Lane-Change in Dense Traffic with Model Predictive Control and Neural Networks

Sangjae Bae, *Member, IEEE,* David Isele, *Member, IEEE,* Alireza Nakhaei, Peng Xu, *Member, IEEE,*
Alexandre Miranda Añon, Chiho Choi, Kikuo Fujimura, *Member, IEEE,* Scott Moura, *Member, IEEE,*

Abstract—This paper presents an online smooth-path lane-change control framework. We focus on dense traffic where inter-vehicle space gaps are narrow, and cooperation with surrounding drivers is essential to achieve the lane-change maneuver. We propose a two-stage control framework that harmonizes Model Predictive Control (MPC) with Generative Adversarial Networks (GAN) by utilizing driving intentions to generate smooth lane-change maneuvers. To improve performance in practice, the system is augmented with an adaptive safety boundary and a Kalman Filter to mitigate sensor noise. Simulation studies are investigated in different levels of traffic density and cooperativeness of other drivers. The simulation results support the effectiveness, driving comfort, and safety of the proposed method.

Index Terms—Autonomous driving, lane-change, cooperation-aware driving, dense traffic, model predictive control, recurrent neural networks, generative adversarial networks.
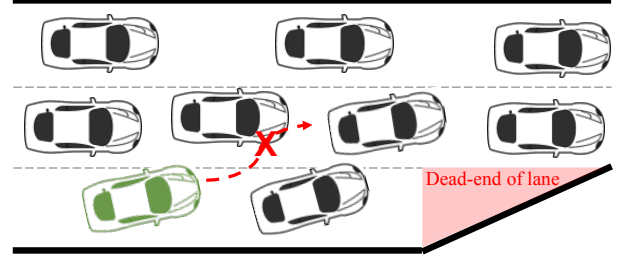

Fig. 1. The autonomous-driving vehicle (in green) intends to change lanes, within a restricted merging area. The traffic is dense with narrow inter-vehicle intervals that are spatially insufficient for a vehicle to merge into. The autonomous-driving vehicle would get stuck in the merging area, unless other drivers slow down to make space for the vehicle.

I. INTRODUCTION

GIVEN the millions of edge cases related to navigating traffic situations safely and efficiently, designing a reliable motion planner is still an open-ended challenge. In this study, we focus on one of the extreme, yet often observed traffic conditions: highly dense traffic.

Advanced Driver Assistance Systems (ADAS), such as Adaptive Cruise Control (ACC) and braking assistance, have effectively enhanced safety [1], [2] by mitigating the impact of human errors and also improving energy efficiency [3], [4]. A large body of literature in the domain of ADAS has focused on lane-change assistance that involves three stages of the decision making process [5]: (i) *WHERE* to change lanes (if desired), (ii) *WHEN* to change lanes, and (iii) *HOW* to change lanes. Decision-making at each stage then engages unique technical challenges and an ample set of literature does exist in ordinary driving conditions. For *WHERE*, if a lane-changing is discretionary, a target lane is often chosen by comparing an average speed or throughput of neighboring lanes with that of a current lane [5]–[7]. If a neighboring

lane has a higher throughput, lane-change is desired and the neighboring lane is set as a target lane. For *WHEN*, inter-vehicle gaps are often investigated to find a safe (target) gap to execute lane-change [8]. For *HOW*, given a target lane and target gap, a control trajectory is found to execute a smooth lane-change [9], [10]. In highly dense traffic, however, the stages are not clearly classified. That is, *HOW* and *WHEN* are closely combined since there is no safe inter-vehicle gap and the gap can be created by how the vehicle is controlled, i.e., *WHEN* is determined by *HOW*. Therefore, to succeed at changing lanes in dense traffic, it is important to estimate other vehicles' behaviors conditioned on how the ego vehicle[1] behaves. Corresponding technical challenges of lane-changing in dense traffic include:

[C1] the absence of safe inter-vehicle gap;
[C2] the need of evaluating "cooperative" behaviors triggered by a complex decision mechanism;
[C3] the presence of multiple vehicles simultaneously playing a role;
[C4] the practicality of *online* computation.

An increasing number of literature on lane-changing in dense traffic addresses the above challenges either partially [11]–[16] or fully [17]–[19]. One recognized study [14] utilizes Inverse Reinforcement Learning (RL) to estimate a reward for a neighboring vehicle being cooperative. The estimated reward is then used to optimize controls. Recently, a model-free RL agent is designed [18] for lane-changing control in dense traffic, proving its effectiveness in learning complex interactive-behavior mechanisms of multiple sur-

S. Bae was with University of California, Berkeley, CA, 95134 USA, at the time of the work, and is currently with Honda Research Institute, CA, 95134 USA e-mail: sbae@honda-ri.com.
D. Isele, A. Miranda Anon, C. Choi, K. Fujimura are with Honda Research Institute, CA, 95134 USA e-mail: {disele, amiranda, cchoi, kfujimura}@honda-ri.com.
A. Nakhaei was with Honda Research Institute, CA, 95134 USA, at the time of the work, and is currently with Toyota Research Institute, CA, 94022 USA e-mail: alireza.nakhaei@tri.global.
P. Xu was with Honda Research Institute, CA, 95134 USA, at the time of the work, and is currently with Amazon, WA, 98109 USA e-mail: xpharry@amazon.com.
S. Moura is with University of California, Berkeley, CA, 94720 USA e-mail: smoura@berkeley.edu.

[1]We refer to the vehicle that is controlled as the "ego vehicle".

rounding vehicles. Still, reliability and interpretability limit the practical use of RL particularly in safety-critical applications like autonomous driving, and these issues remain active research areas [20]–[22].

Apart from RL, an increasing amount of literature has applied Deep Neural Network architectures to autonomous driving for explaining complex environments [23]–[25]. Particularly in predicting motions of humans (drivers), Recurrent Neural Network (RNN) architectures have rapidly advanced, proving their accuracy [26], [27] as well as computational efficiency [28]. Therefore, it would be natural to take advantage of those advances for controller design, in combination with control-theoretic methods for autonomous driving control [29] and well-formulated vehicle dynamics models [30]. That is, a controller based on control theory determines optimal control inputs for autonomous driving, while partially exploiting predictions by RNN. The incorporation thereby would keep the controller interpretable, and tunable, while having the controller responsive to delicate, interactive motions of human drivers. It is still challenging to mathematically incorporate RNN into formal controller design and to solve the control problem effectively and efficiently. We address these challenges in this paper.

We add two original contributions to the literature: (i) We propose a mathematical control framework that systematically evaluates several human drivers' interactive motions, in dense traffic. The framework is built upon Model Predictive Control (MPC) and integrates Recurrent Neural Networks (RNN) as a *sub-model* – which we denote as "NNMPC" for convenience. RNN exclusively predicts future positions of other vehicles, reflecting interactions between vehicles. Then the MPC framework evaluates the predictions through a formalized optimization structure. (ii) We propose a *real-time* numerical algorithm that finds optimal solutions based on driving-intentions. The idea of incorporating RNN as a prediction model into an MPC controller is straightforward. However, a challenge with RNNs is their highly nonlinear structure, which is complex to solve using classical optimization algorithms. We show our algorithm generates smooth lane-change maneuvers while securing a *real-time* computational efficiency. Additionally, we enhance the practicality by accounting for sensor noises, prediction errors, and recursive feasibility. We also adapt a commonly-used two-stage structure of planning & control and validate the proposed framework under the CARLA simulation environment.

This paper incorporates and extends our previous work in [17]. In particular, we develop an action space refinement technique utilizing driving intentions, which improves driving comfort and computational efficiency. Besides, this paper develops the theoretical model of our previous work into a robust system more capable of meeting the demands of a physical system. Finally, the proposed model is compared with other cooperation-aware lane-changing models, including [16], to validate its performance.

The paper is organized in the following manner: Section II gives an overview of the two-stage planning and control framework, followed by detailed formulations and algorithms of the proposed trajectory planner in Section III. Section IV re-
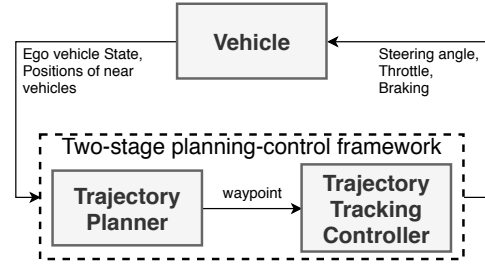

Fig. 2. Diagram for two-stage planning and control framework. The trajectory planner in the upper layer generates waypoints, a sequence of positions coupled with velocities. The trajectory tracking controller in the lower layer determines a pair of steering angle and throttle/braking. The trajectory planner evaluates cooperative behaviors conditioned on possible choice of trajectory.

ports the performance of the proposed method in the CARLA simulation environment. The paper concludes with a summary in Section V.

II. OVERVIEW OF STRUCTURE AND MODEL

A. Structure

In our previous work [18], we proposed a control framework that combines trajectory planning with determining control inputs, i.e., one-stage. This one-stage structure, however, may not be appropriate where a low-level controller already exists and true system plants are barely known. Additionally, the one-stage structure limits the flexibility in tuning the planning and control as they are coupled. Therefore, in many practical scenarios, a two-stage structure is utilized [31], [32], where the upper layer plans trajectories (planner), and the lower layer follows the trajectories (controller). Fig. 2 illustrates the two-stage structure, which we leverage in this work. The planner in the upper layer (bottom left box in Fig. 2) builds an MPC formulations that incorporate RNN as a prediction module. The controller in the lower layer (bottom right box in Fig. 2) consists of two separate control process for longitudinal control and lateral control, respectively. This separation enables ease in implementations. The longitudinal control is canonically designed as a proportional-integral-derivative (PID) controller, and the lateral control is designed as an MPC controller [33]. This paper focuses on the upper layer, i.e., trajectory planner while briefly summarizing the lower layer controller design for completeness of the work.

