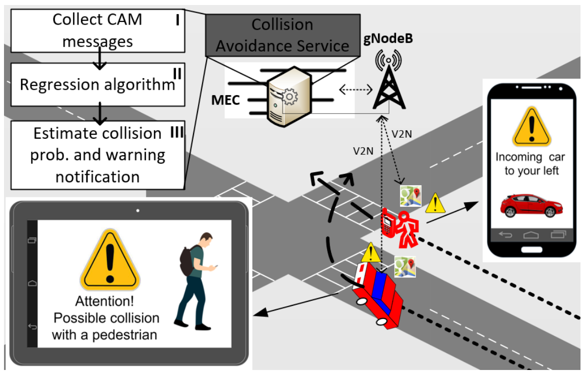
Collect CAM messages I
Regression algorithm II
Estimate collision prob. and warning notification III
Collision Avoidance Service
MEC
gNodeB
V2N
V2N
Incoming car to your left
Attention! Possible collision with a pedestrian

Fig. 1: Collision avoidance system overview

collisions. Asahara et al. [15] proposed a system for the prediction of pedestrians' movements based on the acquisition of geospatial positions and further processing with mixed Markov-chain models. The pedestrians are equipped with a personal digital assistant, a GPS receiver integrated, and their real-time positions are fed into the Markov-chain models to predict their next position. The system was used for human behavioral studies. The precision rate to predict the next position reaches the 64 % by combining the trajectories from almost 700 participants for an average period of 90 minutes.

In this paper, we focus on the prediction of vehicles and VRUs' trajectories by processing geospatial positions with low-complexity regression algorithms [6]. Regression techniques are useful to anticipate a specific event by forecasting future steps based on historical data. As mentioned in Section I, regression techniques have been used to predict trajectories. However, to the best of our knowledge, this manuscript contributes to using regression algorithms to predict two non-correlated attributes instead of one (e.g., weather forecast).

III. System Architecture

The system architecture proposed in this work is shown in Figure 1, where two vehicles communicate via V2X. We consider two types of road users: vehicles (e.g., cars, vans and buses) and VRUs (e.g., scooters, bicycles and pedestrians). A 5G cellular base station (gNodeB) is connected to one MEC server with storage and computing resources. Both vehicles and VRUs periodically transmit Cooperative Awareness Messages (CAMs) via 5G to the MEC server using an on-board unit (OBU) and a smartphone, respectively. In the MEC server there is an application, named Collision Avoidance service (CA), which collects the positions of each vehicle and VRU. These positions are fed into a regression algorithm that predicts the trajectories within a predefined time window. Finally, the Collision Avoidance service estimates the probability of collision between vehicles and VRUs.

Figure 1 shows a typical situation where a car is just starting a turning maneuver while a pedestrian accelerates to cross the road. Both vehicle and pedestrian transmit CAM messages to the MEC, through 5G.

- **I-Collect CAM messages:** This function collects the CAM messages sent by the vehicles and VRUs, via V2N, to the MEC.

- **II-Regression algorithm:** The MEC receives CAMs from the vehicles with geospatial information. This information is fed into regression algorithms to forecast trajectories, i.e., predicted Cartesian coordinates. This function is key to predict an ahead position accurately. In Figure 1 we can observe the road users' historical path used to feed regression algorithms and an expected *n* number of predicted positions, represented as a dashed and a long-dashed black line, respectively.
- **III-Estimate collision probability and warning notification:** Finally, from step **II**, we obtain a predicted trajectory from all vehicles and VRUs. By analyzing if pairs of predicted trajectories overlap in each instant of time, we can detect a collision. Finally, if the probability of collision reaches a given threshold, a warning notification is sent to both vehicle and VRU.

Finally, the vehicle shows the warning in its human machine interface and the VRU is alerted through its smartphone. This solution can be adapted easily to any kind of vehicle. As presented in step **II**, the key function is the trajectory prediction by using machine learning techniques.

IV. Regression algorithms

This section briefly presents the selected algorithms to predict the trajectory of vehicles and VRUs performing common actions in urban environments.

