Safe and Computational Efficient Imitation Learning for Autonomous Vehicle Driving

Flavia Sofia Acerbo, Herman Van der Auweraer, Tong Duy Son

Siemens Digital Industries Software, 3001 Leuven, Belgium
Email: {flavia.acerbo,herman.vanderauweraer,son.tong}@siemens.com

***Abstract*— Autonomous vehicle driving systems face the challenge of providing safe, feasible and human-like driving policy quickly and efficiently. The traditional approach usually involves a search or optimization-based planning followed by a model-based controller. This may prove to be inadequate in some driving scenarios due to disturbance, uncertainties and limited computation time. The more recent end-to-end approaches aim at overcoming these issues by learning a policy to map from sensor data to controls using machine learning techniques. Although being attractive for its simplicity, they also show some drawbacks such as sample inefficiency and difficulties in validation and interpretability. This work presents an approach that attempts to exploit both worlds, combining machine learning-based and model-based control into an imitation learning framework that mimic expert driving behavior while obtaining safe and smooth driving. The dataset is generated from high-fidelity simulations of vehicle dynamics and model predictive control (MPC). A smooth spline-based motion planning represents the policy provided by a constrained neural network exploiting the convex hull property of B-splines. The policy network is trained with few dataset aggregations coming from its induced distribution of states. The learned policy is used as guidance for model-based feedback control and tested on a 15DOF high fidelity vehicle model.**

I. Introduction

Recently, autonomous vehicle technologies have gained significant interest from both industrial and academic research. Generally, autonomous driving systems can be divided into hierarchical blocks such as perception, planning and control [1]. Perception is done through, i.e. segmentation, object classification.... Motion planning algorithms are based on graph-search, rapidly exploring random tree, or optimization-based techniques. And trajectory tracking is designed with feedback controllers acting on steering, throttle and brake actuators. This hierarchical approach may show some drawbacks in safety and computational efficiency. First, the planning and control algorithms often rely on simplified models, which may not represent sufficiently the vehicle dynamics in certain driving scenarios. Second, while optimization-based motion planning or control methods such as model predictive control (MPC) have demonstrated potential capabilities in dealing with complexities of traffic environment, they are sometimes difficult to meet real-time requirements in critical or complex scenarios. Third, it is generally hard to incorporate human-like driving behavior such as comfort and performance aspects into the design.

Considering its ability in learning from large amount of data that is possible to get from human driving, imitation learning techniques for autonomous driving have been investigated. For example, [2]–[4] discussed the promising end-to-end approach that aims at mapping directly control actions from raw sensor readings. The results have shown some successes but their use in production vehicles is limited [5]–[8]. Indeed, this approach suffers from sample inefficiency, lack of interpretability and guarantees on safety. Moreover, the deep neural network structures require large labeled dataset and long training time.

In this paper, we exploit the advantages of both model-based and machine learning-based approaches for a hierarchical mid-to-mid framework. The inputs to the proposed learning model are representative features coming from processed sensor data and the outputs are, in some form, the reference trajectories. This reduces the complexity of the neural network architecture while leaving to model-based feedback techniques to guarantee safety, feasibility and stability. The imitation learning training model is implemented using online dataset aggregation (DAgger) [9]. This is an iterative supervised learning fashion, with an increasing dataset due to the exposure of the expert driver to new states induced by the learner. The dataset for learning is generated from an advanced nonlinear MPC design proposed in [10]. This MPC design guarantees safety in infinite horizon formally using control barrier function. In the training loss function of the neural network, we incorporate from beginning the knowledge of safety objectives such as collision avoidance with road boundaries through barrier function constraints. Combined with DAgger, the proposed loss function will show advantages on both safety improvement and convergence speed.

Moreover, previous imitation learning works usually consider output layers representing directly the trajectory coordinates. This choice may lead to jerky motions that do not represent well the expert behaviors, and also does not scale well for long horizons. An interpolation post-processing method could be used to deal with this problem, which on the other hand, will increase computation and affect safety. Consequently, we propose to use B-spline trajectory parametrization, choosing spline coefficients instead of points as output nodes for learning. Another advantage of this method lies in the fact that a B-spline is always contained

in the convex hull of its coefficients, i.e. a spline function is contained within the minimum and maximum value of its coefficients. Therefore, safety constraints on generated trajectories can be imposed by only constraining the spline coefficients in the barrier function of the loss function.