B. Controller

The longitudinal controller is a PID controller that takes a desired speed and current speed to compute an offset and devises a throttle in the next time step according to PID gains tuned *a priori*.

The lateral controller takes a reference position (from waypoints) in next time step and current speed as input. It finds a steering angle that minimizes a divergence between "next" and "propagated" position through the vehicle rotation model in [33]. The MPC formulation reads:

$$\delta(t) = \text{argmin}_\delta \; (x^{\text{ref}}(t+1) - \tilde{x}(t+1))^2$$
$$+ (y^{\text{ref}}(t+1) - \tilde{y}(t+1))^2$$
$$+ \lambda_\psi (\psi^{\text{ref}}(t+1) - \tilde{\psi}(t+1))^2, \quad (1)$$
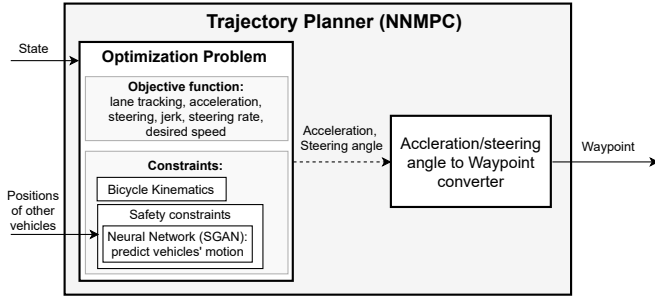
Fig. 3. Diagram of the planning framework with a Recurrent Neural network, SGAN. The optimization is solved with respect to accelerations and steering angles which are then converted to waypoints that the lower level controller in Fig. 2 tracks.

subject to the vehicle rotation model, where $\tilde{\ }$ denotes a predicted value and superscript $^{\text{ref}}$ denotes a reference value. The parameter λ_ψ relatively weights the penalty on heading. Recall, designing a lower-level controller is not the main focus of this paper. Hence, the design is motivated by simplicity. More advanced lower-level controllers can be formulated, e.g., combining longitudinal control with lateral control [34].

C. Vehicle Model

For trajectory planning, we utilize the nonlinear kinematic bicycle model in [35] to represent the vehicle dynamics. For completeness, we re-write the kinematics here:

$$\dot{x} = v\cos(\psi + \beta) \tag{2}$$

$$\dot{y} = v\sin(\psi + \beta) \tag{3}$$

$$\dot{\psi} = \frac{v}{l_r}\sin(\beta) \tag{4}$$

$$\dot{v} = a \tag{5}$$

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan(\delta)\right) \tag{6}$$

where (x,y) is the Cartesian coordinate for the center of vehicle, ψ is the inertial heading, v is the vehicle speed, a is the acceleration of the car's center in the same direction as the velocity, and l_f and l_r indicate the distance from the center of the car to the front axles and and to the rear axles, respectively. The control inputs are: (front wheel) steering angle δ and acceleration a. We use Euler discretization to obtain a discrete-time dynamical model in the form:

$$z(t+1) = f(z(t), u(t)), \tag{7}$$

where $z = \begin{bmatrix} x & y & \psi & v \end{bmatrix}^\top$ and $u = \begin{bmatrix} a & \delta \end{bmatrix}^\top$ for time t. Note that the above simple bicycle kinematics can improve computational efficiency without losing modeling accuracy [35], compared to more complex models, e.g., vehicle dynamics with tire models [36], [37].

III. TRAJECTORY PLANNING

The trajectory planner is composed of two problems: (i) smooth trajectories generations and (ii) optimality and safety evaluations. The first problem is to find a set of smooth trajectories that reflect different driving intentions (safety is not considered). Then the second problem is to evaluate both the
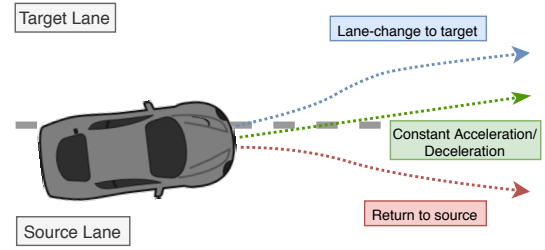

Fig. 4. Examples of driving intentions.

optimality and safety of the pre-computed trajectories (neural networks are applied conditioned on the trajectories). In other words, the first problem (referred to as the pre-computation problem) pre-computes a set of trajectories and the second problem (referred to as the optimization problem) chooses the best one among the trajectories. This hierarchical structure is due to the complexity of the optimization problem that limits both finding solutions itself and computing in real-time. We detail each problem in the following sections.

A. Pre-computation of Smooth Trajectories Reflecting Driving Intentions

Driving in urban areas takes advantage of the presence of lanes. Lanes are used as guidelines to limit the number of intentions available to the vehicle. We specify four intentions: (i) keeping on a current (source) lane, (ii) changing to a next (target) lane, (iii) constantly increasing speed, (iv) and constantly decreasing speed. For each intention, a trajectory is pre-computed and converted to a sequence of accelerations/braking coupled with steering angles. Examples of pre-computed trajectories are illustrated in Fig. 4. It is important to highlight that each trajectory needs to be computed timely for online control while remaining feasible with respect to control limits. To compute trajectories for (i) keeping on a source lane or (ii) moving to a target lane, we utilize polynomial spiral curves [10]. Polynomial spiral curves are commonly used in path-planning problems for their curvature continuity which facilitates smooth-path generation and for their computational efficiency. We briefly review the path-planning problem with cubic spiral curves [10], where a curve $\mathcal{K}(l)$ is formulated:

$$\mathcal{K}(l) = \beta_3 l^3 + \beta_2 l^2 + \beta_1 l + \beta_0, \tag{8a}$$

for arbitrary distance step l. Positions x, y and heading ψ are formulated:

$$x(l) = x_0 + \int_0^l \cos(\psi(l'))dl', \tag{8b}$$

$$y(l) = y_0 + \int_0^l \sin(\psi(l'))dl', \tag{8c}$$

$$\psi(l) = \psi_0 + \int_0^l \mathcal{K}(l')dl'. \tag{8d}$$

Given boundary conditions for positions and heading, numerical approximations using Simpson's rule (for integrals) [38], and parameter mappings (between spiral parameters β and curvature p), the smooth-path optimization problem reads:

$$\min_p f_{be}(\beta_0, \beta_1, \beta_2, \beta_3, l_f)$$
$$+ \lambda_x(x_s([p]_4) - x_f) + \lambda_y(y_s([p]_4) - y_f)$$
$$+ \lambda_\psi(\psi_s([p]_4) - \psi_f), \tag{9a}$$

subject to:

$$|[p]_1| \le \mathcal{K}_{\max}, \tag{9b}$$

$$|[p]_2| \le \mathcal{K}_{\max}, \tag{9c}$$

where p indicates a vector of curvatures with $[p]_i$ indicating the ith element of p. Subscript s denotes an approximation with Simpson's rule, and subscript f denotes a final point. The first term in the objective function is the penalty on bending energy, formulated as:

$$f_{be}(\beta_0, \beta_1, \beta_2, \beta_3, l_f) = \int_0^{l_f} \mathcal{K}(l')^2 dl', \tag{10}$$

and the second through the last term in the objective function respectively represent softened boundary conditions (i.e., soft constraints) for final positions (x_f, y_f) and heading ψ_f. The penalty weights λ_x, λ_y, and λ_ψ on the soft constraints are arbitrarily set to a high value. Note that the curvature constraints (9b),(9c) are applied only to a couple points, which effectively reduces a number of variables to optimize[2]. The maximum curvature $\mathcal{K}_{\max}$ is set to the maximum steering angle $\delta_{\max}$, i.e., $\mathcal{K}_{\max} = \delta_{\max}$. The optimization problem is non-convex and non-linear with respect to the curvature p. To solve the problem, we apply L-BFGS-B [39]. Recall that a resulting path is continuous and only a few equally-spaced points[3] are sampled among the path to evaluate its optimality and feasibility of lane-changing in Section III-B. It is important to note that the pre-computed trajectories are converted to control sequences of acceleration a and steering angle δ.

B. Objective

The planning objective is to merge to the target lane while avoiding collisions with other vehicles. We prefer to change lanes sooner than later. We also prefer smooth accelerations and steering for passenger comfort. The objective function is formulated:

$$J = \sum_{\ell=t}^{t+T} \lambda_{div}(\ell|t)D(\ell|t) \qquad \textbf{target lane} \tag{11}$$

$$+ \sum_{\ell=t}^{t+T} \lambda_v \|v(\ell|t) - v^{\mathrm{ref}}\|^2 \qquad \textbf{target velocity} \tag{12}$$

$$+ \sum_{\ell=t}^{t+T-1} \lambda_\delta \|\delta(\ell|t)\|^2 \qquad \textbf{steering effort} \tag{13}$$

$$+ \sum_{\ell=t}^{t+T-1} \lambda_a \|a(\ell|t)\|^2 \qquad \textbf{accel. effort} \tag{14}$$

$$+ \sum_{\ell=t+1}^{t+T-1} \lambda_{\Delta\delta} \|\delta(\ell|t) - \delta(\ell-1|t)\|^2 \quad \textbf{steering rate} \tag{15}$$

$$+ \sum_{\ell=t+1}^{t+T-1} \lambda_{\Delta a} \|a(\ell|t) - a(\ell-1|t)\|^2 \quad \textbf{jerk} \tag{16}$$

where $(\ell|t)$ indicates time ℓ based on the measurements at time t. Symbol x_{end} is the latitude coordinate of the road-end, $D(\ell|t)$ is the distance norm for the vector between

---

[2]Limiting the number of curvature constraints, the cubic spiral curve still generates curvatures the vehicle can perform [10].

