Autonomous vehicle navigation based in a hybrid methodology: model based and machine learning based

Marcone Ferreira Santos, Alessandro Corrêa Victorino

Autonomous vehicle navigation based in a hybrid methodology: model based and machine learning based

Marcone Ferreira Santos
Mechanical Engineering Department
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil
marconefs@ufmg.br

Alessandro Corrêa Victorino
Computer Engineering Department
Sorbonne Universités - Université de Technologie de Compiègne (UTC)
HEUDIASYC UMR CNRS 7253
Compiégne, France
acorreav@hds.utc.fr

***Abstract*—For decades researchers have attempted to make the car drives autonomously. One of the challenges of developing this kind of system is the environment detection and understanding where the car is supposed to drive, and control the vehicle on a safe way, based upon on the perception of the environment, provided by on-boarded sensors (cameras, LIDAR). The existing navigation methods applied to solve this problem can be organized in two categories : those based on "geometrical or physical models" and those based upon on "machine learning models". Machine Learning based models can have good performance when trained appropriately, however, due to its "black-box" characteristics (lack of physical meaning), it can return non accurate output or even wrong values in unseen situations, leading the vehicle to collision. In the other hand, Geometric or physical approaches are designed under certain approximately modeling assumptions, based on physical/geometrical parameters that are uncertain or can not be identified, and becomes a painstaking work as it gets more complex. This paper presents a hybrid autonomous navigation methodology, which takes advantage of the learning capability of Machine Learning (ML) models, and uses the safeness of the Dynamic Window Approach geometric method. Using a single camera and a 2D LIDAR sensor, the proposed method actuates as a high level controller, finding optimal vehicle velocities to be applied by a low level controller. The system algorithm is validated on CARLA Simulator environment, where a vehicle coupled by this system proved to be capable of achieving the following tasks: Lane keeping and obstacle avoidance.**

***Index Terms*—Mechatronics Systems, Autonomous Vehicles, Autonomous Cars, Machine Learning, Intelligent Vehicle Systems**

I. Introduction

Since the first automobile equipped with a computer vision and automated steering mechatronics systems, called Navlab, appeared in the 1980s at Carnegie Mellon University [1], fully autonomous vehicles have becoming more safe, efficient, environment responsible in our nowadays society. Advanced mechatronics embedded systems make possible new autonomous navigation methodologies [2], with impressive results like in [3] where an autonomous vehicle have driven more than 16 million kilometers autonomously.

[1]This project was partially founded by Labex MS2T and by the Owheel project, Europe H2020 grant agreement No 872907.

Some tasks are necessary when developing an autonomous vehicle system, and usually is summarized as environment perception, mapping and localization, motion planning, decision and control. The perception task basically means understand the environment where the car is driving, and can be tackled by using cameras, LIDARs and/or other sensors. Object detection [4], lane signs identification, road segmentation [5], and even data fusion for better accuracy [6] are some of work done in perception step. These robotic functionalities are specific mechatronics modules embedded in the vehicle, which are constructed applying scientific methodologies based upon on "model based" or upon on "machine learning" control methods.

The model-based navigation control techniques aim to control the displacements of the vehicle in order to safely navigate in a previewed path, while avoiding eventual obstacles. The desired motion is generated by motion planning methods, which propose geometric kinodynamic candidate trajectories and select the best collision-free one, as used for some DARPA Urban Challenge teams [7]. Random search, optimal control and Artificial potential field are some advanced methods for path planing with obstacle avoidance [8]. The Stanley method is one of the popular steering controller that was first introduced in the DARPA Grand Challenge with Stanford University's entry [9]. Model predictive control (MPC), Fuzzy control and preview control are some techniques still in research [10]. All these techniques are based on the geometrical modeling of the environment around the vehicle, requiring, for some ones, a complete localization and mapping solution of the navigation system. On the other hand, advanced methods as machine learning based controller have been in active research area. In [11] an end to end control learning is designed, where the vehicle is totally driven by a Deep Learning model.

Machine Learning based navigation systems can presents good performance when trained appropriately, however non accurate or even wrong output values can be returned in unseen situations, leading to a collision. Geometric based navigation systems are designed under certain assumptions, and becomes more computational expensive and a painstaking work as it

gets more complex. For this reason, this project aims to develop a hybrid system, combining Machine learning and geometric based models in order to take advantage of both methods, using a single camera and 2D LIDAR.

