

Model-based and machine learning-based high-level controller for autonomous vehicle navigation: lane centering and obstacles avoidance

Marcone Ferreira Santos¹, Alessandro Corrêa Victorino², Hugo Pousseur²

¹Department of Mechanical Engineering, Federal University of Minas Gerais, Belo Horizonte, Brazil

²CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems), Université de Technologie de Compiègne, Compiègne, France

Article Info

Article history:

Received Sep 29, 2021

Revised Oct 3, 2022

Accepted Nov 18, 2022

Keywords:

Autonomous car

Autonomous navigation systems

Intelligent vehicles

ABSTRACT

Researchers have been attempting to make the car drive autonomously. Environment perception, together with safe guidance and control, is an important task and is one of the big challenges when developing this kind of system. Geometrical or physical-based models, machine learning-based models, and those based on a mixture of both models are the three types of navigation methods used to resolve this problem. The last method takes advantage of the learning capability of machine learning models and uses the safeness of geometric models in order to better perform the navigation task. This paper presents a hybrid autonomous navigation methodology, which takes advantage of the learning capability of machine learning and uses the safeness of the dynamic window approach geometric method. Using a single camera and a 2D lidar sensor, this method actuates as a high-level controller, where optimal vehicle velocities are found, then applied by a low-level controller. The final algorithm is validated in the CARLA Simulator environment, where the system proved to be capable to guide the vehicle in order to achieve the following tasks: lane keeping and obstacle avoidance.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Marcone Ferreira Santos

Department of Mechanical Engineering, Federal University of Minas Gerais

Belo Horizonte, Brazil Email: marconefs@ufmg.br

1. INTRODUCTION

The first car, named Navlab, coupled with computer vision and a smart steering system, emerged in the 1980s at Carnegie Mellon University [1]. Since then, several attempts have been made to make fully autonomous vehicles become safer, more efficient, and environmentally responsible. One of the most advanced smart embedded systems nowadays is found in [2], where a self-driving car has driven more than sixteen million kilometers autonomously.

The main tasks when developing an autonomous vehicle system are summarized as follows: environment perception, mapping and localization, motion planning, decision, and control. Through images captured by one or more cameras, lidar, and other useful sensors, the perception task is designed to detect and understand the local environment where the vehicle is driving. Some studies comprising the perception subject are found in [3], [5]. These tasks are carried out through modules presented in embedded smart systems equipped in an autonomous vehicle and are developed using scientific methodologies based on "model-based", "machine learning based" or "hybrid-based" control methods.

An optimal collision-free path, from the current vehicle state to the desired state, is generated by motion planning methods and can be divided into global planners or local planners, where local information is considered. These methods consist of candidate trajectories calculation based on the kinodynamic model and then selecting the best one considering the safeness and other relevant assumptions. Such methods were presented in some DARPA Urban Challenge team cars [6]. A classic reactive motion planning dynamic windows approach (DWA) proposed by [7], evaluates pair of velocities selection by an objective function optimization, considering a short time and discarding any velocities which generate a path with collision according to a minimum distance.

In recent years machine learning has gained space in the robot field, with many works in path planning and control. In [8] two neural networks are designed to find the free navigation space and the trajectory for a mobile robot. Reinforcement learning combined with the DWA path planning method is proposed by [9], where the Q-learning algorithm is used for DWA adaptive function weights adjustment for each evaluation task. Related works where machine learning is combined in robot path planning are shown in [10], [12].

Model-based navigation control methods are designed to control the vehicle based on the selected motion paths. Stanley method, first introduced in the DARPA Grand Challenge [13], model predictive control (MPC), fuzzy control and preview control [14] are some techniques still in the research field. Machine learning techniques also have been applied in robot control methodologies, where in [15] an end-to-end navigation control system based fully on machine learning is designed.

Geometric-based autonomous navigation systems are designed assuming certain conditions, getting computationally expensive. Navigation systems based on machine learning can improve performance, but on the other hand, wrong outputs may happen in unseen situations, leading the vehicle to undesired motions. Hybrid methodologies, combining geometric-based navigation models with machine learning can raise good results, taking the advantage of both methods.

This paper presents a hybrid controller methodology for lane centering with obstacle avoidance. This controller is inspired by the image-based dynamic window approach (IDWA) [16], where autonomous navigation is done through visual features and uses a modified DWA function to evaluate the reactive control. Using camera images, the convolutional neural network is trained in order to segment road lane lines. Visual features are extracted from these segmented images, and another neural network model is trained to predict pair of velocities to be applied on the vehicle. When this value leads the vehicle to collision, the reactive control is performed in order to find and select the best collision-free path according to the modified DWA optimization function called the image-based reduced dynamic window approach (IRDWA). A third machine learning model is used for the reduced dynamic-window functionality, which aims to reduce the optimum velocities of search space. Each aspect of the proposed system is presented in the next section.

This paper is structured as follows. Section 2 presents the general aspects of the proposed system, with a brief explanation of the high- and low-level controller. Then, in section 3 the High control block, which is the focus of this work, is explained in more detail. The proposed system is evaluated on simulation, where a car equipped with a camera and a single-layer lidar sensor must accomplish lane-keeping and obstacle avoidance tasks. All the results are presented in section 4 and a conclusion about this work is in section 5.

2. MODELING ASPECTS: GENERAL OVERVIEW OF THE SYSTEM

Two main coordinate frames represent the camera and LIDAR position and are shown in Figure 1. The former is located in the car's front roof (camera frame), while the latter is placed in front of the car (World frame). The relative displacement of the camera frame to the world frame is tz (in Z_W axis), ty (in Y_W axis), and its relative rotation in X_W axis is ϕ . The camera is pointed along the Z_C axis in order to capture images comprising the lanes road.