[3]The number of points is equivalent to the number of time steps in the receding horizon T

---

the ego vehicle's center and the target lane at time ℓ, v^{ref} is the reference velocity. Each penalty is regularized with λ_{div}, λ_v, λ_δ, λ_a, $\lambda_{\Delta\delta}$, and $\lambda_{\Delta a}$, respectively. We incentivize a timely lane change with the dynamic weight $\lambda_{div}(\ell|t) = \frac{1}{\sqrt{(x(\ell|t)-x_{\mathrm{end}})^2+(y(\ell|t)-y_{\mathrm{end}})^2}}$. As the ego vehicle gets closer to the deadend position $(x_{\mathrm{end}}, y_{\mathrm{end}})$, the dynamic weight exponentially increases. The term (11) penalizes the divergence of the center of the vehicle from the vertical center of the target lane. The term (13) and (14) penalize the control effort of steering angle and acceleration, respectively. The term (15) and (16) penalize the steering rate and jerk, respectively, for drive comfort.

C. Safety Constraints with a Recurrent Neural Network

To quantify safety, we consider a distance measure between two vehicles. That is, if the distance between two vehicles is zero, it means they collide with each other. Now, there are two important aspects to set up a safety constraint. First, a mathematical measure of distance between two vehicles depends on how each vehicle is shaped since we do not have direct measurements of inter-vehicle distances over a planning horizon – recall, we have a measurement only at the current time, not every time step over a planning horizon. Second, the distance must be calculated based on predicted positions with respect to the ego vehicle's trajectories. Namely, we need a prediction module that estimates positions of other vehicles in response to the ego vehicle's actions – and we leverage a recurrent neural network.

1) Minimum Inter-vehicle Distance Measure: As illustrated in Fig. 5, we first model the vehicle shape with three circles where the radius is half the vehicle width. Then an Euclidean distance between two circles are analytically obtained (which is computationally efficient). To constrain safety, a minimum distance between any pairs of circles must be greater than a safety bound ϵ. Formally:

$$g_i(x, y) = \min_{p,q \in \{-1,0,1\}} d_i(p, q) \ge \epsilon, \tag{17}$$

where: $d_i(p, q)$
$$= \Big[\big((x + p(h - w)\cos\psi) - (x_i + q(h_i - w_i)\cos\psi_i) \big)^2$$
$$+ \big((y + p(h - w)\sin\psi) - (y_i + q(h_i - w_i)\sin\psi_i) \big)^2 \Big]^{\frac{1}{2}}$$
$$- (w + w_i), \tag{18}$$

and w and h are, respectively, half width and half height of the vehicle, respectively, and ϵ is a safety bound. Subscript i indicates vehicle i. Note that Eqn. (18) represents a distance between two circles.

2) Interactive Motion Prediction: Drivers' motions are responsive to interactions with each other, and consequently, the motions must be predicted simultaneously. We adopt a state-of-the-art RNN structure, Social Generative Adversarial Networks (SGAN) from [28] which efficiently and effectively captures interactions between agents (drivers). SGAN is composed of a generator and discriminator that are adversarial to each other. Both the generator and discriminator are comprised
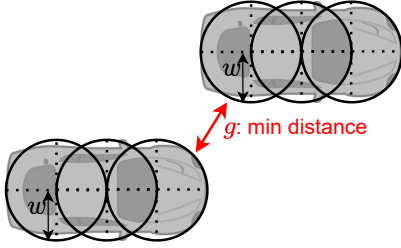
Fig. 5. Vehicle shapes are modeled by three circles. Note that multiple (more than three) circles of various radius can be applied to any vehicle shape and size. The minimum distance g is computed as the minimum Euclidean distance between any pair of circle, as in Eqn. (17)

of long-short term memory (LSTM) networks to account for the sequential nature of agents' motion. The following module aggregates the agents' motion states to evaluate their interactions. This pooling process is essential to share the motion history between agents, generating social interactions as a pooled tensor P_i for *each* agent i. The decoder in the generator then predicts multiple trajectories that are socially interactive with each other. SGAN takes as an input a sequence of positions for each agent within a scene over an observation time horizon T_{obs}. It outputs a sequence of positions for each agent over a prediction time horizon T_{pred}. Interested readers are referred to [28] for more details. It is important to highlight that a trained SGAN will predict the most probable reactions of other vehicles based on the control commands on the ego vehicle, and the history of previous actions. However, in reality, the reactions of other vehicles might be different.

3) Incorporation of Recurrent Neural Network: A trained SGAN predicts the evolution of vehicle i's centroid. These predictions are incorporated into safety constraints (17). Formally, we denote a trained SGAN as a function $\phi(t)$ that maps observed trajectories to predicted trajectories for N nearby vehicles at time t:

$$
\phi(t):
$$
$$
\begin{bmatrix}
(x_1(t), y_1(t)) & \cdots & (x_N(t), y_N(t)) \\
\vdots & \vdots & \vdots \\
(x_1(t - T_{obs} + 1), & \cdots & (x_N(t - T_{obs} + 1), \\
\quad y_1(t - T_{obs} + 1)) & & \quad y_N(t - T_{obs} + 1))
\end{bmatrix}
$$
$$
\Updownarrow
$$
$$
\begin{bmatrix}
(\hat{x}_1(t + 1), \hat{y}_1(t + 1)) & \cdots & (\hat{x}_N(t + 1), \hat{y}_N(t + 1)) \\
\vdots & \cdots & \vdots \\
(\hat{x}_1(t + T_{pred}), & \cdots & (\hat{x}_N(t + T_{pred}), \\
\quad \hat{y}_1(t + T_{pred})) & & \quad \hat{y}_N(t + T_{pred}))
\end{bmatrix},
$$
$$(19)$$

where $\hat{\cdot}$ indicates a predicted value. Given the observations until time t, the coordinates of vehicle i at time $t + 1$ are represented as $\hat{x}_i(t + 1) = \phi_{i,x}(t)$ and $\hat{y}_i(t + 1) = \phi_{i,y}(t)$. Note that the trained SGAN is not time dependent, i.e., the notation (t) in $\phi(t)$ indicates the evaluation at time t without any indication of temporal dependencies.

D. Optimization Formulations

The complete optimization problem for the receding horizon control is:

$$
\min_{z,a,\delta} \; J = \sum_{\ell=t}^{t+T} \Big(\lambda_{div}(x(\ell|t); x_{\text{end}}) D(\ell|t)
$$
$$
+ \lambda_v \| v(\ell|t) - v^{\text{ref}} \|^2 \Big)
$$
$$
+ \sum_{\ell=t}^{t+T-1} \Big(\lambda_\delta \| \delta(\ell|t) \|^2 + \lambda_a \| a(\ell|t) \|^2 \Big)
$$
$$
+ \sum_{\ell=t+1}^{t+T-1} \Big(\lambda_{\Delta\delta} \| \delta(\ell|t) - \delta(\ell - 1|t) \|^2
$$
$$
+ \lambda_{\Delta a} \| a(\ell|t) - a(\ell - 1|t) \|^2 \Big) \qquad (20a)
$$

subject to:

$$
z(\ell + 1|t) = f(z(\ell|t), \delta(\ell|t), a(\ell|t)) \quad (20b)
$$
$$
g_i(z(\ell + 1|t); x_i(t), y_i(t)) \geq \epsilon, \quad \forall i \in \{1, \cdots, N_{veh}\} \quad (20c)
$$
$$
a(\ell|t) \in [a_{\min}, a_{\max}] \quad (20d)
$$
$$
\delta(\ell|t) \in [\delta_{\min}, \delta_{\max}] \quad (20e)
$$
$$
(x(t + T|t), y(t + T|t)) \in \mathcal{S} \quad (20f)
$$

where $(\ell|t)$ denotes time ℓ based on the measurements at time t, z is a stacked variable of states, a denotes acceleration, δ denotes steering angle, and $D(\ell|t)$ is the distance norm for the vector between the ego vehicle's center and the target lane at time ℓ. The objective (20) assesses reference tracking, control effort, and drive comfort. Each penalty in (20) is weighted by λ as described in Section III-B. The constraint (20b) indicates the nonlinear bicycle kinematics [35]. The constraint (20c) indicates collision avoidance, enforcing the Euclidean distance between the ego vehicle and vehicle i to be larger than the safety bound ϵ. Controls (a, δ) are lower and upper bounded in (20d) and (20e). The constraint (20f) ensures the ego vehicle not to get stuck at a road-end positioned at $(x_{\text{end}}, y_{\text{end}})$ where $\mathcal{S}$ denotes a feasible set that avoids freezing behaviors. We detail the constraint for avoiding freezing behaviors in the next section.

1) Terminal Constraint for Avoiding Freezing Problem: To prevent the ego vehicle being stuck at the dead end, we define the feasible set of a terminal position as a half-plane in the two-dimensional Euclidean space:

$$
\mathcal{S}_\theta = \{(x', y') \mid y' \cos \theta - x' \sin \theta
$$
$$
\geq f_{\text{aff}}(y' \sin \theta + x' \sin \theta)\}, \qquad (21)
$$

where:

$$
f_{\text{aff}}(x) = \delta_{\max}(x - x_{\text{end}}) + y_{\text{end}}, \qquad (22)
$$

with road angle θ. Note that bicycle kinematics represents an ellipsoidal trajectory (with a constant velocity and steering angle) and the set bounded by the ellipsoidal trajectory is a convex set. Therefore, by definition, the affine function (22) is enveloped by the trajectory between any two points on the trajectory. Each subset split by the affine function still represents a convex set and the volume of any subset is less than that of the primal set [40]. In other words,

regardless of the road angle and lane-changing direction, the feasible set (21) with the affine function (22) represents a more conservative set of trajectories than that with the bicycle kinematics between two points on the trajectory.