Once we know the algorithms to process our geolocation data, we must implement them. Fortunately, Weka includes all the selected algorithms. Weka is an open-source software provides tools for data preprocessing and implementation of several machine learning algorithms. In addition, its intuitive graphical user interface and flexibility allows a fast algorithm tuning and comparison. Hereby, we will perform the simulations using Weka. We used the following algorithms:

- **Alternating Model Tree (AMT):** In this method, a collection of linear regression models is used to estimate the desired variables for different conditions. The models are organized in a decision tree, where the leaf nodes contain the models while the rest are decision nodes used to decide which one should be used. Forward stage-wise additive modeling is proposed in the literature to construct the tree [16].
- **M5 Rules (M5R):** This one is also a decision tree model with regression models in the leave nodes where a separate-and-conquer paradigm is used to define the tree automatically. In this technique, the standard deviation of the classification error is used to decide which feature will be used to generate the next node and further split the data. The result is a decision tree where the error is minimized.
- **Least Median Square (LMS):** LMS is a variation of traditional regression (i.e., least mean squares) which is more robust against the effect of outliers. Least squared regression functions are generated from random subsamples of the data. The least squared regression with the lowest median squared error is chosen as the final model.
- **MultiLayer Perceptron (MLP):** These are traditional Neural Networks, where layers of artificial neurons are combined to represent complex non-linear functions, like the ones found in classification problems. Gradient-descent and back-propagation algorithms are the usual methods to find the optimal set of parameters during training.
- **Long Short-Term Memory (LSTM):** This is a special type of recurrent neural network where the outputs of the network are as inputs, allowing the network to remember past states and solve sequential problems. LSTM was proposed as an alternative to simple recurrent networks to solve the vanishing gradient problem, where the error signal used for gradient-descent to update the network's weights degraded with time.

Algorithms' performances are usually categorized according to their computational complexity using the big-O notation. Table I presents the above algorithm type and computational complexity. A preliminary analysis indicates the Alternating

TABLE I: Complexity comparison

Method	Type	Complexity
AMT	Ensemble learning	$\mathcal{O}(1)$
LMS	Linear regression	$\mathcal{O}(n^2)$
LSTM	Recurrent Neural Network	$\mathcal{O}(W)$
M5R	Decision Rules	$\mathcal{O}(n^2)$
MLP	Artificial Neural Network	$\mathcal{O}(n^2)$
SMOR	Binary classification	$\mathcal{O}(n - n^2)$

Model Tree over the rest with a O(1) analysis. The next section will show the results of the above algorithms by feeding them with real geospatial data for trajectory prediction.

V. Performance evaluation

In this section, we evaluate the performance of the regression models explained in Section IV for trajectory prediction. Firstly, we describe the methodology followed to train and test the regression algorithms by using data sets based on real-trajectories data. Secondly, we compute the performance of each regression model in terms of position accuracy as a function of the data set length. Then, we analyze the computational complexity of the regression models and select the one that provides the lowest processing time. Finally, we fine-tune the best model to maximize the prediction position accuracy and to minimize the processing time, and we evaluate its performance in a low-cost embedded computer.

A. Models Training and Testing Methodology

A road user, such as a vehicle or VRU, draws a trajectory that can be geospatially represented with a sequence of geospatial samples. Each sample is composed of latitude and longitude coordinates. We have used an open-source Android application, named Ultra GPS Logger, to acquire samples of real trajectories of vehicles and VRUs in the city of Barcelona. The Ultra GPS Logger was executed on an Android smartphone to acquire one position every one second from its integrated GPS receiver. These real trajectories were employed as data sets to train a model from the selected regression algorithms described in Section IV, by using Weka. The data sets were injected into each one of the regression algorithms to train the model in real time and predict future samples within a predefined prediction time window. Figure 2 shows a map with an example of a 15-samples data set, where each sample is depicted as a white-paddle mark. As can be seen, the data set consists in a turn-left maneuver which may result complex for trajectory prediction.

Since the regression models that we have considered are supervised, they require a training phase to provide a prediction of future geospatial samples. The length of the data set used for training might influence the performance of the prediction and the time to train the model. The larger the training data set, the higher will be the prediction accuracy and, so, higher the time to train the model. Hence, data set length for model training will be explored. As an example, Figure 2 shows the samples predicted by each model within a time window of 10 seconds (i.e., 10 samples) and using a 15-samples data set for training. For each regression model, we used the configuration parameters that yield the lowest prediction error in meters. Those parameters were obtained by a hyper parameter optimization process in each model.