The general system can be divided into 3 main blocks as shown in Figure 2. The first block corresponds to all sensors and actuators equipped in the vehicle, providing current state information such as the linear velocity (V_t), yaw rate (W_t), camera image, and 2D point cloud provided by the single layer lidar sensor. Then, based on the kinematic bicycle model [17], the high-level controller processes all this information in order to find the next pair of velocities to be applied in the vehicle. The desired longitudinal velocity is pre-configured, but when the reactive control is triggered, this velocity is adjusted according to the current scenario (IRDWA optimization step, as explained in section 3.4.). The last block, the low-level controller, has the task of controlling the vehicle steering, throttle, and brake aiming to make the vehicle achieves the desired

velocities (V_{t+1} and W_{t+1}). Each block is explained in the next sections. The desired longitudinal velocity is pre-configured, so the system tends to keep this velocity, adjusting its value according to the current scenario.

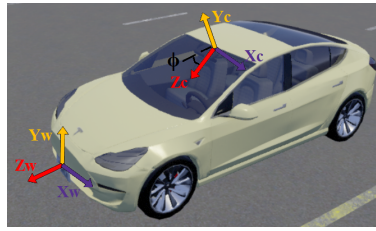


Figure 1. Camera frame ($X_c Y_c, Z_c$), World frames (X_w, Y_w, Z_w) and angle of rotation ϕ (rotation in X_c axis)

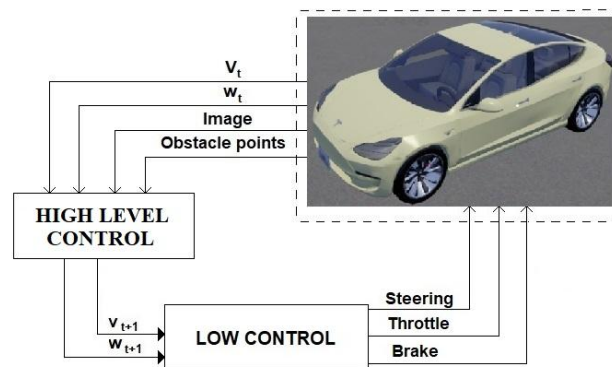


Figure 2. System overview

3. HIGH CONTROL: ENVIRONMENT PERCEPTION AND VELOCITIES ESTIMATION

All the steps compounding the high-level controller are summarized in Figure 3. The diagram shows the sequences of the whole process. There are four steps. It begins with lane line detection and tracking and is followed by control parameters estimation. The next step is yaw rate finding. Finally, it finds the optimal velocities with the IRDWA method.

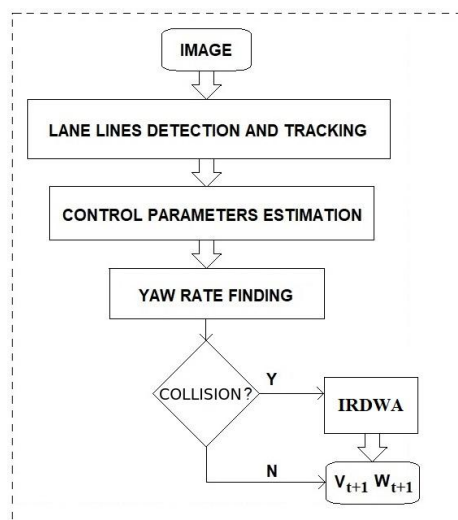


Figure 3. High level control diagram

3.1. Step 1: Lane lines detection and tracking

For the lane lines detection and tracking, the proposed steps are shown in Figure 4. This task is achieved in 2 steps. They are the lane lines detection step and the tracking step.

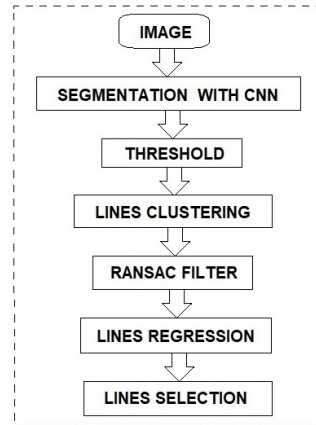


Figure 4. Lane lines detection and tracking diagram

3.1.1. Lane lines' detection

The first step is lane lines' detection in the images provided by the camera and is divided into two parts. The first part creates binary masks, one per line from the raw image, and the second extract a model for each line. The binary masks' prediction process is illustrated in the block diagrams as shown in Figure 5.

