Reachability-based Decision Making for Autonomous Driving: Theory and Experiment

Ahn, Heejin; Berntorp, Karl; Inani, Pranav; Ram, Arjun Jagdish; Di Cairano, Stefano

TR2020-165 December 16, 2020

Abstract

We describe the design and validation of a decision making system in the guidance and control architecture for automated driving. The decision making system determines the timing of transitions through a sequence of driving modes, such as lane following and stopping, for the vehicle to eventually arrive at the destination without colliding with obstacles, hence achieving safety and liveness. The decision making system commands a transition to the next mode only when it is possible for an underlying motion planner to generate a feasible trajectory that reaches the target region of such next mode. Using forward and backward reachable sets based on a simplified dynamical model, the decision making system determines the existence of a trajectory that reaches the target region, without actually computing it. Thus, the decision making system achieves fast computation, resulting in reactivity to a varying environment and reduced computational burden. To handle the discrepancy between the dynamical model and actual vehicle motion, we model it as a bounded disturbance set and guarantee robustness against it. We prove the safety and liveness of the decision making system, and validate it using small-scale car-like robots.

IEEE Transactions on Control Systems Technology

Reachability-based Decision Making for Autonomous Driving: Theory and Experiment

Heejin Ahn, Karl Berntorp, Pranav Inani, Arjun Jagdish Ram, and Stefano Di Cairano

***Abstract*—We describe the design and validation of a decision making system in the guidance and control architecture for automated driving. The decision making system determines the timing of transitions through a sequence of driving modes, such as lane following and stopping, for the vehicle to eventually arrive at the destination without colliding with obstacles, hence achieving safety and liveness. The decision making system commands a transition to the next mode only when it is possible for an underlying motion planner to generate a feasible trajectory that reaches the target region of such next mode. Using forward and backward reachable sets based on a simplified dynamical model, the decision making system determines the existence of a trajectory that reaches the target region, without actually computing it. Thus, the decision making system achieves fast computation, resulting in reactivity to a varying environment and reduced computational burden. To handle the discrepancy between the dynamical model and actual vehicle motion, we model it as a bounded disturbance set and guarantee robustness against it. We prove the safety and liveness of the decision making system, and validate it using small-scale car-like robots.**

***Index Terms*—Decision making, formal methods, motion planning, autonomous driving**

I. Introduction

DECISION making is one of the key components to enable automated driving. Based on a route given by a navigation system, it makes mission-level decisions such as if and when to perform lane changing, stopping, waiting, and intersection crossing. According to the decisions, an underlying motion planning system generates a state trajectory, and a vehicle control system computes the input signals to track the trajectory, as shown in Fig. 1. This modular structure yields several advantages: increased flexibility by executing different algorithms, a more effective usage of information and resources by exploiting different prediction models at different decision rates, easier maintenance and expandability as one layer can be modified without affecting the others, and the parallel and independent development of each layer. On the other hand, the modules have to be carefully designed to guarantee, once integrated, the desired overall system behavior, without side-effects or interference.

This paper proposes the design of the decision making system that guarantees safety and liveness, and retains them

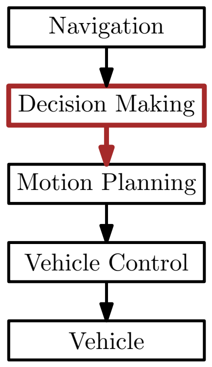


Fig. 1. Example architecture of an automated driving system. The focus of this paper is on decision making and its integration with motion planning.

once it is integrated with the motion planning system. The decision making system is referred to as just *the decision maker* in the rest of the paper. Here, we use the particle filter motion planner in [1], while the proposed design can be applied to any other motion planning method aimed at generating dynamically feasible vehicle trajectories.

The objective of the decision maker design in this paper is to ensure safety and liveness; that is, the decision maker should ensure that the ego vehicle (EV) does not collide with other vehicles (OVs) and reaches a sequence of goal sets, which represents a route given by a navigation system. To guarantee the liveness property of the decision maker, we use *backward reachable sets* of such goal sets., i.e., the set of states from which the goal set can be reached by an appropriate choice of input signals. The key idea of our design is that the decision maker keeps the EV state inside the backward reachable sets of all the subsequent goal sets, thereby ensuring that there exists an input signal to reach the goal sets. A commonly adopted approach for decision maker design is forward simulation [2]–[4]. However, such an approach restricts the prediction of the EV's trajectories to a finite horizon, and often does not guarantee persistent feasibility [5]. Hence, decisions at the current time step may cause infeasibility in subsequent decision-making problems. Our method also uses forward reachability to guarantee the safety property of the decision maker, but in addition uses backward reachability to guarantee persistent feasibility and therefore liveness. More details of the relation between this work and the literature are discussed in Section I-A.

For computationally efficient decision making over a long horizon, we use a simplified vehicle dynamical model, while relying on the underlying motion planner to generate a more

H. Ahn was with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, at the time of this work. She is currently with the University of British Columbia, BC, Canada, e-mail: hjahn@cs.ubc.ca
K. Berntorp, and S. Di Cairano are with MERL, e-mail: berntorp@merl.com, dicairano@ieee.org
P. Inani and A.J. Ram were research interns at MERL at the time of this work. They are now with Torc Robotics, Blacksburg, VA, e-mail: {pranavinani94,arjunjram}@gmail.com.

realistic motion profile using a higher fidelity model. As opposed to our initial study [6], we model the discrepancy between the simplified model in the decision maker and the more accurate model in the motion planner as an additive bounded disturbance. Then, we ensure robustness of the decision maker by applying tools from robust reachability analysis, which amounts to appropriately shrinking the backward reachable sets and enlarging the avoidance sets for preventing collisions with OVs. For validating our approach, we evaluate the decision maker integrated with the motion planner and vehicle tracking controller in simulations and experiments of a small scale testbench for automated driving.

This paper is structured as follows. In the rest of this section, we introduce notations used throughout the paper and explain the relation of our work with existing literature. In Section II, we introduce the discrete and continuous models used in the decision making and motion planning systems. In Section III, we state the problem of designing the decision maker, and in Section IV we present an algorithm that provides the solution of the problem. In Section V, we present some implementation aspects of the algorithm, and detail the integration between the decision making and motion planning systems. We present the simulation and experimental results in Section VI and conclude the paper in Section VII.

Notation: Let $\mathbb{Z}_{\geq 0}$ and $\mathbb{R}$ denote the sets of nonnegative integers and real numbers, respectively. Function composition is denoted by $\circ$. For a discrete time signal, $x_k = x(t_k) = x(k\Delta t)$ where $\Delta t = t_{k+1} - t_k$ is the sampling period. By $\mathbf{x}_{0:N}$, we explicitly denote a sequence $(x_0, x_1, \ldots, x_N)$ over the prediction horizon N. We simply write $\mathbf{x}$ when the prediction horizon is clear from the context. The same notation applies to a sequence of inputs $\mathbf{u}$. Given a system with state x and input u, we denote by $\hat{x}_k(\hat{\mathbf{u}}, \hat{x}_0)$, the state reached at time step k starting from $\hat{x}_0$ with input sequence $\hat{\mathbf{u}}$. A random vector $v \sim \mathcal{N}(m, Q)$ is Gaussian distributed with mean m and covariance Q. Given two sets A and B, the Minkowski set addition is $A \oplus B := \{a + b : a \in A, b \in B\}$, the Pontryagin set difference is $A \ominus B := \{a : a + b \in A, \forall b \in B\}$, the complement of A is A^c, and the origin-symmetric set of A as $-A$, i.e., $-A := \{-a : a \in A\}$.

