Proceedings of the ASME 2018
Dynamic Systems and Control Conference
DSCC2018
September 30-October 3, 2018, Atlanta, Georgia, USA

DSCC2018-9249

A FAST INTEGRATED PLANNING AND CONTROL FRAMEWORK FOR AUTONOMOUS DRIVING VIA IMITATION LEARNING

Liting Sun
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA, 94720
litingsun@berkeley.edu

Cheng Peng
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA, 94720
chengpeng2014@berkeley.edu

Wei Zhan
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA, 94720
wzhan@berkeley.edu

Masayoshi Tomizuka
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA, 94720
tomizuka@berkeley.edu

ABSTRACT

Safety and efficiency are two key elements for planning and control in autonomous driving. Theoretically, model-based optimization methods, such as Model Predictive Control (MPC), can provide such optimal driving policies. Their computational complexity, however, grows exponentially with horizon length and number of surrounding vehicles. This makes them impractical for real-time implementation, particularly when nonlinear models are considered. To enable a fast and approximately optimal driving policy, we propose a safe imitation framework, which contains two hierarchical layers. The first layer, defined as the policy layer, is represented by a neural network that imitates a long-term expert driving policy via imitation learning. The second layer, called the execution layer, is a short-term model-based optimal controller that tracks and further fine-tunes the reference trajectories proposed by the policy layer with guaranteed short-term collision avoidance. Moreover, to reduce the distribution mismatch between the training set and the real world, Dataset Aggregation is utilized so that the performance of the policy layer can be improved from iteration to iteration. Several highway driving scenarios are demonstrated in simulations, and the results show that the proposed framework can achieve similar performance as sophisticated long-term optimization approaches but with significantly improved computational efficiency.

INTRODUCTION

In recent years, autonomous driving has attracted a great amount of research efforts in both academia and industry for its potential benefits on safety, accessibility, and efficiency. Typically, autonomous driving systems are partitioned into hierarchical structures including perception, decision-making, motion planning and vehicle control (see [1, 2]). Among them, planning and control are two core challenging problems that are responsible for safety and efficiency. They should

1. comprehensively consider all possible constraints regarding safety and feasibility based on as much as possible the perceived environment information and reasonable prediction of other road participants' behaviors;
2. generate optimal/near-optimal maneuvers that provide good driving qualities such as smoothness, passengers' comfort and time-efficiency;
3. solve the problem within limited runtime to timely respond to rapid changes in surrounding environment.

Simultaneously satisfying the above requirements can be difficult and many planning and control approaches have been proposed [3]. They can be categorized into four groups: i) graph-search-based, such as A^*(see [4]), ii) sampling-based (e.g., RRT, RRT*, see [5, 6]), iii) interpolating-curve-based [7, 8], iv) optimization-based [9, 10], and v) learning-based approaches [11–13]. Groups i) and ii) are space-discrete in the sense that they discretize the state space into grids/lattices or sampled nodes and search for solutions that build feasible connections among them. Consequently, the resulting paths/trajectories are not continuous but jerky, and a subsequent smoother is necessary for realistic implementation. Approaches in iii) can mostly generate smooth paths, but it is hard to guarantee its optimality, not even locally. Besides, without temporal consideration, dealing with moving obstacles can be time-consuming using such approaches. Optimization-based approaches, on the other hand, formulate all possible cost functions and constraints in a uniform manner as constrained optimization problems with continuous state and action spaces. A popular example utilizes Model Predictive Control (MPC). At each time step, the optimal control actions are generated by solving a constrained optimization problem (in general highly nonlinear and non-convex). A practical issue is that the computational load grows exponentially with the horizon length and the number of obstacles, which makes it hard to be implemented in realtime, particularly when a long horizon length is preferred for persistent feasibility and safety. In order to guarantee the realtimeness, several methods have been proposed. For instance, Ziegler et. al [14] terminates the sequential optimization at pre-determined maximum runtime, and in [15, 16] the unified optimization problem is decomposed into two or more sub-problems with linearized vehicle models and constraints. Both methods sacrifice optimality for computational efficiency.

On the other hand, "End-to-End" learning approaches in group v) are increasingly attracting attentions in recent years due to their powerful representations of the environment and fast forward computation in test phases. Typically, such learning-based approaches take raw images as inputs, and directly output the driving actions or a driving policy net by training a deep neural network via supervised learning [12] or deep reinforcement learning [17, 18]. As pointed by [19], however, the outputs of such "End-to-End" networks are hard to yield any guarantees for feasibility, safety and smoothness. Also, such "End-to-End" structure is hard to incorporate *a priori* knowledge on vehicle models, which makes their training not only data-hungry, but also time-consuming, i.e., hours/days of training time on multiple GPUs. This makes the "End-to-End" learning system hard to adapt to new situations. This can be dangerous in practice since real-time deviations from the training set/simulation environment can lead to unpredictable error propagation and cause severe consequences if the training set/simulation environment cannot be augmented by realistic test results.

In this paper, we propose a hierarchical structure that combines the advantages of both learning- and optimization-based methods via safe imitation mechanism. The framework is consist of two layers. The first layer, defined as the ***policy layer***, is represented by a small-size fully-connected neural network trained via imitation learning. The inputs of the network is a set of highly representative features that describe the environment information, and the output is an instructive trajectory imitating a long-term expert driving policy. The second layer, called the ***execution layer***, is formulated as a short-horizon optimal controller that tracks and further fine-tunes the instructive trajectories from the ***policy layer*** to guaranteed short-term obstacle avoidance. *Our key point is to replace traditional time-consuming long-horizon online optimizations with offline training and efficient online generation and adjustment*. In such sense, not only the long-term optimality is preserved, but also *a priori* knowledge on vehicle models can be fully utilized for better tracking and control performance. Moreover, via the selected set of representative features in ***policy layer***, the network structure and size are maintained simple and small, which allows us to quickly improve the learning performance via Data Aggregation (DAgger) from online test data.