The proposed methodology is based on local navigation with reactive strategy requiring only the environment perception for control objectives achieving. The method is inspired in IDWA controller [12], where in our approach vehicle velocities are learned by supervised learning through the visual features, and a new objective function (IRDWA) for the reactive control is proposed. Lane lines segmentation model benchmark proposed by [13] is integrated in the system in order to extract the visual features.

This paper is structured as follows. Section 2 presents the proposed methodology, explaining each block that composes the general system: High level control (perception task and velocities generation) and low level control (Steering, acceleration and brake control). Each aspect involved in the High control block design are presented with more detail in section 3. The low level control block is deeper explained In section 4. Finally, section 5 shows the results obtained with the final algorithm implemented in a simulation environment, including Lane following task and obstacle avoidance task.

II. Modeling aspects: general overview of the system

A schematic of the vehicle with its sensors is shown in figure 1, the camera is attached on the car roof and the LIDAR on the front side. The LIDAR position is considered to be the World frame (reference frame used when distance to collision are estimated).

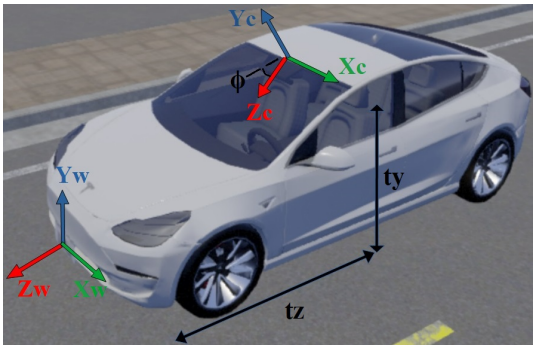


Fig. 1: Camera frame (Xc, Yc and Zc) and World frames (Xw, Yw and Zw), angle of rotation ϕ (rotation in Xc axis) and the relative distance ty and tz between base frames origin.

Figure 2 shows the general systems in 3 main blocks. LIDAR sensor and a single camera compound the sensory system attached on the vehicle. The vehicle block sends some information to the high control, where Vt is the current vehicle longitudinal velocity and Wt the current yaw rate. Image and obstacle points are provided by the camera and LIDAR.

Then, the High control block process all the given data, as a way to find the next longitudinal V_{t+1} and yaw rate W_{t+1} to be applied on the vehicle through the low control block.

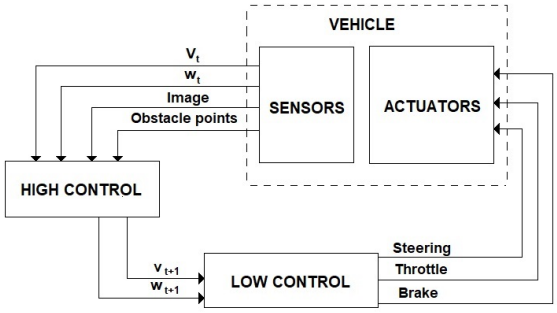


Fig. 2: Example of a figure caption.

As the dynamics is not considered in this work, the kinematic bicycle model is used to estimate the vehicle behavior [14]. Therefore, the vehicle is evaluated in low speed (3 m/s).

III. High control: Environment perception and velocities estimation

The process to find V_{t+1} and W_{t+1} is summarized in Figure 3. Firstly, existing lane lines are extracted from the image. Then these extracted lines are processed in the control parameters estimation block. The Yaw Rate Finding block estimates the velocity W_{t+1}, using V_t and the visual parameters found in the previous block. The V_{t+1} value is predefined according to a desired longitudinal velocity based on traffic rules. Then these pair of velocities (V_{t+1} and W_{t+1}) are checked if leads the vehicle to collision, if so new values are find taking into consideration the Yaw Rate Finding output, the current vehicle state and the visual features.

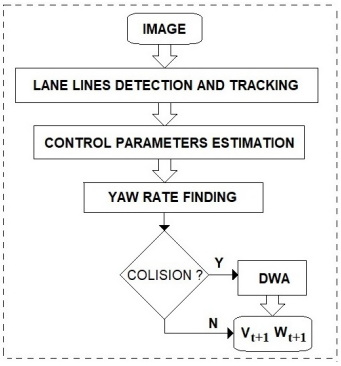


Fig. 3: High control diagram

Each block of the High control diagram, presented in Figure 3, are described in the following sections.