As can be observed in Figure 2, the trajectory predicted by the AMT model (in blue-square samples) follows a similar trend to the real trajectory (in white-star paddle samples). The trajectory predicted the LMS model (in red-circle samples) clearly separates from the real trajectory; it tends to the corner and crashes against the building. The trajectories predicted by MLP (in green-diamond samples), SMOR (in yellow-triangle samples) and M5R (in black-hexagon samples) clearly draw a straight-line trajectory and do not predict the left-turn action. Finally, the LSTM is clearly not able to predict future samples due to its inefficiency on detecting mobility. This fact might be due to a short-training data set. It is important to highlight that the trajectory presented in Figure 2 is a challenging situation where the training data set lacks left-turning positions.

B. Prediction Accuracy over the Data set Length

The aim of this subsection is to analyze how the data set length affects the prediction error. The prediction error at time t_p is defined as the lineal distance between a predicted position at time t_p and the correct position (in the real trajectory) at the same time instant.

Table II shows the prediction errors obtained for each regression model, within a time window of 5 seconds (i.e., 5 samples predicted), and considering a data set length of 15, 30, 60 and 120 samples. As can be observed, the larger the data set length, the worse the prediction error for all the regression models. This is because of the non-periodicity of the geospatial data along the time and the non-correlation factor between latitude and longitude. Furthermore, the neural network-based models (LSTM and MLP) do not provide better results by increasing the length. That fact lies in the vulnerability of those techniques when new data is added, against a classification problem where all data are similar (e.g., classify cats and dogs). Again, we can observe how the AMT model provides the most accurate prediction with respect to the positions of the real trajectory.

Results on Table II show that prediction errors for dataset lengths of 30, 60 and 120 samples are not acceptable for a collision avoidance service. The prediction error is larger than 7 m for AMT, larger than 19 m for LMS, and larger than 44 m for LSTM, M5R, MLP and SMOR. Thus, we decided to further analyze the prediction error for shorter lengths in the range between 10 and 20 samples. The six plots in Figure 3 show the prediction errors (in the y-axis) against the length from 10 to 20 samples (in the x-axis). Each plot is for one regression model and shows five different curves that represent the prediction error of every sample predicted within a time window of 5 seconds (i.e., 5 samples). The prediction error for each sample is represented with a symbol: Circle, square, star, diamond and hexagon, respectively, for samples 1 to 5. As can be observed in Figure 3, the AMT model provides the lowest prediction error (lower than 10 m) among all the regression models. In addition, except for LSTM, all the regression models have a minimum prediction error when using a data set length of 14 and 15.

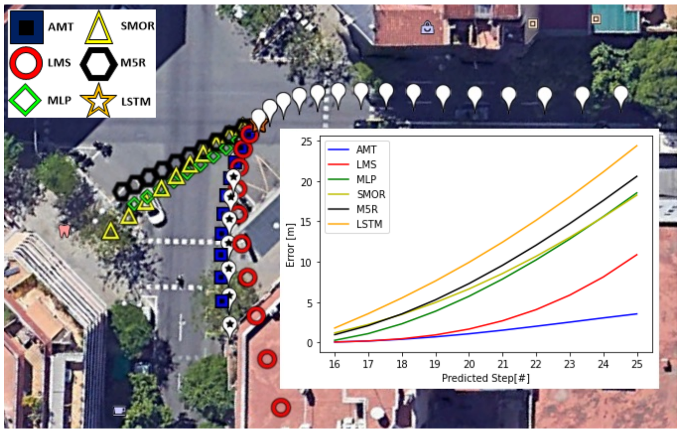

Fig. 2: Prediction accuracy illustration

C. Computational Complexity over the Data set Length

The aim of this subsection is to evaluate the computational complexity of the regression models by measuring the processing time required by each model to predict a trajectory with a prediction time window of 5 seconds. To this end, we executed the regression models 100,000 times on a Monte Carlo basis for different data set lengths, which grow exponentially from 15 to 480 samples. The computations were performed on a laptop Intel(R) Core(TM) i7-10610U CPU at 1.80 GHz - 2.30 GHz with 16 GB of RAM. Figure 4 shows the average processing time for each model as a function of the data set length. As can be observed in Figure 4, the curves match with the expected big-O notation presented in Table I. As could be expected, the average processing time of the AMT model maintains a horizontal trend, i.e., O(1), and its value is below 10 ms to predict 5 samples.