PRELIMINARIES

Overview of the Hierarchical Structure

Figure 1 shows the overview of the proposed planning and control framework for autonomous driving. It consists of three hierarchical modules: perception, decision-making, and planning and control. At each time step, the perception module detects the surrounding environment via on-board sensors (e.g., GPS, LiDAR), and yields measurements/estimates of all necessary states of the ego vehicle such as in-map location, orientation, velocity and its relative positions and velocities to all other visible road participates (static or moving). Based on the perception results and pre-defined driving tasks, the decision-making module will set the reference lane to instruct the next-level planning and control.

In this paper, we focus on the planning and control module. To satisfy all the three requirements discussed above, the planning and control module consists of two layers in test phase[2]: the ***policy layer*** and the ***execution layer***. First, based on the driving decisions (e.g., target lane) and the perception results, a set of highly representative features are extracted as inputs of the ***policy layer*** that is trained to yield trajectories imitating that of a long-term expert driving policy (for instance, such a policy can be directly obtained via long-term MPC or sampling approaches such as RRT*). Then, with the imitated reference trajectory, the ***execution layer*** computes the optimal control actions (steering

[2]Note that in the training phase, no execution layer is need and the training of the policy layer is a typical behavior cloning process.

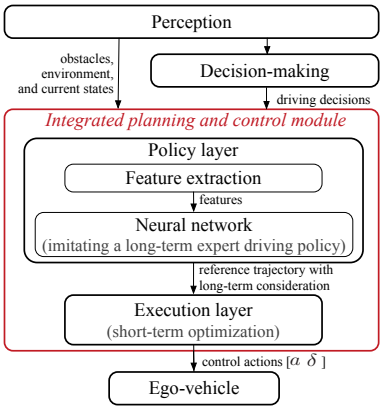

FIGURE 1. The overall hierarchical structure

angles and accelerations) that considers model constraints, feasibility and short-term obstacle avoidance so that the ego vehicle can be safely driven. To assure that the planning and control module is environmentally responsive, we adopt the re-planning scheme, i.e., at each time step, only the first control actions from the execution layer are sent to the actuators to drive the ego vehicle.

Short-Term Optimal Controller in Execution Layer

We formulate the short-term safe trajectory tracking and adjustment process in the ***execution layer*** as a MPC controller that 1) tracks as much as possible the trajectories generated by the policy layer and 2) makes safe adjustments when necessary to guarantee short-term collision avoidance.

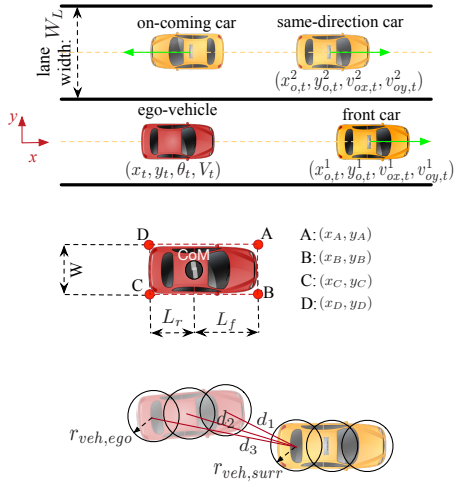

FIGURE 2. An example driving scenario for short-term MPC

Take the driving scenario in Fig. 2 as an example. Assume that at each time instant t, states of the ego-vehicle are represented by $z_t=[x_t, y_t, \theta_t, V_t]^T$, where (x_t, y_t) is current car position, θ_t is the yaw angle and V_t is the speed. Similarly, states of all surrounding cars are denoted by $(x^i_{o,t}, y^i_{o,t}, v^i_{ox,t}, v^i_{oy,t})$ $(i=1,2,\cdots,n)$. Control actions of the ego vehicle include the longitudinal acceleration and steering angle, i.e., $u_t=[a_t, \delta_t]^T$. Define the horizon length in execution layer as N_e, and at time t, the within-horizon predicted ego-vehicle's states and control variables are respectively $z^p_{t,k+1}=[x^p_{t,k+1}, y^p_{t,k+1}, \theta^p_{t,k+1}, V^p_{t,k+1}]^T$ and $u^p_{t,k}=[a^p_{t,k}, \delta^p_{t,k}]^T$ with $k=0,1,2,\cdots,N_e-1$, and the within-horizon surrounding cars' states are denoted by $(x^{p,i}_{o,t,k+1}, y^{p,i}_{o,t,k+1}, v^{p,i}_{ox,t,k+1}, v^{p,i}_{oy,t,k+1})$ for $i=1,2\cdots,n$.

Remark 1: The prediction of the within-horizon states of all surrounding cars is never trivial work and goes beyond the focus of this paper. Typically, they can be predicted using simple rule-based behavior models (e.g., zero-input or constant velocity assumptions) or advanced prediction algorithms such as deep learning. In this paper, we assume that the prediction is available.

Objective Function Suppose that the first N_e trajectory points given by the policy layer are $[(x^r_{t,1}, y^r_{t,1}), (x^r_{t,2}, y^r_{t,2}), \cdots, (x^r_{t,N_e}, y^r_{t,N_e})]$, then a quadratic cost function is defined as

$$J^e_t = \sum_{k=1}^{N_e} \begin{bmatrix} x^e_{t,k} \\ y^e_{t,k} \end{bmatrix}^T W_e \begin{bmatrix} x^e_{t,k} \\ y^e_{t,k} \end{bmatrix} + u^T_{t,k} W_u u_{t,k} + \triangle u^T_{t,k} W_{\sigma u} \triangle u_{t,k} \quad (1)$$

where $[x^e_{t,k}, y^e_{t,k}] \triangleq [x^p_{t,k} - x^r_{t,k}, y^p_{t,k} - y^r_{t,k}]$ represents the tracking error and $\triangle u_{t,k} \triangleq [a_{t,k} - a_{t,k-1}, \delta_{t,k} - \delta_{t,k-1}]$ is the derivatives of the control inputs. $W_e \in \mathscr{R}^{2\times 2}$, $W_u \in \mathscr{R}^{2\times 2}$ and $W_{\sigma u} \in \mathscr{R}^{2\times 2}$ are all positive definite matrices that tune the weights of tracking error, control effort and comfort level of the passengers.

