System and Experiments of Model-Driven Motion Planning and Control for Autonomous Vehicles

Shaobing Xu, Robert Zidek, Zhong Cao, *Member, IEEE*, Pingping Lu, Xinpeng Wang, *Graduate Student Member, IEEE*, Boqi Li, and Huei Peng

***Abstract*—This article presents a model-based motion planning and control system for autonomous vehicles and its experimental validation. The system consists of four modules: 1) global routing; 2) behavior planner; 3) local trajectory generation; and 4) trajectory tracking. The algorithm and software of each module are detailed, including a behavior planner with unified models to handle typical scenarios in both highway and urban driving, a deterministic sampling algorithm for robust responsive trajectory generation, and a dynamics-and-delay-aware preview algorithm to achieve accurate trajectory tracking. The developed system is implemented and tested at the Mcity test facility with a full-size automated car and a dozen of challenging traffic scenarios.**

***Index Terms*—Autonomous vehicles, behavior planner, deterministic sampling, motion planning, preview control.**

I. Introduction

A. Motivation

AUTONOMOUS vehicles saw significant development efforts over the last two decades [1]–[3]. Wide deployments are not happening both because of high hardware costs and performance challenges in perception, prediction, and motion planning in complex situations.

This article focuses on motion planning and control, two of the crucial functions for autonomous driving. Various algorithms were proposed in the literature [4]–[8], which can be roughly divided into model-based and data-driven approaches. For example, the reinforcement learning design for highway driving in our papers [5], [6]. The approach we follow in this article is model based. Model-based approaches usually solve the motion planning and control problem sequentially, e.g., stack of routing, decision making, trajectory planning, and control [4]. In the literature, many papers focused on a certain algorithm or improvement targeting one or two modules, or concentrated on a specific scenario (e.g., left-turn, cut-in, or merge) [4], [7]. However, gaps from the nearly independent various algorithms to a unified system that covers the full stack and can handle a variety of realistic driving situations do exist. This article targets this gap and presents a model-based motion planning and control stack for autonomous vehicles, including algorithms, software, and simulation/field experiments. Compared to the existing reports [1]–[3], this article presents different algorithms and more experiments in complex and challenging scenarios.

Manuscript received November 12, 2020; revised May 24, 2021; accepted November 17, 2021. This work was supported by the TRI-Sponsored Project "Intelligent and Automatic Motion Planning for Self-Driving Vehicles." This article was recommended by Associate Editor R. Roberts. *(Corresponding author: Shaobing Xu.)*

Shaobing Xu, Pingping Lu, Xinpeng Wang, Boqi Li, and Huei Peng are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: xushao@umich.edu; pingpinl@umich.edu; xinpengw@umich.edu; boqili@umich.edu; hpeng@umich.edu).

Robert Zidek is with the Toyota Research Institute, Ann Arbor, MI 48105 USA (e-mail: robert.zidek@tri.global).

Zhong Cao is with the School of Vehicle and Mobility, Tsinghua University, Beijing 10083, China (e-mail: caoc15@mails.tsinghua.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2021.3131141.

Digital Object Identifier 10.1109/TSMC.2021.3131141

B. Literature Review of Motion Planning

There exist several reviews that summarize the state of the art of motion planning for autonomous vehicles [4], [7], [8]. On the whole, time-independent path planning has been well-solved using algorithms such as the graph-based planner (e.g., the Dijkstra algorithm, A* family, and state lattices) [9] and the interpolating curve planner (e.g., Bezier, polynomial, and spline) [10], [11]. For safety-critical tasks, the trajectory-based planner, which specifies both path and speed profiles that connect the initial state and a predefined terminal state, is usually required [8]. One approach is to use numerical nonlinear optimization [12], e.g., MPC [13]. Another typical approach is the rapidly exploring random tree (RRT) and its variants [14]. They can search nonconvex, high-dimensional spaces by randomly building a space-filling tree to reach the goal terminal state, and can also be considered as a Monte-Carlo method to bias search into the largest Voronoi regions of a graph in a configuration space. RRT had been implemented in self-driving cars successfully, but the random search does not well match the requirements of self-driving on structured roads, which demands planned optimality (not randomness) and human-like smooth operations. Different from the random search and targeting a single fixed terminal state, Lacaze *et al.* proposed a multiresolutional architecture for obstacle avoidance in outdoor mobility [15]; this architecture uses offline dynamical simulations to build massive precalculated trajectories, called ego-graph, corresponding to multiple permissible terminal states. The planner runs recursively with a receding horizon and the resulted trajectory can respond to obstacles robustly and smoothly. Werling *et al.* leveraged this strategy and proposed an optimal-control-based solution. It decouples lateral and longitudinal motions in the Frenet coordinates of

the street [16] and shows that a 5th-order polynomial connecting a given pair of initial/terminal states is optimal when a simplified kinematic model and a jerk-based cost function are used. In the implementation, the trajectory is solved by sampling in the terminal manifolds. This article will leverage and improve this approach to generate local trajectories.

Decision making refers to the strategy of interacting with multiple other road users under the roadway constraint over a time horizon. Existing decision-making algorithms can be classified as semi-interactive [1], [2], [17]–[19] and interactive [20], [21]. The former follows a two-layer structure, i.e., prediction first and then decision [17]. It considers other agents' responses once in each cycle and ignores interactions among the agents. In this strategy, model-based approaches are frequently used, e.g., finite-state machine [1], [2] and risk potential optimization [18]. The latter learns interactions from data or experiences, e.g., using the partially observable Markov decision process (POMDP) [21] or reinforcement learning [5], in which historical data or failure cases parameterize the involved hidden probability matrix or neural network. This article adopts the semi-interactive strategy.

C. Contribution

The main contribution of this article is the systematic design and simulation/field experiments of a model-based motion planning and control stack that can be used to drive safely and efficiently in multiple urban and highway traffic scenarios. It consists of a hierarchical framework which comprises: 1) an A*-based shortest time/distance routing module; 2) a behavior planner with unified models that handle typical scenarios; 3) a deterministic sampling algorithm for trajectory generation; and 4) a preview control algorithm that considers vehicle dynamics, future road curvature, and system lag/delay for accurate trajectory tracking. We will present experimental results on a test vehicle interacting with both simulated and real traffic.

The algorithms are declared as secondary contributions. The A* is a widely used existing algorithm. The unified models of the behavior planner and dynamics-and-delay-aware preview tracking control are designed in our work. The deterministic sampling trajectory generation leverages the approach in [16], but we propose several improvements: new cost functions, sampling strategy, inequality constraints based on collision and road-departure considerations, and efficient constraint checking methods. In addition, the approach in [16] handles driving on highways only, or more accurately, on roads, while our system can handle both highway and urban driving including intersections.

The remainder of this article is organized as follows: Section II presents the system framework. Sections III–VI present the routing, trajectory generation, behavior planning, and control algorithms, respectively. Experimental results are reported in Section VII, and Section VIII concludes this article.

II. Motion Planning and Control Framework

As shown in Fig. 1, a self-driving system can include several subsystems. This article focuses on the routing, behavior planning, trajectory generation, and motion control modules; other modules are not covered.

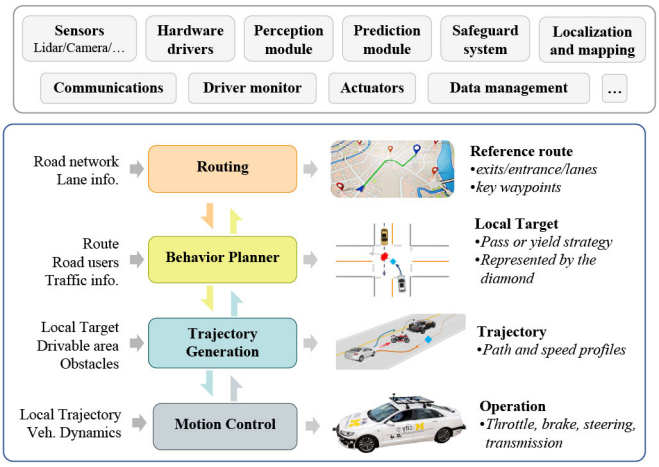


Fig. 1. Framework of our motion planning and control system, i.e., the four modules in the blue box.

The routing module is responsible for finding a feasible or optimal reference path from the origin to a given destination using digital maps. The reference path is represented by a sequence of lanes with entrance and exit, or a list of waypoints.

The behavior planner, also known as the decision maker in some papers, provides farsighted higher level strategies over a horizon for given scenarios such as yield, stop, lane change, merge, and car following. These behavior-level strategies are represented by a local target to be reached a few seconds later, with reference latitude, longitude, speed, and acceleration information to guide the vehicle's movement.

The trajectory generation module optimizes the ego car's path and speed profile over a horizon (6 s in this article) to pursue the local target given by the behavior planner under the constraints of surrounding objects, drivable area, and vehicle dynamics.

Given the planned trajectory, the motion control module generates throttle/brake and steering commands to smoothly track it. This article designs an optimal preview control that considers vehicle dynamics, input delay, and steering lag. It is able to achieve accurate and smooth tracking and increase the stability margin. In the following four sections, we will describe the design of each module.

III. Map and Routing

A. Digital Map

High-definition (HD) maps provide useful prior information for highly automated vehicles. Here, we select the Mcity as the target testing track and develop its HD map. As shown in Fig. 2, it digitizes the following.

1) Road segments and type, i.e., highway, ramp, roundabout, and normal roads.
2) Lane features, i.e., centerline, width, lane marking type, and speed limit.
3) Connection information, i.e., exit, entrance, priority, and direction of turning of each connector.

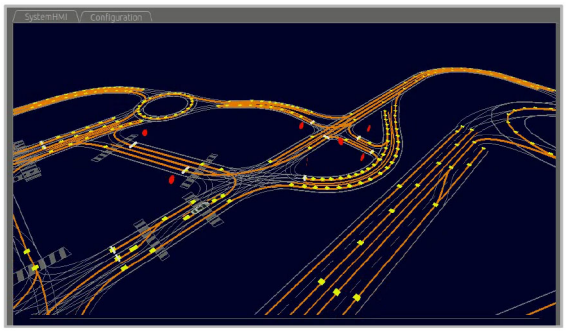


Fig. 2. Developed HD Map of Mcity. The yellow dots are the waypoints and the orange curves are the center lines of lanes or the yellow lane markings.