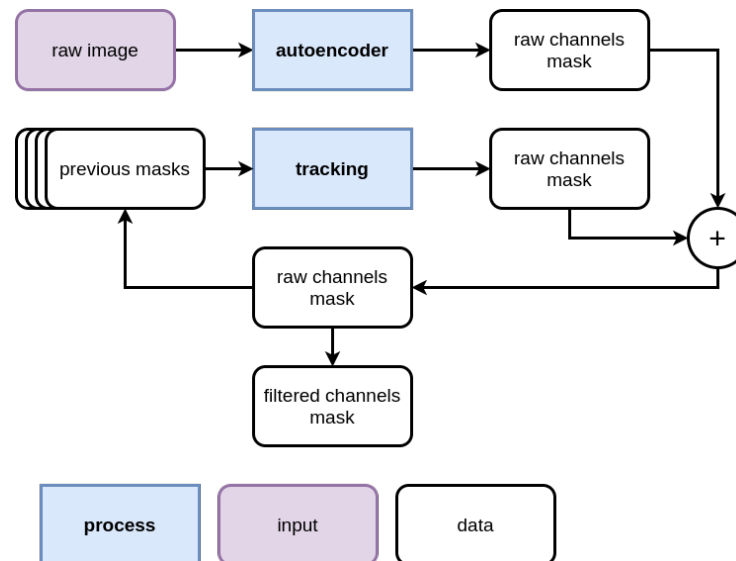


Figure 5. Binary masks predictions process

The binary masks are predicted by two deep-learning models. These models are trained and tested on the CuLane dataset [18]. Before the prediction, the image is preprocessed, pixel values are normalized and only the region of interest, the lower part of the image, is kept.

The first model predicts the future binary masks, this prediction is carried out using the autoencoder model. The autoencoder is composed of two parts: the encoder part and the decoder part. The encoder part compresses and selects information inside the image with successive convolution/pooling layers. Once features are extracted from the raw image, the decoder replaces this information in the initial space dimension and creates 4 binary masks, each one representing a lane line. Figure 6 shows the input and output shape of these masks.

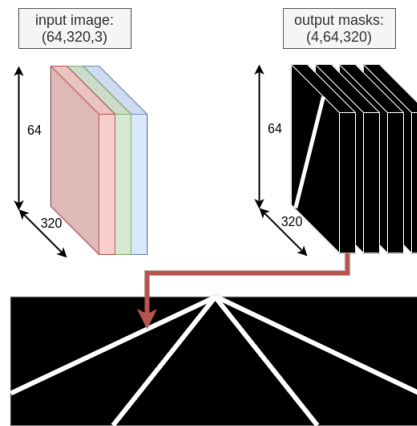


Figure 6. Input and output shapes

The second model was trained to predict the future binary masks from previous masks binary. This part allows us to take into consideration previous predictions and improve the robustness of the autoencoder model. The model is a convolutional LSTM network [19].

$$masks_{final} = \frac{w_{auto}.masks_{auto} + w_{track}.masks_{track}}{w_{auto} + w_{track}} \quad (1)$$

Figure 7 shows an example of prediction, each color line represents activated pixels from the same mask. As explained above, the tracking process improves the robustness of the prediction. This process can help to fill partial prediction (Figure 8) or remove outliers.



Figure 7. Prediction lines example



Figure 8. Tracking robustness example, top image shows predictions without tracking and bottom image shows predictions with tracking.

3.1.2. Lane lines tracking

In situations like when the vehicle is in the middle of two lanes, in the case where the car is changing lanes (for overtaking an obstacle for example), the segmented image tends to present more than one lane line,

needing a selection of the correct one. Thus, on each segmented image, all activated pixels are clustered using the agglomerative clustering algorithm [15], where each cluster is treated as a line candidate and the best one is chosen to represent the correct line points. To keep tracking the 4 lane lines along the way, the best line means the one with less error ($error_1$, defined by 2) according to the corresponding line position and inclination calculated in the previous image frame.

$$error_1 = e_{coef} + \alpha \cdot e_{int} \quad (2)$$

Lane lines that are not detected in the segmentation task are discarded and the most distant detected lines are used as road obstacles. These obstacles mean the road limits, image frame to world frame conversion is necessary to transform these lines into obstacles (road plane surface is considered).

3.2. Step 2: Control parameters estimation

Right after lane lines are selected in the previous step, visual parameters are extracted from the image as it is shown in Figure 9. These visual parameters are those used in the visual-based controller proposed by [16]. Point P is localized in the middle lane line (where the vehicle must keep driving) at a predefined vertical distance Y to the image bottom. Parameter X is the horizontal distance of the point P to the image center. The angle between the lane line center tangent and the vertical axis is the last parameter θ . X and θ values are then sent to the yaw rate finding, which is explained in the next section.

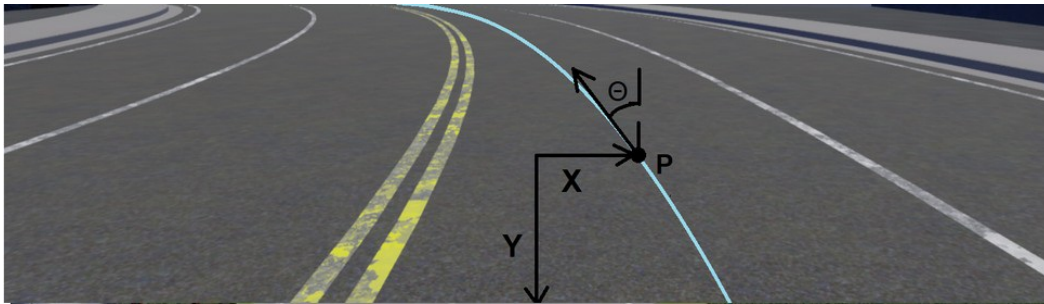


Figure 9. Visual parameters

3.3. Step 3: Yaw rate finding

The velocity W_{t+1} that the vehicle must achieve in order to keep driving on the desired lane, given the desired longitudinal velocity, is estimated using a neural network model. This machine learning model was trained by supervised machine learning with data gathered using CARLA simulator [20] (on map 07), where a car drove along a road keeping in the lane center. This model has as input the extracted visual parameters X and θ , the current linear velocity V_t , and the current yaw rate W_t .