D. AMT Model Optimization

From the results presented before, we concluded that the AMT model is the one that provides the lowest prediction

TABLE II: Prediction error for larger data set lengths

	15					30					60					120				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
AMT	0.048	0.173	0.389	0.686	1.064	7.836	14.116	19.099	22.968	25.864	12.998	23.185	31.150	37.344	42.121	31.907	56.983	76.778	92.442	104.889
LMS	0.043	0.174	0.446	0.921	1.648	19.006	19.724	24.769	28.995	26.545	29.315	30.065	38.239	45.731	43.352	67.368	71.244	91.885	111.366	108.901
LSTM	1.776	3.559	5.506	7.613	9.926	45.498	48.903	52.154	55.252	58.135	94.350	96.317	98.292	100.262	102.201	229.067	231.780	234.554	237.401	240.315
M5R	0.955	2.077	3.524	5.273	7.291	44.727	47.309	49.789	52.132	54.280	93.614	94.745	95.908	97.091	98.268	228.345	230.218	232.165	234.193	236.293
MLP	0.255	1.086	2.298	3.853	5.716	44.246	46.124	48.518	50.927	53.052	93.502	93.266	94.333	95.732	96.949	229.984	228.891	230.600	232.927	235.026
SMOR	1.173	2.218	3.466	4.936	6.623	44.106	46.625	49.389	52.177	54.671	91.539	92.397	94.139	96.461	98.728	223.138	224.086	227.068	231.677	236.477

error as well as the lowest computational complexity. Since previous results were obtained using the default configuration parameters of the AMT model available in Weka, the aim of this subsection is to find the optimal parameters of AMT that minimize the prediction error and reduce the processing time. The AMT model contains two main configuration parameters to train the model [16]: the iteration number and the shrinkage coefficient. The iteration number determines the number of runs and its value by default is 10. The shrinkage coefficient controls over fitting and its value by default is 1. Since the main concern is the prediction error, we tune the shrinkage

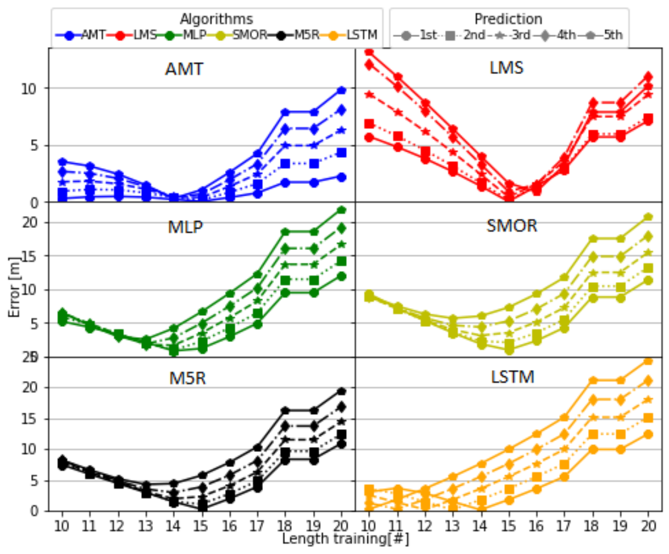

Fig. 3: Comparison of algorithms. Error prediction Vs. Data set length for training

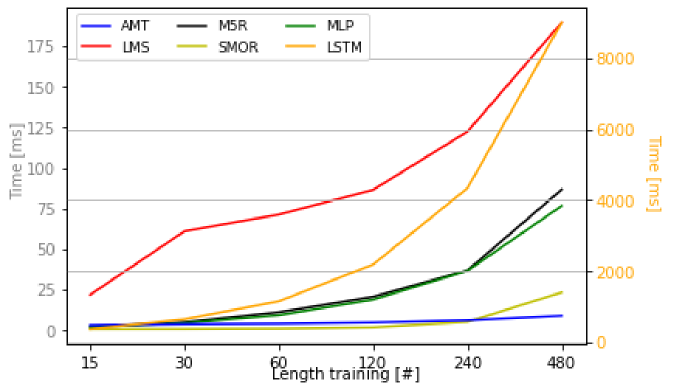

Fig. 4: Processing time Vs. multiple data set lengths

coefficient to find out its optimal value that minimizes the prediction error by using a data set length of 15 samples, which provides forecasting stability and low prediction error. Figure 5 shows the prediction error (in the x-axis), as a function of the shrinkage coefficient from 0.1 to 1.9 (in the y-axis), for each of the 5 predicted samples. As can be observed, when the shrinkage coefficient is very low, the prediction error is very high. As the shrinkage coefficient increases, the prediction error for all five predicted samples decreases, reaching a minimum when the value of the coefficient is 1.1. Then, the prediction error increases almost exponentially with unacceptable errors above 30 meters. Thus, we fix the value of shrinkage coefficient in 1.1, where the prediction errors in the first four predicted positions are below 56 cm, i.e., 1.9, 9.2, 24.8 and 55.7 cm. These results fit as expected in [5] where a predicted position of less than 55 cm minimizes the risk of accident in a VRU environment.