A. Relation with Existing Literature

Our decision maker is designed to ensure liveness and safety of the overall automated driving system, when the remaining components operate normally. Currently, the methods for decision making can be categorized into three groups: rule-based, reactive, and interactive planning.

Rule-based planning refers to algorithms that make decisions using the current states of OVs, without considering their dynamics and hence the prediction of their trajectories. In the DARPA Urban Challenge [7], most teams implemented a rule-based decision making system involving hand-tuned heuristics for different city-driving scenarios.

Reactive planning refers to algorithms that make decisions in response to predicted trajectories of OVs over a finite horizon. A commonly used approach is based on forward simulation of the EV. For example, decision makers based on forward simulation allow the action of lane change when there is a trajectory of the EV that reaches the adjacent lane without overlapping with the predicted (forward simulated) trajectories of OVs. The work [2] optimizes trajectories, represented by cubic splines, that satisfy a required probability of collision with OVs. Instead of using deterministic mode transitions, it lets multiple possible trajectories share a single trajectory segment at the beginning and applies an input that tracks the shared segment. The works [3], [4] classify trajectories based on their topological property for different decisions and choose the best decision based on some metrics. A limitation of some forward simulation approaches is not to guarantee persistent feasibility.

Some reachability-based approaches to reactive planning have also been studied before. The work [8] uses similar tools to the ones we use in this paper, but for a different purpose. The method in [8] verifies if a trajectory, which is assumed to be generated by an external motion planner, is safe to execute by checking for the intersection of the forward reachable sets of the EV and OVs over a prediction horizon. The method accepts a trajectory if it maintains the EV in a safe condition at all future time steps, for example, by bringing the EV to a stop in a safe location. The approach of planning until stop was adopted for rapidly-exploring random trees (RRT) in urban scenarios [9]. Backward reachability tools are used in [10] to determine the drivable area, where safety is guaranteed at all future times by preventing the EV state from entering the backward reachable sets of the occupied region by OVs.

Interactive planning incorporates the reaction of OVs into the plans of the EV. One approach, based on game theory, selects the actions of the EV and OVs that maximize rewards by evaluating all possible combinations of actions [11] or assuming that agents have different interaction levels [12]. Another approach based on partially observable Markov Decision Processes (POMDPs) determines the optimal action of the EV in response to the unobservable OVs' intentions. The works [13], [14] present computationally tractable POMDPs by capturing regular behavior of drivers as a finite set of policies and by planning the longitudinal input along a finite set of predetermined paths through intersections, respectively.

The approach developed in this work is an instance of the reactive planning category. Our decision maker uses both forward and backward reachability to ensure persistent feasibility. While we also use tools from reachability analysis, our goal is fundamentally different from [8]. Rather than verifying a specific trajectory, we determine the existence of trajectories that safely reach certain target regions before the motion planner actually computes them. This allows the motion planner to avoid unnecessary computations in trying to accomplish unachievable goals. Similar to [10], we ensure safety with respect to OVs by commanding a mode transition only if it is possible to avoid their *capture sets*, the sets where collision is unavoidable. In addition, a transition is activated only if the EV state is within the backward reachable set of corresponding target region, which guarantees liveness. As opposed to known methods for interactive planning, our work considers a sequence of decisions to ensure liveness by appropriately concatenating the reachable sets.

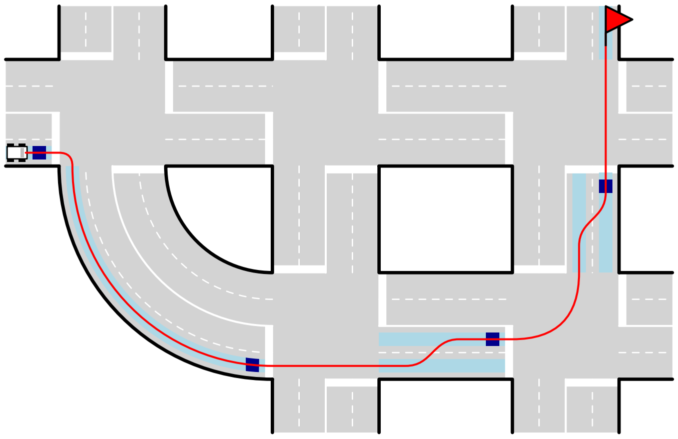


Fig. 2. Example of city driving. A route to the destination (red flag) from the navigation system involves a sequence of driving modes such as lane following (with light blue target regions around centerlines) and stopping (with dark blue rectangles target regions before stop lines). The motion planner iteratively generates a trajectory (red line) that reaches the sequence of target regions, while avoiding collisions with traffic (not shown). The decision maker determines the current driving mode and target region.

II. Vehicle Modeling for Decision Making and Motion Planning

The decision maker in Fig. 1 determines discrete mode transitions based on continuous-valued signals of a simplified dynamic model, while the underlying motion planner generates the actual vehicle trajectories using a more accurate dynamic model. In this section, we describe the models of discrete modes and continuous states of the decision maker and motion planner. We also describe the model of OVs.

A. Discrete Mode Transitions in the Decision Maker

In the architecture in Fig. 1, the route of the EV is determined by the navigation system; see for instance Fig. 2. From the route, a sequence of driving modes is generated,

$$q_1 \to q_2 \to \ldots \to q_M, \tag{1}$$

where $q_i \in Q$ for $i = 1, \ldots, M$, Q is the finite set of driving modes, and $q_i \to q_{i+1}$ denotes the transition from q_i to q_{i+1}. The sequence in (1) can be constructed in different ways. For example, it may be the sequence of driving lanes, stop areas, and turns, which is directly given by the navigation system. Alternatively, it may be a sequence of discrete actions, such as lane changing, stopping, waiting and crossing an intersection, generated usually from an automaton [7].

The navigation system may plan the route based on static information, such as static maps and road rules, but the timing of the mode switches must be determined in real time, based on the current road and traffic conditions. Furthermore, since the traffic conditions may cause scenarios where the predetermined mode sequence cannot be achieved, switching to an alternative sequence may be necessary. This may happen when the EV cannot achieve the appropriate lane before an intersection, because such lane is jammed with OVs. In this case, the EV may maintain the current lane, and the navigation system will provide an alternative route. In the general case,

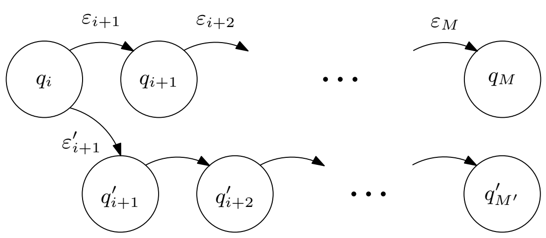


Fig. 3. If q_{i+1} become impossible to achieve in the future, for instance due to presence of OVs, the decision maker transitions to a backup route given by the navigation system and returns ε'_{i+1} when q'_{i+1} can be achieved.

the mode sequence may not be entirely determined *a priori*, but may be partially modified in real time, with enough preview for the proper operation of the decision maker.

The decision maker in Fig. 1 connects the long-term, static information-based, navigation system, with the short-term, dynamic information-based, motion planning by executing two actions. First, the decision maker determines when to transition to the next driving mode. Second, the decision maker determines whether the next driving mode has become unachievable, and if so, it determines when to transition to the next mode of an alternative sequence, which is available from the navigation system; see Fig. 3.