The predicted yaw rate and the desired longitudinal velocity are then checked if they drive the car to the collision. This is done by calculating the distance to collision, as proposed by [21], with all obstacles in the 2D point cloud provided by the lidar. If the distance is less than a threshold value, a reactive control (IRDWA block) finds optimum velocities in order to avoid obstacles and keep the vehicle the closest to the lane center. Otherwise, if the distance to collision is bigger enough, the predicted velocities are considered.

3.4. Step 4: Finding optimal velocities with IRDWA method

When the collision is detected, a new pair of velocities (V_{t+1} and W_{t+1}) are found by maximizing the proposed IRDWA objective function as shown in 3. V_{min} and V_{max} are the minimum and maximum desired velocity, W_{max} is the maximum desired yaw rate, D_{coll} is the distance to collision and D_{min} is the minimum allowed distance to collision. For each possible velocity belonging to a search space (4) as shown in Figure 10, the distance to the collision on each obstacle point is calculated, and the minimum value is considered to be the D_{coll} value for these velocities. V_{br} and W_{br} are the maximum break accelerations, ac is the maximum acceleration and dt the time between two high control timestamps.

$$IRDWA = gain_{V1} \cdot Velocity_1 + gain_{Dist} \cdot Dist + gain_{V2} \cdot Velocity_2 \quad (3)$$

where

$$\begin{aligned}
 Dist &= \frac{D_{coll}}{D_{min}} \\
 Velocity_1 &= 1 - \frac{|W_t - W_{t+1}|}{W_{max}} \\
 Velocity_2 &= \begin{cases} \frac{V_{max} - V_t}{V_{max} - V_{t+1}} & , \text{ if } V_t > V_{t+1} \\ \frac{V_t}{V_{t+1} - V_{min}} & , \text{ if } V_t < V_{t+1} \end{cases} \\
 \{(V_{t+1}, W_{t+1})\} &\in V_{max} \cap V_a \cap V_s \cap W_{max} \cap W_s
 \end{aligned} \tag{4}$$

where

$$\begin{aligned}
 V_a &< V_t + ac \cdot dt \\
 V_t + ac \cdot dt &> V_a > V_t - V_{br} \cdot dt \\
 V_s &< \sqrt{2 \cdot D_{coll} \cdot V_{br}} \\
 W_s &< \sqrt{2 \cdot D_{coll} \cdot W_{br}}
 \end{aligned}$$

As the number of obstacles increases more calculations must be done for each evaluated velocity. For this reason, a reduction in the velocities search space can optimize this process by reducing the total number of calculations (less computational expense), or by making a more accurate search as the search space is reduced including the optimal values. An example of this search space is shown in Figure 10, where a heatmap shows the regions with higher IRDWA values.

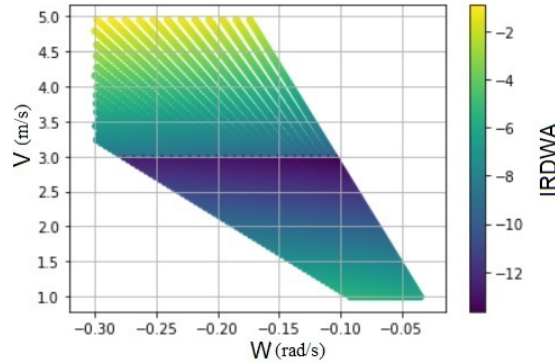


Figure 10. Search space

Supervised machine learning is used to accomplish the task of the proposed search window reduction. Therefore, a scalable tree boosting algorithm called XGBoost classifier [22] model is trained in order to predict the range of yaw rate values containing the optimum values. This machine learning-based model receives as input data: a flattened 2D occupancy grid containing all obstacle points around the vehicle, the current velocities (V_t and W_t), and the values X and $Theta$ provided by the control parameters estimation block. The occupancy grid is formed by the collected single-layer LIDAR points, in 2D space, covering a plane surface around the vehicle where the sensor is placed in front of the car.

4. VALIDATION RESULTS

The final system is evaluated using CARLA environment simulation on map 04. A computer with the following configuration is used to run the simulation together with the implemented proposed method: i7-6700HQ Processor and Nvidia Geforce GTX 970M graphic card. Using 0.1s and 0.5s for the low and high

controller timestep respectively, a minimum distance to the collision of 10 m and desired velocity equal to 4 m/s are considered to evaluate the results.

In the following subsection, the results of the learning step for the yaw rate prediction are shown, where, among different trained models, the best one is selected and tested in simulation for the lane centering task. In the next subsection, obstacles are placed in the road, where the high controller must drive the vehicle along the lane performing obstacle avoidance. Firstly, to make use of the reduced dynamic window from the IRDWA controller, a machine-learning model is trained and its results are shown. Then, different optimization methods for IRDWA maximization are used for comparison purpose, where each method are tested with and without the reduced dynamic window (activated and not activated).

4.1. Lane keeping

The training dataset containing visual features and the current velocities for yaw rate prediction was gathered from driving a vehicle on the lane center along the road of map 07, from the CARLA simulator. Situations, where the car is not in the lane center and must return to it, were also considered. Figure 11 shows the yaw rate histogram, which dataset has 6,804 data.