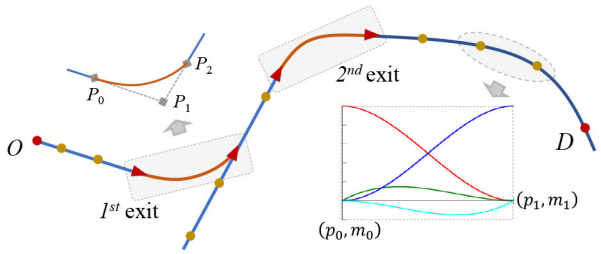


Fig. 3. Schematic diagram of the routing result (red circles/triangles) and path smoothing.

4) Traffic signal information, i.e., stop line, light id, and channel id corresponding to each lane.
5) Waypoints, including their id and position.
6) Crosswalk, including their position and shape.
7) Stop and yield signs.

This map may be different from the commercial HD maps some companies are using (not yet standardized), but it provides the necessary information for our purposes.

B. Routing and Path Smoothing

We implemented the shortest-distance and shortest-time routing using the A* algorithm [9]. For the latter, the time cost on each lane is estimated by its length and speed limit. Temporary lane closure information is used to dynamically correct the map. This article does not consider dynamic traffic, but the routing algorithm considering the traffic, energy consumption, and time/battery constraints was separately reported in our previous paper [23].

As shown in Fig. 3, the routing module outputs a sequence of exits or connectors, which specify the exiting waypoint of the first lane and the entering waypoint of the next lane. We then extract all waypoints from the origin to the destination. The path of a connector is not available in the map; thus, we use the Bezier curve to connect the exit and entrance to form the path

$$P(z) = (1-z)^2 P_0 + 2(1-z)zP_1 + z^2 P_2 \tag{1}$$

where $P \in \mathbb{R}^2$, P_0/P_2 stands for the exit/entrance, P_1 is the intersection of their tangents, and $z \in [0, 1]$ is the variable of the function. Usually, the waypoints within a lane are sparse, refer to Fig. 2, so we use the Hermite spline to connect any two adjacent waypoints to smooth the reference path

$$P(z) = \left(2z^3 - 3z^2 + 1\right)p_0 + \left(z^3 - 2z^2 + z\right)m_0 + \left(-2z^3 + 3z^2\right)p_1 + \left(z^3 - z^2\right)m_1 \tag{2}$$

where $p/m \in \mathbb{R}^2$ is the position/tangent of a waypoint. With these interpolation schemes, a smooth reference path is obtained. In addition, the reference speed of each lane is set to be the speed limit; the reference speed of a connector changes linearly from the speed limit of the previous lane to the next.

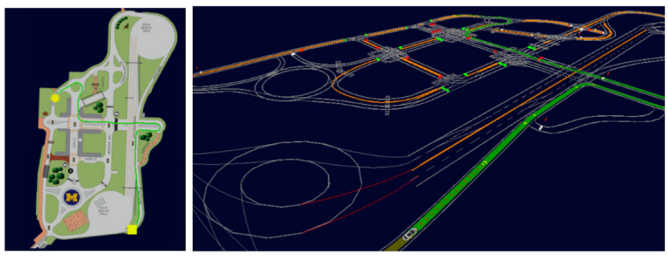


Fig. 4. Routing and path smoothing results. The green curve is the planned reference path and the orange curves are road markings.

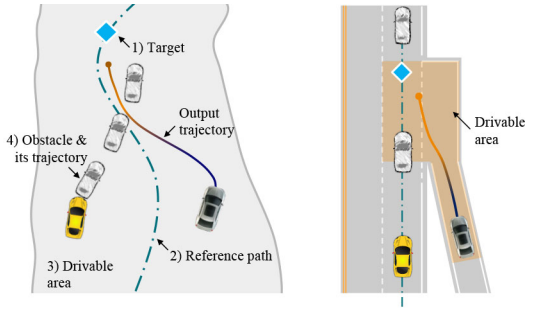


Fig. 5. Definition of the trajectory generation problem. The blue diamonds stand for the local target. The orange spots mean the terminal position of a trajectory.

C. Implementation

Fig. 4 shows our implementation of the routing and path smoothing algorithms using C++, Qt, and OpenGL. Once a destination is selected by clicking the map, the software computes a smooth reference path. This process takes less than 5 ms to finish. The rendering of the reference path and the map is shown in Fig. 4.

IV. Trajectory Generation

This section describes the deterministic sampling algorithm for local trajectory generation. Note that we intentionally present this section before the behavior planning in Section V for better readability.

A. Problem Formulation

As shown in Fig. 5, the trajectory generation module aims to optimize the local trajectory using four inputs: 1) local target from the behavior planner; 2) reference path from the routing module; 3) drivable area from the digital map;

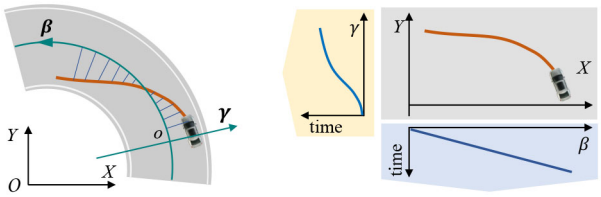


Fig. 6. Frenet (or street) coordinate system and motion decomposition.

and 4) information of obstacles including size and all potential trajectories from the perception, communication, or the prediction module.

We formulate the trajectory generation as a receding horizon optimal control problem (OCP)

$$\begin{aligned} &\mathcal{J} = \int \mathcal{L}(\mathbb{x}, \mathbb{u})\mathrm{d}t + \emptyset(t_f, \mathbb{x}_f) \\ &\text{s.t.} \\ &\qquad \dot{\mathbb{x}} = f(\mathbb{x}, \mathbb{u}) \\ &\qquad \mathcal{C}(\mathbb{x}, \mathbb{u}) \leq 0 \\ &\qquad \mathbb{x}(t_0) = \mathbb{x}_0 \end{aligned} \tag{3}$$

where $\mathcal{J}$, $\emptyset$, f, and $\mathcal{C}$ are the cost function, endpoint cost, system dynamics, and constraints, respectively. $\mathbb{x}_0$ and $\mathbb{x}_f$ are the (given) initial and the (free) terminal states. To better describe a trajectory, we introduce the Frenet (or street) coordinate system $\beta o \gamma$, as shown in Fig. 6. It can decouple the longitudinal and lateral motion, which are parallel to and perpendicular to the reference path, denoted by $\beta(t)$ and $\gamma(t)$, respectively. Note that this longitudinal/lateral motion differs from the concepts of vehicle dynamics defined in the vehicular coordinate system. With the decoupling, the longitudinal or lateral kinematic motion is modeled as an integrator control system,

$$\dot{\mathbb{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \mathbb{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mathbb{u}(t) \tag{4}$$

where $\mathbb{x} = [\beta, \dot{\beta}, \ddot{\beta}]$ or $[\gamma, \dot{\gamma}, \ddot{\gamma}]$ and $\mathbb{u} = \dddot{\beta}$ or $\dddot{\gamma}$ is the control input, the so-called jerk. Both β and γ are defined in the time domain; their synthesis forms the final path in the spatial domain. Note that this is a simplified kinematic model for trajectory generation only, the resulted trajectory will be checked by vehicle dynamics constraints and tracked by controllers considering dynamics, refer to Section VI. The hard constraints $\mathcal{C}$ we consider include the following six items.

1) Maximal/minimal vehicle speed $v \in [v_{\min}, v_{\max}]$.
2) Longitudinal acceleration $a_x \in [a_{x,\min}, a_{x,\max}]$.
3) Curvature of trajectory $c_t \in [c_{t,\min}, c_{t,\max}]$.
4) Lateral acceleration $a_y \in [a_{y,\min}, a_{y,\max}]$.
5) Confined inside the drivable area.
6) No collision with other objects.

The constraints 1)–4) relate to the vehicle dynamics. The cost function $\mathcal{J}$ is a critical component for trajectory planning since it directly identifies the quality of a trajectory. In our design, five factors are considered, i.e., smoothness, mobility (deviation from the desired target state), risk of leaving the drivable area, risk of being too close to obstacles,

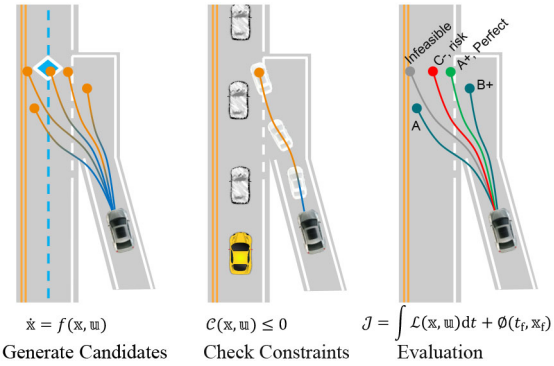


Fig. 7. Concept of deterministic sampling-based trajectory generation.

and consistency of the trajectories generated at two adjacent time steps. The detailed design of $\mathcal{J}$ will be discussed in Section IV-D.

B. Deterministic Sampling for Fast Computation

The trajectory generation problem is a nonlinear constrained OCP. To achieve real-time performance, we leverage the sampling strategy to solve it. The fundamental idea is to generate a finite number of candidate trajectories first and then search the optimal solution within the candidate set. This strategy can deliver a near-optimal solution (within grid accuracy) efficiently. It contains three steps, as shown in Fig. 7: 1) generate candidate trajectory set; 2) check the constraints $\mathcal{C}$ to filter out infeasible ones; and 3) evaluate all feasible candidates and select the optimal one. The development of emergency maneuvers when no feasible trajectory is available is left for future work.