A. Lane lines detection and tracking

For the lane lines detection and tracking the proposed steps are shown in figure 4. For each image captured by the camera, lane lines segmentation is done with a trained convolution neural network model [13].

This segmentation model returns 5 images as shown in Figure 5, where each image corresponds to a specific lane line: the left line from the left lane, the left line from the

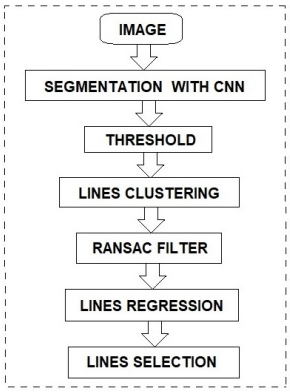


Fig. 4: Lane lines detection and tracking diagram

central lane, the right line from the central lane and the right line from the right lane. Each image presents activated pixels when the presence of the respective lane line is recognized by the model (Figure 5), These images are 1 channel, and its activated pixels are those with high values.

The model also returns the confidence probability for each image to have a lane line, so images with a probability above to 0.6 are considered that the respective lane line exists. If so, these images are normalized and all pixels with values above a threshold are considered to be activated points. Some values was tested and 0.6 was choose as it performed well.

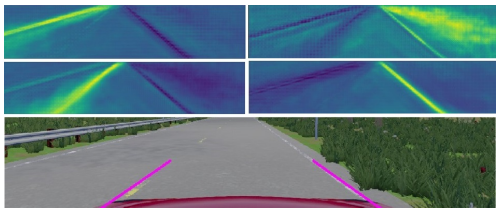


Fig. 5: Lane lines segmentation (upper images) and final selection and tracking (lower image)

When the vehicle is in the middle of two lanes, situation where the car is changing of lane (for overtaking an obstacle), the segmented image tends to present more than one lane lines, needing a selection of the correct one. To overcome this, and for time integrating purpose, on each segmented image all activated pixels are clustered using Agglomerative Clustering algorithm [15]. One ore more clusters can be found, so each cluster is treated as a line candidate, where the best one is chosen to represent the correct line points. The best candidate lines are selected getting the lowest error (called $error_1$), defined as:

$$error_1 = e_{coef} + \alpha \cdot e_{int} \tag{1}$$

where e_{coef} and e_{int} are the angular coefficient and intercept differences between the previous line and the candidate line, and a weight factor (7.0 was found to be a good weight value). Ransac Linear regression is used as the model fitting. Each selected lane lines (right and left lane side) is compared with the previous frame, by calculating an error, called $error_2$, as follows:

$$error_2 = \frac{|e_{coef}|}{e_{max-coef}} + \frac{|e_{int}|}{e_{max-inter}} \tag{2}$$

where $e_{max-coef}$ and $e_{max-inter}$ are the maximal admissible angular coefficient and intercept difference. In case these lines differ to much with each other ($error_2$ greater than 1.0) the lane line selected in the previous frame keeps in the current one. This time integration step ensures the desired lane lines tracking.

1) Non existing lane invasion: When the car face an obstacle while driving on a road, sometimes it is necessary to get a bit away from the lane center, resulting in an other lane invasion. To avoid the vehicle drives to a non existing lane, the lane line to be crossed which its confidence probability is bellow to 0.5 is considered as obstacle. To perform this task, world coordinates points must be found from line image pixels. Considering the road as a plane, we have:

$$F = \begin{pmatrix} f & 0 & C_x \\ 0 & f & C_y \\ 0 & 0 & f \end{pmatrix} \tag{3}$$

$$T = \begin{pmatrix} 1 & 0 & tx \\ 0 & -\sin\phi & ty \\ 0 & \cos\phi & tz \end{pmatrix} \tag{4}$$

$$p = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{5}$$

$$P = \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{6}$$

$$p = F \cdot T \cdot P \tag{7}$$

$$P = (F \cdot T)^{-1} \cdot p \tag{8}$$

Parameter f is the camera focus distance, C_x and C_y image center corrections; tx, ty and tz relative distance from the world frame to the camera frame. World frame origin is stetted to be at the LIDAR position (in the vehicle front side). P is the coordinates point in world frame, and p is the coordinates in image frame. Each image lane line point that the car must not cross, is converted to world coordinated and is counted as an obstacle.