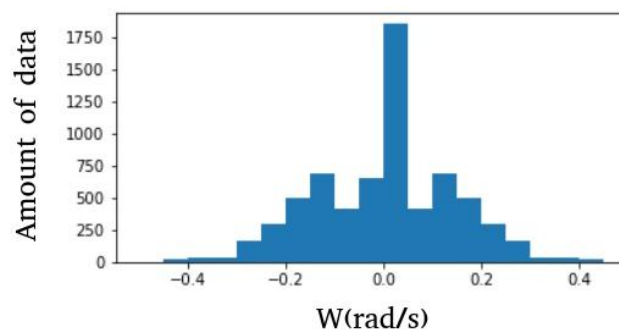


Figure 11. Yaw rate histogram

Supervised machine learning models were trained with the dataset applying different methods, and the results are shown in Table 1. Mean squared error (MSE) and accuracy (Acc) are the scores used for the best model selection. The Acc corresponds to the model response rate with an error of less than 0.5 m/s. The scores are evaluated both on training data and test data, preprocessing is done in the dataset (normalization or standardization) and the best hyperparameters were found by trying different settings and selecting the ones resulting in higher test accuracy.

Table 1. ML scores

Method	Preproc.	Parameters	Train score		Test score	
			MSE	Acc	MSE	Acc
SVR	Norm.	C=100, degree=0.5, kernel=rbf	0.011	0.878	0.011	0.880
Ridge	Norm.	alpha=1	0.021	0.776	0.021	0.779
K Neighbors	Stand.	Leaf size=3, neighbors=8	0.008	0.900	0.011	0.892
Random Forest	Norm.	Max. depth=13, estimators=300	0.002	0.965	0.010	0.884
Elastic Net	Stand.	Alpha=0.1, l1 ratio=0.5	0.262	0.744	0.272	0.751
Neural Network	Stand.	activation=tanh, batch size=64, learning rate=0.01, solver=sgd, hidden layer _s sizes = (500, 400, 200, 50)	0.119	0.893	0.118	0.886

Higher scores on test data mean a better generalization. As K-neighbors, random forest and neural network regression models presented good scores compared to the other methods. These three methods were then tested in a lane-keeping task. The results are shown in Table 2.

Table 2. Lane keeping results

Method	Maximum offset	Offset mean
K Neighbors	1.275	0.139
Random Forest	1.097	0.235
Neural Network	0.960	0.240

During each simulation, the vehicle drove on a 2800,00 meters road long. For all three tests, the vehicle was capable to keep the lane with a maximum offset to the lane center as shown in Table 2. The method which presented the lowest maximum offset was the neural network, and for this reason, was chosen to be used in the High control during the obstacle avoidance test.

Figure 12 shows the vehicle trajectory and its respective lane center offset in meters using a neural network model. The trajectory color represents the lane center offset, and its values are according to the sidebar scale. The points where the car is more distant from the lane center are colored yellow, and the points where the car is centered in the lane are purple. The visual parameter values X and Θ and the output from the high control (Yaw rate), collected in the trajectory segment with the maximum offset, are shown in Figure 13.



Figure 12. Trajectory

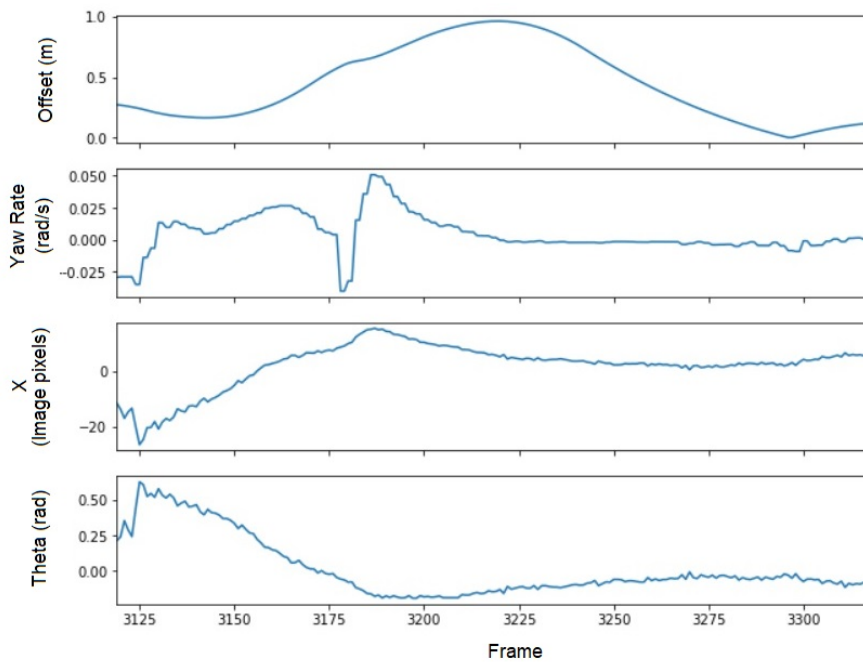


Figure 13. Maximum offset trajectory segment

4.2. Obstacle avoidance task

Dataset histograms used for the reduced dynamic window model training are displayed in Figure 14. Confusion matrix as shown in Figure 15 illustrates the performance of the final model, where the numbered vertical and horizontal axis represent the classes corresponding to their respective range of yaw rate values. The correct number of predictions are shown in diagonal from the confusion matrix. The F1 scores achieved by this model with train data is 0.996 and with test data is 0.826, which are good scores as the highest value is

1 (100% assertiveness). This model is then used to perform obstacle avoidance tasks, where the controller is validated and compared with and without the reduced dynamic window module activated.