We denote the sampling space as $\Theta = [t, \beta, \dot{\beta}, \ddot{\beta}, \gamma, \dot{\gamma}, \ddot{\gamma}]^T$, the local target as $\Theta_r = [t_r, \beta_r, \dot{\beta}_r, \ddot{\beta}_r, \gamma_r, \dot{\gamma}_r, \ddot{\gamma}_r]^T$, and its upper and lower bounds as $\Theta_{\max}$ and $\Theta_{\min}$. Note that the target Θ_r is a reference goal only; the vehicle does not need to hit it exactly. To generate different candidate trajectories, the main strategy is to sample various terminal states Θ_f around the target Θ_r, which then parameterize $\beta(t)/\gamma(t)$ and form the set of trajectories. To determine the trajectory β/γ that connects the initial state and a sampled terminal state, we leverage the terminal manifold method proposed in [16], which indicated that the optimal trajectory is a 5th-degree polynomial if the goal is to minimize the control input and deviation from the terminal states.

For each sampling cycle, the initial state in the space Θ, denoted by Θ_0, is known *a priori*; what we need is to sample Θ_f in Θ. Here, we use an asymmetrical rule to sample, i.e., split the sampling space $[\Theta_{\min}, \Theta_{\max}]$ into $[\Theta_{\min}, \Theta_r)$ and $(\Theta_r, \Theta_{\max}]$, and then independently sample in the two subspaces with equal gaps. For instance, sampling γ in $[\gamma_{\min}, \gamma_r)$ obeys

$$\{\gamma_i | \gamma_i = \eta\gamma_{\min} + (1-\eta)\gamma_r, \eta = i/N_l, i = 1, \ldots, N_l\}. \tag{5}$$

Note that $\gamma = \gamma_r$ (the local target Θ_r) is always sampled. Thus, the total number is $N_l + N_u + 1$, where N_l and N_u are the number of sampling points in $[\gamma_{\min}, \gamma_r)$ and $(\gamma_r, \gamma_{\max}]$. One example of sampling in a subspace $[t, \beta, \gamma]$ is shown in Fig. 8.

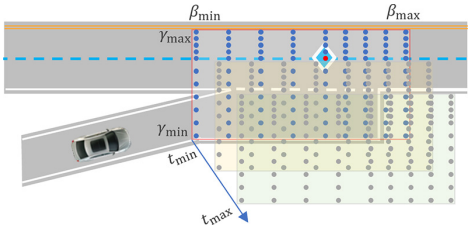


Fig. 8. Example of sampling in the subspace $[t, \beta, \gamma]$. The blue diamond stands for the local target, each blue/gray point stands for a terminal state sampled in the 3-D space $[t, \beta, \gamma]$.

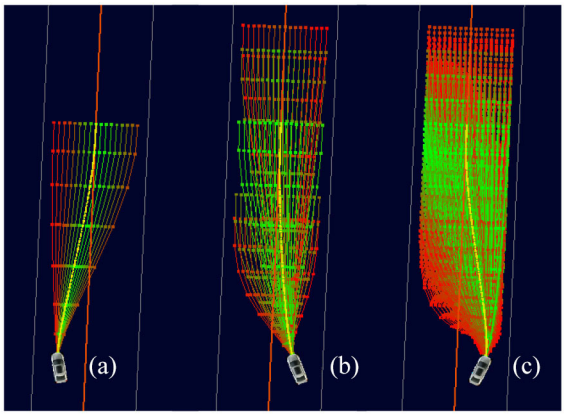


Fig. 9. Sampled trajectories. (a) 17 samples with 17 different γ_f. (b) 85 samples. (c) Full 1224 trajectories. The curve's color indicates the cost, green for lower, and red for higher cost. The bold yellow is the optima. The bold solid red curve is the reference path. The white curves are the boundaries of the drivable area.

Once Θ_f is sampled, the motion γ is described by

$$\gamma(t) = \sum_{i=0}^{5} \rho_i t^i \tag{6}$$

where $\rho_0 = \gamma_0$, $\rho_1 = \dot{\gamma}_0$, and $\rho_2 = 0.5\ddot{\gamma}_0$. The other three coefficients are determined by the sampled terminal state γ_f, $\dot{\gamma}_f$, and $\ddot{\gamma}_f$. $\beta(t)$ follows the same mechanism.

In our implementation, t_r is set to 6 s; namely, planning for 6 s into the future. The sampling parameters are listed in Table I. We sample in a 3-dimensional (3-D) subspace $[t, \beta, \gamma]$; $\dot{\gamma}_f$ and $\ddot{\gamma}_f$ are set to 0, meaning no lateral movement at $t = t_f$. β_f and $\ddot{\beta}_f$ are set by the behavior planner according to the specific scenarios. In total, 1224 trajectories are sampled, as shown in Fig. 9. This algorithm is called *deterministic sampling trajectory generation* in the rest of this article.

Note that γ and β are defined in the Frenet coordinate; they should be merged with the global reference path that can have arbitrary shapes and is usually given by a set of discrete points. Our strategy is to discretize the sampled trajectory with time gap Δt and then merge γ, β, and the reference path point by point. In this article, Δt is set to 0.25 s. An example of the merged trajectories in the global earth coordinate and the curvature c_t of the optimal trajectory is shown in Fig. 10.

C. Constraint Checking

The constraints $\mathcal{C}$ are used to remove infeasible trajectories before searching for the optimal trajectory. We check pointwise for each discretized trajectory. For constraints (1)

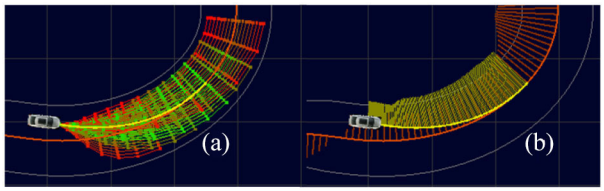


Fig. 10. Merge with the reference path: (a) merging the sampled trajectories with the curved reference path at a roundabout, 85 samples and (b) the calculated curvature of the optimal trajectory.

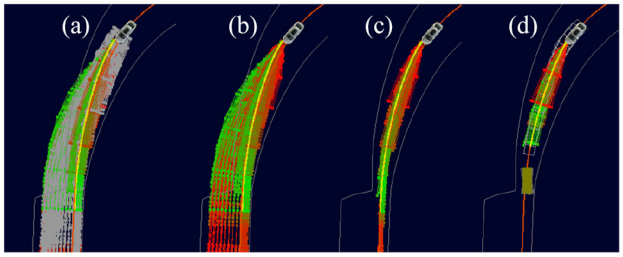


Fig. 11. Constraint check. (a) Full set of 1224 trajectories; the gray curves are the 611 infeasible trajectories identified by the constraints (1)–(4). (b) 613 feasible trajectories. (c) 274 feasible trajectories after checking the drivable area constraint 5. (d) 66 feasible trajectories after checking the collision constraint 6.

TABLE I
PARAMETER SETTING OF DETERMINISTIC SAMPLING

Variable	Sample or Not	$\Theta_{\min}$	$\Theta_{\max}$	N_l	N_u	Total Number	Default Value
t_f	Y	$0.55t_r$	$1.6t_r$	3	4	8	-
γ_f	Y	Right bound	Left bound	8	8	17	-
$\dot{\gamma}_f$	N	-	-	-	-	-	0
$\ddot{\gamma}_f$	N	-	-	-	-	-	0
β_f	Y	$0.6\beta_r$	$1.4\beta_r$	4	4	9	-
$\dot{\beta}_f$	N	-	-	-	-	-	-
$\ddot{\beta}_f$	N	-	-	-	-	-	-
Total Sampling		8×17×9=1224					

and (2), the closed form of vehicle speed v is

$$v = \left(\left[1 - c_r \gamma\right]^2 \dot{\gamma}^2 + \dot{\beta}^2 \right)^{0.5} \tag{7}$$

where c_r is the curvature of the reference path. To reduce computation load, $\dot{\beta}$ and $\ddot{\beta}$ are used to approximate the vehicle speed v and the longitudinal acceleration $\dot{v}$. For the constraint (4), the lateral acceleration a_y is estimated by $\dot{\beta}^2 c_t$. Fig. 11(1) and (2) shows an example, in which 611 out of 1224 trajectories were found to be infeasible by the constraints (1)–(4).

To check constraint (5) for staying inside drivable areas, we assume the reference path near the car has a fixed radius R_r, as shown in Fig. 12; then use a circle with radius R_v to represent the frontal area of the car. The gap from the car to the boundary of the drivable area is estimated by the distance d_b from the circle with radius R_v to the boundary

$$\begin{aligned} d_b &= R_b - R_v - \sqrt{\xi^2 + l_c^2 + 2 l_c \xi \sin(\Delta\theta_t + \theta_s)} \\ \xi &= \gamma + R_r \\ \theta_s &= -\text{asin}\left(l_f / R_t\right) \end{aligned} \tag{8}$$

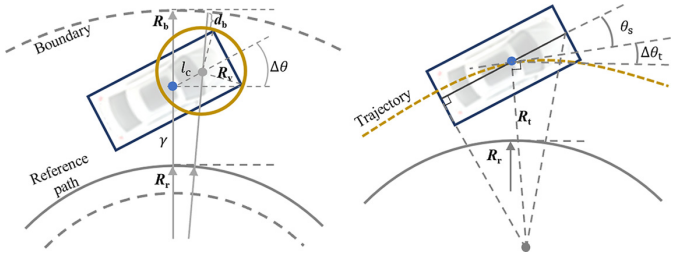


Fig. 12. Left: gap from the vehicle to the boundary of the drivable area. Right: vehicle slip angle estimated by the kinematics model.