2) Adaptive Safety with Prediction Errors: Prediction errors of neural networks are often evaluated with Average Displacement Error (ADE) computed as a Euclidean distance between "predicted" and "actual" positions over a prediction horizon. The ADE can be used as a distance buffer in the safety constraint (20c) to mitigate the prediction error. The ADE from training, however, may differ from testing and therefore the ADE needs to be adaptively evaluated in real-time. Therefore, in real-time, we evaluate a maximum prediction error among all subject vehicles' position at each time and add additional distance buffer in longitudinal and lateral directions, respectively. Namely:

$$\epsilon_i^{\text{lng}} = \left(\max_{i=1,\cdots,N_{\text{veh}}} \text{ADE}_i \right) \cos \psi_i,$$

$$\epsilon_i^{\text{lat}} = \left(\max_{i=1,\cdots,N_{\text{veh}}} \text{ADE}_i \right) \sin \psi_i, \qquad (23)$$

where:

$$\text{ADE}_i = \sum_{\ell=t-T+1}^{t} \frac{\|(\hat{x}_i(\ell), \hat{y}_i(\ell)) - (x_i(\ell), y_i(\ell))\|_2}{T}, \qquad (24)$$

$(\hat{x}_i(t), \hat{y}_i(t))$ is a predicted position of vehicle i at time step t. The real-time prediction error is then applied in longitudinal and lateral directions, respectively. That is, the safety constraint (20c) now reads:

$$\min_{p,q \in \{-1,0,1\}} d_i(p, q) \cos \psi_i \geq \epsilon_0 + \epsilon_i^{\text{lng}}, \qquad (25)$$

$$\min_{p,q \in \{-1,0,1\}} d_i(p, q) \sin \psi_i \geq \epsilon_0 + \epsilon_i^{\text{lat}}, \qquad (26)$$

where ϵ_0 is a default distance buffer. Note that the real-time prediction error is applied in longitudinal and lateral directions, respectively, to avoid over-conservative behaviors (e.g., pushing brake when a vehicle is on the next lane as the lateral distance is within the safety margin). Note that the distance buffer can apply distributionally robust optimization methods [41] to provably improve safety, which remains for future work.

E. Complete Algorithm

The complete Intention-based NNMPC algorithm with localization and perception filtering is structured in Algorithm 1. The algorithm proceeds in the following manner. Suppose a target lane is given. The state of the ego vehicle and the positions of other drivers are initialized with initial measurements. While the divergence D of the ego vehicle's position from the target lane is larger than a minimum distance threshold $\epsilon_{\min}$, control sequences are pre-computed by (9). Then each control sequence is evaluated by (20) (detailed in Fig. 6). With an optimal control sequence, waypoints are generated and sent to a lower-level controller. Meanwhile, new states of the ego vehicle and positions of other drivers are measured and updated.

Algorithm 1: Intention-based NNMPC algorithm

Init : states $z = z_0$,
other vehicles' position $(x_i, y_i) = (x_{i0}, y_{i0})$ for all
$i \in \{1, \cdots, N_{veh}\}$

1 **while** $D \leq \epsilon_{min}$ **do**
2   Generate control candidates for different driving intentions $\mathcal{U} = \{U_{\text{target}}, U_{\text{source}}, U_{\text{acc}}, U_{\text{dcc}}\}$
3   Find the optimal sequence that minimizes cumulative cost over T and that is feasible with (20c) (Fig. 6)
4    $U_* \leftarrow \text{argmin}_{U \in \mathcal{U}}$ (20)
5   Propagate through dynamics (20b) with U_* to generate waypoints
6    $z^{\text{ref}} \leftarrow f(z, U_*)$
7   Observe state of ego vehicle at the current time step t
8    $z \leftarrow \hat{z}$
9   Observe positions of other vehicles at the current time t
10    $(x_i, y_i) \leftarrow (\hat{x}_i(t), \hat{y}_i(t))$ for all i
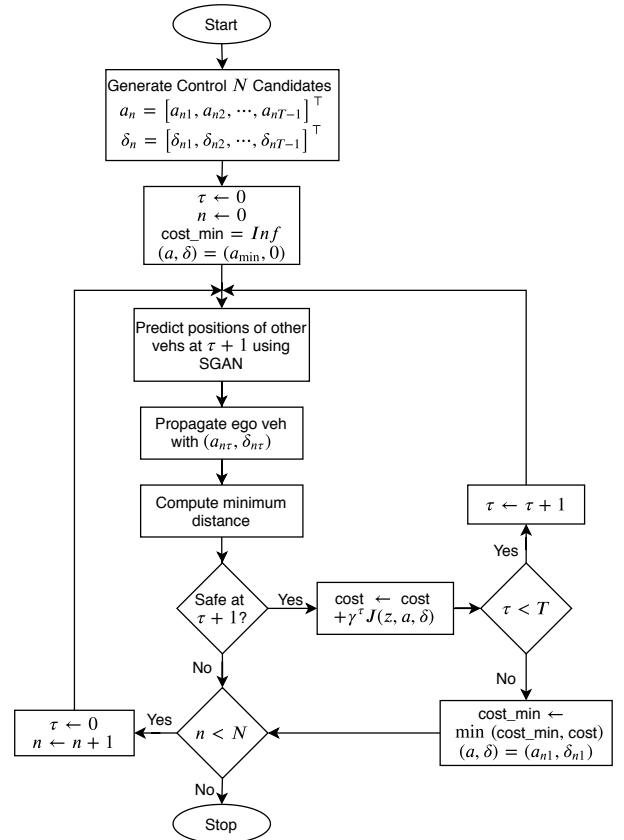11 **end**

Fig. 6. Finding optimal controls of NNMPC. At each time step τ, (i) predict other drivers' position (for the next time step, i.e., $\tau+1$), (ii) propagate the ego vehicle with a pre-computed control at time τ through the dynamics (20b), (iii) check the constraints (20c)-(20f), and discards the candidate sequence if violated, and otherwise (iv) update the cumulative cost (20) with discounting factor $\gamma \in [0, 1]$. Repeat (i)-(iv) for all control candidates and return the candidate that has a minimum cost. If no candidate is feasible, the ego vehicle takes the maximum brake with zero steering.Note, the choice of action when no feasible candidate exists can be extended depending on traffic and road, e.g,. forced returning back to the source lane at high-speed.
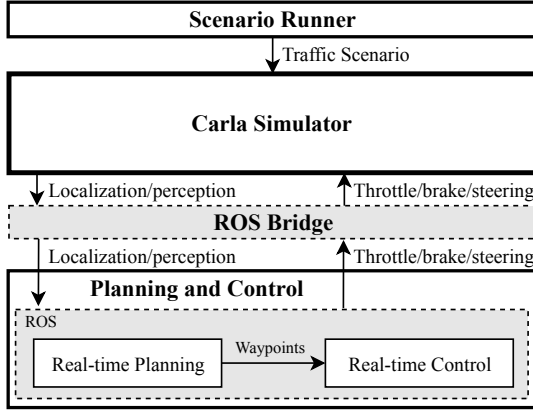
Fig. 7. Simulation Setup. The upper three components (Scenario runner, CARLA simulator, and ROS bridge) are open source.
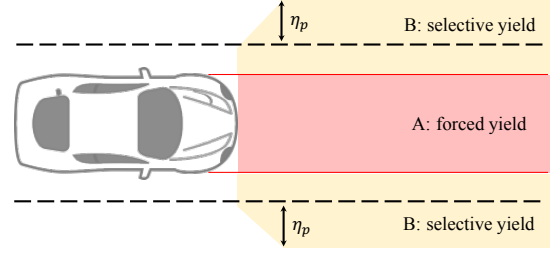

Fig. 8. Forced yield zone (red area labeled with A) and selective yield zone (yellow area labeled with B). The dashed lines indicate the boundaries of the center lane. If a vehicle from the next lane intersects with the path of the vehicle, i.e. in zone A, the vehicle in the center lane must stop and wait until the other vehicle cuts into the center lane. If the other vehicle intersects with zone B, then the vehicle in the center lane decides to either yield or not, according to the cooperativeness parameter η_c. Zone B corresponds to a vertical perception range, and the range can be adjusted by η_p.

IV. SIMULATIONS

A. Simulation Setup

Shown in Fig. 7, the simulation setup is composed of three main components: (i) CARLA simulator [42], (ii) Scenario runner, and (iii) Planning and control. The CARLA simulator is an open-source simulation platform for autonomous driving research, which has gained significant attention due to its scalability with vehicles/pedestrians, flexibility in traffic scenarios, and practicality in implementations with Robot Operating Systems (ROS). The Scenario runner is also an open-source script that enables customization of traffic scenarios in CARLA simulations. The planning and control components represent the proposed methods. We run the simulations on Ubuntu 16.04 LTS (Intel Xeon CPU ES-2640 v4 @2.40GHz x 20, GeForce GTX TITAN).

B. Training Networks

1) Driver Model: We train SGAN through simulations. We first define a driver model in which the longitudinal dynamics are controlled by an intelligent driver model (IDM) from [43]. The lane-changing behavior is governed by the strategy of Minimizing Overall Braking Induced by Lane changes (MOBIL) from [44]. The driver model is also based on the bicycle kinematics in Section II-C. Additionally, we introduce a parameter for cooperativeness $\eta_c \in [0, 1]$ to the driver model. If $\eta_c = 1$, a vehicle stops and waits until another vehicle within the selective yield zone (zone B in Fig. 8) overtakes. If $\eta_c = 0$, then the vehicle ignores neighboring vehicles and drives forward. If $0 < \eta_c < 1$, then the yield action is randomly sampled from the Bernoulli distribution with probability $p = \eta_c$.