Constraints The short-term tracking problem comprehensively considers all constraints from the system kinematics feasibility (nonlinear equality constraints), dynamics feasibility and collision-free safety (inequality constraints).

Kinematics feasibility: Bicycle model [20] is adopted in this paper to express the kinematic model of the ego vehicle, i.e.,

for k=$0,1,\cdots,N_e-1$ within the preview horizon:

$$x^p_{t,k+1} = x^p_{t,k}+V^p_{t,k}\cos(\theta^p_{t,k}+\tan^{-1}\frac{L_r\tan\delta^p_{t,k}}{L})dt \quad (2)$$

$$y^p_{t,k+1} = y^p_{t,k}+V^p_{t,k}\sin(\theta^p_{t,k}+\tan^{-1}\frac{L_r\tan\delta^p_{t,k}}{L})dt \quad (3)$$

$$\theta^p_{t,k+1} = \theta^p_{t,k}+\frac{V^p_k\tan\delta^p_{t,k}}{L}\cos(\tan^{-1}\frac{L_r\tan\delta^p_{t,k}}{L})dt \quad (4)$$

$$V^p_{t,k+1} = V^p_{t,k}+a^p_{t,k}dt \quad (5)$$

and $x^p_{t,0}$=$x_t, y^p_{t,0}$=$y_t, \theta^p_{t,0}$=$\theta_t, V^p_{t,0}$=V_t. L=L_r+L_f is the total car length as shown in Fig. 2 and dt is the discrete time interval.

Dynamics feasibility: The dynamics feasibility is guaranteed via constraints from the G-G diagram, as shown in Fig. 3, where $a^+_{max} < a^-_{max}$ represent, respectively, the maximum acceleration input and deceleration input. Therefore, the feasible region (shadow part in Fig. 3) can be expressed in terms of decision variables as:

$$\left(\frac{x^p_{t,k+2}-2x^p_{t,k+1}+x^p_{t,k}}{dt^2}\right)^2 + \left(\frac{y^p_{t,k+2}-2y^p_{t,k+1}+y^p_{t,k}}{dt^2}\right)^2 \leq (a^-_{max})^2 \quad (6)$$

$$a^p_{t,k} \triangleq a^p_{lon,t,k} \leq a^+_{max} \quad (7)$$

Safety constraints: Safety is assured by constraining the ego-vehicle's configurations (position and orientation) and its distances to surrounding cars. As shown in Fig. 2, we decompose each of the vehicle into m circles of radius r_{veh}, which are laid out equidistantly along its longitudinal axis as explained in [14]. To assert collision free, we require that the distances from the ego vehicle's circles to all surrounding vehicles' circles to be larger than $r_{veh,ego}+r_{veh,surr}+\varepsilon$ where ε is a pre-determined safety buffer distance between two cars. Mathematically, for n surrounding vehicles, this brings in $N_e nm^2$ safety constraints

$$d^{i,j}_{t,k} \geq r^j_{veh,ego}+r^{i,j}_{veh,surr}+\varepsilon, \quad (8)$$
$$i=1,2,\cdots,n, j=1,2,\cdots,m^2, k=0,1,\cdots,N_e-1.$$

Moreover, other safety constraints to ensure that the ego vehicles does not hit the curb are also considered. By limiting the corner points (A,B,C,D in Fig.2) of the ego vehicle within the road boundaries, another $2N_e$ nonlinear inequalities are generated. Hence, the total number of safety constraints is $N_e nm^2+2N_e$.

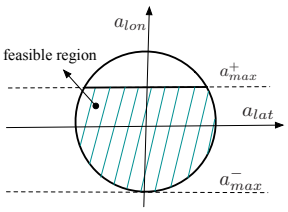

FIGURE 3. Acceleration boundaries from G-G diagram

In summary, the short-term time-varying optimal control problem in the execution layer at each time instant t is given by:

$$P_t := \min_{(z^p_{t,0},\cdots,z^p_{t,N_e},u^p_{t,0},\cdots,u^p_{N_e-1})} J^e_t \quad (9)$$
$$s.t \quad z^p_{t,k} \in \mathscr{Z}, \forall k=0,1,\cdots,N_e$$
$$u^p_{t,k} \in \mathscr{U}, \forall k=0,1,\cdots,N_e-1$$
$$\text{constraints Eqns. (2)-(8)}$$

where $\mathscr{Z}$ and $\mathscr{U}$ are the bounded set of the ego vehicle's states and control variables. By solving this optimization problem online, we can get the optimal action sequence $u^p_{t,0},\cdots,u^p_{N_e-1}$ at time t, execute the first action $u^p_{t,0}$ and re-plan at time $t+1$ in a manner of receding horizon control.

Note that commonly, a longer horizon is preferred to generate safe and comfort driving policies. The computational load of the optimization problem in Eqn. (9), however, grows exponentially as the preview horizon N_e increases. Hence, in order to keep N_e small while preserving good performance, we propose to use the policy layer to generate guidance trajectories considering long-term planning performance.