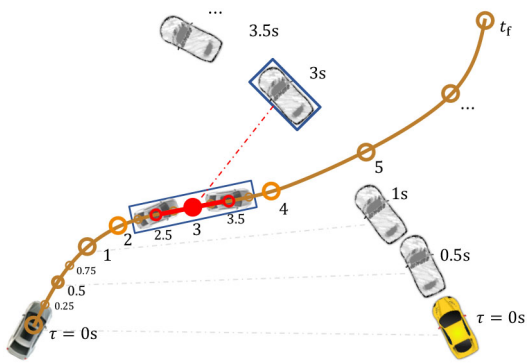


Fig. 13. Instant-to-period strategy for collision checking.

where R_b and R_t are the radii of curvature of the boundary and the planned trajectory, respectively, l_c/l_f is the distance from the ego car's center of gravity (c.g.) to the circle with radius R_v/rear axle; θ_s is the slip angle of the car and estimated by the kinematic model shown in Fig. 12; $\Delta\theta_t$ is the heading angle gap between the trajectory and the reference path. Fig. 11(c) shows 274 trajectories which pass checking the constraint (5).

For constraint (6), several methods have been proposed for fast collision check, e.g., using circles to approximate the objects' and the ego car's geometries [24]. In their implementations, the ego car at every future discretized time instant t is checked against all objects at the same instant t, called *instant-to-instant* checking. To avoid omissions, frequent checking is required; namely, Δt should be much smaller than 0.25 s used in this article. Different from the numerous instant-to-instant checking, we propose an *instant-to-period* checking strategy. As shown in Fig. 13, at instant t, the ego car is represented by one rectangle over the period $[t-\Delta t, t+\Delta t]$, namely the rectangle with a minimum area that covers the ego car's geometry at instant $t-\Delta t$, t, and $t+\Delta t$. Then, check if this larger rectangle intersects with the rectangle at instant t for each potential path of each object, see Fig. 14. When calculating the geometry in the unified earth coordinate, the slip angle of ego car is estimated by the kinematic model shown in Fig. 12. The checking window moves forward with a step Δt, which is set to 0.25 s. This strategy significantly reduces the number of checking and then improves real-time performance. Fig. 11(d) shows the final 66 feasible trajectories after the collision check.

D. Design of the Cost Function

The cost function considers the following five criteria.

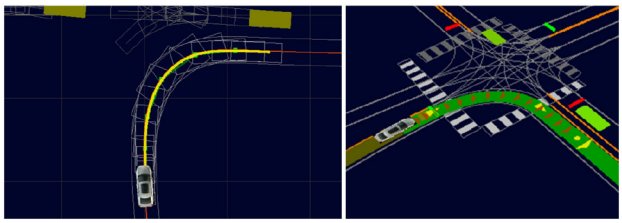


Fig. 14. Collision check of one trajectory during a right turn. The varying sizes of the rectangles in the left figure are caused by the vehicle slip angle and the road curvature.

1) *Smoothness*, measured by the jerk and the maximum lateral acceleration $a_{ym} = \max\{a_y(t), t\in[0, t_f]\}$, i.e.,

$$\mathcal{J}_s = w_a a_{ym}^2 + \int_0^{t_f} \left(w_\gamma \dddot{\gamma}^2 + w_\beta \dddot{\beta}^2 \right) d\tau \tag{9}$$

where w_a, w_γ, and w_β are weighting coefficients. Note γ and β are polynomials; thus, their jerk has closed-form expressions.

2) *Mobility or Deviation From the Target State:* If a trajectory exactly hits the given target, it is considered to have the best mobility. Otherwise, the deviation is penalized

$$\mathcal{J}_m = \left(\Theta_r - \Theta_f\right)^T \mathbf{w}_m \left(\Theta_r - \Theta_f\right) \tag{10}$$

where $\mathbf{w}_m$ is the weight matrix.

3) *Risk of Leaving the Drivable Area:* Being too close to the boundary of the drivable area is considered a risk. We quantify this risk using the minimal distance d_{bm} from a trajectory to the boundary over $[0, t_f]$

$$\mathcal{J}_d = \begin{cases} \text{discarded}, & d_{bm} < 0.1 \text{ m} \\ 0, & d_{bm} > \overline{d}_{bm} \\ w_d\left(\overline{d}_{bm} - d_{bm}\right)^2, & \text{else} \end{cases} \tag{11}$$

where $\overline{d}_{bm}$ is a constant and d_{bm} is calculated by (8).

4) *Risk of Being Too Close to Obstacles:* Being too close to an obstacle is also penalized, i.e.,

$$\mathcal{J}_o = \begin{cases} \text{discarded}, & d_{om} < 0.2 \text{ m} \\ 0, & d_{om} > \overline{d}_{om} \\ w_o\left(\overline{d}_{om} - d_{om}\right)^2, & \text{else} \end{cases} \tag{12}$$

where d_{om} is the minimal distance from a trajectory to all objects over $[0, t_f]$ and $\overline{d}_{om}$ is a constant. Equation (12) mainly considers the lateral gap, as a trajectory leading to small time headway has been discarded in the collision check.

5) *Consistency of the Optimal Trajectories:* Frivolous change of the optimal trajectories of two adjacent steps should be avoided. This is measured by the variation of terminal states between a candidate Θ_f and the previous optimal trajectory Θ_p

$$\mathcal{J}_\Delta = \left(\Theta_f - \Theta_p\right)^T \mathbf{w}_\Delta \left(\Theta_f - \Theta_p\right) \tag{13}$$

where $\mathbf{w}_\Delta$ is the weight matrix.

The sum of these five sub costs yields the final cost

$$\mathcal{J} = \mathcal{J}_s + \mathcal{J}_m + \mathcal{J}_d + \mathcal{J}_o + \mathcal{J}_\Delta. \tag{14}$$

TABLE II
SYMBOLS AND VALUES USED IN THE TRAJECTORY GENERATION

Symbol	Value	Sym.	Val.	Sym.	Val.
$v_{\max}/v_{\min}$	30/0 m/s	l_c	1.65 m	w_d	30
$a_{x,\max}/a_{x,\min}$	2.5/-7 m/s^2	l_f	1.2 m	w_o	150
$a_{y,\max}/a_{y,\min}$	4.0/-4.0 m/s^2	w_a	20	$\bar{d}_{om}$	1.2 m
$c_{t,\max}/c_{t,\min}$	0.16/-0.16	w_γ	3	$\bar{d}_{om}$	1.5 m
R_v	1.2 m	w_β	1		
$\mathbf{w}_m$	[50, 180, 0, 0, 2, 0, 0]				
$\mathbf{w}_\Delta$	[0, 0.2, 0, 0, 1.5, 0, 0]				

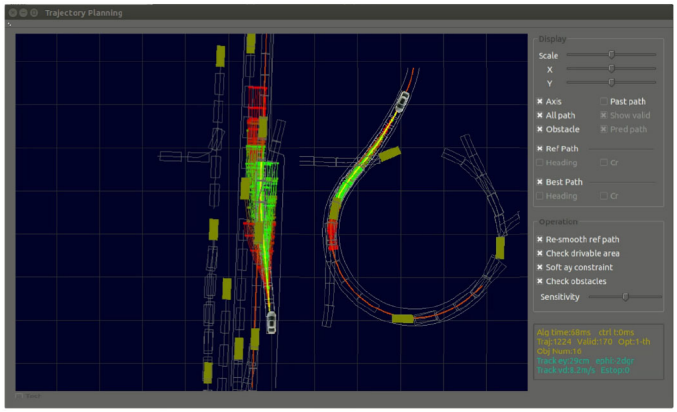


Fig. 15. Software of the deterministic sampling trajectory generation. The white rectangles show the multiple potential trajectories of each object (note: two scenarios are presented to provide more information).

All feasible trajectories are evaluated based on this cost function and the optimal one is selected. The related parameters are set in Table II. Examples can be found in Figs. 9–11.

E. Implementation

We use C++ and the robot operating system (ROS) to implement the deterministic sampling trajectory generation, as shown in Fig. 15. This software receives the ROS messages of the local target, reference path, and other objects from the corresponding modules and then broadcasts the optimal trajectory. The computation time per step is usually less than 100 ms. Refer to Fig. 23 for computation times from the field tests.

V. BEHAVIOR PLANNER

This section describes the target driving scenarios and behavior planning strategies. We cover the typical scenarios of: 1) driving on highways, including ramp merging, lane change, exiting from highways, car-following and 2) driving in urban areas, including responding to traffic lights, stop signs, pedestrians, unprotected left turns, and entering/exiting roundabouts and intersections. In the following, three unified models are presented to handle these scenarios.

A. Merge, Exit, and Lane Change

Merge and exit are treated as special lane changes within limited drivable areas. Thus, we consider them as one model.

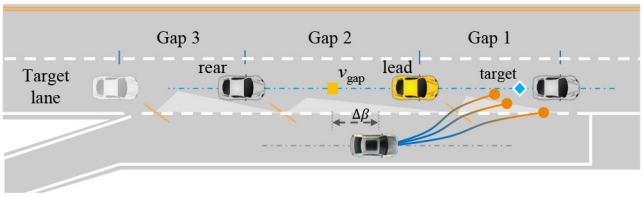


Fig. 16. Target setting of merge, exit, or lane change.

The first decision is when to trigger a lane change. For the merge scenario, a lane change is triggered when the ego car enters the ramp; the exit is triggered when the distance from the vehicle to the exit ramp is less than

$$d_{\text{exit}} = \varsigma N_L t_L v_{Lm} \tag{15}$$

where N_L is the number of lanes to cross, v_{Lm} is the speed limit, ς is a factor proportional to traffic density, and t_L is the response time.

Once a lane change is triggered, the local target, reference path, and drivable area should be set. The reference path is switched from the ego lane to the target lane and the drivable area is set by the map. Then, a key decision is to identify the merge gap to take for the lane change. As shown in Fig. 16, all observable gaps in the target lane are identified first. Their validity is then checked using the gap length d_g and vehicles' speeds, i.e., valid if

$$\begin{aligned} d_g &> d_{\text{thd}} \\ d_{\text{thd}} &= \min\big(d_{\max}, \max\big(d_{\min}, t_m v_g\big)\big) \\ v_g &= \big(v_{gf} + v_{gr}\big)/2 \end{aligned} \tag{16}$$

where v_{gf}/v_{gr} is the front/rear car's speed of a gap, v_g is called the gap's speed, and t_m is a preset time constant.