The paper is organized as follows. In Section II, previous works are presented and compared to ours. Section III presents some relevant background and Section IV discusses architecture of the proposed network. The simulation framework and settings for the offline and online learning illustrated in. Finally, Section VI comments the results obtained.

II. Related Works

Since the late 1980s, end-to-end imitation learning for self-driving cars was investigated by Pomerleau, with his work on ALVINN [2]. More recently, this approach has gained new attention from Nvidia [4], where a CNN was trained to steer a car directly from camera images, after 3000 miles of driving. This approach was also presented in Pan et al. [3] for high-speed off road driving of a small experimental rally car. However, the end-to-end approach is generally not recommended for production vehicles due to sample inefficiency [6] or safety and validation reasons [7]. A complete framework in this fashion has been developed by Sun et al. [5], where a shallow fully-connected neural network is employed in the long-term planning, while a low level MPC guarantees short-term safety and control. The work by Zhan et al. [11] showed the potential of training losses different from L_2 in case of safety-critical single maneuvers. The use of additional training losses has been also exploited by Waymo in ChaffeurNet [8], where a RNN was trained to predict the trajectory to be consumed by a controller. In that case it has been chosen not to train the network in an online learning manner, and synthetized perturbations have been created to avoid using algorithms like DAgger [9].

Our work demonstrates a mid-to-mid approach with simple fully-connected networks trained using online learning and an augmented training loss, investigating the interaction between them. We propose a combination of different techniques learned from model-based control community such as safety-critical nonlinear MPC in data generation, barrier functions in loss functions, and different choice of output variables parameterizing the predicted trajectory to reduce the number of output nodes. The solution is practically promising to improve driving safety, comfort, convergence speed, and computational effort.

III. Background

This section presents some background and notions used in the proposed motion planning framework. First, we discuss briefly imitation learning and how it differs from pure supervised learning. Second, B-splines and motivation to consider B-spline are presented. Third, model predictive control design as expert planner is given.

A. Imitation Learning

Imitation learning is a type of machine learning that, given demonstrations, is able to directly learn how to mimic a certain behavior. The expert policy is defined as $a_t = \pi^*(o_t)$, i.e. the mapping between observations and actions that needs to be learned. First of all, the policy π^* is run and a dataset of observations and actions is collected for each visited state s of the system. We can call this dataset $D^* = \{(o_1, a_1), ..., (o_N, a_N)\}$ and define the set of visited states as S^*. Applying supervised learning on D^*, a policy $\hat{\pi}$ is learned such that $\hat{\pi} = \arg\min_\pi loss(\pi, \pi^*)$. This is also known as behavioral cloning. The supervised learning is not always accurate, and $\hat{\pi}$ will present imperfections that will lead the system to reach states not included in S^*, for which the policy behavior becomes unpredictable. This is usually called compounding of errors problem and it is due to the fact that the statistical i.i.d. assumption between training and testing data is not valid for sequential predictions.

To overcome this issue, several algorithms have been proposed and one common solution is DAgger (Dataset Aggregation) proposed by Ross et al. [9]. Starting from the policy trained on D^*, we let this policy control the system (policy rollout), while the expert labels each new visited state with the actions it would have taken. This generates a new dataset $D_1 = \{(o_1, a_1^*), ..., (o_N, a_N^*)\}$ of observations coming from the states visited by the learner and actions labeled by the expert. The two datasets are aggregated and a new policy is trained on $D = \{D^* \cup D_1\}$. The algorithm is repeated until the policy performance on the learner distribution of states, based on a certain metric, does not improve anymore. The algorithm guarantees that the policy will converge and its loss will be inversely proportional to the number of policy rollouts N.