IMITATION LEARNING OF A LONG-TERM EXPERT DRIVING POLICY

As discussed above, as the preview horizon and the number of surrounding obstacles increase, the computational loads of direct optimization-based planning approaches increase exponentially, which makes them impractical in real-time online applications. In fact, even solving the safe tracking problem in Eqn. (9) with a quadratic cost function is not easy in realtime when N_e is fairly large, not to mention an optimization-based planning problem with more complicated cost function that considers all necessary driving qualities. To address such time efficiency challenge while maintaining its optimality (to some extent), we proposed to design a policy layer to imitate an expert long-term driving policy as a fast online reference trajectory generator, and let the execution layer safely track the reference trajectory. Note that such an expert policy can be obtained via either direct optimization or other approaches (e.g., rapidly-exploring random tree (RRT) and

RRT*) offline. In this section, details with respect to the imitation learning process will be discussed.

Feature Extraction

For an efficient and compact network in the policy layer, feature selection is of key importance. The features we selected can be divided into three groups at each time instant t:

on-map motion features: ego-vehicle's current location and speed within the abstract map settings;
goal features: ego-vehicle's lateral distance to the target lane set by the decision-making module at current time instant;
safety features: ego-vehicle's current relative positions, orientations and velocities with respect to surrounding obstacles/cars.

On-Map Motion Features To effectively describe the *on-map motion features* on curvy roads, we re-express all the coordinates of vehicles in the Frenet Frame, as shown in Fig. 4. The position (x_t, y_t) at time t is translated to (s_t, d_t) where s_t is the longitudinal travelled distance along the road and d_t is the lateral deviation from the lane center.

First the lateral distance of the ego-vehicle to curbs are given, namely, $f_t^{map,1} \triangleq d_{t,curb}^{+} - d_t$ and $f_t^{map,2} \triangleq d_t - d_{t,curb}^{-}$. Furthermore, we spatially discretize the proximate reference lane center, the coordinates of which are represented by $r_{unit} \triangleq [x_1^c, y_1^c, x_2^c, y_2^c, \cdots, x_j^c, y_j^c, \cdots, x_m^c, y_m^c]$. Then the current deviations of the ego-vehicle from the reference lane center can be represented by $d_t^y \triangleq [d_t^{y,1}, d_t^{y,,2}, \cdots, d_t^{y,j}, \cdots, d_t^{y,m}]$, where each element can be calculated by

$$d_t^{y,j} = FF(y_t + (x_j^c - x_t)\tan\theta_t - y_j^c), \forall j = 1, 2, \cdots, m. \quad (10)$$

where $FF(\cdot)$ represents the function to convert from $x-y$ coordinate to Frenet Frame coordinate.

In this work, m=10 is set, i.e., ten on-map-motion-related features $f_t^{map,j+2} \triangleq d_y^{t,j}$ for j=1,2,⋯,10 are generated.

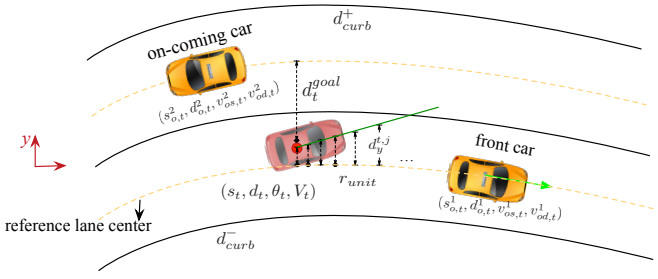

FIGURE 4. Definitions for feature selection

As for the speed-related feature, it should indicate the current ego-vehicle's speed V_t as well as its margin to the pre-defined speed limit. Hence, we define $f_t^{map,13} \triangleq V_t^{mar} = V_t - V_{max}$.

Remark 2: r_{unit} is not necessarily evenly sampled in distance. Actually, noting the fact that closer proximity matters more, r_{unit} can be sampled spatially with increasing intervals. Such configuration allows r_{unit} to cover a longer distance, which helps improve the features' sensitivity to small yaw angle changes.

Goal Features Given the current goal position d_t^{goal}, as shown in Fig. 4, the goal feature is defined as

$$f_t^{goal} \triangleq d_t^{goal}. \quad (11)$$

Safety Features Considering the states of all surrounding vehicles, the following features are extracted, for i=1,2,⋯,m,

$$f_{t,i}^{safety} = \left[s_t - s_{o,t}^i, d_t - d_{o,t}^i, v_{t,s} - v_{os,t}^i, v_{t,d} - v_{od,t}^i\right]^T \quad (12)$$

where $v_{t,s}$ and $v_{t,d}$ are, respectively, the longitudinal velocity and the lateral velocity, and similarly, $v_{os,t}^i$ and $v_{od,t}^i$ are the velocities for the i-th surrounding vehicle. For m surrounding vehicles, $4m$ features are generated.

All above defined features f_t^{map}, f_t^{goal} and f_t^{safety} generate a set of highly representative features $f_t = [(f_t^{map})^T, f_t^{goal}, (f_t^{safety})^T]^T$ and will be used as inputs of the neural network in the policy layer.

Efficient Proximate Learning

To obtain an efficient neural network in the policy layer, we recall the first-action-only principle in receding horizon control. At each time step t, although a sequence of trajectories/control actions are generated, only the first one will be applied to drive the system. Similar concept can be adopted in the imitation process. Suppose that the length of the long-term trajectory given by an expert policy is N, it makes more practical sense if the execution layer can track the first several reference points than the later ones. As a matter of fact, if the execution layer can track the first several reference points precisely, the influence of the remaining ones are ignorable. Hence, instead of imitating all N-length reference trajectories given by an expert policy, we introduce the concept of "proximate learning" in the policy layer. Namely, as shown in Fig. 5, the training loss is defined as a weighted sum of Euclidean distances between the network outputs and the refer-

ence expert trajectory only within the proximate area:

$$l = \sum_{[s_k^{net}, d_k^{net}]}^{\text{proximate area}} W_d \left(d_k^{net} - d_k^{ref}\right)^2 + W_s \left(s_k^{net} - s_k^{ref}\right)^2 \qquad (13)$$

where (s_k^{ref}, d_k^{ref}) represents the expert trajectory in the Frenet Frame and (s_k^{net}, d_k^{net}) is the output of the network in the policy layer, which is later converted into (x_t, y_t) coordinate for the execution layer to safely track. In this paper, we set the proximate areas to contain 10 sequential points, i.e., equal to the horizon length N_e in the execution layer.