Specifically, in this work we design the decision maker π_ε that issues transition commands $\varepsilon_{i+1} \in \mathcal{E}$ based on current mode, EV state, and environment information. Here, $\mathcal{E}$ is the finite set of all discrete inputs. The discrete input $\varepsilon_{i+1} \in \mathcal{E}$ causes the transition from q_i to q_{i+1}, denoted by $q_i \to q_{i+1}$, and the void input $\varepsilon_0 \in \mathcal{E}$ causes the mode to remain the same, i.e., $q_i \to q_i$. When the decision maker determines that $q_i \to q_{i+1}$ has become impossible to achieve in the future, it begins considering the alternative sequence also provided by the navigation system,

$$q'_i \to q'_{i+1} \to \ldots \to q'_{M'}, \tag{2}$$

where $q'_i = q_i$ and $q'_{M'} = q_M$, and issues the discrete input $\varepsilon'_{i+1} \in \mathcal{E}$ when $q_i \to q'_{i+1}$ is feasible. Note that we assume q_M is reachable from any mode, possibly via re-routing executed by the navigation system.

For the simplicity of exposition, here we consider the case where the driving modes are the driving lanes and stop areas

$$Q = \{LF_i, S_i\}_{i=1}^{n_{\text{seg}}}$$

along the route, where n_{seg} is the number of road segments in the route. The sequence (1) is determined in real time based on route information, as we do for experimental validation in Section VI. Furthermore, we consider a single backup sequence (2), which amounts to continuing in the current lane and stopping in the stop area at the end of such lane.

Although this is a slightly simplified case, it still allows to represent several basic driving operations. A lane change is modeled as the transition from following a lane to following an adjacent lane, stopping as the transition from a lane to a stop area, and intersection crossing as the transition from a stop area to a lane. Considering all combinations, we can model lane following and changing, stopping and waiting at an intersection, and performing left and right turns at the intersection.

Thus, the majority of operations in city and highway driving can be decomposed into these basic driving modes. In this paper, we consider only intersections with all-way stop, but other intersection operations can be handled with additional information. For example, for vehicles to share intersection occupancy, it requires the definition of additional right-of-way rules, and signalized intersections require either traffic signal timing preview or that there is enough time to stop the vehicle or complete the intersection during the transitioning from when crossing is allowed (green) to forbidden (red). Similarly, multiple alternative sequences can be considered instead of one, by simply repeating the approach for each of them, and determining the most suitable one either by a static priority or by estimating their fitness to the current scenario.

B. Ego Vehicle Model in the Decision Maker

The decision maker determines the mode transition based on the discrete-time real-valued EV motion model

$$\hat{x}_{k+1} = \hat{f}(\hat{x}_k, \hat{u}_k), \tag{3}$$

where $k \in \mathbb{Z}_{\geq 0}$ is the sampling instant, $\hat{x}_k \in \hat{X} \subseteq \mathbb{R}^{\hat{n}_x}$ is the EV state, and $\hat{u}_k \in \hat{U} \subseteq \mathbb{R}^{\hat{n}_u}$ is the EV input. For (3), we use the discrete-time unicycle model

$$\begin{aligned}
p_{x,k+1} &= p_{x,k} + v_k \cos(\theta_k)\Delta t, \\
p_{y,k+1} &= p_{y,k} + v_k \sin(\theta_k)\Delta t, \\
v_{k+1} &= \begin{cases} v_{\max} & \text{if } v_k + \hat{u}_{v,k}\Delta t \geq v_{\max}, \\ 0 & \text{if } v_k + \hat{u}_{v,k}\Delta t \leq 0, \\ v_k + \hat{u}_{v,k}\Delta t & \text{otherwise,} \end{cases} \\
\theta_{k+1} &= \begin{cases} \theta_k + \hat{u}_{\theta,k}\Delta t & \text{if } v_k \geq v_{\min}, \\ \theta_k & \text{otherwise.} \end{cases}
\end{aligned} \tag{4}$$

Here, the EV state is $\hat{x} = (p_x, p_y, v, \theta)$ where (p_x, p_y) are the x and y positions in the global frame, v is the longitudinal velocity and θ is the heading angle, and the EV input is $\hat{u} = (\hat{u}_v, \hat{u}_\theta)$ where $\hat{u}_v$ is the longitudinal acceleration and $\hat{u}_\theta$ is the heading angular rate. In (4), $v_{\max}$ is the maximum velocity, imposed by road regulations or personal driving preference, and $v_{\min}$ is the minimum velocity necessary to achieve any admissible orientation rate $\hat{u}_\theta$, according to the actual behavior of vehicles. In (4), we impose input constraints

$$\hat{u}_k \in \hat{U} := [\hat{u}_{v,\min}, \hat{u}_{v,\max}] \times [\hat{u}_{\theta,\min}, \hat{u}_{\theta,\max}],$$

where $\hat{u}_{v,\min} \leq 0$ so $\hat{x}_{k+1} = \hat{x}_k$ is allowed when $v_k = 0$.

Let h be the output function of (3) that returns the longitudinal component of the EV state,

$$y_k = h(\hat{x}_k) \tag{5}$$

where the longitudinal state y consists of the longitudinal position p_l and velocity v. Function (5) maps the EV position (p_x, p_y) into the longitudinal position p_l by projecting onto its nominal path and computing the traveled distance between the projected point and a reference point. Thus, by composing (3) and (5), the longitudinal dynamics is

$$y_{k+1} = \hat{g}(\hat{x}_k, \hat{u}_{v,k}) := h \circ \hat{f}(\hat{x}_k, \hat{u}_k). \tag{6}$$

C. Ego Vehicle Model in the Motion Planner

The motion planner uses another discrete-time model for the EV motion,

$$x_{k+1} = f(x_k, u_k) \tag{7}$$

where $x_k \in X \subseteq \mathbb{R}^{n_x}$ and $u_k \in U \subseteq \mathbb{R}^{n_u}$ are the EV state and input, respectively. In general, $\hat{n}_x < n_x$, while usually $\hat{n}_u = n_u$. Models (3) and (7) are different due to their different purpose. Model (3) is used over a long horizon to determine the feasibility of a certain goal, i.e., if there exists a trajectory that achieves such a goal. Model (7) is used over a shorter horizon to compute a trajectory for the EV to achieve the goal. Thus, (3) should be simple, to allow fast computations over long horizons, although possibly with some approximations, while (7) should be precise to produce drivable trajectories that the vehicle can follow closely.

Given model (7), the goal of the motion planner is to generate an input trajectory $\mathbf{u}_{0:N-1}$ over the planning horizon N such that when applied to (7), leads to a safe, collision-free trajectory $\mathbf{x}_{0:N}$ that reaches the goal set selected by the decision maker. In this paper, we integrate the decision maker with the motion planner presented in [1], where we developed a particle filter (PF) based motion planner for computationally efficient generation of vehicle trajectories. The planner relies on *a priori* defined requirements c_k, modeled as

$$c_k = r(x_k) + d_k, \tag{8}$$

where r is the known requirement function and d_k is interpreted as a stochastic disturbance determining the tolerance to deviations. The PF based motion planner in [1] constructs the state trajectory probability density function given the requirements, that is, $p(\mathbf{x}_{0:N}|\mathbf{c}_{0:N})$, and extracts the state trajectory from the posterior density, which is subsequently used as the motion plan. The motion planning problem can be interpreted as an estimation problem, where in (8) c_k is the the measurement vector, $d_k \sim \mathcal{N}(0, R)$ is the measurement noise, and $u_k \sim \mathcal{N}(0, Q)$ in (7) is the process disturbance.