To find the best gap, the following optimization problem is designed, where the criterion $\mathcal{J}_g$ assesses each valid gap:

$$\begin{aligned} \max_{\text{gap}} \ \mathcal{J}_g &= \min\big(\mathbb{T}_g, \mathbb{T}_{\max}\big) - w_a|a_e| \\ \mathbb{T}_g &= d_g/v_g \end{aligned} \tag{17}$$

where $\mathbb{T}_g$ is the time headway of the gap and w_a is the weight set to 5 if $a_e > 0$, otherwise set to 2. a_e is called the equivalent acceleration, by which the ego car can reach the reference point of the gap (the yellow square in Fig. 16) after t_e (set to 6 s here) assuming other cars cruise at constant speeds, i.e.,

$$\begin{aligned} a_e &= 2\frac{\Delta\beta + \Delta v t_e}{t_e^2} \\ \Delta\beta &= \beta_g - \beta \\ \Delta v &= v_g - \dot{\beta}. \end{aligned} \tag{18}$$

Once the optimal gap is selected, the local target after t_r seconds (the blue diamond) is set to

$$\begin{aligned} \beta_r &= t_r v_g + \Delta\beta \\ \dot{\beta}_r &= v_g, \gamma_r = 0, t_r = 6 \text{ s}. \end{aligned} \tag{19}$$

It is worth noting that even if the target is not well selected, the deterministic sampling-based trajectory generation will select the optimal trajectory among the candidate set and

Fig. 17. Target setting of car following.

ensure safety. One example can be found in Fig. 19, where the ego car decides to merge into the front gap due to its higher speed.

B. Car Following

In the car following scenario, we predict the lead vehicle's longitudinal motion β_{lv} by a fixed-acceleration rule:

$$\begin{aligned}\dot{\beta}_{lv}(t) &= \min\left(\dot{\beta}_{lv}, \max,\ \dot{\beta}_{lv}(0) + a_{lv}t\right)\\ \beta_{lv}(t) &= \int_0^{t_r} \dot{\beta}_{lv}(t)dt\end{aligned} \tag{20}$$

where a_{lv} is the acceleration of the lead vehicle. If a_{lv} is not available, we assume it is 0. The ego car is set to maintain a safe time headway to the lead vehicle, as shown in Fig. 17, so the local target is set to

$$\begin{aligned}\dot{\beta}_r &= \dot{\beta}_{lv}(t_r) - \mathbb{T}_{cf}a_{lv}\\ \beta_r &= \beta_{lv}(t_r) - \beta_s - \mathbb{T}_{cf}\dot{\beta}_r\\ t_r &= 6 \text{ s},\\ \gamma_r &= 0\end{aligned} \tag{21}$$

where $\mathbb{T}_{cf}$ is the desired time headway and β_s is a constant.

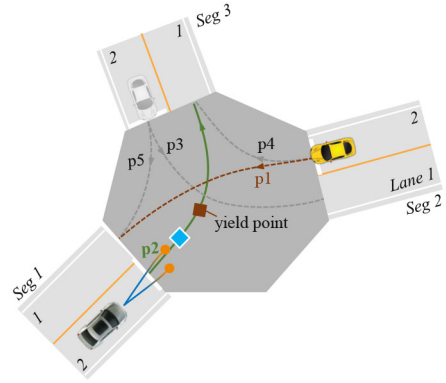
C. Yielding

Different from driving on highways or other local roads, the key of driving around intersections is to properly yield to other road users. Here, we generalize the typical scenarios of driving at intersection or junction (e.g., protected or unprotected left turns, right turns, yield to stop signs/pedestrians/traffic lights, and entering/exiting roundabouts) as a unified model shown in Fig. 18, called the yielding model. To decide the behavior, three steps are considered: 1) identify the priority (or right-of-way) of each connector by traffic rules; 2) determine yield or not; and 3) set the local target point.

1) Priority Identification: The priority of a connector can be set dynamically or statically. At a signalized intersection, the priority is controlled by the traffic lights. At an all-way stop intersection, vehicles follow the first-come–first-go principle. Intersections without traffic lights or stop signs follow fixed priority order defined in the digital maps, e.g., p1–p5 in Fig. 18(a), the smaller the number, the higher the priority.

2) Yield or Not: Two modes of yield are considered, called stop- and no-stop-to-yield. In the former, the ego vehicle needs to completely stop at a stop line before the yield strategy applies, e.g., facing a red light or a stop sign. In this case, if there are any valid vehicles on the connector (including part of its starting lanes) with higher priority, then the ego car will yield. In the no-stop mode, e.g., merging into a roundabout, the vehicle does not have to stop first. Instead, it can adjust its speed to yield. The yield decision is based on the time to collision (*ttc*), i.e., yield if

$$ttc_{\text{ego}} - ttc_{\text{obj}} \in [ttc_{\min}, ttc_{\max}]. \tag{22}$$



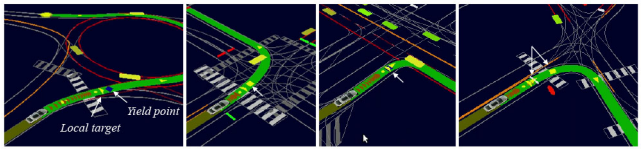


Fig. 18. Model of yield at intersections. (a) Schematic diagram. (b) Example scenarios: merging into a roundabout, left turn, right turn, and stop sign; noting that the blue/yellow squares are the calculated yield points, the yellow triangles are the set local targets, which are different from (a).

TABLE III
SYMBOLS AND VALUES USED IN THE BEHAVIOR PLANNER

Symbol	Value	Sym.	Val.	Sym.	Val.
t_l	30 s	$\mathbb{T}_{\max}$	4.0 s	$ttc_{\min}$	-3 s
t_m	1.5 s	t_e	6 s	$ttc_{\max}$	+3 s
$d_{\max}$	40 m	w_a	5 or 2	a_d	0.75 m/s^2
$d_{\min}$	15 m	$\mathbb{T}_{cf}$	2.0 s		

Remark: The decision to yield to pedestrians is made more conservatively. Not only the time to collision but also the distance to collision and the state of approaching or leaving are considered.

3) Set Local Target: In the stop-to-yield mode, the target is

$$\begin{aligned}\dot{\beta}_r &= 0\\ \beta_r &= \beta_{\max} = \beta_d\\ t_r &= \dot{\beta}_0/|a_d|,\\ \gamma_r &= 0\end{aligned} \tag{23}$$

where β_d is the distance to the stop line and a_d is the desired acceleration. For the no-stop-to-yield mode, in theory, it is not necessary to adjust the default target because the trajectory generation algorithm has the ability of speed adaptation to other objects. However, to enhance robustness, we add a rule to adjust the target: calculate a safe yield point that keeps a safe distance to the conflicting connector, as shown in Fig. 18(a), and then use it to replace the stop line in the stop-to-yield model. The setting of local target keeps the same. If the conflict in (22) disappears, then the yielding ends.

In summary, the presented unified models provide higher level behavior guidance in the selected typical scenarios. The targets act as reference points, and improve the rationality of driving behaviors, while the underlying collision avoidance is handled by the trajectory generation module. All the parameter values are shown in Table III.

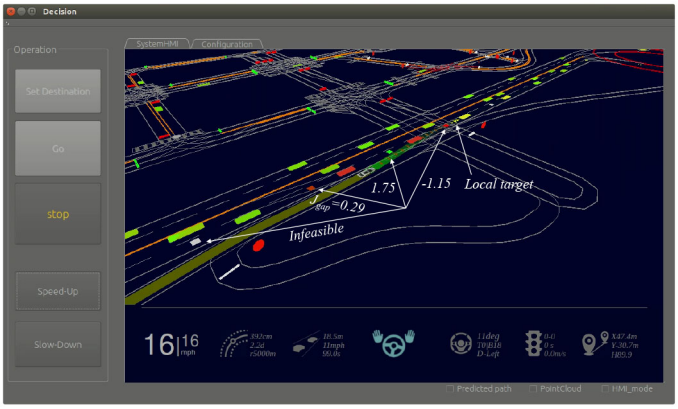


Fig. 19. Interface of our routing and behavior planning software. It shows the gap selection when merging into a highway. The small green square indicates the best gap with the highest score of 1.75, the red squares indicate gap candidates with lower scores, and the gray square indicates an infeasible gap. The yellow triangle indicates the local target.

D. Implementation

The routing and behavior planning software is shown in Fig. 19. This software combines information from the map, perception, prediction, and communication modules, then determines the scenarios and outputs the local target and reference path for trajectory generation.

VI. Motion Control

We design a dynamics-and-delay-aware preview control for accurate and smooth trajectory tracking. The trajectory tracking is split into path and speed tracking. For the path tracking or steering control, the model we use is [25]

$$\dot{x} = \overline{A}x + \overline{B}\delta(t-\tau_d) + \overline{D}c_t$$

where

$$\begin{aligned}
x &= \left[\, e_y, \dot{e}_y, e_\varphi, \dot{e}_\varphi, \delta_r \right]^T \\
\overline{A} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{-\sigma_1}{mv} & \frac{\sigma_1}{m} & \frac{\sigma_2}{mv} & \frac{2\mathcal{C}_{\alpha f}}{m} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{\sigma_2}{I_z v} & \frac{-\sigma_2}{I_z} & \frac{\sigma_3}{I_z v} & \frac{2l_f\mathcal{C}_{\alpha f}}{I_z} \\ 0 & 0 & 0 & 0 & -1/\tau \end{bmatrix}, \\
\overline{D} &= \begin{bmatrix} 0 \\ \sigma_2/m - v^2 \\ 0 \\ \sigma_3/I_z \\ 0 \end{bmatrix} \\
\overline{B} &= \left[0, 0, 0, 0, 1/\tau\right]^T \\
\sigma_1 &= 2\left(\mathcal{C}_{\alpha f} + \mathcal{C}_{\alpha r}\right) \\
\sigma_2 &= -2\left(l_f\mathcal{C}_{\alpha f} - l_r\mathcal{C}_{\alpha r}\right) \\
\sigma_3 &= -2\left(l_f^2\mathcal{C}_{\alpha f} + l_r^2\mathcal{C}_{\alpha r}\right).
\end{aligned} \tag{24}$$

In this model, e_y and e_φ are the lateral offset and the heading angle error. τ_d is the time delay and τ is the time constant of the first-order lag of the steering system, i.e.,

$$\tau\dot{\delta}_r(t) = -\delta_r(t) + \delta(t-\tau_d) \tag{25}$$

TABLE IV
Symbols and Definitions of the Vehicle Dynamics Model

Definition	Symbol	Value & Unit
Vehicle mass	m	1800 kg
Yaw moment of inertia of the vehicle	I_z	3270 kg·m²
Distance from c.g. to the front/rear axle	l_f/l_r	1.2/1.65 m
Cornering stiffness of the front wheel	$\mathcal{C}_{\alpha f}$	70000 N/rad
Cornering stiffness of the rear wheel	$\mathcal{C}_{\alpha r}$	60000 N/rad
Pure time delay	τ_d	0.2 s
Time constant	τ	0.2 s
Sampling period	$\Delta\tau$	0.04 s

where δ_r is the actual steering angle and δ is the steering command. Parameter values are listed in Table IV. Note that all vehicle dynamics, input delay (about 200 ms), and steering lag are considered for better performance, which makes our control different from the typical controls in the literature.