TABLE I lists hyperparameters for the SGAN trained through simulations. Training and validation datasets are generated by simulations with only the driver model from Section IV-B1, with heterogeneous parameters from TABLE II. The dataset is collected from multiple scenarios in various traffic densities, from free flow to dense traffic. In the dataset, we also add noise to the positions so that the trained neural network becomes more robust to noisy perceptions. With a total of 27550 data points, training the SGAN with a GPU takes

approximately 18 hours. An example of motions predicted by SGAN, compared to ground truth, is illustrated in Fig. 10.

The SGAN has 1.872 $[m]$ of average displacement error and 2.643 $[m]$ of final displacement error after $T_{pred} = 2$, for the training data. It is important to note that the training dataset does not include the testing scenarios. Figure 9 shows the *real-time* time-averaged errors that are comparable with the aforementioned training error in the specific testing scenario (discussed in the next section). The advantage of the adaptive safety boundary (based on the real-time prediction errors) is not obvious against the static safety boundary, however, it should not be underestimated especially under unknown conditions where the real-time observation can enhance safety.

TABLE I
SGAN PARAMETERS

Param	Description	Value
T_{obs}	Observation time horizon	8
T_{pred}	Prediction time horizon	2
N_b	Batch size	64
D_{emb}	Embedding dimension	64
D_{mlp}	MLP dimension	256
$D_{G,e}$	Hidden layer dimension of encoder (generator)	32
$D_{G,d}$	Hidden layer dimension of decoder (generator)	64
$D_{D,e}$	Hidden layer dimension of encoder (discriminator)	64
D_{bot}	Bottleneck dimension in Pooling module	1024
α_G	Generator learning rate	$5 \cdot 10^{-4}$
α_D	Discriminator learning rate	$5 \cdot 10^{-4}$

TABLE II
DRIVER MODEL DESIGN PARAMETERS

| Param	Description	Value
$\tilde{v}^{\text{ref}}$	Reference velocity $[m/s]$	$\mathcal{U}(2, 5)$
$\tilde{T}$	Safe time headway $[s]$	$\mathcal{U}(1, 2)$
$\tilde{a}_{\max}$	Maximum acceleration $[m/s^2]$	$\mathcal{U}(2.5, 3.5)$
$\tilde{b}$	Comfortable deceleration $[m/s^2]$	$\mathcal{U}(1.5, 2.5)$
$\tilde{\delta}$	Acceleration exponent	$\mathcal{U}(3.5, 4.5)$
$\tilde{s}_0$	Minimum distance to front vehicle $[m]$	$\mathcal{U}(1, 3)$
η_c	Cooperativeness $\in [0, 1]$	$\mathcal{U}(0, 1)$
η_p	Vertical perception range $[m]$	$\mathcal{U}(-0.15, 0.15)$
w	Length from center to side of vehicle $[m]$	0.9
h	Length from center to front of vehicle $[m]$	2
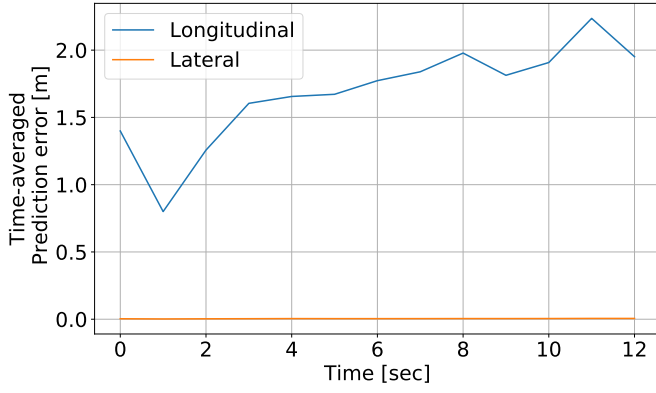
Fig. 9. Real-time (time-averaged) prediction errors in longitudinal and lateral directions. The prediction in lateral direction is very accurate while that in longitudinal direction is not as accurate. To mitigate the prediction error, the safety margin ϵ in (20c) is set accordingly in each longitudinal and lateral direction. Note that the mean is $1.703\,[m]$, which is comparable with the training error of the SGAN ($1.872\,[m]$).
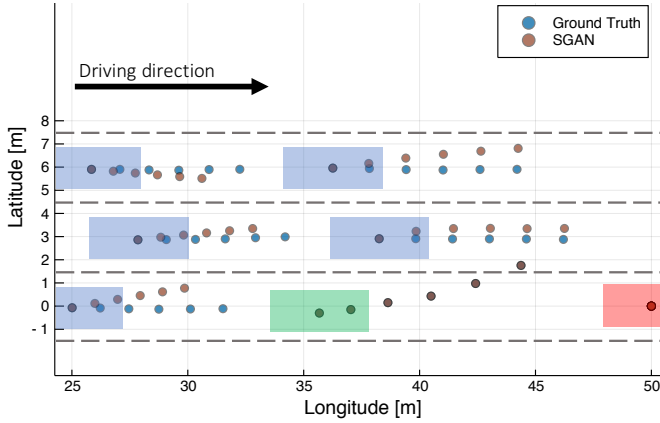

Fig. 10. Motions predicted by SGAN. Rectangles in green, blue, and red indicate the ego vehicle, other vehicles, and the stopped vehicles respectively. Dashed lines represent the lane boundary. Each circular point indicates the vehicle center at each time step.

Recall, the prediction of SGAN is obviously not perfect. Still, we leverage its capability of evaluating interactive behaviors, which is the key to changing lanes under highly dense traffic. The prediction errors are embedded into the safety bound (23). Also, our ultimate goal is a safe lane change, as discussed next, not "zero-error" motion prediction.

Note that SGAN can provide a distribution of predicted motions, which can be incorporated into the optimization problem as chance constraints, thereby enabling a robust formulation [41]. Also, different methods for designing loss functions for SGAN training can be applied, which are topics for future work.

C. Testing Scenario

Figure 11 illustrates a traffic scenario. Only the ego vehicle applies the proposed method. The other drivers are modeled as IDM and their parameters are randomly sampled from uniform distributions as in TABLE II. We examine four traffic scenarios with two different levels of density and cooperative-
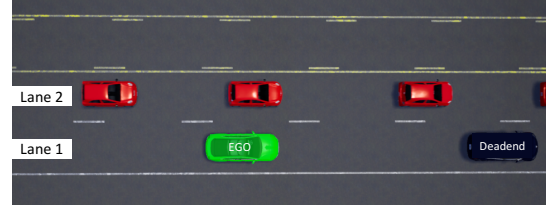

Fig. 11. Lane-changing scenario in CARLA. The **green** vehicle is the ego vehicle and the **black** vehicle is a broken car. The **red** vehicles are other traffic participants. Lane 1 is the source lane and Lane 2 is the target lane. The objective of the ego vehicle is to lane-change to the target lane before the dead end.

ness: (i) sparse-cooperative, (ii) sparse-aggressive, (iii) dense-cooperative, and (iv) dense-aggressive. The average initial inter-vehicle gap is $10.0\,[m]$ with $1.75\,[s]$ of time headway in sparse scenarios and $7.75\,[m]$ with $0.875\,[s]$ of time headway in dense scenarios. Note that both sparse and dense scenarios do not have a "free space" where the ego vehicle can merge-in without interacting with other vehicles (as we target scenarios that require cooperation). The cooperativeness is measured as the likelihood of slowing down when another vehicle is perceived to change lanes. In aggressive scenarios, other drivers do not slow down unless another vehicle overlaps the trajectory. Zero mean Gaussian noises are added to localization and perception to simulate sensor noise. For localization, by default, the standard deviation (std) in (longitudinal position, lateral position) is set to $(0.3, 0.1)\,[m]$ and in heading is $0.08\,[rad]$. For perceptions, by default, std is $(0.1, 0.1)\,[m]$ in positions and $0.08\,[rad]$ in heading.

D. Simulation Results

1) Qualitative Analysis: In most cases, NNMPC successfully finds lane-change maneuver and one common procedure is observed: (i) the ego vehicle gets close to the target lane; (ii) a rear vehicle on the target lane slows down; (iii) the ego vehicle changes the lane. The success in lane-change is mainly led by accurate predictions of whether other drivers are being cooperative or not. However, predictions can often be inaccurate, and consequently, the maneuver fails to induce other drivers to slow down. In these instances, the NNMPC maneuvers the ego vehicle back to the source lane and tries to lane-change again. See the demonstrations in Fig. 12 (video available at https://github.com/bsj1216/NNMPC). To measure the optimality of the trajectory, we compare an average jerk and steering rate. Simulation studies show that utilizing driving intentions to generate a smooth-path yields a small average jerk ($0.14\,[\frac{m}{s^2}]$) and average steering rate ($0.37\,[\frac{rad}{s^2}]$).