B. B-splines parametrization

Splines are piecewise polynomial function that can easily represent trajectories as smooth, continuous functions by only a limited number of variables. The points where the pieces meet are the knots which are sorted in non-decreasing order and not necessary to be distinct. A spline is expressed as a linear combination of B-splines basis function $B_i(\tau)$ with spline coefficients α_i as

$$s(\tau) = \sum_{i=1}^{n} \boldsymbol{\alpha}_i B_i(\tau). \tag{1}$$

The number of coefficients n depends on the basis function degree d and the number of spline knots m, that is, $n = m - d$. The coefficients $\boldsymbol{\alpha}_i$ can be points in the x, y plane. As discussed, the main reason that we adopt B-spline parametrization is the convex hull property: the spline always lies within the convex hull of the control polygon. This is explained visually in Fig. 1. Consequently, a constraint on the values of the $\boldsymbol{\alpha}_i$ coefficients would imply constraint on the amplitude of the spline function $s(\tau)$ for any τ. B-splines parametrization has been exploited for real-time optimal motion planning of robotics applications in order to

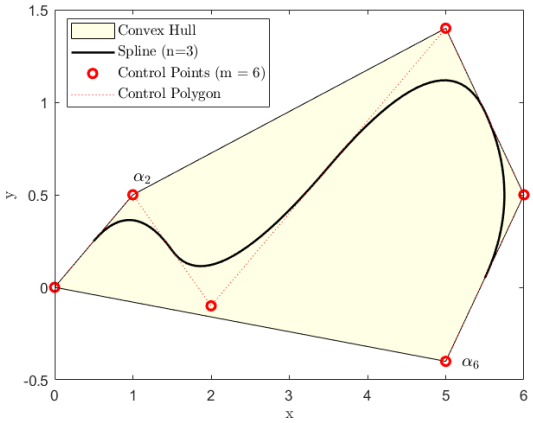

Fig. 1: An example of a spline of order 3 constrained inside its convex hull

enforce constraints at all times rather than conventional time-gridding constraints approach [12]. Note that this B-spline relaxations also introduce some conservatism coming from the distance between the control polygon and the spline.

Apart from imposing vehicle position constraints over the entire time horizon, we could also impose other kinematic limits such as velocity, acceleration and jerk constraints in the similar manner. It is because the derivative of a B-spline of degree d is a function of B-splines of degree $(d-1)$. As a result, the velocity bounds $v_{min} \leq \dot{s}(\tau) \leq v_{max}$ can be replaced by constraints on the spline coefficients of $\dot{s}(\tau)$.

C. Model predictive control (MPC)

MPC is a model-based control algorithm that optimizes the control action on a certain cost function while satisfying constraints on inputs and states [13]. The model is usually given as a state-space nonlinear/linear model. The optimization problem is solved at each time step and, according to the receding horizon principle, only the first computed control input is applied. A MPC problem can be written as following,

$$\begin{aligned} \underset{x(.),\, u(.)}{\text{minimize}} \quad & \sum_{k=0}^{N-1} C_k(x_k, u_k) + V_N(x_N) && (2)\\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k) && k = 0, \dots, N-1 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U} && k = 0, \dots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(t_0). \end{aligned}$$

The cost function $C_k(x_k, u_k)$ and constraints represent design objectives (trajectory tracking, stability) and may also indicate driving style preferences such as comfort or aggressive. MPC has been applied successfully in different industries such as chemical plants and smart building. And recently, thanks to the advancement in both theory and algorithms for solving optimization problems in real-time environment, the controller have been shown also capable for fast dynamics systems including autonomous driving. Still, it is hard to guarantee real-time optimal solution in all driving scenarios. In this work, MPC is investigated to be the expert driver, generating dataset for the imitation learning.

We apply the recent proposed nonlinear safety-critical MPC [10] to deal with safety state and input constraints. As it can be seen by the problem formulation, MPC does not only produces the optimal control input, but also the optimal trajectory of the states. Since we are not interested in learning the control inputs, we consider the MPC only as an optimal trajectory planner. The considered output is the state $x(t), \forall t \in [t_0, t_0 + H]$, where t_0 is the current time instant when we perform the optimization and $H_p \leq N$ is the time horizon for the trajectory that the learner predicts.

IV. Spline-Constrained Policy Network

A policy network aims at producing motions that can mimic the behavior of an expert. The inputs to the policy are the observations o_t, which can be only a part of the state $x(t)$ observed by the expert. The outputs a_t can be either controls or trajectories. As discussed in the introduction, we desire to prioritize safety and interpretability of the framework, hence consider using policy networks to generate state trajectories and let them be tracked by a model-based feedback control. The main characteristics of the studied policy network are the use of B-splines coefficients to parametrize the trajectory and the addition of barrier-function constraints in the backpropagation loss.