To facilitate the controller design, the continuous-time system is converted into a discrete-time system with a fixed sampling time $\Delta\tau$ and zero-order hold

$$x(k+1) = Ax(k) + B\delta(k-N) + Dc_t(k) \tag{26}$$

where N is the number of delay steps. Based on this model, we formulate the path-tracking task as an OCP with cost function

$$\begin{aligned}
\mathcal{J}(x,\delta) &= 0.5\sum_{k=0}^{\infty} x^T(k)\mathcal{Q}x(k) + \mathcal{R}\delta^2(k) \\
\mathcal{Q} &= \mathrm{diag}[q_1, \ldots, q_4, 0] \\
c_t(i) &= 0,\, i \in [k+\mathbb{N}+1, \infty)
\end{aligned} \tag{27}$$

where $\mathcal{R} \in \mathbb{R}$ is a positive weighting factor and $c_t(i)$ denotes the curvature of the planned trajectory at time step i; curvatures beyond $\mathbb{N}$ time steps in the future are assumed to be 0.

The key to the control design is to solve this nonlinear time-delay problem efficiently. Different from numerical optimizations, we pursue closed-form control laws and leverage the preview control theory to solve it. To do so, all delayed inputs and the previewed c_t are converted into system states, i.e.,

$$\begin{bmatrix} x(k+1) \\ \Psi(k+1) \\ \mathcal{C}_t(k+1) \end{bmatrix} = \begin{bmatrix} A & B_\Psi & D_\mathcal{C} \\ O & A_\Psi & O \\ O & O & A_\mathcal{C} \end{bmatrix} \begin{bmatrix} x(k) \\ \Psi(k) \\ \mathcal{C}_t(k) \end{bmatrix} + \begin{bmatrix} O \\ \mathbb{B}_\Psi \\ O \end{bmatrix} \delta(k)$$

with

$$\begin{aligned}
\Psi(k) &= [\delta(k-N), \ldots, \delta(k-2), \delta(k-1)]^T \in \mathbb{R}^N \\
\mathcal{C}_t(k) &= [c_t(k), c_t(k+1), \ldots, c_t(k+\mathbb{N})]^T \in \mathbb{R}^{\mathbb{N}+1} \\
B_\Psi &= [B, O] \in \mathbb{R}^{5\times N}, \\
D_\mathcal{C} &= [D, O] \in \mathbb{R}^{5\times\mathbb{N}} \\
A_\Psi &= \begin{bmatrix} O & I_{(N-1)} \\ 0 & O \end{bmatrix} \in \mathbb{R}^{N\times N}, \\
A_\mathcal{C} &= \begin{bmatrix} O & I_{(\mathbb{N}-1)} \\ 0 & O \end{bmatrix} \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}} \\
\mathbb{B}_\Psi &= [0, 0, \ldots, 1]^T \in \mathbb{R}^N
\end{aligned} \tag{28}$$

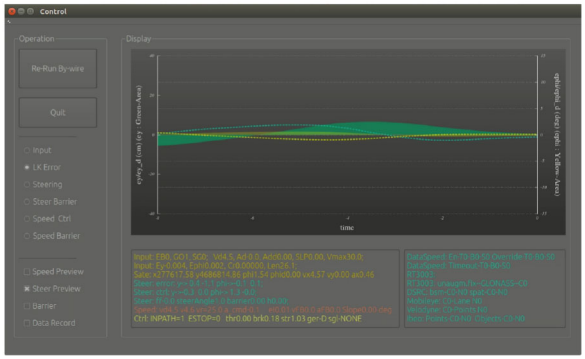


Fig. 20. Motion control software. In the left section, users can select what information to display (e.g., input or tracking errors) and enable/disable preview control, barrier control, and data record. In the right section, the system states and control results are displayed.

where x, B, $D \in \mathbb{R}^{5\times 1}$, and O/I stands for the zero/identity matrix. This augmentation strategy converts the original nonlinear time-delay OCP into an augmented linear delay-free system with the state dimension increasing from 5 to $6+N+\mathbb{N}$. As a result, the augmented problem becomes a standard linear quadratic regulator (LQR), whose optimal control is

$$\delta^*(k) = -\overbrace{K_{b,1-4}\, x_e(k)}^{\text{errors}} - \overbrace{K_{b,5}\, \delta_r(k)}^{\text{steering angle}} - \underbrace{\sum_{i=0}^{N-1} K_{b,6+i}\delta(k-N+i)}_{\text{delayed commands}} - \underbrace{\sum_{i=0}^{\mathbb{N}} K_{f,i}c_t(k+i)}_{\text{feedforward}} \quad (29)$$

where $x_e = [e_y, \dot{e}_y, e_\varphi, \dot{e}_\varphi]^T$ are the tracking errors, K_b is the feedback gain vector, and K_f is the feedforward/preview gain vector. This control law has a closed-form expression and is computationally efficient.

Note that this control did not consider constraints in tracking errors. To bound the tracking error, refer to our previous paper [26], in which a barrier control is developed to supervise the proposed control. It is activated if and only if the error is approaching the boundary, and then adds well-calculated steering to guarantee bounded errors. Finally, we use C++ and ROS to implement and integrate the designed trajectory tracking controllers, as shown in Fig. 20. It runs at a cycle time of 40 ms.

VII. Experimental Results

The above four modules are implemented and tested on the Mcity self-driving platform, a hybrid Lincoln MKZ, as shown in Fig. 21. It is a shared platform we developed for researchers at the University of Michigan. This car is equipped with a set of sensors (e.g., 32-channel LiDAR, Radars, cameras, Mobileye, Ibeo, and RTK) and DSRC/4G-based communication. The by-wire control enables automatic manipulation of throttle/brake pedals, transmission, steering wheel, and turn signals.

In the following, we first test the motion control module, then test the motion planning and control system in both highway and urban driving in Mcity, as reported in Section B and

Fig. 21. Mcity automated vehicle platform—a hybrid Lincoln MKZ.

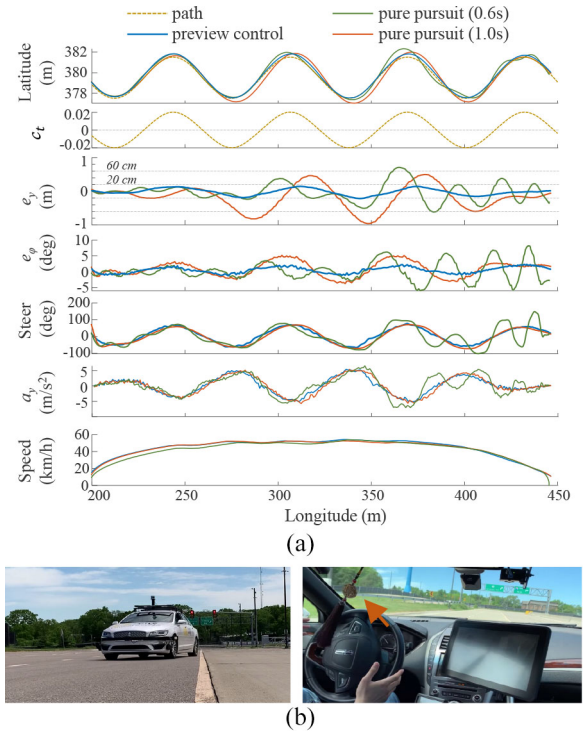


Fig. 22. Testing results of the preview control and the pure pursuit control. (a) Test results. (b) Snapshot. The floating pendant indicates a radical lateral acceleration.

the supplementary file. In the test, the SUMO is used to generate virtual surrounding cars and challenging scenarios [22]. In Section C, another real self-driving car is used to challenge the ego car. It is recommended to watch the four supplemental videos in the Appendix to better understand the following content.

A. Test of Motion Control Module

To validate the designed dynamics-and-delay-aware preview control, we apply it to track a sinusoidal trajectory (amplitude 2 m, period 20π m), as shown in Fig. 22. The vehicle speed is set to 60 km/h, corresponding to a maximal lateral acceleration of about 5 m/s^2, which is high for typical human driving. Compared to the classic pure pursuit control with look-ahead time being 0.6 and 1.0 s [27], the designed control achieved much lower tracking errors, i.e., within $\pm$20 cm, see the blue e_y profile in Fig. 22(a). The pure pursuit control (0.6 s) is nearly unstable due to the smaller look-ahead distance, and the maximal tracking error is $>$60 cm.