Such configuration helps reduce the number of output nodes in the output layer of the network and consequently the network size, so that training time can be significantly reduced, which allows fast adaptation of the system via online DAgger, as discussed in Section -C. Moreover, it also reduces the horizon length for the following execution layer, i.e., reducing the overall computational load of the proposed framework.

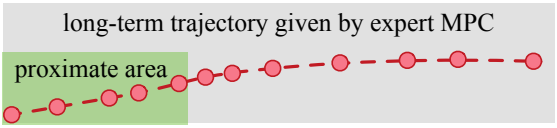

FIGURE 5. Illustration of the proximate learning concept

Learning via Sampled-DAgger Process

It is well known that, behaviour cloning itself is often not enough since it is never practical to generate a training set that satisfies the same states distribution as the real-world test data. Such distribution mismatch may make the learned policy biased and cause severe problems during execution, see, e.g. [21]. Once the encountered states slightly deviate from the training set (i.e., unfamilar features), the learned policy might yield wrong output, and such error will propogate and eventually fails the system.

One effective way to solve this is to use DAgger [21]. As an iterative algorithm, it adds all on-policy data into the training set so that, over iterations, the training set can cover most scenarios that the learned policy might encounter based on previous experience.

Motivated by this, a customized DAgger, defined as Sampled-DAgger, is proposed to improve the performance of the policy layer. As shown in Fig. 6, the process of the Sampled-DAgger are given as follows:

1. Gather an initial training set $\mathscr{D}_0$ by running a long-term expert planning algorithm π_{expert} for randomly generated scenarios in simulation and train the network to yield an initial policy π_0.
2. Run the autonomous driving architecture shown in Fig. 1 with the learned policy π_0 in policy layer. Meanwhile, run π_{expert} in parallel at a slow rate to periodically label the features with expert outputs. As shown in Fig. 6, π_{expert} is running every M time steps, i.e., $T_{interval}^{expert}=Mdt$, which is set to be long enough to find the solution given by π_{expert}.
3. Compare the proximate loss (i.e., Eqn. (13)) of current policy $\pi_0(f_{t=kT_{interval}^{expert}})$ w.r.t the expert policy π_{expert}. If the normalized loss is larger than the pre-defined safety criterion, label the features $f_{t=kT_{interval}^{expert}}$ with corresponding expert outputs and push them into a new training set $\mathscr{D}'$. Once the size of $\mathscr{D}'$ reaches a pre-defined threshold, go to step 4).
4. Aggregate previous training set with $\mathscr{D}'$, i.e., $\mathscr{D} = \mathscr{D} \cup \mathscr{D}'$ and re-train the network to yield a new policy π_{new} and set $\pi_0 = \pi_{new}$.
5. Repeat 2) to 4) until the performance of the learned policy π_0 achieves similar performance as π_{expert}.

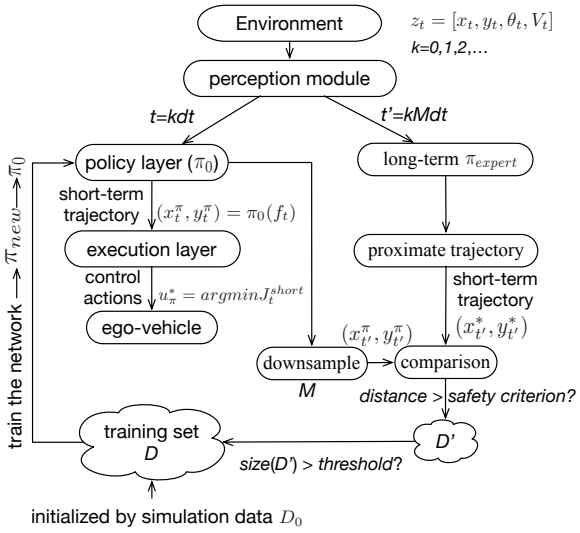

FIGURE 6. The Sampled-DAgger procedure

ILLUSTRATIVE EXAMPLES IN HIGH-WAY DRIVING

In this section, the proposed efficient planning and control framework is applied on a simple high-way driving scenario for verification. We consider two typical high-way driving behaviors with two surrounding vehicles ($m = 2$ with one front car and another adjacent-lane car): 1) overtaking and 2) car following.

Simulation Settings The simulation environment is established based on a 1/10 scale ratio controlled (RC) car with a sampling period of dt=0.1s. The speed limit is set as 1 m/s for the RC car. Regarding to the network in the policy layer, we utilize a fully-connected neural network with configurations given in Tab. I. For the safe execution layer, a horizon length of N_e=10 is selected for solving the constrained optimization problem in Eqn. (9). More detailed parameters for the simulation environment can be found in Tab. II.

All simulations are performed using Julia on a Macbook Pro (2.5 GHz Intel Core i7) using the standard Ipopt solver.