In this work, we want to reduce the processing time as much as possible, and at the same time, keep the prediction error very low. The next step is to reduce the processing time of the AMT model by reducing the number of iterations. Figure 6 shows the prediction error (in the y-axis) as a function of the number of iterations from 1 to 10 (in the x-axis). The blue curves are the results of the prediction error for each predicted position, while the grey line are the results of the average processing time after 100,000 K runs for each number of iterations used to train the model. As can be observed in Figure 6, one iteration is not enough to provide low prediction errors. However, using only 4 iterations, the prediction error drops considerably for all five predicted positions with values of 3.25, 12.26, 28.5, 60 and 108.6 cm. Indeed, these prediction errors are larger than using an iteration value of 10, nonetheless, the processing time decreases an 80.27 % (1.26 against 6.4 ms) with a penalization in accuracy of 68.39, 33.7, 14.92, 7.72 and 3.82 %, respectively.

Finally, we measured the processing time of the AMT model on a low-cost embedded computer widely used in many real implementations and proof-of-concepts. The embedded computer is a Raspberry Pi 2b Rev. 1.1 (1GB RAM) with 32bits-raspbian OS. The AMT model was executed 100,000 times on Weka resulting in an average CPU time of 5.5 ms. Summarizing, the AMT model can predict at least five future positions with a prediction error lower than 1 meter in only one millisecond. With respect to the previous research works, we have reduced the processing time by half the results presented in [4] by using a data set length of 15 samples for training. In addition, we have predicted more than 2 future positions,

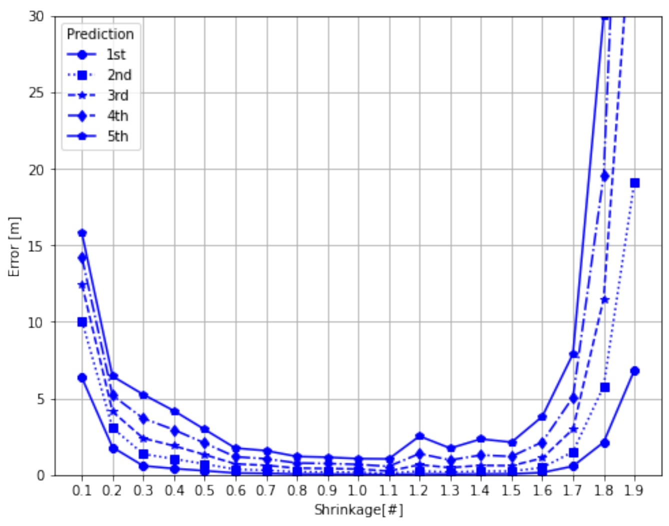

Fig. 5: Prediction error of AMT vs. the shrinkage parameter

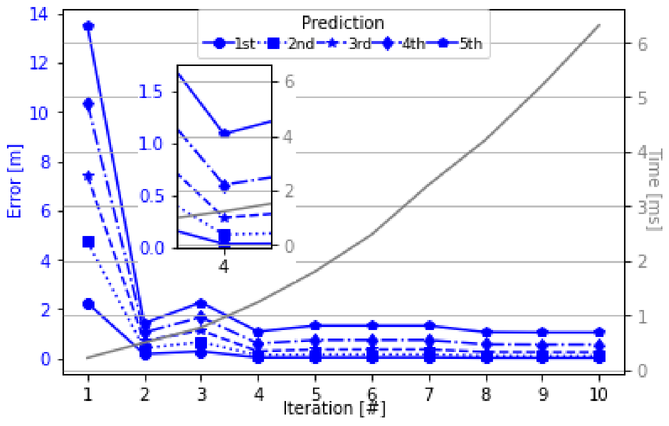

Fig. 6: Prediction error of AMT vs. the iteration parameter

improving by more than 90 % the prediction time window considered in [6], which fulfills the requirements proposed in [5] to protect VRUs.