B. Control parameters estimation

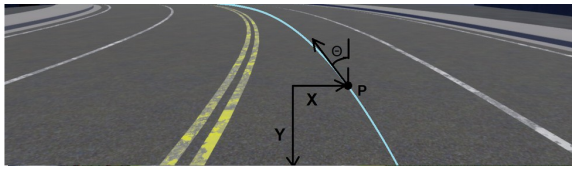


Fig. 6: Image frame with point P in the lane line center (in blue), at fixed distance Y and the control parameters θ and X

The visual parameters generated in control parameters estimation block are shown in Figure 6. θ corresponds to the angle of line lane center tangent at point P from the image vertical axis. X is the image horizontal distance between image center and point P. These parameters values are sent to the Yaw Rate Finding block.

C. Yaw Rate Finding

This block task is done by supervised machine learning technique. For a regression model to be trained, a dataset is required, therefore data containing control parameters and the current velocities are gathered in a vehicle simulation environment, where a car is driving along a road keeping in the lane center. Then the collected data are used as the dataset.

As the vehicle is supposed to drive with a specific longitudinal velocity, according to the road limits, only the yaw rate velocity is predicted by regression model.

As outliers are not desired, the dataset is filtered with Z-score (7), where μ is the mean of the time interval yaw rate values and σ the standard deviation, such that outliers are avoided on training step. When yaw rate difference values is less than 0.01 rad/s for a time interval, values with Z-score grater than a threshold (1.0) is not considered. The dataset is then augmented by multiplying all set of values by -1 but velocity. Then the dataset is normalized and split into training (70%) and test (30%) dataset. Different supervised machine learning models is trained, and the bet one is chosen according to its score.

$$Z = \frac{value - \mu}{\sigma} \tag{9}$$

CARLA simulator [16] is chosen to gather the dataset and to test the final algorithm, as it is well documented and is available for python language code. A vehicle is driven along lanes on map 07 (available in the simulator) while necessary data are collected and stored.

D. Image-based Reduced Dynamic Window Approach (IRDWA)

The predicted W_{t+1} from Yaw Rate Finding block and the desired velocity V_{t+1}, this pair of values is then checked. If it drives the vehicle to a collision, new values are found by maximizing the objective function $IRDWA$:

$$\begin{aligned} IRDWA = {} & gain_{V1} \cdot Velocity_1 + gain_{Dist} \cdot Dist \\ & + gain_{V2} \cdot Velocity_2 \end{aligned} \tag{10}$$

where,

$$Dist = \frac{D_{coll}}{D_{min}}$$

$$Velocity_1 = 1 - \frac{|W_t - W_{t+1}|}{W_{max}}$$

$$Velocity_2 = \begin{cases} \dfrac{V_{max} - V_t}{V_{max} - V_{t+1}} & \text{, if } V_t > V_{t+1} \\ \dfrac{V_t}{V_{t+1} - V_{min}} & \text{, if } V_t < V_{t+1} \end{cases}$$

$Dcoll$ is defined as the minimum distance to collision as proposed by [17], being calculated for polygonal robots. $Dmin$ is a predefined value, where $Dcoll$ must be grater than $Dmin$, otherwise it means that the vehicle drives to collision.

Gain parameters ($gain_{V1}$, $gain_{Dist}$ and $gain_{V2}$) must be settled according to desired behavior preferences. Increasing $gain_V1$ the pair of values optimized will be nearest to the predicted values by Yaw Rate Finding block, but if $gain_{Dist}$ is increased the vehicle tends to go more distant from obstacles, and, and if $gain_{V2}$ is bigger vehicle longitudinal velocity is preferred.