A. B-spline coefficients

To be comparable with the expert performance, the output trajectory of the policy network should be as smooth and safe as the expert one. Usually, policy networks have a certain number of vehicle poses as output nodes $a_t = \{(x_1, y_1), \dots, (x_i, y_i), \dots\}$ and an interpolation is performed between them to obtain the complete trajectory. This may show some drawbacks due to possible jerky motions, difficulty in scalability for longer horizons and the impossibility to impose constraints in between the predicted poses. For this reason, the outputs of the policy network have been chosen to be the coefficients of a spline written in B-form, with predefined knots:

$$a_t = \{\boldsymbol{\alpha}_{i=1:n}\}.$$

Using coefficients as outputs allows a forced smooth trajectory, imposition of constraints at each time instant just by limiting the coefficient values and few output nodes. In this way, it is more scalable for longer horizons.

B. Constraints

The expert behavior that the policy network has to mimic is made of individual characteristics but also of hard constraints, such as collision avoidance or lane boundary limits. As demonstrated by previous works [8], [11], the pure imitation loss given by the mean squared error between the expert and policy network outputs is not sufficient to obtain the compliance to constraints. Indeed, the MSE does not incorporate any information about safety, and in case the vehicle is near the boundaries of the constraints, the

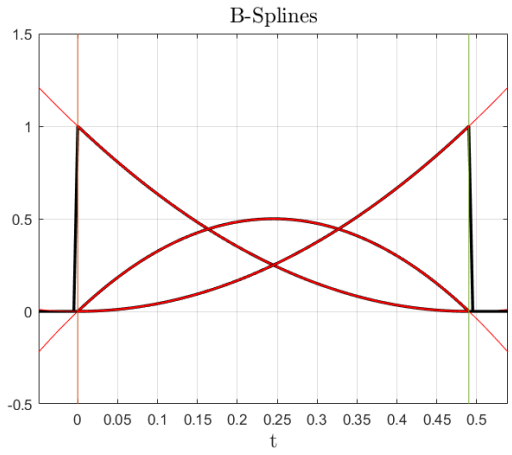

Fig. 2: B-Splines parametrizing the trajectory (0.5s)

policy network may violate them. To overcome this issue, we add constraints in the form of barrier functions to the loss function used in training. The pure imitation loss can be represented as $MSE = \frac{1}{dim(a_t)}\sqrt{(a_t - a_t^*)^2}$, while the constrained loss can be seen as the addition between the imitation loss and a barrier function, yielding

$$CL = MSE + I(o_t, a_t, a_t^*). \tag{3}$$

Note that the barrier function $I(o_t, a_t, a_t^*)$ can reflects not only safety conditions but also design objectives on velocity, acceleration and jerk states. In this way, the constrained policy network can incorporate constraints while imitating the high quality driving behaviour of the expert.

Combining B-splines and constrained loss, we can obtain a policy network providing safe and smooth trajectories with computational efficiency both in training and deployment. The barrier functions can be written as ReLu functions limiting the values of the coefficients according to collision avoidance or lane boundaries constraints and, thanks to the convex hull property, the overall trajectory will be contained inside the safe area.

V. Simulation and Data Generation Settings

In order to test the validity of the spline-constrained policy network, we implemented a lane keeping scenario in a simulated environment. In this scenario, we performed the DAgger algorithm using a high fidelity vehicle dynamics, realistic sensor data, a NMPC expert and a simple PID decoupled control.

The vehicle is represented by a 15DOF high-fidelity model implemented in Simcenter Amesim, including chassis, steering, braking, suspensions, and tyre dynamics model. This allows to easily get a fairly big amount of high quality training data in terms of physical accuracy. We suppose that all the vehicle states can be measured and the environment data from sensors models is obtained via Simcenter Prescan. More details on this simulation toolchain for autonomous driving testing are presented in [1].

The expert is made by a safety-critical nonlinear MPC producing an optimal trajectory based on a cost function that maximizes comfort while keeping the vehicle on its lane and with hard-constraints on the lane boundaries. The NMPC is implemented using CasADi with IPOPT solver [14]. The sampling time is $0.01s$. The low-level control to track the trajectory is realized by a PID controller for the longitudinal dynamics and a pure pursuit for the lateral dynamics. These controllers are simple for computational efficiency reasons.