D. Disturbance Set for Model Discrepancy

To determine mode transitions based on (3) that the motion planner can accomplish based on (7), we model the discrepancy between (3) and (7) as an additive disturbance $w \in W$. We use a bounded set W and a well-defined, usually non-invertible, function Ψ that maps an element of X into an element of $\hat{X}$ such that

$$\Psi(x) - \hat{x} = w \in W.$$

An example of Ψ is the projection operator onto $\hat{X}$. Formally, the decision maker needs Ψ and W that satisfy the following.

Assumption 1. There exist a bounded set $W \subset \mathbb{R}^{\hat{n}_x}$ and a well-defined function $\Psi : X \to \hat{X}$ such that for all $\hat{x} \in \hat{X}$, $\hat{u} \in \hat{U}$, and $x \in \{x \in X : \Psi(x) - \hat{x} \in W\}$, there exists $u \in U$ that satisfies

$$\Psi\left(f(x, u)\right) - \hat{f}(\hat{x}, \hat{u}) \in W.$$

According to Assumption 1, if there exists a sequence $\hat{\mathbf{x}}_{0:N}$, then there exists a sequence $\mathbf{x}_{0:N}$ within a tube obtained by

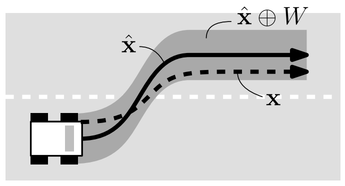


Fig. 4. Projection of $\hat{\mathbf{x}}, \hat{\mathbf{x}} \oplus W$, and $\mathbf{x}$ onto the position space. Considering the disturbance $w \in W$ allows the motion planner to find a trajectory $\mathbf{x}$ according to the more precise model (7) inside the tube surrounding $\hat{\mathbf{x}}$, which is based on the simplified model (3).

$\hat{\mathbf{x}}_{0:N}$ and W. In other words, the set $\{(\hat{x}, x) : \Psi(x) - \hat{x} \in W\}$ is robust control invariant [5] for (3) and (7) with respect to disturbance $\hat{u} \in \hat{U}$ and control $u \in U$. The effects of Assumption 1 are depicted in Fig. 4 using the notation of $\hat{\mathbf{x}} \oplus W = (\hat{x}_k \oplus W)_{k=0}^{N}$. In this paper, we consider W that is the symmetric hyperrectangle

$$\begin{aligned} W := [w_{x,\min}, w_{x,\max}] \times [w_{y,\min}, w_{y,\max}] \\ \times [w_{v,\min}, w_{v,\max}] \times [w_{\theta,\min}, w_{\theta,\max}], \end{aligned}$$

where $w_{\star,\min} = -w_{\star,\max}$ with $\star \in \{x, y, v, \theta\}$. The disturbance set W can be either constructed from the models (3) and (7) or numerically through extensive simulation. The bounded set W exists because the input and state sets of (3) and (7) are bounded. However, if the set W is large, the decision maker may be conservative as the motion planner trajectory may significantly deviate from the decision maker expectation. In this case, the models (3) and (7) should be changed to be more similar, yielding a smaller set W. Conditions similar to Assumption 1 have already been considered, for instance for interconnecting motion planning with vehicle control [15], and shown to be satisfiable in automated driving applications.

E. Modeling of Other Vehicles

We assume that OVs behave non-cooperatively yet rationally, meaning that they do not aim at causing collisions on purpose, but do not make excessive efforts in avoiding them either if the EV does something improper. The OV immediately behind the EV is responsible for maintaining a safety distance from the EV, and the OVs on adjacent lanes do not suddenly change lanes, i.e., without proper notification, if that results in a likely collision with the EV. Thus, we focus on avoiding rear-end collisions with the preceding OV in the EV lane, and collisions during lane changing. We assume that positions and velocities of OVs are measured with no noise and the spatial extent of OVs is known. For the o-th OV, we compute the predicted occupancy set $\mathcal{S}_{k,o} \subset \mathbb{R}^{\hat{n}_x}$ for each $k \in [0, N]$ based on the road geometry, on the spatial extent, and on the available OV position and velocity measurement.

The area that the motion planner must avoid is

$$\mathcal{S}_k = \bigcup_{o=1}^{n_{OV}} \mathcal{S}_{k,o}, \tag{9}$$

where n_{OV} is the number of OVs. Although the predicted occupancy sets can be estimated [16], here we robustify the decision maker by using capture sets, rather than explicitly using the occupancy sets, as described in Section IV.

With respect to the OV immediately in front, the EV is responsible for avoiding rear-end collisions. For that, we define the longitudinal error between the preceding OV and EV as $e = (e_p, e_v) \in E \subseteq \mathbb{R}^{n_e}$, where e_p and e_v are the longitudinal position and velocity differences, respectively. By using as OV motion model the same model used for the EV (3) the longitudinal error dynamics are

$$e_{k+1} = g_e(e_k, \hat{u}_{v,k}, \hat{u}_{v,k}^{OV}), \tag{10}$$

where $\hat{u}_{v,k}^{OV}$ is the longitudinal acceleration of the preceding OV at time step k. In (10), we need to know the relative longitudinal velocities and positions between EV and OV, while we assume that $\hat{u}_{v,k}^{OV}$ is not directly known, yet it is bounded in a known range $\hat{u}_{v,k}^{OV} \in [\hat{u}_{v,\min}^{OV}, \hat{u}_{v,\max}^{OV}]$. Here, $\hat{u}_{v,\min} \leq \hat{u}_{v,\min}^{OV}$ to ensure that the EV can prevent collisions even when the OV brakes at full force. An alternative assumption is that the minimum EV speed is no larger than that of the OVs, which however seems more conservative. Using the same model for OV and EV simplifies the construction of (10), but it is not strictly necessary as additional states can be allowed in (10).

In this paper, we consider a rear-end collision to occur if two vehicles are in the same lane and the position error is less than the minimum safety distance $d_{\min}$. The configurations that cause such rear-end collisions define the *bad set* B,

$$B := \{e = (e_p, e_v) \in E : e_p < d_{\min}\}.$$

The error should be kept out of the bad set B at all times to prevent any rear-end collision.

III. Decision Making: Problem Definition

Before formally stating the decision maker design problem, we define the goal sets. Each mode is associated with a goal set $\mathcal{G}(q_i) \subset \hat{X}$. When the current state $\hat{x}$ is inside the goal set $\mathcal{G}(q_i)$, we say that the EV *has reached* mode q_i. The lane-following goal set $\mathcal{G}(LF)$ is the set of states around the lane's centerline with velocity between $[0, v_{\max}]$, and the stopping goal set $\mathcal{G}(S)$ is the set of states with zero velocity before the stop line. The projections of these goal sets onto the position space are depicted in Fig. 2, where the light blue regions around centerlines represent the projections of the LF goal sets, and the dark blue regions before stop lines represent the projections of the S goal sets.

We design the decision maker such that when it commands a mode transition, there is a sequence of inputs $\hat{\mathbf{u}}_{0:N-1}$ that makes the state reach the next goal set for all disturbances $w \in W$. The decision maker is based on the map information $\mathcal{M} \in \mathbf{M}$, which includes the state of OVs and the mode sequences (1) and (2).