A search space must be considered while optimizing $IRDWA$:

$$\begin{gathered} \{(V_{t+1}, W_{t+1})\} \in V_{max} \cap V_a \cap V_s \cap W_{max} \cap W_s \\ V_a < V_t + ac \cdot dt \\ V_t + ac \cdot dt > V_a > V_t - V_{br} \cdot dt \\ V_s < \sqrt{2 \cdot D_{coll} \cdot V_{br}} \\ W_s < \sqrt{2 \cdot D_{coll} \cdot W_{br}} \end{gathered}$$

Where V_{max} and W_{max} are the maximum allowed velocities, ac the maximum acceleration, V_{br} and W_{br} the maximum break acceleration, dt the time between two command controls. Exhaustive search optimization is the method used to find the optimum values for V_{t+1} and W_{t+1}, where less variation between searched values the better results should have.

Collision checking is done for all obstacles points. To save computational cost, as it gets more expensive with more obstacles points, 2D occupancy grid is considered in this work.

IV. Low level control: input to the actuators

The Low level control has the optimum velocities generated by the High control block as its input. And has the task of control the actuators presented in the vehicle: Throttle, Brake and Steering. In order to make the car drives with the desired velocities. Two Proportional and Integrative (PI) controllers are used for this task, one for the longitudinal velocity and the other for yaw rate velocity.

V. Validation results

The final algorithm is tested on map 04 (Figure 8) for different tasks: Lane keeping and obstacle avoidance. A computer with the following configuration is used on this work: i7-6700HQ Processor and Nvidia Geforce GTX 970M graphic card. All the results are avalueted considering a minimum distance to collision equal to 10 m and desired velocity 3 m/s.

1) Yaw Rate Finding: The training dataset was gathered driving a vehicle on the road of map 07 from CARLA simulator, as showed in Figure 5. Different situations were considered: lane keeping and returning to the goal lane from a distance to its center. Figure 7 shows the dataset distribution, which has 6804 collected and augmented data.

Supervised machine learning models were trained with the dataset applying different methods, and the results are shown

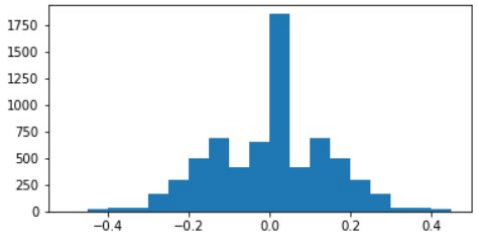


Fig. 7: Yaw rate histogram

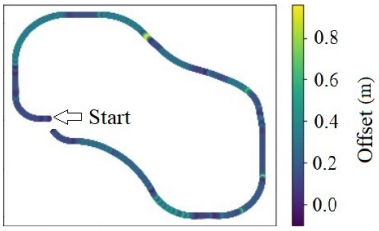


Fig. 8: Trajectory

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 error less than 0.5 m/s. The scores are evaluated both on training data and test data, preprocessing are done in the dataset (Normalization or Standardization) and the best hyperparameters were found by trying different settings and selecting the ones resulting on higher test accuracy.

TABLE I: 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$_sizes = (500, 400, 200, 50)$	0.119	0.893	0.118	0.886

Higher scores on test data means a better generalization, and as K Neighbors, Random Forest and Neural Network Regression models presented good scores compared to the other methods it is chosen to be tested in the vehicle High controller. The results are shown in the following sections.

A. Lane keeping

With both machine learning trained models (K Neighbors, Random Forest and Neural Network) the vehicle was capable to drive keeping itself on the same lane in the map as shown in Figure 8. The vehicle mean offset and the maximum offset to the lane center for each tested method are shown in table 2.

TABLE II: 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 the each simulation the vehicle runned on a 2800,00 meters long road, and was capable to keep on the same lane during the whole trajectory with a maximum offset as shown in the table above. The method which presented the lowest maximum offset was Neural Network, for this reason was chosen to be used in the High control during the obstacle avoidance test (Section 5.3).

Figure 10 shows the vehicle trajectory and its respective lane center offset in meters using Neural Network model.

The trajectory color presented in Figure 8 represents the lane center offset, and its values are according to the side bar scale. The points where the car is more distant to the lane center is colored by yellow, and the points where the car is centered in the lane are in purple. Visual parameters (X and Θ) and the output from the high control (Yaw rate) collected in the trajectory segment with the maximum offset point are shown in Figure 9, which presents the graph for each of these values by the current frames.