The policy networks, both constrained and unconstrained, are implemented and trained using Keras. To ensure a fair comparison, all networks share common characteristics:

A. NN Architecture

Two hidden layer fully-connected architecture with 20 nodes in each hidden layer.

B. Inputs

The features shall take into account the road ahead and the states of the vehicle. The inputs are expressed as the current longitudinal velocity and the coordinates of the road boundaries (right and left) in the local reference frame of the vehicle, with origin in its center of mass and rotated according to the yaw angle. Defining H_o as the observation horizon:

$$\begin{aligned} o_t = \{&v_x(t), r_0^j + \Delta_i = (x_i^j, y_i^j)\}, \\ &\forall j = right, left; i = 0, \dots, H_o. \end{aligned} \tag{4}$$

The Δ_i intervals don't need to be all equal. Indeed, as noted in [5], the closest points matter more and so the intervals can increase along with i. Furthermore, if the speed has a high dynamics, the amplitude of the intervals shall be proportional to it. In our case, we considered 10 road points with fixed increasing intervals (small interval of possible speeds: $(10 \div 8.5)m/s$). The first road point corresponds to $\Delta_0 = 0$, and so the coordinates x_0^j for j = right,left are always null and neglected. The total number of features is 39.

C. Outputs

The networks output the coefficients of the B-splines fitting the trajectory. Since the trajectory is short ($0.5s$ at a maximum of $10m/s$) and the curvature is limited, a spline of order 3 is sufficient to describe it. The chosen knot vector is $T = (0, 0, 0, 0.49, 0.49, 0.49)$. The resulting three B-splines basis functions can be seen in Figure 2. Since the trajectory always starts from the origin of the current local reference frame, the coefficient multiplying the first B-spline is always null and can therefore be neglected. The total number of outputs is then 4: $a_t = \{\alpha_1^x, \alpha_1^y, \alpha_2^x, \alpha_2^y\}$. The described components are imported into the simulation toolchain implementation.

VI. Simulation Results

In this section, we compare different policy networks performance in a lane keeping scenario according to the settings defined previously in V. The metric that we consider to evaluate the performance of the learner is:

$$e = \frac{1}{T}\sum_{t=1}^{T} max(||s^*(t) - s(t)||_2) + L_{fail}, \tag{5}$$

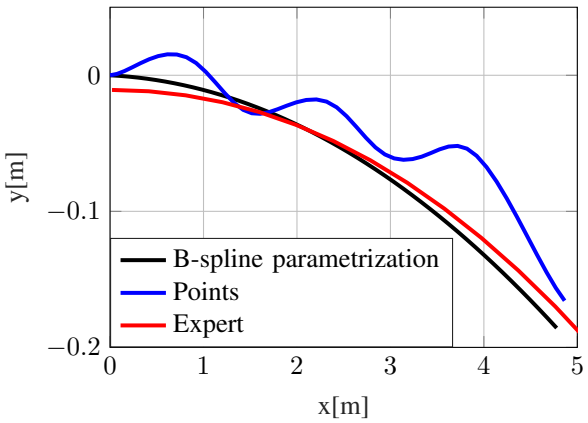

Fig. 3: Comparison between different choices of the policy outputs

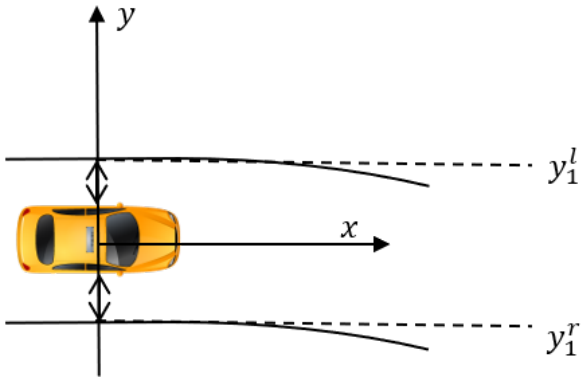

Fig. 4: Lane boundaries coordinates constraints used in the lane keeping scenario

i.e., the maximum euclidean distance between the points of the expert and learner trajectories, averaged on the time length of the policy rollout, summed to an additional loss in case the learner fails the lane keeping task, violating the hard constraints of the expert.