TABLE 1. Parameters in the simulation environment

layers	number of nodes	activation function
Input layer	22	sigmoid
hidden layer 1	50	sigmoid
hidden layer 2	50	sigmoid
output layer 2	20	none

TABLE 2. Parameters in the simulation environment

W_L	L_r	L_f	W	ε
0.38 (m)	0.19 (m)	0.21 (m)	0.19 (m)	0.02 (m)
N	N_e	V_{max}	a^+_{max}	a^-_{max}
30	10	1.0 (m/s)	0.5 (m/s^2)	-1.0 (m/s^2)

Training Dataset We adopt a long-term MPC-based planning strategy to collect the training data with a preview horizon of N=30. Initially, the size of the training dataset $\mathscr{D}_0$ is 20k. For each iteration in the DAgger process, we run the trained policy with randomized vehicle states settings for 100 roll-outs/trajectories, where each rollout/trajectory period is 10s (i.e., 1000 points for dt=0.1s).

Performance for Overtaking Maneuver

Figure 7 and Fig. 8 show the results of an overtaking maneuver with the proposed hierarchical structure and a short-term MPC only. In this scenario, the ego vehicle (red) should bypass the slowly moving front car (blue) when the left lane is clear (left-lane car is also represented by blue rectangles). It can be seen that, in Fig. 7, the policy layer effectively imitated the expert long-term planning which enables the ego vehicle to accelerate and turn left at an early stage for a smoother and safe overtaking trajectory. When the yaw angle of the ego vehicle is relatively large, the policy layer brakes a bit to avoid possible collision with curbs. On the contrary, without the policy layer, the short-term MPC in Fig. 8 behaves "blindly" due to a too-short planning horizon: acceleration without long-prepared turning, sharp brakes and large steering angles when it is too close to the front car and finally repeated curb hitting and stopping.

Performance for Car Following

In car-following scenarios, no overtaking is allowed (this is set by the decision-making module considering environment settings, for instance, not enough safe margin for overtaking or adjacent lanes are blocked). To utilize the same policy layer as in overtaking maneuveur, a virtual adjacent-lane car is introduced in the feature extraction part.

The results are shown in Fig. 9, where the red, blue and yellow rectangles represent the ego-vehicle, the front car and the virtual car on adjacent lane, respectively. The blue bars are the rear positions of the front car. The corresponding velocity profiles show that with the proposed hierarchical structure, the ego vehicle can decelerate in-time and perform car-following maneuver with the end speed equal to that of the front car. On the other hand, with a short-term MPC, the ego vehicle collides with the front car due to its short preview horizon, even a sharp brake is attempted in the end.

Imitation Performance Improvement with DAgger

In this section, the imitation learning performance with DAgger is presented. As mentioned above, the initial size of the training dataset is 20k, and in each iteration 10k on-policy data is generated. To evaluate the imitation performance, we adopt two emergent metrics: the collision rate (the fraction of trajectories where the ego vehicle intersects with another road participate) and the hard brake rate (the frequency at which the ego vehicle brakes harder than $0.9a^-_{max}$). Results are shown in Fig. 10, where it can seen that both collision and hard brake rates significantly reduce as the DAgger iteration increases. An example of performance improvement via DAgger is also presented in Fig. 11, where part (a) represents the on-policy running results with the initial policy π_0 on training set D_0. Due to the data distribution mismatch between the training set and test environment, with π_0, the policy layer cannot yield correct on-policy reference trajectory. As a result, the ego vehicle fails to perform a smooth lane-keeping maneuver. On the other hand, part (b) shows the results after the DAgger process converges. It can be seen that with more on-policy running data aggregated into the training set D, the performance of the policy layer is effectively improved so that the ego vehicle can smoothly go straight as fast as possible when it is safe.

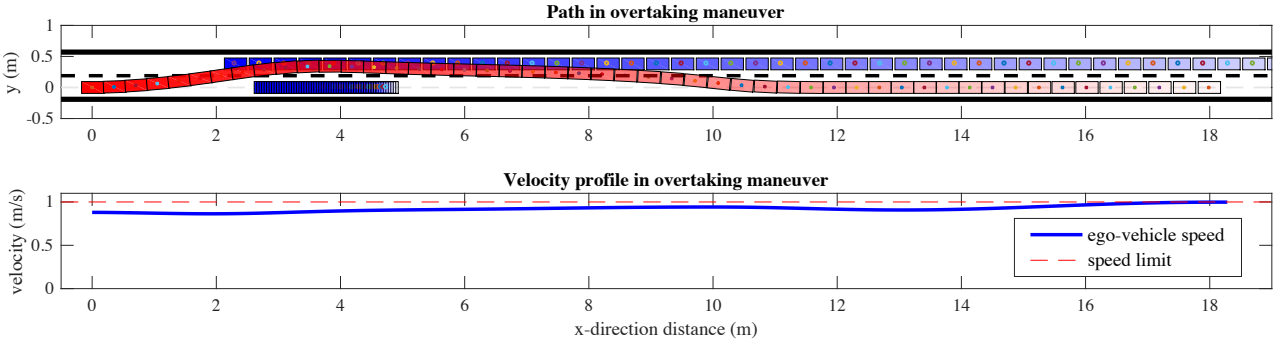

FIGURE 7. Overtaking maneuver using the proposed planning and control framework

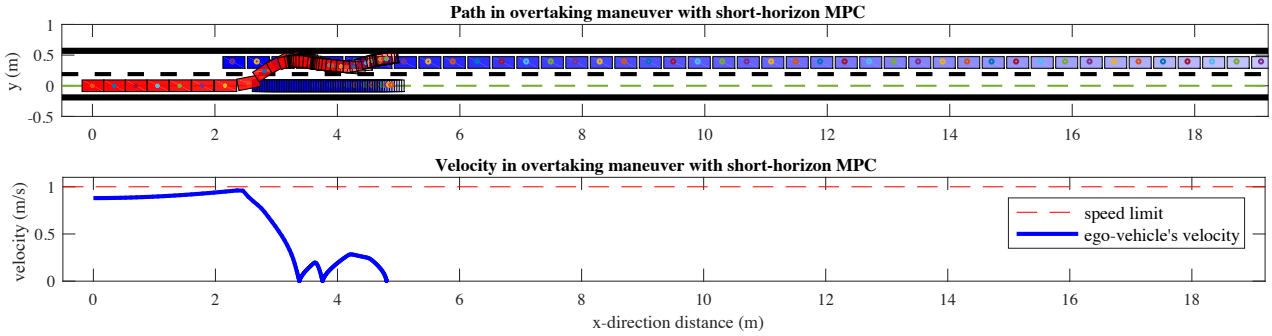

FIGURE 8. Overtaking maneuver using only a short-horizon MPC without the learned policy