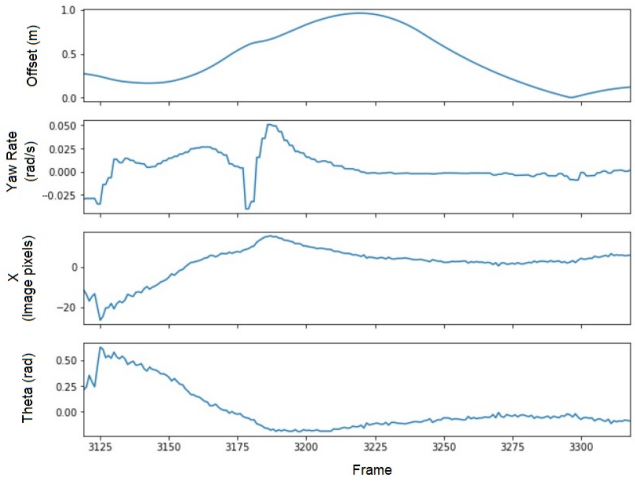
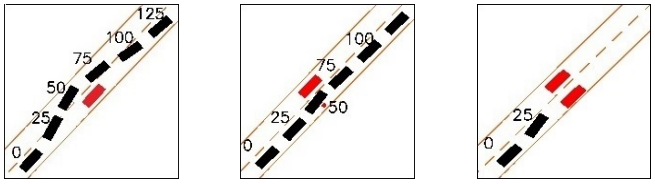


Fig. 9: Maximum offset trajectory segment

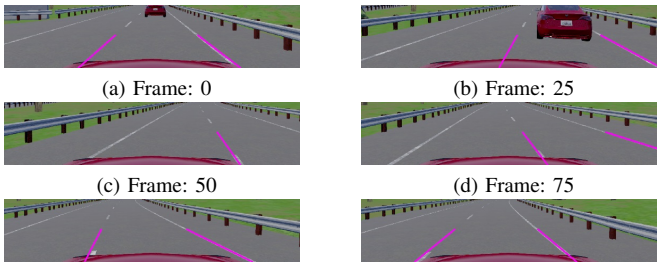
B. Obstacle avoidance

Three different cases are proposed for obstacle avoidance task analysis. The former consider 1 obstacle on the lane center (case 1), the second one has 2 obstacles and a narrow way between them (case 2), and the third case is set to have obstacles totally obstructing the road (case 3). The results for each of these cases are shown bellow. Figure 10 presents the trajectory made by the vehicle for each situation, with their position marked by their corresponding image frame (Figure 11), obstacles are painted in red color. Lidar output can be seen in Figure 12, where obstacles are painted in yellow color. Finally, visual features X and Θ, the vehicle velocities and the high control outputs (Yaw rate values) are presented in figure 13. The DWA weights were manually found with an empirical procedure.