Problem 1. Design a decision maker

$$\pi_\varepsilon : Q \times X \times \mathbf{M} \to \mathcal{E} \tag{11}$$

that, given the current discrete mode $q_i \in Q$, the EV state $x \in X$, and the map information $\mathcal{M} \in \mathbf{M}$, issues the mode transition command $\varepsilon \in \mathcal{E}$ such that there exists an input sequence $\mathbf{u}$ that satisfies:

- **Liveness**: For all q_j, $j \geq i$ there exists a time step k such that $\Psi(x_k(\mathbf{u}, x)) \in \mathcal{G}(q_j)$ or $\Psi(x_k(\mathbf{u}, x)) \in \mathcal{G}(q'_j)$.
- **Safety**: For all time steps k, $\Psi(x_k(\mathbf{u}, x)) \cap \mathcal{S}_k = \emptyset$.

Given a route (1) in the map information $\mathcal{M}$, the liveness property imposes that the EV reaches each mode in the sequence. If it is impossible to execute the mode transition $q_i \rightarrow q_{i+1}$ in route (1), the EV instead should reach the modes of the new route (2). Therefore, the liveness constraints can keep changing as the EV travels along a route due to uncertain behavior of OVs. The safety property imposes that there exists a motion planner trajectory that does not overlap with the occupancy sets of OVs. The decision maker (11) only determines the timing of mode transitions and the switch to the backup sequence to guarantee the existence of an input sequence $\mathbf{u}$ computed by the underlying motion planner to achieve the corresponding goal sets.

In Problem 1, we assume that at any time the normal and alternative sequences both reach the final mode and at least one of them is feasible. If both become infeasible, the navigation system must be capable of planning a different route, yielding a different sequence of driving modes to the final mode. This assumption is commonly satisfied by a navigation system that allows for re-routing when requested by the decision maker.

IV. Decision Maker Design

In this section, after some preliminaries, we present an algorithm that solves Problem 1.

A. Preliminaries

1) Forward and Backward Reachable Sets: The backward reachable set of a goal set is the set of initial states for which there exists an input sequence that reaches the states in the goal set. A forward reachable set of an initial set is the set of states that is reachable with an input sequence starting from the states in the initial set.

Definition 1. For a given set $K \subseteq \hat{X}$, the *one-step backward reachable set* $\mathrm{Pre}(K)$ and the *one-step forward reachable set* $\mathrm{Reach}(K)$ are

$$\mathrm{Pre}(K) := \{\hat{x} \in \hat{X} : \exists \hat{u} \in \hat{U}, \hat{f}(\hat{x}, \hat{u}) \in K\},$$
$$\mathrm{Reach}(K) := \{\hat{x} \in \hat{X} : \exists \hat{x}_0 \in K, \exists \hat{u} \in \hat{U}, \hat{x} = \hat{f}(\hat{x}_0, \hat{u})\}.$$

The k-steps backward reachable set, $k \in \mathbb{Z}_{\geq 0}$, is recursively defined as $\mathrm{Pre}^0(K) = K$, $\mathrm{Pre}^k(K) = \mathrm{Pre}\left(\mathrm{Pre}^{k-1}(K)\right)$. Similarly, the k-steps forward reachable set, $k \in \mathbb{Z}_{\geq 0}$, is recursively defined as $\mathrm{Reach}^0(K) = K$, $\mathrm{Reach}^k(K) = \mathrm{Reach}\left(\mathrm{Reach}^{k-1}(K)\right)$. □

The one-step backward reachable set is also called one-step predecessor set [5], motivating the notation $\mathrm{Pre}(K)$. The overall backward reachable set of $\mathcal{G}(q_i)$ is the union of backward reachable sets for different numbers of steps

$$\bigcup_{k=0}^{\infty} \mathrm{Pre}^k(\mathcal{G}(q_i)).$$

Our design for decision maker (11) maintains the EV state within the backward reachable set of the next goal set until the mode transition is triggered. Thus, it ensures that there exists an input sequence $\hat{\mathbf{u}}$ for the EV to reach the next mode.

Given a route (1), we define the new goal set $\mathcal{G}^*(q_i, q_{i+1})$ as the intersection between the original goal set and the backward reachable set of the subsequent goal sets,

$$\begin{aligned} &\mathcal{G}^*(q_M, \cdot) := \mathcal{G}(q_M), \\ &\mathcal{G}^*(q_i, q_{i+1}) := \mathcal{G}(q_i) \cap \bigcup_{k=0}^{\infty} \mathrm{Pre}^k(\mathcal{G}^*(q_{i+1}, q_{i+2})), \end{aligned} \tag{12}$$

for $i = 1, 2, \ldots, M-2$, and $\mathcal{G}^*(q_{M-1}, q_M) := \mathcal{G}(q_{M-1}) \cap \bigcup \mathrm{Pre}^k(\mathcal{G}^*(q_M, \cdot))$ for $i = M-1$. In other words, $\hat{x}_0 \in \mathcal{G}^*(q_i, q_{i+1})$ implies that the state is inside the original goal set $\mathcal{G}(q_i)$ and there is an input sequence $\hat{\mathbf{u}}$ that makes $\hat{x}_k(\hat{\mathbf{u}}, \hat{x}_0) \in \mathcal{G}^*(q_{i+1}, q_{i+2})$ for some $k \in \mathbb{Z}_{\geq 0}$. In Section V, we explain how to efficiently compute $\mathcal{G}^*(q_i, q_{i+1})$ or its under-approximation, based on computing the intersection in (12), which is more efficient than computing the backward reachable sets separately.

For an intersection crossing, transition from S to LF,

$$\mathcal{G}^*(S, LF) = \mathcal{G}(S), \tag{13}$$

because the road intersections are assumed to be constructed such that there exists an input to travel through them. This allows the modular computation of backward reachable sets in (12). Specifically, although by definition, $\mathcal{G}^*(q_i, q_{i+1})$ is the intersection between $\mathcal{G}(q_i)$ and the backward reachable sets of all the subsequent goal sets, we can compute it by considering the subsequent modes until the stopping mode appears.

Also, as discussed in Section II, when the mode transition $q_i \rightarrow q_{i+1}$ is impossible for any input due to OVs, the alternative mode sequence (2) provided by the navigation system is used, in which a new mode transition $q_i \rightarrow q'_{i+1}$ is possible to execute. This implies that the current state $\hat{x}$ is outside of $\mathcal{G}^*(q_i, q_{i+1})$ but inside $\mathcal{G}^*(q_i, q'_{i+1})$, so that the EV is able to reach q'_{i+1} at some future time step.

2) Control Invariant Sets: A control invariant set is the set of states for which there exists an input that keeps the state evolution inside the set.

Definition 2. A set $K \subseteq \hat{X}$ is *control invariant* for (3) if for any $\hat{x} \in K$ there exists $\hat{u} \in \hat{U}$ such that $\hat{f}(\hat{x}, \hat{u}) \in K$. □

For ensuring safety, i.e., collision avoidance, we use the *capture set* of the bad set B.