Firstly, we validate the use of B-splines parametrization comparing it to the trajectory points prediction, without the use of constraints in the training loss.In the case of predicted points, they are interpolated using cubic splines. As it can be seen from Figure 3, the trajectory obtained by the points prediction is sometimes jerky. It does not incorporate well the comfort parameters of the expert, leading to an uncomfortable driving. On the other hand, the B-splines based approach produces a smooth trajectory and similar to expert driving.

Secondly, we introduce barrier function constraints in the training loss of the networks. The choice of the barrier function is not unique and therefore influences the performance of the constrained policy network. The function must be coherent with the hard constraints of the expert:

$$y^r(k) + 0.45 \leq y(k) \leq y^l(k) - 0.45, \forall k = 0, \ldots, N-1,$$

and it should also take into account the conservatism introduced by the convex hull. Several functions have been tested and it must be noted that some of them even worsened the performance with respect to the unconstrained network. Here, we show the results for the following barrier function,

$$\begin{aligned} I(o_t, a_t) = \quad & C \cdot max(0, \alpha_1^y - y_1^l) + C \cdot max(0, \alpha_2^y - y_1^l) \\ & - C \cdot min(0, \alpha_1^y - y_1^r) - C \cdot min(0, \alpha_2^y - y_1^r), \end{aligned} \quad (6)$$

where $C = 1000$. The limits in the y-coordinate can be seen graphically in Figure 4. The function grows linearly with a high slope when the coefficients violate the lane boundaries.

In Figure 5, we can see the performance evolution of the unconstrained and constrained networks across the various policy rollouts of the DAgger algorithm. On the y-axis, it is represented the error of the learner, defined according to the metric (5). The unconstrained policy (UPN), at its first iteration, causes a crash and a subsequent interruption of the rollout. The constrained policy (CPN), instead, is already able to keep the car on the track and so it allows us to gather more data from the first rollout. After the second rollout, the constrained policy worsens its performance both in terms of mean and standard deviation, even leading to a crash after some iterations. From this, we can deduct that the theoretical guarantees of the DAgger algorithm may not hold in case of other loss functions other than the standard imitation ones (Mean Squared Error, L2...). To keep improving across the rollouts, an adaptive loss function may be used (ACPN): at each iteration the slope of the barrier functions is decreased. In this way, the adaptive constrained policy network is able to perform better than the UPN at almost every rollout. The difference between them is anyway really small, in the order of 10^{-3}. Figure 6 and Figure 7 demonstrate the euclidean error to the lane center and the vehicle trajectory, respectively, after the 10th rollout. The results show that the vehicle can follow the lane center using the proposed

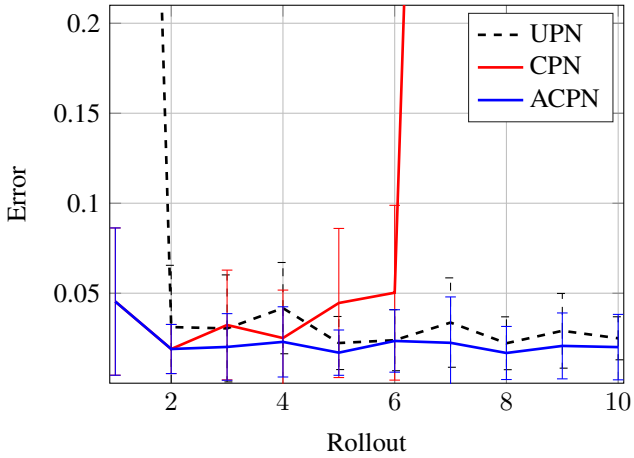

Fig. 5: Average maximum error between expert and learner: Error of the learner over ten DAgger policy rollouts, using Unconstrained (UPC), Constrained (CPN) and Adaptive Constrained (ACPN) Policy Networks