2) Quantitative and Comparative Analysis: For the four traffic scenarios with different levels of density and cooperativeness, we examine a quantitative analysis to validate the proposed method. For comparative analysis, we consider two baseline methods: (i) a game-theoretic framework that estimates driving intentions of other drivers using a *game tree* [16] and (ii) a simpler *probabilistic* method that classifies "yielding" behavior by projecting a relative acceleration onto a pre-determined Gaussian distribution of yielding [45]. Note, both models interact with one vehicle at a time (i.e., not
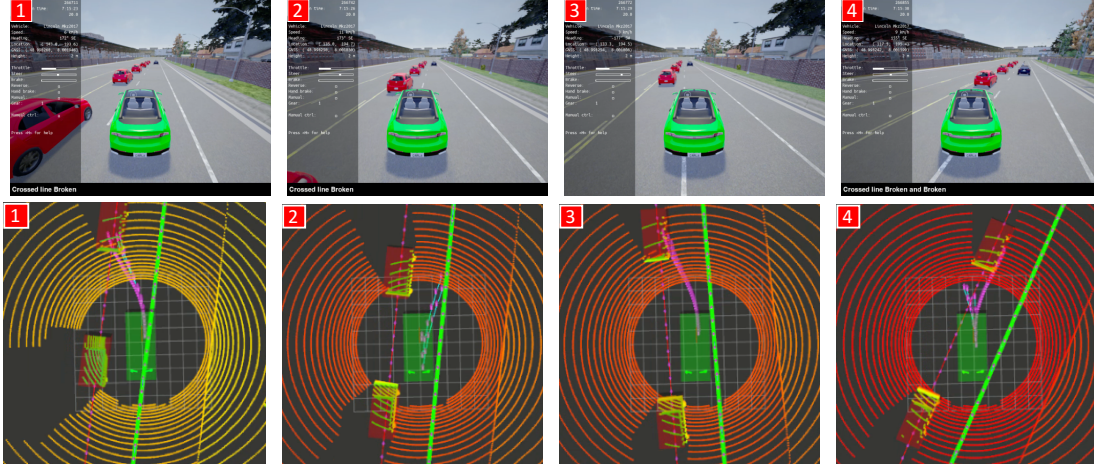
Fig. 12. CARLA simulation snapshots. The top row illustrates third-person view of the ego vehicle (green). The bottom row shows lidar scans, source lane (green line), target lane (red line), and waypoints (purple dots connected by cyan lines). Each snapshot indicates, in order, that the ego vehicle: (1) tries to lane-change to target lane, expecting cooperative behavior of the rear vehicle, (2) returns to source lane as the rear vehicle gets too close without slowing down, (3) again tries to lane-change to target lane as the rear vehicle starts slow down, and (4) successfully changes lane.

interacting with multiple vehicles at once). That said, each interaction is time-efficient and a target vehicle of interaction is updated quickly (to address **[C3]** in Section I).

We formally define that a lane-change is called "Success" if the ego vehicle moves to the target point (50 [m] ahead start point on the target lane) within a time-limit (80 [s]) without collisions[4]. A throttle and angular acceleration are in the range of $[0, 1]$, and a brake is in the range of $[-1, 0]$, where ± 1 indicates a maximum/minimum. Each model (i.e., NNMPC, Game Tree, and probabilistic) has a total of 50+ runs for each traffic scenario, and other vehicles are randomly positioned at each run.

TABLE III reports the performance results for each traffic scenario. We highlight three findings from the analysis. First, the proposed method is robust to both the traffic density and cooperativeness of other drivers. Consequently, the 100% success rate is consistently achieved in any traffic scenario. Second, the average brake, throttle, and angular acceleration indicate that a lane-change maneuver gets aggressive (yet not significant) with non-cooperative behaviors of other drivers and density of traffic. Third, compared to the baseline methods (game tree and probabilistic), NNMPC (Ours) also secures the smoother steering and takes 27% shorter time[5] to merge. In contrast, the game tree model out-performs the smoothness in longitudinal planning (albeit not in lateral). In short, we emphasize that NNMPC secures the highest success rate with proactive coordination with other drivers (hence shorter time to merge), while securing smooth lateral maneuvers.

It is also notable that NNMPC takes more time on average in the sparse cases (than the dense cases) and cooperative cases (than the aggressive cases). Recall that the "Time" in TABLE III is the "scenario completion time" not "time to merge" (as defined above). Meaning, after the ego vehicle

merges into the target lane, the ego vehicle has to slowly cruise with traffic until it passes the goal point (50 meters ahead of the starting point). Therefore, if the ego vehicle merges into the further inter-vehicle gaps at the front, the time will take less (it will be minimum if the ego vehicle merges ahead of the first vehicle in the queue). The results indicate that our method: (i) chooses one of the closer gaps from the starting point when other vehicles are cooperative and sparse (thus a longer completion time) and (ii) chooses one of the further gaps from the starting point when vehicles are aggressive and dense as the ego vehicle decides to cruise in the source lane and explore further gaps (thus a shorter completion time). We found that lane-changing generally takes more time when the ego vehicle stays and waits until enough space is found or other vehicles yield (while not). These behaviors are often found from the baseline methods (game tree and probabilistic), while our method proactively and successfully induces other vehicles to make room for the ego vehicle (consequently, less time while more aggressive brakes).

3) Real-time Applicability of NNMPC: The computation efficiency of the pipeline in Fig. 3 is dominated by the implementation of the neural networks. Thus, the choice of neural networks is crucial for the real-time applicability of this work. SGAN is known for its computational efficiency [28], compared to its variants, e.g., S-LSTM [27]. The time complexity of SGAN corresponds to the matrix multiplications from LSTM and is asymptotically approximated to $\mathcal{O}(n^3)$ [46] (although it can run even faster [47]). Given the input size $(N_p \times N_{veh})$, where $N_p = 8$ and $N_{veh} <= 3$ in our experiments, we evaluated the computation time in three different implementation settings: (i) Dell Optiplex 7070 Micro (6-Core, 9MB Cache, 2.2GHz to 3.7GHz, 35W, 8GB RAM), (ii) GT75 Titan 8RG (Intel CM246, GeForce GTX 1080, 64GB RAM), and (iii) Intel Xeon CPU ES-2640 v4 2.40GHz x 20, GeForce GTX TITAN. In summary, the computation time is approximated as 0.1 [sec] on average which was sufficient for the ROS communication with 10 Hz frequency. In particular, the first setting (i) is the on-board device for on-road driving

---

[4]Note that, a collision can occur either by the ego vehicle crashing into a front vehicle or by a rear vehicle crashing into the ego vehicle.

[5]The absolute value of time may not represent the real-world driving behaviors, but the simulation environment represents extremely dense yet moving traffic to challenge lane-changing.
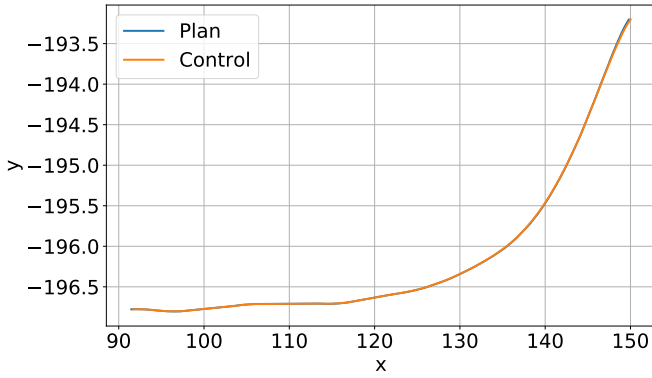
Fig. 13. Planned versus executed trajectory. The planning trajectory only shows the first element of the entire trajectory, which the controller actuates at each time step.

and its computation time is 0.07 [sec] on average. The last setting (iii) is the setting we ran for the simulations, of which most cores were occupied by the Carla simulator and scenario runner. Recall that the computation efficiency highly depends on the choice of prediction modules as the computation time is dominantly consumed from iteratively evaluating a prediction module. Therefore, if a prediction module (not necessarily neural networks) is time-efficient, the overall computation time of the proposed pipeline can be as efficient.

4) Planning versus Control: Figure 13 shows the planning trajectory from the planner and the actuated trajectory by the (speed and steering) controllers. It is clearly seen that the planning trajectory is well tracked by the controllers without delays. In fact, the controllers run much faster than the planner by a factor of two. The stability of the controllers is also mathematically proved in Appendix A.

5) Sensitivity Analysis on Noise: To assess the impact of localization and perception noises on the performance of the proposed method, we investigate a sensitivity analysis with and without noises. The noise is set to 0.3 $[m]$ of standard deviation for both localization and perceptions. There are two main findings: (i) NNMPC is robust to perception noises while (ii) being susceptible to localization noises. NNMPC is robust to perception noises as SGAN was trained with noises in position. Consequently, the perception noises only marginally degrade the prediction accuracy. In fact, the localization noises also marginally degrade the prediction accuracy of SGAN for the same reason. However, the localization noises directly perturb the safety measure in (20c), causing either significant over-estimation or under-estimation. As a consequence, we see a more pronounced decrease of the success rate in TABLE IV. The safety bound in (23) can be jointly adaptive to localization noises, which remain for future work.

6) Performance at Edge Corner and Infeasible Solutions: As mentioned, autonomous driving is one of the most safe-critical applications and thus the planner should be able to timely plan for corner cases, such as unexpected changes in the environment and stochastic behaviors of other traffic participants. In our lane changing scenario, the main edge case can be represented by incorrect evaluations on the other vehicles.

The incorrectness is mainly with the failure of predictions, for example, by estimating a vehicle to slow down while it actually speeds up. Under dense traffic, the incorrect prediction may directly result in collisions. In our simulation runs, we occasionally observed the faulty predictions on the aggressive behaviors and consequently the ego vehicle had to return to the source lane (as illustrated in Fig. 12). Another corner case could be the ego vehicle being stuck at the dead end (i.e., stopped vehicle) without having enough space to escape even with the max steering angle. Our method prevents the state of being stuck successfully with the feasible set in (21), as opposed to our baseline (probabilistic) that occasionally gets stuck under dense and aggressive traffic conditions.

Although our method shows robustness to these corner cases, the scenario is relatively simple and accordingly corner cases are limited. Thus, validating algorithms under more complex scenarios with different types/behaviors of traffic participants remain for future work.