Definition 3. Given (10) and the set B, the *one-step capture set* $\mathcal{C}(B) \subseteq \mathbb{R}^{n_e}$ is

$$\begin{aligned} \mathcal{C}(B) = \{&e \in E : \\ &\forall \hat{u}_v \in \hat{U}_v, \exists \hat{u}_v^{OV} \in \hat{U}_v^{OV}, g_e(e, \hat{u}_v, \hat{u}_v^{OV}) \in B\}, \end{aligned}$$

The k-steps backward reachable set, $k \in \mathbb{Z}_{\geq 0}$, is recursively defined as $\mathcal{C}^0(B) = B$, $\mathcal{C}^k(B) = \mathcal{C}\left(\mathcal{C}^{k-1}(B)\right)$. □

When $e \in \mathcal{C}^0(B)$, regardless of the EV's control input $\hat{u}_v \in \hat{U}_v$ there is an input of the preceding OV that results in a collision at the next time step. The overall capture set is the union of the capture sets for any number of steps

$$\bigcup_{k=0}^{\infty} \mathcal{C}^k(B).$$

By Definition 3, if the error is outside the capture set, there must exist a control input $\hat{u}_v \in \hat{U}_v$ that keeps the error outside of the capture set for any $\hat{u}_v^{OV} \in \hat{U}_v^{OV}$ for all future times. Thus, the complement of the capture set is control invariant for the system describing the longitudinal error (10).

Also, the stopping mode goal set $\mathcal{G}(S)$ is control invariant, since if the state is within $\mathcal{G}(S)$, the vehicle is stopped, i.e., $v = 0$, and zero acceleration $\hat{u}_v = 0$ keeps it within the set.

B. Decision Maker Design

The disturbance $w \in W$ models the difference between (3) and (7), so that if a trajectory of (3) achieving the properties in Problem 1 for all $w \in W$ exists, then a trajectory of (7) satisfying such properties is guaranteed to exists.

For the goal set, the effect of W is to tighten the set

$$\hat{\mathcal{G}}(q_i, q_{i+1}) := \mathcal{G}^*(q_i, q_{i+1}) \ominus W.$$

Similarly, W_E tightens the complement of the capture set

$$\hat{\mathcal{I}} := \left(\bigcup_{k=0}^{\infty} \mathcal{C}^k(B) \right)^c \ominus W_E,$$

where $W_E := \{e = h(\hat{x}_k) - h(\Psi(x_k)) \in \mathbb{R}^{n_e} : \exists \hat{x} \in \hat{X}, x \in X,\ \hat{x}_k - \Psi(x_k) \in W\}$, that is, W_E is the set of disturbances that represents the discrepancy between the longitudinal states of (3) and (7). In proving the main theorem presented later in this section, we let the set W_E be an hyperrectangle $W_E := [w_{e_p,\min}, w_{e_p,\max}] \times [w_{e_v,\min}, w_{e_v,\max}]$ that over-approximates the actual disturbance set W_E, for simplicity.

The computations for $\hat{\mathcal{G}}(q_i, q_{i+1})$ and $\hat{\mathcal{I}}$ mostly consist in the computations of $\mathcal{G}^*(q_i, q_{i+1})$ and $\left(\bigcup_{k=0}^{\infty} \mathcal{C}^k(B)\right)^c$, respectively, because the Pontryagin set difference is a computationally efficient operation. Thus, in Section V, we compute the efficient approximations of $\hat{\mathcal{G}}(q_i, q_{i+1})$ and $\hat{\mathcal{I}}$ by determining the set boundaries from the extreme values of $w \in W$.

To describe the solution, we define a sequence of states over the planning horizon that are considered to be safe.

Definition 4. Given the current state x_0 and next mode q_{i+1}, a *safe reference state sequence*, denoted by $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$, is a finite sequence $\hat{\mathbf{x}}_{0:N}$ that satisfies

(i) $\Psi(x_0) - \hat{x}_0 \in W$;
(ii) $\hat{x}_k \notin \mathcal{S}_k \oplus W$ for all $k \in \{0, \ldots, N\}$;
(iii) $\hat{x}_{N_{\text{reach}}} \in \hat{\mathcal{G}}(q_{i+1}, q_{i+2})$ for some $N_{\text{reach}} \in \{0, \ldots, N\}$;
(iv) If $q_{i+1} = S$, then $\hat{x}_k \in \hat{\mathcal{G}}(S, \cdot)$ for all $k \in \{N_{\text{reach}}, \ldots, N\}$. If $q_{i+1} = LF$, then $e_k \in \hat{\mathcal{I}}$ for all $k \in \{0, \ldots, N\}$. □

When the decision maker considers the mode transition $q_i \to q_{i+1}$, a safe reference state sequence is a sequence of states over the planning horizon N that does not collide with the OVs (Condition (ii)), reaches the goal set of the next mode q_{i+1} at some time step N_{reach} (Condition (iii)), and remains inside a control invariant set, which is either $\hat{\mathcal{G}}(S, \cdot)$ or $\hat{\mathcal{I}}$ (Condition (iv)). Recall that the sets $\mathcal{G}^*(S, \cdot)$ and $(\cup \mathcal{C}^k(B))^c$ are control invariant. It is not difficult to prove that the shrunken sets $\hat{\mathcal{G}}(S, \cdot)$ or $\hat{\mathcal{I}}$ are still control invariant; for $\hat{\mathcal{I}}$, the proof is in Lemma 2 in Appendix A. Condition (iv) ensures liveness by imposing that in stopping mode the EV is in the stopping goal set, and that in lane following mode the error state is outside the capture set to prevent rear-end collisions with OVs.

Remark 1. If the predicted OVs occupancy sets $\mathcal{S}_k$ are significantly wrong, the performance may degrade. However, the motion planner can ultimately avoid collisions, at the expense that the goal may be missed and the current mode may become infeasible. In those cases, the decision maker switches to the alternative sequence, or a different emergency route may be needed, yet the vehicle safety will be preserved. Moreover, the occupancy sets $\mathcal{S}_k$ in Condition (ii) of Definition 4 are effectively used only to avoid "corner collisions" in two cases: when the EV changes between two lanes and when it turns inside an intersection. Thus, they are unnecessary if a safety distance during lane change is imposed, and if at most one vehicle is allowed to be inside the intersection. In all other cases, collision avoidance, and hence Condition (ii), is enforced by the constraints on the capture set in Condition (iv). The capture set needs the position and velocity of OVs, but is robust to their longitudinal input range $\hat{U}_v^{OV}$. Moreover, capture sets are based on closed-loop prediction subject to uncertainty [17], and provide a less conservative method than open-loop predictions of OV motion.

Thus, the algorithm for solving Problem 1 is as follows.

Algorithm 1 Decision maker $\pi_\varepsilon(q_i, x_0, \mathcal{M})$

```
• If $\Psi(x_0) \in \mathcal{G}^*(q_i, q_{i+1})$,
    – If there exists $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$, then return $\pi_\varepsilon(q_i, x_0, \mathcal{M}) = \varepsilon_{i+1}$.
    – Else if $\text{Reach}(\{\hat{x}_0\}) \cap \hat{\mathcal{G}}(q_i, q_{i+1}) = \emptyset$ for all $\hat{x}_0$ such that $\Psi(x_0) - \hat{x}_0 \in W$,
        ∗ If there exists $\hat{\mathbf{x}}_{\text{safe}}(q'_{i+1}, x_0)$, then return $\pi_\varepsilon(q_i, x_0, \mathcal{M}) = \varepsilon'_{i+1}$.
        ∗ Otherwise, return $\pi_\varepsilon(q_i, x_0, \mathcal{M}) = \varepsilon_0$ and request $\Psi(x_1) \in \mathcal{G}^*(q_i, q'_{i+1})$.
    – Otherwise, return $\pi_\varepsilon(q_i, x_0, \mathcal{M}) = \varepsilon_0$ and request $\Psi(x_1) \in \mathcal{G}^*(q_i, q_{i+1})$.
• Otherwise, return $\pi_\varepsilon(q_i, x_0, \mathcal{M}) = \varepsilon_0$.
```