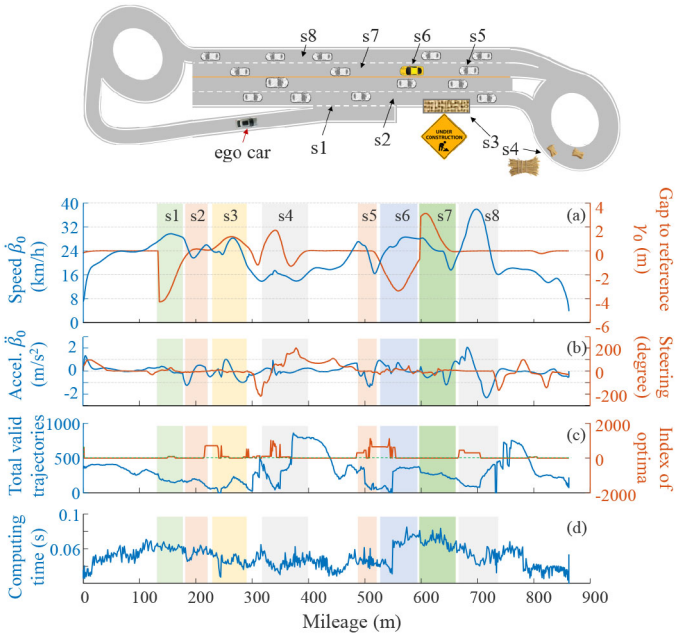


Fig. 23. Scenarios and testing results of driving on Mcity highway with simulated traffic.

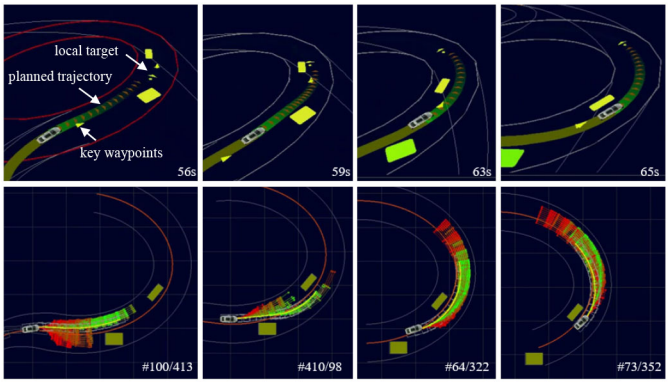


Fig. 24. Snapshots of scenario s4, i.e., avoid two static obstacles. 1) The first row shows the behavior planning, where the green/yellow *rectangles* stand for surrounding obstacles. The yellow *triangles* are the local targets and key waypoints and 2) the bottom row shows the trajectory generation. The bold yellow curve is the optimal trajectory, the bold red curve is the reference path. The white curves define the drivable area.

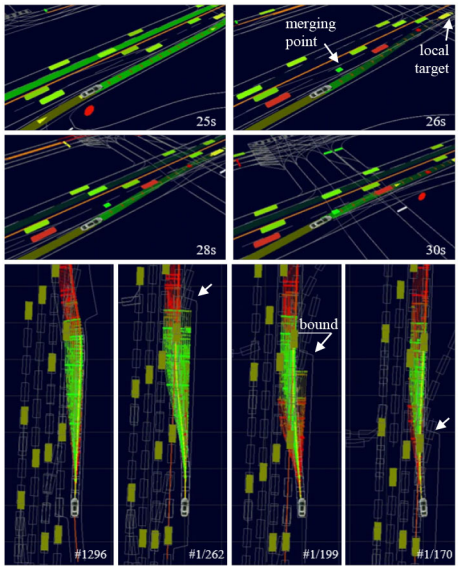


Fig. 25. Snapshots of scenario s1, i.e., merge into the highway. 1) The first two rows show the behavior planning. The red rectangles indicate the target merging gap. The small green squares show the target merging point and 2) the bottom row shows the trajectory generation. The white arrows highlight the boundaries of the changing drivable area.

B. Highway Driving

As shown in Fig. 23, the track consists of four carriageways and one merging ramp. The length of the highway is about 250 m, limiting the reference speed to 30 km/h. The traffic is randomly generated by SUMO with a density up to 32 cars per lane per km. Their driving behaviors are governed by the SUMO default models with stochastic initializations.

In the test, the ego car launches from the merging ramp and experiences 8 scenarios, i.e., merge into highway (denoted as s1 in Fig. 23), car-follow (s2), avoid two static obstacles (straw bales, s4), avoid a car generated suddenly (s5), exit the highway (s7), and avoid an aggressive following car (s8). To further increase the challenge, we add two *phantom* objects: 1) a road construction area partially blocking the rightmost lane (s3) and 2) a car parked on the second lane (s6). The phantom objects can impede the ego car but are transparent to other cars, which will create challenging scenarios. The surrounding cars' motion is predicted on top of two assumptions: 1) they keep constant acceleration and 2) they could go in all possible lanes defined by the map, and aim to approach the lane centers within a limited time; namely, multiple paths are allowed for each object. The test results are shown in Fig. 23.

To better understand the result, we start from s4, i.e., avoiding two static obstacles. Some key moments are highlighted by the snapshots in Fig. 24, in which the first row shows the interface of the behavior planner, marked by timestamps at the right-bottom corner. The bottom row shows the interface of trajectory generation; the two numbers at the right-bottom corners represent the index of the optimal trajectory and the number of total valid trajectories out of the 1224 samples. Figs. 25 and 26 follow the same definition as Fig. 24.

In Fig. 24, we can see the flexible lateral motion, the cost of each valid trajectory indicated by color, and how the two obstacles trigger rejection of the sampled trajectories that will lead to collisions. The ego car avoids the two static obstacles safely with a hat-shaped lateral movement without significant speed reduction, refer to γ_0 and $\dot{\beta}_0$ profiles in Fig. 23 (s4).

Different from s4, other scenarios involve dynamic traffic. In s1, the ego car merges into the traffic successfully with a smooth path and speed profile, as shown in Figs. 23 and 25. In Fig. 23 (s1), the step change of γ_0 indicates that the lane change occurred and the reference path is switched to the new target lane. The profile of $\dot{\beta}_0$ shows that the vehicle accelerates from 24 to 30 km/h to merge into the traffic. In Fig. 25, the small green squares indicate the selected merging gap and the target merging point. The bottom row shows all valid trajectories at each moment, all of them are inside the changing drivable area and have no collision with other cars.

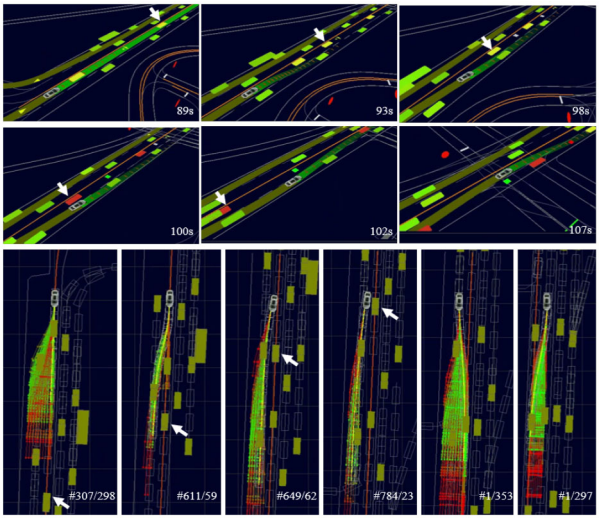


Fig. 26. Snapshots of scenarios s5, s6, and s7, i.e., avoid a car generated suddenly (at 89 s), avoid a *phantom* car parked in the lane (from 93 to 102 s), and exit the highway (at 107 s). The first two rows show the behavior planning. The white arrows highlight the static phantom obstacle. The yellow rectangles indicate they are the leading cars. The red rectangles indicate the merging gap.

Fig. 26 show three scenarios, i.e., s5 (at 89 s), s6 (93–102 s), and s7 (at 107 s). In s5, the SUMO environment suddenly generates a car driving 28 km/h, 5 m before the ego car. The ego car avoids collision and brakes with -1.2 m/s^2 to increase the time headway. Similar to s1, in s7 the car executes another lane change to exit the highway. The vehicle decelerates to 24 km/h to merge into the selected gap. The duration of the lane change is about 5 s, close to the typical human drivers' operation.

s6 is an atypical and challenging scenario, refer to Fig. 26 from 93 to 102 s. The ego car needs to avoid not only the static phantom obstacle (highlighted by the white arrows in Fig. 26) but also other moving cars. We can observe the active and smooth adjustment in both vehicle speed $\dot{\beta}_0$ and lateral offset γ_0 in Figs. 23 and 26. The setting of s3 is similar to s6, but their speed and path profiles are significantly different, see Fig. 23, indicating the good adaptability and robustness of the system. In both cases, we did not change the reference path (i.e., no lane change command); thus, the behavior can be regarded as two overtake maneuvers. The maximal lateral offset γ_0 in s3 and s6 are 1.2 and -3.2 m, respectively, which achieve maintaining safe lateral gaps to the obstacles and minimizing the deviation from the reference path. In s8, a following car is approaching aggressively, the ego car responds to it properly with a strong acceleration.

In summary, the developed system delivered reasonable, responsive, and smooth behaviors in these scenarios, while safely avoiding static and moving obstacles. The computation time is less than 0.1 s per cycle as shown in Fig. 23(d). We also test the system in urban driving with another 8 selected scenarios, refer to the supplementary file (limited by the length of the manuscript). This test further illustrates the system's capability of flexibly adjusting vehicle path and speed profiles against dynamic traffic (even when other cars violate traffic rules or act improperly), and validates the behavior competence of driving in urban areas.

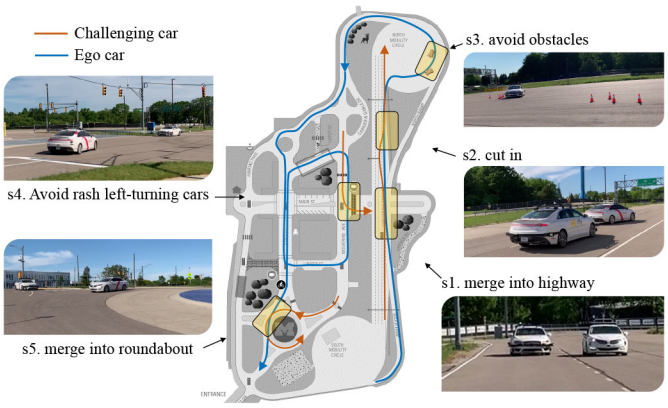


Fig. 27. Scenario definition of interacting with a real surrounding car.