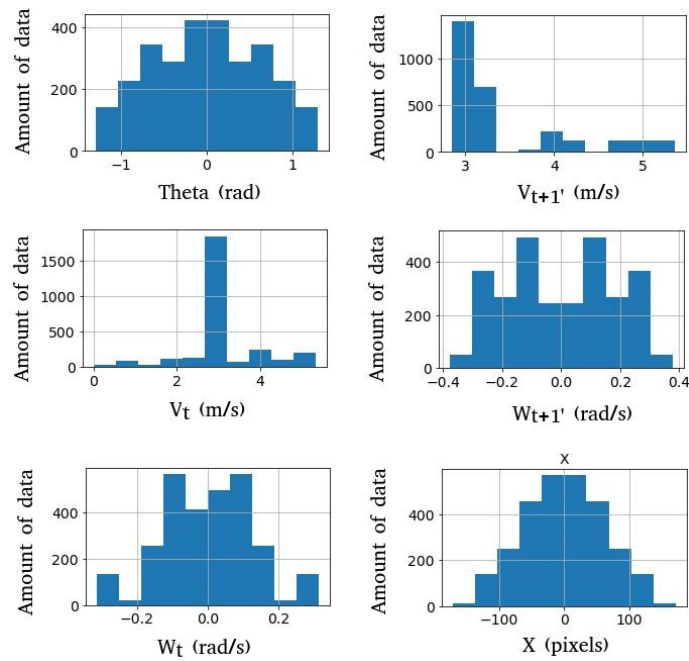


Figure 14. Dataset histograms

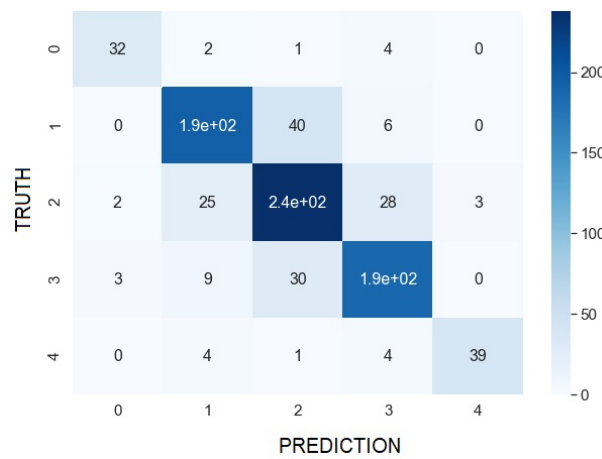


Figure 15. Confusion matrix

Tables 3 and 4 show the results achieved by running the final system on a simulated car driving along a road avoiding a single obstacle and multiple obstacles. Different optimization methods for DWA maximization are used, where different parameters are tested: Velocities value step between iterations for exhaustive optimization and population size for the other methods. The first column displays the computer processing time in seconds spent by each test set (optimization method, its parameters, and if the reduced dynamic window module is activated or not), and the corresponding mean of IRDWA values during the trajectory is shown in the second column.

Table 3. Optimization results: single obstacle

Case	Time (s)	IRDWA	Method	Parameters	SWR
1	02.835	12.220	Exhaustive	W step: 0.05 V step: 0.15	No
2	01.298	10.151	Exhaustive	W step: 0.05 V step: 0.15	Yes
3	04.426	10.404	Dif. Evol.	pop 15	No
4	18.298	12.631	Dif. Evol.	pop 15	Yes
5	01.858	10.760	Dif. Evol.	pop 03	No
6	04.639	13.031	Dif. Evol.	pop 03	Yes
7	01.177	09.936	Part. Swarm.	pop 25	No
8	03.371	11.829	Part. Swarm.	pop 25	Yes
9	00.521	10.100	Part. Swarm.	pop 05	No
10	00.776	10.605	Part. Swarm.	pop 05	Yes

Table 4. Optimization results: multiple obstacles

Case	Time (s)	IRDWA	Method	Parameters	SWR
1	02.456	12.257	Exhaustive	W step: 0.05 V step: 0.15	No
2	01.207	10.087	Exhaustive	W step: 0.05 V step: 0.15	Yes
3	05.161	12.401	Dif. Evol.	pop 15	No
4	15.955	12.659	Dif. Evol.	pop 15	Yes
5	01.693	10.849	Dif. Evol.	pop 03	No
6	05.181	13.019	Dif. Evol.	pop 03	Yes
7	00.817	09.201	Part. Swarm.	pop 25	No
8	01.380	11.325	Part. Swarm.	pop 25	Yes
9	00.471	08.582	Part. Swarm.	pop 05	No
10	00.830	11.152	Part. Swarm.	pop 05	Yes

Analyzing these tables and comparing the optimizer methods' performance with and without search window reduction, it is clear that the search window reduction enables a faster optimization in some cases (Case 2 on tables 3 and 4), and also with an increase on IRDWA mean value (Case 7 and 10 on Table 3) when population size is reduced. In some cases, higher IRDWA mean values are achieved with small changes in time (Case 7 and 10 on Table 4) when making the optimizer parameter simpler.

Figure 16 presents the trajectory made by the vehicle for some of the cases as shown in the tables, where the numbers represent the current frame or timestamp. Figure 16(a) is the resulted trajectory of Case 1 from Table 3, 16(b) is the resulted trajectory of Case 2 from Table 3, Figure 16(c) is the resulted trajectory of Case 7 from Table 4, and Figure 16(d) is the resulted trajectory of Case 10 from Table 4. The vehicle trajectory as well as the position of the obstacles were generated according to the coordinate points as collected during simulations. The velocities along the trajectory are shown in Figure 17, where Figure 17(a) corresponds to Case 1 from Table 3, Figure 17(b) to Case 2 from Table 3, Figure 17(c) to Case 7 from Table 4, and Figure 17(d) to Case 10 from Table 4. Samples of captured image frames are shown in Figure 18 and Figure 19, where the timestamps order is from top left to bottom right.