(a) Single obstacle (b) Two obstacles (c) Obstructed road

Fig. 10: Trajectory

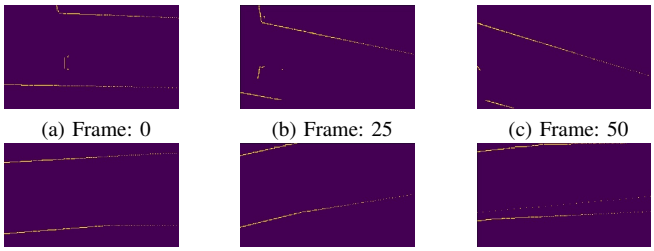


(a) Frame: 0 (b) Frame: 25

(c) Frame: 50 (d) Frame: 75

(e) Frame: 100 (f) Frame: 125

Fig. 11: Single obstacle: Camera images



(a) Frame: 0 (b) Frame: 25 (c) Frame: 50

(d) Frame: 75 (e) Frame: 100 (f) Frame: 125

Fig. 12: Single obstacle: LIDAR points

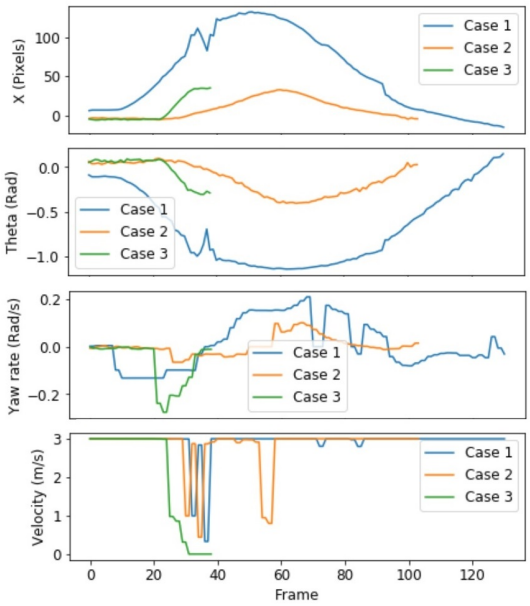


Fig. 13: Single obstacle: Collected data

VI. Conclusion

The results showed that the vehicle was capable not to only drive keeping on same lane, but when facing an obstacle could avoid it with no collision (Case 1 and 2), and when there was not free space on the road to continue driving, the vehicle could stop until a collision happen (Case 3).

References

[1] Pomerleau, D., & Jochem, T. (1996). Rapidly adapting machine vision for automated vehicle steering. IEEE Expert, 11(2), 19–27.

[2] Fagnant, D. J., & Kockelman, K. (2015). Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. Transportation Research Part A: Policy and Practice, 77, 167–181.

[3] Building the World's Most Experienced Drive. 2019. Available in: rhttps://waymo.com/safety/. Accessed in October 2019.

[4] Li, Y., & Ibanez-Guzman, J. (2020). Lidar for Autonomous Driving: The principles, challenges, and trends for automotive lidar and perception systems. Retrieved from http://arxiv.org/abs/2004.08467

[5] Van Brummelen, J., O'Brien, M., Gruyer, D., & Najjaran, H. (2018). Autonomous vehicle perception: The technology of today and tomorrow. Transportation Research Part C: Emerging Technologies, 89(July 2017), 384–406. https://doi.org/10.1016/j.trc.2018.02.012

[6] Zhao, G., Xiao, X., Yuan, J., & Ng, G. W. (2014). Fusion of 3D-LIDAR and camera data for scene parsing. Journal of Visual Communication and Image Representation, 25(1), 165–183. https://doi.org/10.1016/j.jvcir.2013.06.008

[7] Urmson, C.; Anhalt, J.; Bagnell, D.; Baker, C.; Bittner, R.; Clark, M.N.; Dolan, J.; Duggins, D.; Galatali, T.; Geyer, C.; et al. Autonomous driving in urban environments: Boss and the Urban Challenge. J. Field Robot. 2008, 25, 425–466

[8] Rasekhipour, Y., Khajepour, A., Chen, S., & Litkouhi, B. (2017). A Potential Field-Based Model Predictive Path-Planning Controller for Autonomous Road Vehicles. 18(5), 1255–1267.

[9] Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; et al. Stanley: The robot that won the DARPA Grand Challenge. J. Field Robot. 2006, 23, 661–692.

[10] Amer, N. H., Zamzuri, H., Hudha, K., & Kadir, Z. A. (2017). 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: Theory and Applications, 86(2), 225–254. https://doi.org/10.1007/s10846-016-0442-0

[11] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., … Zieba, K. (2016). End to End Learning for Self-Driving Cars. 1–9. Retrieved from http://arxiv.org/abs/1604.07316

[12] De Lima, D. A., & Victorino, A. C. (2014). A Visual Servoing approach for road lane following with obstacle avoidance. 2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014, 412–417. https://doi.org/10.1109/ITSC.2014.6957725

[13] Hou, Y., Ma, Z., Liu, C., & Loy, C. C. (2019). Learning Lightweight Lane Detection CNNs by Self Attention Distillation. Retrieved from http://arxiv.org/abs/1908.00821

[14] Francis, B. A., & Maggiore, M. (2016). Models of Mobile Robots in the Plane. 7–23. https://doi.org/10.1007/978-3-319-24729-8

[15] Müllner, D. (1984). Modern hierarchical, agglomerative clustering algorithms. (1973), 1–29.

[16] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. (CoRL), 1–16. Retrieved from http://arxiv.org/abs/1711.03938

[17] Arras, K.,Persson,J.,Tomatis,N.,andSiegwart,R.(2002).Real time obstacle avoidance for polygonal robots with a reduced dynamic window. In Proceedings of the IEEE International Conference on Robotics and Automation, volume 3, pages 3050 – 3055.