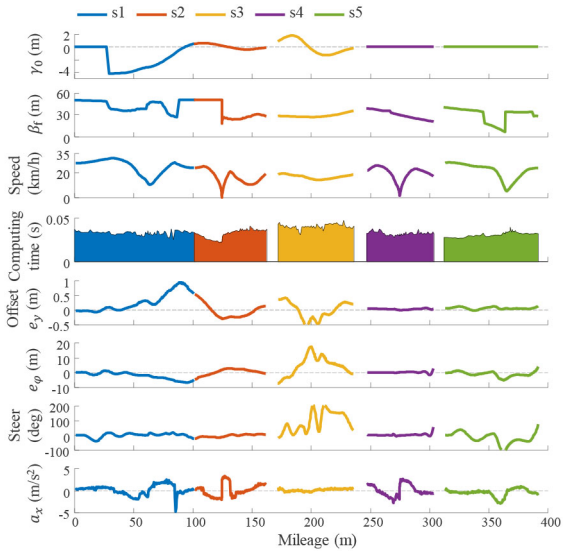


Fig. 28. Testing results of interacting with a real car.

C. Experiment Using Real Surrounding Car at Mcity

As shown in Fig. 27, the ego car runs along the blue path at Mcity. Five scenarios are highlighted, i.e., merge into highway (s1), handle an aggressive cut-in (s2), avoid two static real obstacles (s3), avoid a left-turning car at an unprotected intersection (s4, traffic lights are disabled), and merge into a roundabout (s5). Different from the previous tests, we use another real self-driving Lincoln MKZ to challenge the ego car. This car is specifically developed to test other autonomous vehicles [28], called the challenger in the following. It adjusts its motion to achieve a preset challenge level, quantified by criteria such as speed gap and time margin.

The experimental results are shown in Fig. 28. The ego car safely avoids all potential collisions. For example, in s1, the challenger deliberately blocks the ego car in the merging area. The ego car decelerates to about 10 km/h and merges successfully before reaching the end of the lane. In s2, the challenger cuts in and starts decelerating when the distance gap is only 7 m; the ego car brakes to avoid a collision. In s4, the challenger takes an aggressive illegal left turn in front of the ego car. The ego car does brake timely to avoid the conflict even when it has the right of way. In s5, the ego car yields to the challenger in the roundabout by slowing down to 5 km/h and then enters the roundabout safely without stopping.

VIII. Conclusion

This article presented the algorithms, software, and experimental validations of a model-driven motion planning and control system for self-driving amidst dynamic and stationary obstacles. The system was designed in a hierarchical framework that consists of routing, behavior planning, trajectory generation, and trajectory tracking modules. The system considers safety constraints and achieves smooth operations due to the deterministic sampling trajectory optimization and the dynamics-and-delay-aware preview tracking control. The implementation was tested both through simulations and experiments.

Note that the designed system does not cover the prediction of other objects' motion and intent nor the uncertainties due to sensing/perception/prediction, both are potential future research topics. The design of emergency maneuvers when no feasible trajectory is available is also left for future work.

Appendix
Videos

Four videos of the tests are available in the manuscript system, or access with the following YouTube links.

1) Comparison of the preview control and the pure pursuit control (https://youtu.be/27JR1kpHcL4).
2) Test on highway (https://youtu.be/b9aLQYevdRw).
3) Test of urban driving (https://youtu.be/5Gz9mqVdsyM).
4) Test using a real other car (https://youtu.be/Rr5MxSeoCvM).

Acknowledgment

Toyota Research Institute (TRI) provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] S. Thrun *et al.*, "Stanley: The robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006.
[2] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.
[3] S. Kato *et al.*, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst.*, 2018, pp. 287–296.
[4] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, pp. 187–210, May 2018.
[5] Z. Cao *et al.*, "Highway exiting planner for automated vehicles using reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 990–1000, Feb. 2021.
[6] Z. Cao, S. Xu, H. Peng, D. Yang, and R. Zidek, "Confidence-aware reinforcement learning for self-driving cars," *IEEE Trans. Intell. Transport. Syst.*, doi: 10.1109/TITS.2021.3069497. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9397429
[7] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.
[8] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C Emerg. Technol.*, vol. 60, pp. 416–442, Nov. 2015.
[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
[10] W. Xu, J. Wei, J. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Proc. IEEE ICRA*, May 2012, pp. 2061–2067.
[11] D. Gonzalez, J. Perez, R. Lattarulo, V. Milanes, and F. Nashashibi, "Continuous curvature planning with obstacle avoidance capabilities in urban scenarios," in *Proc. IEEE 7th Int. ITSC*, 2014, pp. 1430–1435.
[12] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—A local, continuous method," in *Proc. IEEE Intell. Veh. Symp.*, 2014, pp. 450–457.
[13] S. Xu, S. E. Li, K. Deng, S. Li, and B. Cheng, "A unified pseudospectral computational framework for optimal control of road vehicles," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 4, pp. 1499–1510, Aug. 2015.
[14] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
[15] A. Lacaze, Y. Moscovitz, N. DeClaris, and K. Murphy, "Path planning for autonomous vehicles driving over rough terrain," in *Proc. IEEE ISIC/CIRA/ISAS Joint Conf.*, Gaithersburg, MD, USA, Sep. 1998, pp. 50–55.
[16] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 346–359, 2012.
[17] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 101–115, Feb. 2012.
[18] P. Raksincharoensak, T. Hasegawa, and M. Nagai, "Motion planning and control of autonomous driving intelligence system based on risk potential optimization framework," *Int. J. Autom. Eng.*, vol. 7, no. 1, pp. 53–60, 2016.
[19] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 5, pp. 1782–1797, Sep. 2018.
[20] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Auton. Robots*, vol. 41, no. 6, pp. 1367–1382, 2017.
[21] R. A. E. Zidek and I. V. Kolmanovsky, "Optimal driving policies for autonomous vehicles based on stochastic drift counteraction," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 290–296, 2017.
[22] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO—Simulation of urban mobility: An overview," in *Proc. SIMUL 3rd Int. Conf. Adv. Syst. Simulat.*, 2011, pp. 1–6.
[23] B. Li, S. Xu, and H. Peng, "Eco-routing for plug-in hybrid electric vehicles in large urban transportation network," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1508–1513.
[24] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. IEEE Intell. Veh. Symp.*, 2010, pp. 518–522.
[25] S. Xu and H. Peng, "Design, analysis, and experiments of preview path tracking control for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 48–58, Jan. 2020.
[26] S. Xu, H. Peng, P. Lu, M. Zhu, and Y. Tang, "Design and experiments of safeguard protected preview lane keeping control for autonomous vehicles," *IEEE Access*, vol. 8, pp. 29944–29953, 2020.
[27] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-RI-TR-92-01, 1992.
[28] X. Wang, Y. Dong, S. Xu, H Peng, F. Wang, and Z. Liu, "Behavioral competence tests for highly automated vehicles," in *Proc. IEEE Intell. Veh. Symp.*, 2020, pp. 1323–1329.

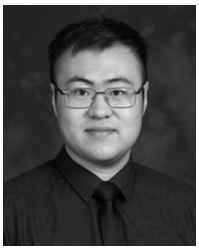
Shaobing Xu received the Ph.D. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2016.

He is currently an Assistant Research Scientist with the Department of Mechanical Engineering and Mcity, University of Michigan, Ann Arbor, MI, USA. His research focuses on vehicle motion control, decision making, and path planning for autonomous vehicles.

Dr. Xu was a recipient of the outstanding Ph.D. Dissertation Award of Tsinghua University and the Best Paper Award of AVEC'2018.

Robert Zidek received the Ph.D. degree in aerospace engineering from the University of Michigan, Ann Arbor, MI, USA, in 2017.

He currently develops autonomous driving software as a Research Scientist and a Manager with the Toyota Research Institute, Ann Arbor. His team focuses on building decision-making capabilities for L4 and L5 autonomous driving systems, including higher level behavior planning and navigation, generating trajectory planning problems, and final trajectory selection.

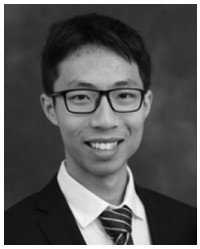


Zhong Cao (Member, IEEE) received the B.S. degree in automotive engineering from Tsinghua University, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree in automotive engineering.

He was a visiting Ph.D. student with the University of Michigan, Ann Arbor, MI, USA, from 2017 to 2019. His research interests include connected automated vehicles, driving environment modeling, and driving cognition.

Pingping Lu received the Ph.D. degree in communication and information system from the Institute of Electronics, Chinese Academic of Sciences, Beijing, China, in 2016.

She is currently a Postdoctoral Researcher with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA. Her research interests include object detection and scene understanding for autonomous vehicles, and high-resolution synthetic aperture radar image processing.



Xinpeng Wang (Graduate Student Member, IEEE) received the B.S. degree in automation from Tsinghua University, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree in mechanical engineering with the University of Michigan, Ann Arbor, MI, USA.

His research focuses on the control and evaluation of highly automated vehicles.

Boqi Li received the B.S. degree in mechanical engineering from the University of Illinois Urbana–Champaign, Champaign, IL, USA, in 2015, and the M.S. degree in mechanical engineering from Stanford University, Stanford, CA, USA, in 2017. He is currently pursuing the Ph.D. degree in mechanical engineering with the University of Michigan, Ann Arbor, MI, USA.

Huei Peng received the Ph.D. degree in mechanical engineering from the University of California, Berkeley, CA, USA, in 1992.

He is currently a Professor with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA, where he is the Director of Mcity. His current research focuses include design and control of electrified vehicles, and connected/automated vehicles. His research interests include adaptive control and optimal control, with emphasis on their applications to vehicular and transportation systems.

Prof. Peng is both an SAE Fellow and an ASME Fellow.