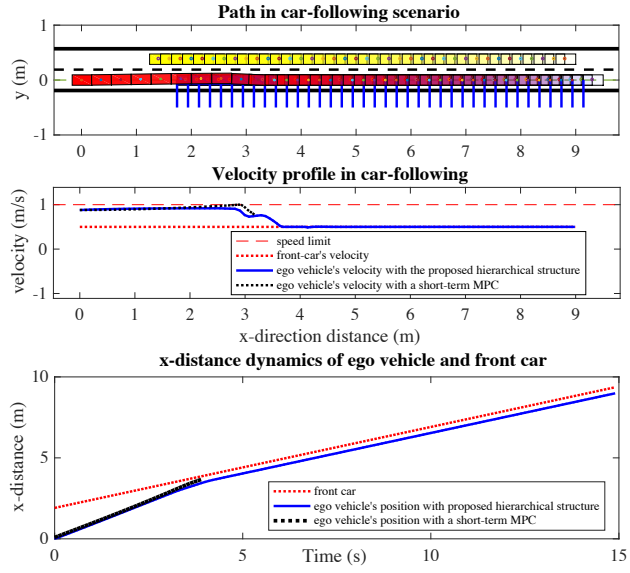

FIGURE 9. Car following Performance with the proposed framework

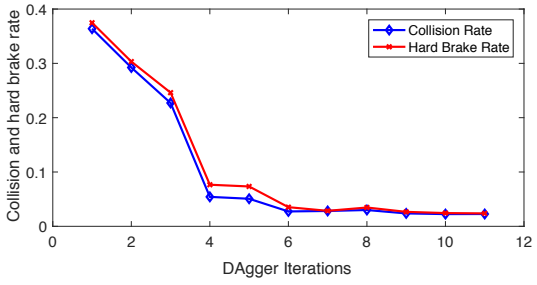

FIGURE 10. Improvement of Collision and Hard Brake Rate

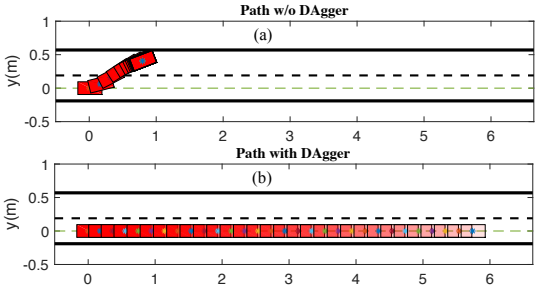

FIGURE 11. Learned policy improvement with customized DAgger

Comparison in terms of Realtimeness and Optimality

We also compare the online running time for both the proposed hierarchical structure and the baseline expert MPC-based

long-term planning policy. With 10 rollouts (i.e., 10k planning events), the worst-case runtime for the expert MPC takes 2.5487s, while the policy layer and the safe execution layer takes 6.42×10^{-6}s and 0.0766s, respectively. This means that the proposed hierarchical structure allows us to do planning with a frequency of 10Hz. Even if we try to optimize the problem formulation in the expert MPC policy by constraints approximation, as addressed in [14], a maximum of 2Hz planning is allowed. Therefore, in term of efficiency, the proposed hierarchical structure significantly reduced the runtime so that real-time requirement can be satisfied.

Moreover, we have also compared the performance of the proposed framework with that of the long-term expert MPC policy for trajectories in different homotopy. As shown in Fig. 12 and 13, five example trajectories are given: lane change, successful and unsuccessful overtake, car following and emergency brake (note that the vehicles are not shown in this plot to make the trajectories clear). We can see that the policy layer and the safe execution layer can achieve similar performance as the long-term MPC policy (the Euclidean distance between the two sets of trajectories are small as shown in Fig. 13). The proposed framework, however, is much more efficient for online planning since the the forward computation of policy layer is very fast, and the formulated optimization problem in the execution layer is small due to a short preview horizon.

CONCLUSIONS

In this paper, a fast integrated planning and control framework for autonomous driving was proposed based on a combination of imitation learning and optimization. Using a hierarchical structure, the framework can efficiently generate a long-term planning trajectory via imitation, and execute the control actions with guaranteed short-term collision avoidance. Moreover, a sampled-DAgger procedure was also introduced for good imitation performance. Two high-way driving examples are demonstrated in simulations as verification, where the results showed that via DAgger process, the proposed framework can effectively generate safe and efficient planning for real-time applications.

The structure of the policy layer can be further extended to cover more complicated driving scenarios within this framework. For example, instead of manual feature extraction, autoencoder techniques can be utilized to automatically generate highly representative features. Also, the output of the policy layer can be distributions of possible trajectories instead of the optimal ones. All these interesting directions can be explored in future studies.

REFERENCES

[1] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. Springer Science & Business Media, 2007, vol. 36.

[2] ——, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

[3] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[4] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[5] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[6] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7681–7687.

[7] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 167–172, 2010.

[8] Z. Liang, G. Zheng, and J. Li, "Automatic parking path optimization based on bezier curve fitting," in *Automation and Logistics (ICAL), 2012 IEEE International Conference on*. IEEE, 2012, pp. 583–587.

[9] D. Madås, M. Nosratinia, M. Keshavarz, P. Sundström, R. Philippsen, A. Eidehall, and K.-M. Dahlén, "On path planning methods for automotive collision avoidance," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 931–937.