Also, it is important to point out that Eqn. (20) can be infeasible. In particular, the safety constraint in Eqn. (20c) can be violated depending on the safety bound size ϵ. To avoid the infeasibility, we can reformulate the hard constraint Eqn. (20c) as a soft constraint and add it as an additional penalty in the objective function (20), i.e.,

$$+\lambda_g \sum_{\ell=t}^{t+T} \|g_i(z(\ell+1|t); x_i(t), y_i(t)) - \epsilon\|^2. \qquad (27)$$

Now, the safety constraint is not hard and therefore the feasibility associated with it can be guaranteed. With this soft constraint, the problem (20) is feasible with all candidates and will choose the best candidate. Depending on the optimality of the trajectory candidates, the best solution could be either safe or colliding with other vehicles. In this specific case of a scenario where the traffic is very slow (~3m/s), we consider that "stopping and waiting" is a good strategy that reflects real-world driving behaviors. Therefore, we intentionally remained the safety constraint as hard, so that when it does not seem to be safe, the ego vehicle stops and waits. Nevertheless, there could be alternatives when solutions become infeasible and the examples include: (i) increasing the speed reference or (ii) decreasing the speed reference, which enables explorations of other gaps. It is also possible to consider these exploring trajectories as additional "trajectory candidates" with extra computational expenses, which is left to a control engineer as a design choice.

E. Limitation, Extension, and Future Work

The primary limitation of this study is related to its validation. Recall that the validation is under the recognized CARLA simulation with the practical implementation setup for planning and control (using ROS nodes). However, traffic vehicles are simulated using IDM, where the driving maneuvers may significantly differ from that of real vehicles on road. Furthermore, our validation scenarios are with the extremely dense traffic (for both "sparse" and "dense" cases) that may not represent real-world scenarios (where a "free space" occasionally exists). As a next step, we plan to incorporate a

TABLE III
QUANTITATIVE/COMPARATIVE ANALYSIS

Model	Scenario	Num. Runs	Time	Succ. (%)	Coll. (%)	Time out (%)	Brake Avg	Thr. Avg	Acc. Max	Brake Jerk Avg	Thr. Jerk Avg	Ang. Acc. Avg	Ang. Acc. Max	Ang. Jerk Avg	Ang. Jerk Max
Ours	coop-sparse	100	42.29	100	0	0	-0.215	0.175	0.675	-0.175	0.18	0.41	2.61	0.375	3.845
	coop-dense	100	40.875	100	0	0	-0.205	0.195	0.825	-0.185	0.17	0.385	2.46	0.355	3.345
	agg-sparse	100	41.065	100	0	0	-0.18	0.175	0.765	-0.165	0.15	0.41	2.705	0.37	3.145
	agg-dense	100	40.01	100	0	0	-0.2	0.17	0.73	-0.17	0.17	0.46	2.845	0.43	4.28
Game Tree	coop-sparse	50	57.88	100	0	0	-0.1	0.11	0.58	-0.09	0.08	0.52	4.31	0.42	5.42
	coop-dense	50	51.44	96	4	0	-0.11	0.08	0.53	-0.06	0.08	0.53	3.92	0.42	5.75
	agg-sparse	50	48.87	96	4	0	-0.13	0.12	0.6	-0.1	0.09	0.57	4.32	0.46	5.46
	agg-dense	50	51.58	98	2	0	-0.14	0.07	0.53	-0.06	0.08	0.51	3.98	0.4	5.68
Probabi-listic	coop-sparse	50	42.34	100	0	0	-0.18	0.19	0.81	-0.18	0.16	0.73	4.18	0.61	4.27
	coop-dense	50	46.16	100	0	0	-0.18	0.18	0.79	-0.18	0.15	0.62	3.66	0.5	4.02
	agg-sparse	50	41.21	96	4	0	-0.18	0.2	0.91	-0.18	0.16	0.86	4.65	0.74	5.44
	agg-dense	50	45.21	98	0	2	-0.17	0.18	0.83	-0.18	0.15	0.62	3.61	0.62	5.07

TABLE IV
SENSITIVITY ANALYSIS WITH LOCALIZATION AND PERCEPTION NOISE, IN
AGGRESSIVE-DENSE SCENARIO.

Noise Scale	Success Rate (%)	
	W/ Localization noise	W/ Perception noise
1x	95	100
2x	85	95
5x	60	95

human driver in the simulation to drive one of the neighboring vehicles and to validate it under diverse scenarios (including moderate traffic scenarios) to reduce sim-to-real gaps. We are also in the progress of running on-road experiments at a test facility in Japan.

The second limitation is related to safety. Although safety is not guaranteed in dense traffic, there exist several approaches to enhancing safety: (i) improving the prediction accuracy of other vehicles' cooperative behavior (as shown in [17]), (ii) integrating distributionally robust methods [41] that learns prediction errors online, (iii) similarly, integrating a distribution of predicted positions instead of single position prediction, and (iv) dynamic adjustments of a selected candidate based on predictions/errors, e.g., reducing a steering angle to increase the inter-vehicle gap to a rear vehicle. These are under active research.

Lastly, the optimal solutions from Eqn. (9) do not necessarily represent the optimal solutions from Eqn. (20), i.e., the optimal solution in Eqn. (9) may be suboptimal in Eqn. (20). Recall, the optimization in Eqn. (20) is mainly to evaluate the safety (with predictions) while the optimization in Eqn. (9) is to generate smooth maneuvers for each intent. That said, the optimality of Eqn. (20) can enhance by generating multiple candidates for each intention. For example, for the lane-change intent, three maneuvers can be generated to represent immediate-change, move-forward-and-change, and slow-down-and-change. Generating multiple maneuvers for each intent also remains for future work. All that being said, the proposed method is a framework that combines optimization-based models with data-driven models (e.g., neural networks) which is not limited to a specific choice of methods.

V. CONCLUSIONS

This paper presents mathematical formulations for smooth lane-change control in dense traffic where inter-vehicle gaps are narrow. We build upon an MPC formulation for trajectory planning with moving objects, by integrating a state-of-the-art Recurrent Neural Network, Social Generative Adversarial Networks (SGAN) to estimate interactions between objects – together being called NNMPC. NNMPC generates smooth lane-change trajectories to improve passenger comfort. In particular, possible driving intentions are specified and refined to control candidates. Additionally, sensor noises, SGAN prediction errors, and recursive feasibility for control are incorporated to increase robustness. The simulation studies with CARLA validate the smoothness of the trajectory and computational efficiency as well as the robustness to traffic density and cooperativeness of other drivers.

ACKNOWLEDGEMENT

REFERENCES

[1] V. L. Neale, T. A. Dingus, S. G. Klauer, J. Sudweeks, and M. Goodman, "An overview of the 100-car naturalistic study and findings," *National Highway Traffic Safety Administration, Paper*, vol. 5, p. 0400, 2005.

[2] M. Peden, R. Scurfield, D. Sleet, D. Mohan, A. A. Hyder, E. Jarawan, C. D. Mathers, *et al.*, "World report on road traffic injury prevention," 2004.

[3] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annual Reviews in Control*, vol. 45, pp. 18–40, 2018.

[4] S. Bae, Y. Kim, Y. Choi, J. Guanetti, P. Gill, F. Borrelli, and S. J. Moura, "Ecological adaptive cruise control of plug-in hybrid electric vehicle with connected infrastructure and on-road experiments," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 1, p. 011109, 2022.

[5] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson, "If, when, and how to perform lane change maneuvers on highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 68–78, 2016.

[6] D. J. Sun and L. Elefteriadou, "Lane-changing behavior on urban streets: A focus group-based study," *Applied ergonomics*, vol. 42, no. 5, pp. 682–691, 2011.

[7] M. Rahman, M. Chowdhury, Y. Xie, and Y. He, "Review of microscopic lane-changing models and future research opportunities," *IEEE transactions on intelligent transportation systems*, vol. 14, no. 4, pp. 1942–1956, 2013.

[8] G. Schildbach and F. Borrelli, "Scenario model predictive control for lane change assistance on highways," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 611–616.

[9] J. Chen, P. Zhao, T. Mei, and H. Liang, "Lane change path planning based on piecewise bezier curve for autonomous vehicle," in *Proceedings of 2013 IEEE International Conference on Vehicular Electronics and Safety*. IEEE, 2013, pp. 17–22.

[10] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 583–601, 2003.

[11] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," *arXiv preprint arXiv:1906.11021*, 2019.

[12] J. E. Naranjo, C. Gonzalez, R. Garcia, and T. De Pedro, "Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 438–450, 2008.

[13] F. You, R. Zhang, G. Lie, H. Wang, H. Wen, and J. Xu, "Trajectory planning and tracking control for autonomous lane change maneuver based on the cooperative vehicle infrastructure system," *Expert Systems with Applications*, vol. 42, no. 14, pp. 5932–5946, 2015.

[14] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, vol. 2, 2016.

[15] Y. Hu, A. Nakhaei, M. Tomizuka, and K. Fujimura, "Interaction-aware decision making with adaptive strategies under merging scenarios," in *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2019.

[16] D. Isele, "Interactive decision making for autonomous vehicles in dense traffic," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3981–3986.

[17] S. Bae, D. Saxena, A. Nakhaei, C. Choi, K. Fujimura, and S. Moura, "Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent neural network," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1209–1216.

[18] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in dense traffic with model-free reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5385–5392.

[19] K. Lee, D. Isele, E. Theodorou, and S. Bae, "Spatiotemporal costmap inference for mpc via deep inverse reinforcement learning," *IEEE Robotics and Automation Letters*, 2022.

[20] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[21] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Conference on Artificial Intelligence*. AAAI, 2018.

[22] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2018.

[23] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*. ACM, 2018, pp. 303–314.

[24] 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.

[25] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[26] C. Choi, A. Patil, and S. Malla, "Drogon: A causal reasoning framework for future trajectory forecast," in *Proceedings of the Conference on Robot Learning 2020*. IEEE, 2020.