VI. Conclusion

Indeed, it is possible to predict a window of future positions using state-of-the-art regression techniques by unique using Cartesian coordinates. Thanks to Weka, which includes hundreds of regression techniques, we could compare different ones easily. This comparison resulted in Alternating Model Tree outperforming in terms of the prediction time window, position accuracy and processing time. AMT returned a prediction time window of 4 seconds with a position accuracy below 55 cm with a processing time of only 1.26 ms (for a predicted time windows of 5 seconds). These results improve the requirements for a safer VRU environment. The future work includes the migration of the Alternating Model Tree to Python, the performance analysis of a larger amount of real-trajectory, the estimation of the collision probability between vehicular trajectories and the design of a multi-platform web-app to receive warning notification.

Acknowledgment

This work has been partially funded by the SPOT5G project (TEC2017-87456-P), the FEMIoT project, which is funded by the European Regional Development Fund (ERDF) and Generalitat de Catalunya under Grant 2017 SGR 891.

References

[1] F. Vázquez-Gallego, R. Vilalta, A. García, F. Mira, S. Vía, R. Muñoz, J. Alonso-Zarate, and M. Catalan-Cid, "Demo: A Mobile Edge Computing-based Collision Avoidance System for Future Vehicular Networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 904–905.

[2] World Health Organization. (2020) Road traffic injuries. (Accessed Apr. 2021) Available at: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[3] K. Rumar, "The basic driver error: late detection," *Ergonomics*, vol. 33, no. 10-11, p. 1281–1290, oct 1990, https://doi.org/10.1080/00140139008925332.

[4] E. Alemneh, S. Senouci, and P. Brunet, "An Online Time Warping based Map Matching for Vulnerable Road Users' Safety," in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2018, pp. 910–915.

[5] P. Themann, J. Kotte, D. Raudszus, and L. Eckstein, "Impact of positioning uncertainty of vulnerable road users on risk minimization in collision avoidance systems," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1201–1206.

[6] Y. L. Murphey, C. Liu, M. Tayyab, and D. Narayan, "Accurate pedestrian path prediction using neural networks," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–7.

[7] University of Idaho - Transportation Engineering. (2003) Brake Reaction Time. (Accessed Apr. 2021) Available at: https://www.webpages.uidaho.edu/niatt_labmanual/chapters/geometricdesign/theoryandconcepts/BrakeReactionTime.htm.

[8] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, and F. Nashashibi, "Relational recurrent neural networks for vehicle trajectory prediction," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1813–1818.

[9] Y. Huang and Y. Chen, "Survey of State-of-Art Autonomous Driving Technologies with Deep Learning," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2020, pp. 221–228.

[10] T. Q. Pham, C. Nakagawa, A. Shintani, and T. Ito, "Evaluation of the Effects of a Personal Mobility Vehicle on Multiple Pedestrians Using Personal Space," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2028–2037, 2015.

[11] M. Muzammel, M. Z. Yusoff, and F. Meriaudeau, "Rear-end vision-based collision detection system for motorcyclists," *Journal of Electronic Imaging*, vol. 26, no. 3, pp. 1 – 14, 2017. [Online]. Available: https://doi.org/10.1117/1.JEI.26.3.033002

[12] Y. Kusakari, W. H. Chin, and N. Kubota, "A Deep Neural Model for Pedestrians Detection with Danger Estimation," in *2020 International Symposium on Community-centric Systems (CcS)*, 2020, pp. 1–6.

[13] R. Raman, P. K. Sa, B. Majhi, and S. Bakshi, "Direction Estimation for Pedestrian Monitoring System in Smart Cities: An HMM Based Approach," *IEEE Access*, vol. 4, pp. 5788–5808, 2016.

[14] N. Matsunaga, R. Kimura, H. Ishiguro, and H. Okajima, "Driving Assistance of Welfare Vehicle with Virtual Platoon Control Method which has Collision Avoidance Function Using Mixed Reality," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 1915–1920.

[15] A. Asahara, K. Maruyama, A. Sato, and K. Seto, "Pedestrian-Movement Prediction Based on Mixed Markov-Chain Model," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 25–33. [Online]. Available: https://doi.org/10.1145/2093973.2093979

[16] E. Frank, M. Mayo, and S. Kramer, "Alternating model trees," Salamanca, Spain, pp. 871–878, Apr 2015, conference Contribution. [Online]. Available: https://hdl.handle.net/10289/9398