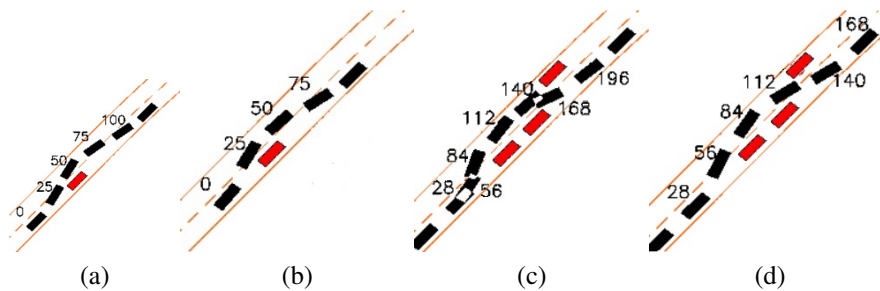


Figure 16. Resulted trajectory from tests in (a) case 1 from Table 3, (b) case 2 from Table 3, (c) case 7 from Table 4, and (d) case 10 from Table 4

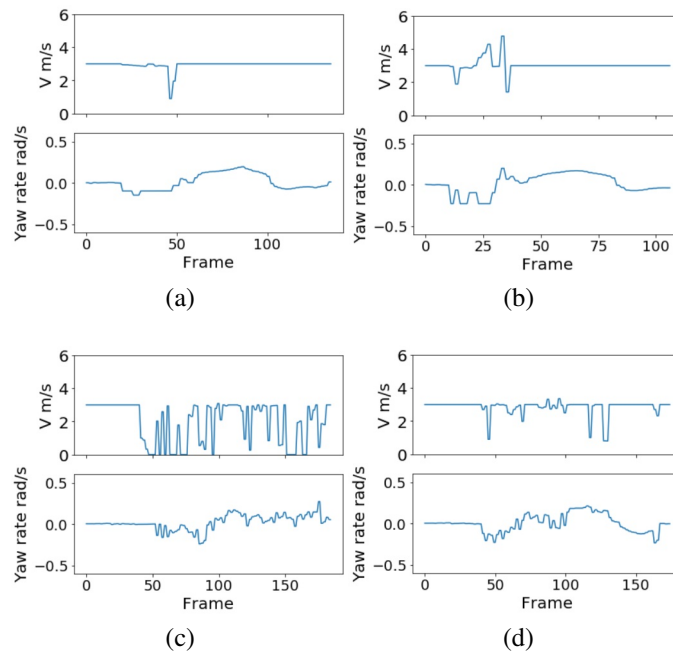


Figure 17. Resulted vehicle velocities from tests in (a) Case 1 from Table 3, (b) Case 2 from Table 3, (c) Case 7 from Table 4, (d) case 10 from Table 4)

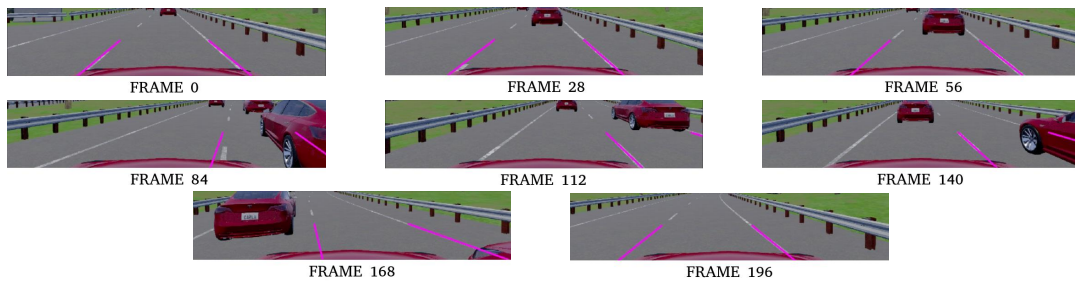


Figure 18. Images of case 7 from Table 4

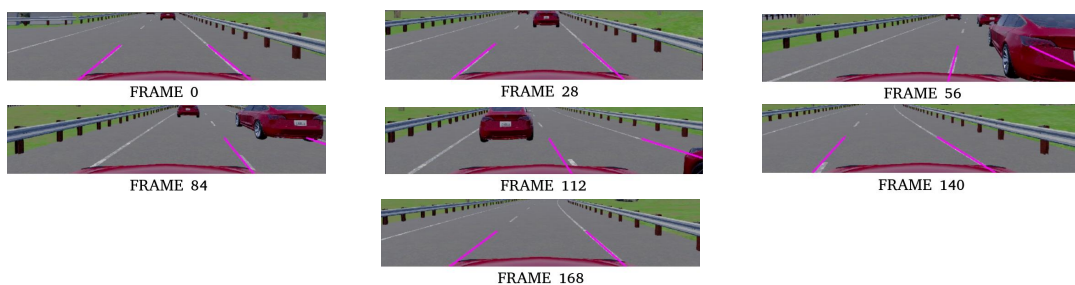


Figure 19. Images of case 10 from Table 4

5. CONCLUSION

This work proposed a combination of machine learning and a model-based controller. Successfully completing the validation tests, the results showed the learning capability for both lanes centering tasks and velocities search window reduction. The proposed system also showed its safety against obstacles collision,





carried out by its reactive functionality. More work must be done to make this system more robust, for example, consider vehicle dynamics in the IRDWA optimization step, enabling the vehicle drives with higher velocities. Also, more data and features can be added to the training data set, for both machine learning used in the system. For the yaw rate prediction, more vehicle dynamics information can be extracted. And for the reduced dynamic window machine learning model, complex scenarios can be covered in the data set.