[27] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[28] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[29] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback control theory*. Courier Corporation, 2013.

[30] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August, pp. 1094–1099, 2015.

[31] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 812–818.

[32] H. Febbo and D. Isele, "Accurate trajectory following for automated vehicles in dynamic environments," in *2020 American Control Conference (ACC)*, 2020.

[33] T. Tashiro, "Vehicle steering control with mpc for target trajectory tracking of autonomous reverse parking," in *2013 IEEE International Conference on Control Applications*. IEEE, 2013.

[34] R. Attia, R. Orjuela, and M. Basset, "Combined longitudinal and lateral control for automated vehicle guidance," *Vehicle System Dynamics*, vol. 52, no. 2, pp. 261–279, 2014.

[35] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1094–1099.

[36] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 2335–2340.

[37] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *Dynamic systems and control conference*, vol. 44175, 2010, pp. 265–272.

[38] K. E. Atkinson, *An introduction to numerical analysis*. John wiley & sons, 2008.

[39] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[40] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[41] A. Kandel, S. Park, H. E. Perez, G. Kim, Y. Choi, H. J. Ahn, W. T. Joe, and S. J. Moura, "Distributionly robust surrogate optimal control for large-scale dynamical systems," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 2225–2231.

[42] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[43] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[44] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[45] J. Wei, J. M. Dolan, and B. Litkouhi, "Autonomous vehicle social behavior for highway entrance ramp management," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 201–207.

[46] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[47] J. Alman and V. V. Williams, "A refined laser method and faster matrix multiplication," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 522–539.

[48] J. T. Wen and S. H. Murphy, *PID control for robot manipulators*. Rensselaer Polytechnic Institute, 1990.

[49] S. Valluri and V. Kapila, "Stability analysis for linear/nonlinear model predictive control of constrained processes," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, vol. 3. IEEE, 1998, pp. 1679–1683.

APPENDIX
PROOF OF CONTROL STABILITY

We refer to [48] for the stability of the PID controller (for the speed control) and we hereby show the stability

of the steering control which is based on MPC. MPC with constraints (i.e., the nonlinear vehicle rotation model and curvature constraints from [33]) does not typically guarantee stability and requires significant efforts to find a Lyapunov function [49] – without any guarantee of existence. Thus, we show that the deviation of the steering control with respect to a reference is bounded. Also, recall that the reference trajectory is obtained from (9), not from (20).

Proposition 1:

Given a feasible solution from (9), the maximum deviation of steering control $\tilde{\delta}$ from reference δ^{ref} is bounded by a finite value K over a control horizon T, i.e.,

$$\max_{\tau \in \{0,\dots,T\}} \|\delta^{\text{ref}}(\tau) - \tilde{\delta}(\tau)\| \le K \quad (28)$$

Proof. At any time step $\tau \in \{0,\dots,T\}$, the reference steering angle from (9) is bounded by $\mathcal{K}_{\max}$, i.e., $-\mathcal{K}_{\max} \le \delta(\tau) \le \mathcal{K}_{\max}$ (The reference trajectory is feasible and therefore (9b) and (9c) must suffice). Now, at any time step τ, the maximum deviation is achieved when the curvature constraints (9b) and (9c) are active, i.e., $\delta^{\text{ref}} = \mathcal{K}_{\max}$ or $\delta^{\text{ref}} = -\mathcal{K}_{\max}$. And we will check if the deviation at each case is bounded. First, if $\delta^{\text{ref}} = \mathcal{K}_{\max}$,

$$\max_{\tau \in \{0,\dots,T\}} \|\delta^{\text{ref}}(\tau) - \tilde{\delta}(\tau)\| \quad (29)$$

$$= \max_{\tau \in \{0,\dots,T\}} \|\mathcal{K}_{\max} - \tilde{\delta}(\tau)\| \quad (30)$$

$$\le \mathcal{K}_{\max} + \max_{\tau \in \{0,\dots,T\}} \|-\tilde{\delta}\| \quad (\because \mathcal{K}_{\max} > 0) \quad (31)$$

$$= \mathcal{K}_{\max} + \tilde{\delta}_{\max}. \quad (32)$$

Similarly, if $\delta^{\text{ref}} = -\mathcal{K}_{\max}$,

$$\max_{\tau \in \{0,\dots,T\}} \|\delta^{\text{ref}}(\tau) - \tilde{\delta}(\tau)\| \quad (33)$$

$$= \max_{\tau \in \{0,\dots,T\}} \|-\mathcal{K}_{\max} - \tilde{\delta}(\tau)\| \quad (34)$$

$$\le \mathcal{K}_{\max} + \max_{\tau \in \{0,\dots,T\}} \|-\tilde{\delta}\| \quad (\because \mathcal{K}_{\max} > 0) \quad (35)$$

$$= \mathcal{K}_{\max} + \tilde{\delta}_{\max}. \quad (36)$$

Therefore, the following suffices

$$\max_{\tau \in \{0,\dots,T\}} \|\delta^{\text{ref}}(\tau) - \tilde{\delta}(\tau)\| \le \mathcal{K}_{\max} + \tilde{\delta}_{\max} = K, \quad (37)$$

which concludes the proof. ∎

David Isele received his B.E. degree in electrical engineering from The Cooper Union: Albert Nerken School of Engineering, in New York City. He received his M.S.E. degree in Robotics and his Ph.D. degree in Computer and Information Science from The University of Pennsylvania.

David is currently a senior scientist at Honda Research Institute US. His research interests include applications of machine learning and artificial intelligence to robotic systems, with a focus on strategic decision making for autonomous vehicles.

Alireza Nakhaei received his PhD degree in computer science and robotics from LAAS-CNRS in Toulouse, France.

Alireza's research interests include employing artificial intelligence in decision making algorithms for self-driving vehicle and leveraging the autonomous driving capabilities to improve the performance of driver-assistance systems (ADAS).
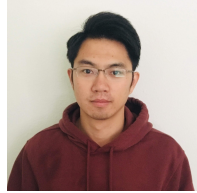
Peng Xu received his B.E. degree in Aircraft Power Engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2010 and his Master degree in Naval Architecture and Ocean Engineering from Shanghai Jiao Tong University, Shanghai, China in 2013. He also received his second Master degree in Electrical Engineering from Case Western Reserve University, Ohio in 2018.

His research interests include applying probabilistic methods, optimization based methods and learning based methods on decision making and motion planning for autonomous vehicles and mobile robots.

Alexandre Miranda Añon received his Dipl. Ing degree from the School of Industrial Engineering of Barcelona (ETSEIB-UPC), Catalonia. He received his M.S.E. degree in Robotics from the University of Pennsylvania.

Alex is currently a Research Engineer and the Assistant Lead Engineer for the System Engineering team at Honda Research Institute USA. His interest include the use of artificial intelligence for robotics, autonomous systems and self-driving vehicles with a focus on decision making and motion planning algorithms.

Sangjae Bae received the B.S. degree in electronics engineering from Kyung Hee University, Korea, in 2015, and received the Ph.D. degree in systems engineering from the University of California, Berkeley, in 2020.

Dr. Bae is currently a senior scientist at Honda Research Institute, USA, where he works on applications of optimization, control, and machine learning, with a focus on autonomous vehicles.

He was nominated for the IFAC Young Author Award at the 2nd IFAC Workshop on Cyber-Physical and Human Systems in 2018.

Chiho Choi received a B.S. from Hanyang University, Korea, and a M.S. in electrical engineering from University of Southern California. He received his Ph.D. in electrical and computer engineering from Purdue University in 2018.

Dr. Choi is currently a senior scientist at Honda Research Institute USA. His research interests span the fields of computer vision, machine learning, and robotics with a focus on understanding and prediction of human behavior for the safe operation of vehicles and robots designed to cooperate with humans.
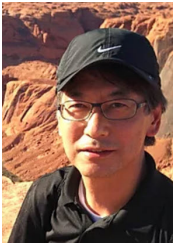
Kikuo Fujimura received the B.S. and M.S. degrees in Information Science from the University of Tokyo and the Ph.D. degree in Computer Science from the University of Maryland, College Park in 1989. After working at Oak Ridge National Laboratory and Ohio State University (Columbus), he joined Honda R&D in 1998, where he was engaged in research on intelligent systems and human-robot interaction with Honda's humanoid robot ASIMO.

He is currently Director Innovation at Honda Research Institute USA in San Jose, California, where he directs teams of researchers working on automated driving, knowledge discovery and informatics, human machine interfaces, and intelligent robotics. His research interests include artificial intelligence for mobility, human robot interaction, and HCI.

He has authored/co-authored one book and over 100 research papers in refereed conferences and journals and has been granted over 20 patents. He currently serves as an Associate Editor of IEEE Transactions on Intelligent Vehicles.

Scott Moura received a B.S. degree (2006) from the University of California, Berkeley, and M.S. (2008) and Ph.D. (2011) degrees from the University of Michigan, Ann Arbor. He was a Postdoctoral Fellow at the University of California, San Diego in 2011-2013.

Dr. Scott Moura is currently the Clare and Hsieh Wen Shen Endowed Distinguished Professor, Director of eCAL, PATH Faculty Director, and Chair of Engineering Science at UC Berkeley. From 2015 – 2021 he was PI at the Tsinghua-Berkeley Shenzhen Institute. In 2013, he was a Visiting Researcher at MINES ParisTech, France. His research includes control, optimization, and data science for batteries, electrified vehicles, and climate-tech.

Dr. Moura has received the ASME Dynamic Systems and Control Division Young Investigator Award, National Science Foundation CAREER Award, Carol D. Soc Distinguished Graduate Student Mentor Award, O. Hugo Shuck Best Paper Award, UC Presidential Postdoctoral Fellowship, and NSF Graduate Research Fellowship.