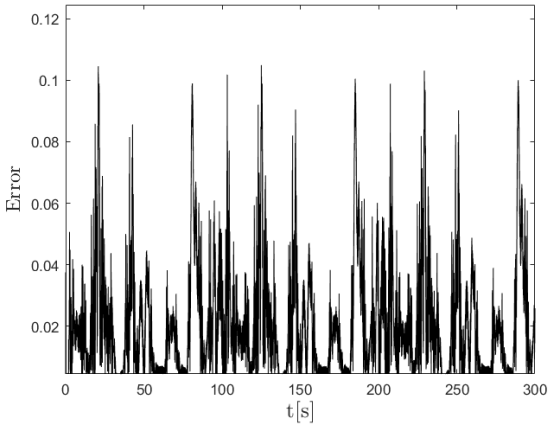

Fig. 6: Euclidean distance error between the ACPN and the expert on a 300s trajectory after the 10^{th} rollout

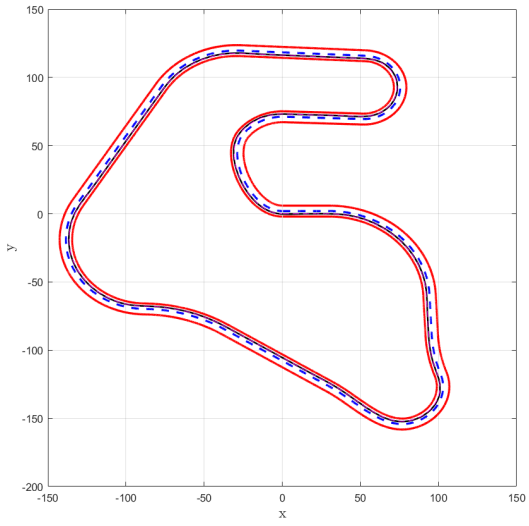

Fig. 7: (x, y) trajectory of the ACPN (black line) after the 10^{th} rollout

algorithm.

The main advantage of the CPN may be the incorporated knowledge of the task objective that allows the vehicle not to fail and its speed in reaching very good performance after just one dataset aggregation. This may be useful in case of more complex scenarios and larger datasets, where each policy rollout is expensive, or in case of a human-in-the-loop, where the vehicle is required not to fail its basic task.

VII. Conclusions and Further Work

The work proposes a framework for autonomous vehicle control relying on imitation learning-based planning and model-based control, and the algorithm is tested for a lane keeping application. The planner can be a shallow fully-connected neural network. It is shown how a parametrization of the planned trajectory can improve the prediction characteristics in terms of smoothness and scalability for longer horizons. This choice may show more improvements on the computational efficiency of the training phase for larger datasets. The results also show that the convergence speed of the learner during online training (following DAgger algorithm) can be enhanced by the use of constraints in the training loss, incorporating the task objectives in advance. However, a theoretical analysis of the DAgger guarantees in convergence with augmented losses should be studied more in depth. Furthermore, the influence of the additional losses definition on the performance requires furthur investigation. It would also be of interest to apply the proposed approach on human driving datasets and other driving scenarios.

Acknowledgments

This work has been conducted within the ITEA3 EMPHYSIS (Embedded systems with physical models in the production code software) and ConACon (Context Aware Control) projects. For both projects, the financial support from the Flemish Agency for Innovation and Entrepreneurship (VLAIO) is gratefully acknowledged.

References

[1] T. D. Son, A. Bhave, and H. V. der Auweraer, "Simulation-based testing framework for autonomous driving development," in *IEEE International Conference on Mechatronics*, Mar. 2019.

[2] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, pp. 305–313, 1989.

[3] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," in *Robotics: science and systems*, 2018.

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, 2016.

[5] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A fast integrated planning and control framework for autonomous driving via imitation learning," in *ASME 2018 Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, 2018.

[6] S. Shalev-Shwartz and A. Shashua, "On the sample complexity of end-to-end training vs. semantic abstraction training," *CoRR*, 2016.

[7] R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of ISO 26262: Machine learning and safety in automotive software," in *WCX World Congress Experience*, SAE International, April 2018.

[8] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

[10] T. D. Son and Q. Nguyen, "Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle," in *58th IEEE Conference on Decision and Control, Nice*, 2019.

[11] W. Zhan, J. Li, Y. Hu, and M. Tomizuka, "Safe and feasible motion generation for autonomous driving via constrained policy net," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 4588–4593, Oct 2017.

[12] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, vol. 26, pp. 2182–2189, Nov 2018.

[13] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2nd ed., 2017.

[14] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, In Press, 2018.