Algorithm 1 keeps the state inside the backward reachable set of all the subsequent goal sets and changes the discrete mode to q_{i+1} when the EV can safely reach the next goal set $\hat{\mathcal{G}}(q_{i+1}, q_{i+2})$ within the N-step planning horizon. That is, Algorithm 1 returns:

- ε_{i+1}, if there exists $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$.
- ε'_{i+1}, if the state at the next time step cannot be inside the backward reachable set of the goal set of q_{i+1} for any control input, i.e., $\text{Reach}(\{\hat{x}_0\}) \cap \hat{\mathcal{G}}(q_i, q_{i+1}) = \emptyset$, and there exists a state trajectory that reaches the goal set of q'_{i+1} within the planning horizon. In this case, the EV will be able to reach the backup mode q'_{i+1} at some future time step.
- ε_0, i.e., the void input that maintains the same mode, if the next goal set cannot be reached within the planning

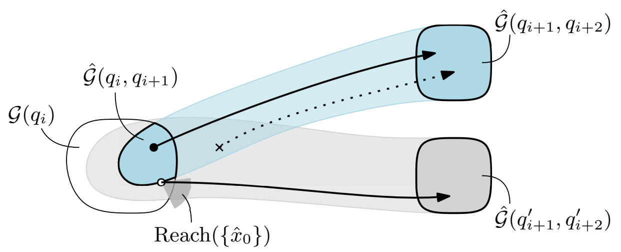

$\hat{\mathcal{G}}(q_{i+1},q_{i+2})$
$\hat{\mathcal{G}}(q_i,q_{i+1})$
$\mathcal{G}(q_i)$
$\hat{\mathcal{G}}(q'_{i+1},q'_{i+2})$
$\text{Reach}(\{\hat{x}_0\})$

Fig. 5. Illustration of the decision maker in Algorithm 1. When the state is at •, the decision maker returns ε_{i+1} because there exists a safe reference state sequence that reaches $\hat{\mathcal{G}}(q_{i+1},q_{i+2})$. At ∘, the decision maker returns ε'_{i+1} because the state will go outside $\hat{\mathcal{G}}(q_i,q_{i+1})$ regardless of control input and there exists a safe reference state sequence that reaches $\hat{\mathcal{G}}(q'_{i+1},q'_{i+2})$. At ×, the decision maker returns ε_0 because the state is outside of $\hat{\mathcal{G}}(q_i,q_{i+1})$ and in the middle of moving towards the next goal set $\hat{\mathcal{G}}(q_{i+1},q_{i+2})$.

horizon. In this case, the algorithm enforces that the next state $\Psi(x_1)$ is inside the current goal set $\mathcal{G}^*(q_i,q_{i+1})$ or $\mathcal{G}^*(q_i,q'_{i+1})$ to keep the state in the current goal set. Hence, the decision maker restricts the range of velocities for the motion planner, as described in Section V-B.

- ε_0 also if $\Psi(x_0) \notin \mathcal{G}^*(q_i,q_{i+1})$, as the state is inside the backward reachable set $\bigcup_k \text{Pre}^k(\mathcal{G}^*(q_{i+1},q_{i+2}))$, but outside the current goal set $\mathcal{G}(q_i)$. Since the transition to q_i is not complete, the current goal must be maintained.

Some operations of Algorithm 1 are shown in Fig. 5.

The decision maker in Algorithm 1 returns only one discrete input value, as this makes the decision maker deterministic and simplifies its analysis. However, the decision maker can be extended to return a set of discrete inputs, leaving the final choice to a motion planner that can handle multiple modes in parallel, e.g., [1]. In this case, when $\Psi(x) \in \mathcal{G}^*(q_i,q_{i+1})$ and there exists $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1},x)$, the decision maker may provide to the motion planner $\pi_\varepsilon(q_i,x,\mathcal{M}) = \{\varepsilon_{i+1},\varepsilon_0\}$ and let the motion planner select the one mode that results in the best performance. Still, the decision maker guarantees the existence of a feasible trajectory for all the discrete inputs provided to the motion planner, which then will not waste computation or get stuck in trying to achieve an unfeasible goal.

Next, we state the main result of this paper.

Theorem 1. *Under Assumption 1, if at some time instant Algorithm 1 returns ε_1 to initiate the first mode q_1 of a route* (1)*, then Algorithm 1 provides the solution of Problem 1.*

The proof of Theorem 1 is in Appendix A.

V. Implementation of the Decision Maker

In this section, we present our approach to the approximate computation of the reachable sets in Algorithm 1, based on model (3), which is efficient and can be executed at high rates. This allows the decision maker to retain reactivity to a changing environment, and the possibility to implement the method in automotive-grade embedded platforms that are more limited than standard desktop computers [18]. We also discuss how to integrate the decision maker with the motion planner.

A. Efficient Computation of the Conditions in Algorithm 1

In this section, we present our approach to the computation of the sets $\mathcal{G}^*(q_i,q_{i+1})$ and $\left(\bigcup \mathcal{C}^k(B)\right)^c$. The evaluation of all the other conditions will be discussed in Appendix B. The main idea is to use the extreme inputs $\hat{u}_{v,\min}, \hat{u}_{v,\max}, \hat{u}_{\theta,\min}$, and $\hat{u}_{\theta,\max}$ to characterize the boundaries of the sets. Although the extreme inputs generate sharp trajectories, these are only used in determining the feasibility of the mode transitions. The motion planner will generate significantly smoother trajectories by exploiting the tube around the sharp trajectories of the decision maker; see Fig. 4.

For a lane change, where we suppose that lane 1 is on the left side of lane 2, we compute the sharp lane-changing trajectory computed with $\hat{u}_{\theta,\min}$ and $\hat{u}_{\theta,\max}$ in the lateral dynamics. Consider the lane changing $LF_2 \to LF_1$, for some nonnegative integers k_1 and k_2, define $\hat{\mathbf{u}}^{LF_2,LF_1}_{0:N-1} = (\hat{u}_{v,k},\hat{u}_{\theta,k})^{N-1}_{k=0}$ as

$$\begin{aligned}\hat{u}_{v,k} &:= \begin{cases}\hat{u}_{v,\min} & \text{if } v_k > v_{\min}\\ 0 & \text{if } v_k = v_{\min}\end{cases}\\ \hat{u}_{\theta,k} &:= \begin{cases}\hat{u}_{\theta,\max} & \text{for } 0 \le k \le k_1-1,\\ \hat{u}_{\theta,\min} & \text{for } k_1 \le k \le k_2-1,\\ \hat{u}_{\text{lane-following},k} & \text{for } k_2 \le k \le N-1.\end{cases}\end{aligned} \tag{14}$$

Here, $\hat{u}_{\text{lane-following},k}$ denotes the orientation rate that follows the centerline of the current lane. The integers k_1 and k_2 can be computed by a simple numerical search (e.g., bisection methods) to satisfy conditions given later in this section. For the opposite lane change, $LF_1 \to LF_2$, the computations are as in (14) but swapping the roles of $\hat{u}_{\theta,\max}$ and $\hat{u}_{\theta,\min}$.