[10] X. Li, Z. Sun, Q. Zhu, and D. Liu, "A unified approach to local trajectory planning and control for autonomous driving along a reference path," in *Mechatronics and Automation (ICMA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1716–1721.

[11] D. A. Pomerleau, "Alvinn, an autonomous land vehicle in a neural network," Carnegie Mellon University, Computer Science Department, Tech. Rep., 1989.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[13] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 204–211.

[14] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha - a local, continuous method," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 450–457.

[15] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss,

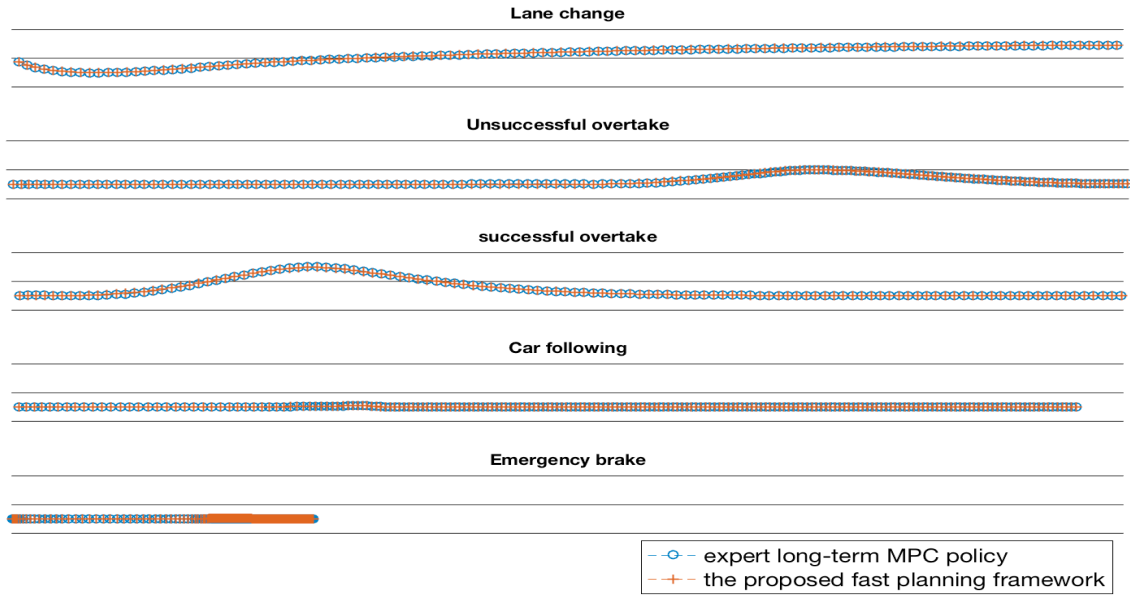

FIGURE 12. Performance comparison of the proposed fast planning framework with long-term expert MPC planner on test trajectories with different homotopy

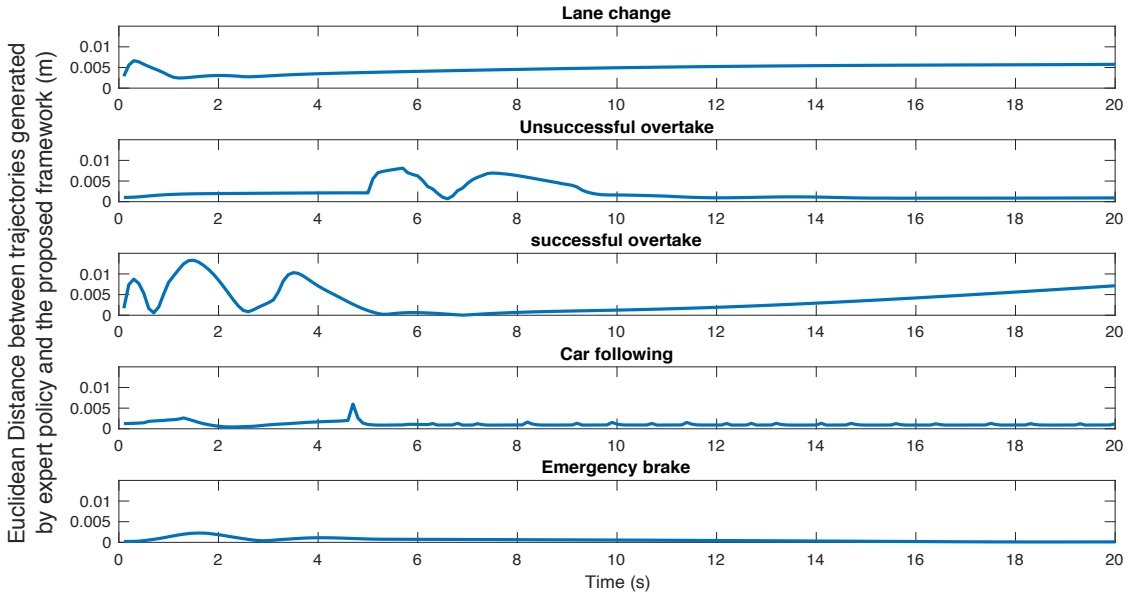

FIGURE 13. Euclidean distance between the proposed fast planning framework with long-term expert MPC planner on test trajectories with different homotopy

C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[16] W. Zhan, J. Chen, C.-Y. Chan, C. Liu, and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 632–639.

[17] C. Wu, A. Kreidieh, E. Vinitsky, and A. M. Bayen, "Emergent behaviors in mixed-autonomy traffic," in *Conference on Robot Learning*, 2017, pp. 398–407.

[18] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3389–3396.

[19] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[20] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[21] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning." in *AISTATS*, vol. 1, no. 2, 2011, p. 6.