REFERENCES





- [1] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert*, vol. 11, no. 2, pp. 19-27, Apr. 1996, doi: 10.1109/64.491277.
- [2] Waymo. <https://waymo.com/safety/> (accessed Oct. 05, 2019).
- [3] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: the principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50-61, Jul. 2020, doi: 10.1109/MSP.2020.2973615.
- [4] G. Zhao, X. Xiao, J. Yuan, and G. W. Ng, "Fusion of 3D-LIDAR and camera data for scene parsing," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 165-183, Jan. 2014, doi: 10.1016/j.jvcir.2013.06.008.
- [5] J. Van Brummelen, M. O. Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: the technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 384-406, Apr. 2018, doi: 10.1016/j.trc.2018.02.012.
- [6] C. Urmson *et al.*, "Autonomous driving in urban environments: boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425-466, Aug. 2008, doi: 10.1002/rob.20255.
- [7] K. O. Arras, J. Persson, N. Tomatis, and R. Siegwart, "Real-time obstacle avoidance for polygonal robots with a reduced dynamic window," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 3, pp. 3050-3055, doi: 10.1109/ROBOT.2002.1013695.
- [8] D. Janglová, "Neural networks in mobile robot motion," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, Mar. 2004, doi: 10.5772/5615.
- [9] L. Chang, L. Shan, C. Jiang, and Y. Dai, "Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment," *Autonomous Robots*, vol. 45, no. 1, pp. 51-76, Jan. 2021, doi: 10.1007/s10514-020-09947-4.
- [10] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 3654-3661, doi: 10.1109/IROS.2018.8594028.
- [11] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3671-3678, doi: 10.1109/ICRA.2012.6224742.
- [12] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: fixed time, near-optimal path generation via oracle imitation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3965-3972, doi: 10.1109/IROS40897.2019.8968089.
- [13] S. Thrun *et al.*, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661-692, Sep. 2006, doi: 10.1002/rob.20147.
- [14] N. H. Amer, H. Zamzuri, K. Hudha, and Z. A. Kadir, "Modelling and control strategies in path tracking control for autonomous ground vehicles: a review of state of the art and challenges," *Journal of Intelligent and Robotic Systems*, vol. 86, no. 2, pp. 225-254, May 2017, doi: 10.1007/s10846-016-0442-0.
- [15] M. Bojarski *et al.*, "End to end learning for self-driving cars," Apr. 2016, *arXiv:1604.07316*.
- [16] D. Alves de Lima and A. C. Victorino, "A visual servoing approach for road lane following with obstacle avoidance," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 412-417, doi: 10.1109/ITSC.2014.6957725.
- [17] B. A. Francis and M. Maggiore, *Models of mobile robots in the plane*. Springer, 2016, doi: 10.1007/978-3-319-24729-8_2
- [18] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: spatial CNN for traffic scene understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.12301.
- [19] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: a machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems*, pp. 802-810, Jun. 2015.
- [20] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: an open urban driving simulator," Nov. 2017, *arXiv:1711.03938*.
- [21] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 785-794, doi: 10.1145/2939672.2939785.
- [22] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection CNNs by self attention distillation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 1013-1021, doi: 10.1109/ICCV.2019.00110.

BIOGRAPHIES OF AUTHORS







Marcone Ferreira Santos     received a B.S. in mechanical engineering from the Pontifical Catholic University of Minas Gerais, Brazil, in 2013 and an M.Sc. degree in mechanical engineering from the Federal University of Minas Gerais, Brazil, in 2021. His current research interests are in the field of robotics, covering the areas of artificial intelligence and autonomous navigation system for intelligent vehicles, where his work has focused on integrating model-based approaches with learning-based approaches. He can be contacted at marconefs@ufmg.br.



Alessandro Corrêa Victorino     received a B.S. degree in mechanical engineering from Federal University of Espírito Santo (UFES) in 1996, an M.Sc. degree in mechanical engineering from the State University of Campinas (UNICAMP) in 1998; and his Ph.D. degree in robotics automation and control from National Institute for Research in Computer Science and Control (INRIA) in 2002. He defended a "Habilitation iriger des Recherches" (HDR) at University of Technology of Compiègne (UTC), Compiègne, France, in 2012. Since 2006, he has been an associate professor with the Department of Computer Engineering, UTC, and a member of the Heudiasyc Laboratory UMR CNRS 7253. His research interests include robotic and mechatronic systems, more precisely in the area of intelligent and robotic vehicles, the development of autonomous navigation and driver assistance systems, integrating advanced control techniques, the fusion of sensor data, observers of dynamic states, and exteroceptive perception of the environment. He can be contacted at acor-reav@hds.utc.fr.



Hugo Pousseur     is a Ph.D. student at Université de Technologie de Compiègne, France. He works on research in the fields of autonomous vehicles and artificial intelligence. His research focuses on the fusion of commands between a human driver and an autonomous driver. More precisely, he works on vision-based autonomous driving, human driving prediction, quantification, and command fusion. These different research areas revolve around artificial intelligence (deep learning, CNN, RNN, image processing) and visual control. He can be contacted at hugo.pousseur@hds.utc.fr.