1) Computation of $\mathcal{G}^*(q_i,q_{i+1})$: Recall from definition (12) that $\mathcal{G}^*(q_i,q_{i+1})$ is the intersection between the goal set $\mathcal{G}(q_i)$ and the backward reachable set of $\mathcal{G}^*(q_{i+1},q_{i+2})$. Since the goal set $\mathcal{G}(q_i)$ has simple geometry, the direct computation of $\mathcal{G}^*(q_i,q_{i+1})$ is easier than computing its subsets separately. In fact, the goal set $\mathcal{G}(q_i)$ is determined only by the longitudinal dynamics, where in details $\mathcal{G}(LF)$ and $\mathcal{G}(S)$ consist of the position (p_x and p_y) around the lane's centerline, and the position right before the stop line, respectively, with corresponding yaw angles and velocities.

As the longitudinal dynamics are monotone with respect to the input, the boundary of reachable sets is determined only by the extreme inputs [17]. For braking, $LF \to S$, the boundary of $\mathcal{G}^*(LF,S)$ is

$$\{\hat{x} \in \mathcal{G}(LF) : \exists k \in \mathbb{Z}_{\ge 0} \text{ s.t. } y_k(\hat{\mathbf{u}}_{v,\min},\hat{x}) \in h(\mathcal{G}(S))\}, \tag{15}$$

where $y_k(\hat{\mathbf{u}}_{v,\min},\hat{x})$ is the longitudinal state reached at time step k starting from $\hat{x}$ with a sequence of inputs all equal to $\hat{u}_{v,\min}$, and $h(\mathcal{G}(S))$ is defined as $\{h(\hat{x}) : \hat{x} \in \mathcal{G}(S)\}$. According to (15), at the boundary of $\mathcal{G}^*(LF,S)$, the state enters the goal set $\mathcal{G}(S)$ at some time step k by maximum braking. Fig. 6a shows an illustration of the set $\mathcal{G}^*(LF,S)$.

For lane changes, the dynamics are not always monotone, and hence we under-approximate the boundary of the set

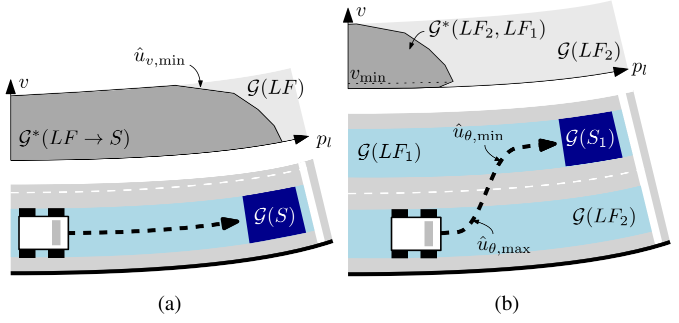
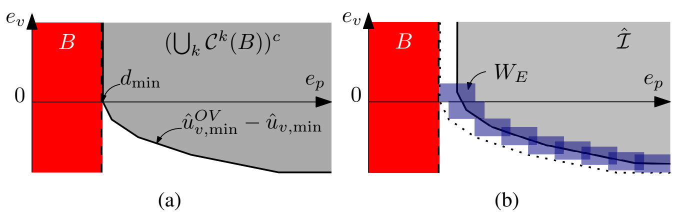


Fig. 6. Projections of $\mathcal{G}(q_i)$ and (approximation of) $\mathcal{G}^*(q_i, q_{i+1})$ onto the p_l–v plane (upper graphs) and the p_x–p_y plane (lower figures). The extreme inputs are used to compute the boundaries of such sets.



e_v
B
$(\bigcup_k \mathcal{C}^k(B))^c$
$d_{\min}$
e_p
0
$\hat{u}^{OV}_{v,\min} - \hat{u}_{v,\min}$
(a)
e_v
B
$\hat{\mathcal{I}}$
W_E
e_p
0
(b)

Fig. 7. Maximum braking of the EV and preceding OV determines the complement of the capture set (gray region in (a)) and its shrunken set $\hat{\mathcal{I}}$ (gray region in (b)).

$\mathcal{G}^*(LF_2, LF_1)$ as

$$\{\hat{x} \in \mathcal{G}(LF_2) : \\ \exists k, k_1, k_2 \in \mathbb{Z}_{\geq 0} \text{ s.t. } \hat{x}_k(\hat{\mathbf{u}}^{LF2,LF1}_{0:N-1}, \hat{x}) \in \mathcal{G}^*(LF_1, S_1)\}. \tag{16}$$

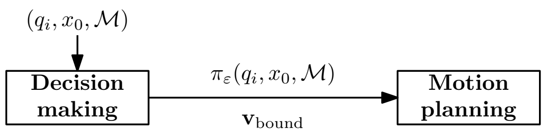
According to (16), at the boundary of $\mathcal{G}^*(LF_2, LF_1)$, the sharp trajectory $\hat{x}_k(\hat{\mathbf{u}}^{LF2,LF1}_{0:N-1}, \hat{x})$ reaches at some time step k the next lane and also the backward reachable set of the stopping goal set. Fig. 6b shows the under-approximation of $\mathcal{G}^*(LF_2, LF_1)$.

It is worth noting that the combined longitudinal-lateral dynamics (3) are not monotone with respect to $(\hat{u}_v, \hat{u}_\theta)$, whereas the longitudinal only dynamics (6) are monotone with respect to $\hat{u}_v$. Thus, the computation of the boundary of $\mathcal{G}^*(LF_2, LF_1)$ by (16) is an under-approximation as we consider only the extreme inputs rather than the entire input set, whereas the computation of the boundary of $\mathcal{G}^*(LF, S)$ by (15) is exact. Under certain driving conditions, the combined longitudinal-lateral dynamics of vehicles are in fact monotone [19], in which case also the computation (16) becomes exact.

2) Computation of $\left(\bigcup \mathcal{C}^k(B)\right)^c$: We compute the boundary of the complement of the capture set based on the extreme braking of the preceding OV and EV. Specifically,

$$\left(\bigcup \mathcal{C}^k(B)\right)^c = \{e \in E : e_k(\hat{u}_{v,\min}, \hat{u}^{OV}_{v,\min}, e) \notin B, \forall k \geq 0\},$$

where $e_k(\hat{u}_{v,\min}, \hat{u}^{OV}_{v,\min}, e)$ is the longitudinal error reached at time step k by applying $\hat{u}_v = \hat{u}_{v,\min}$ and assuming $\hat{u}^{OV}_v = \hat{u}^{OV}_{v,\min}$, starting from e. Hence, inside the complement of the capture set, applying maximum braking of the EV can always avoid rear-end collisions regardless of the input of the preceding OV. Fig. 7a shows an illustration of $\left(\bigcup_j \mathcal{C}^j(B)\right)^c$.



$(q_i, x_0, \mathcal{M})$
Decision making
$\pi_\varepsilon(q_i, x_0, \mathcal{M})$
$\mathbf{v}_{\text{bound}}$
Motion planning

Fig. 8. Integration between the decision making and motion planning systems. The decision maker propagates the discrete input $\pi_\varepsilon(q_i, x_0, \mathcal{M})$ and a profile for the velocity range $\mathbf{v}_{\text{bound}}$ that bounds the range of velocities of the motion planner.

B. Integration with the Motion Planner

The motion planner in Section II-C generates a state trajectory and corresponding input signals over the planning horizon that connects the initial state x_0 with the goal set $\mathcal{G}^*(q_{i+1}, q_{i+2})$. Specifically, a state sequence of the motion planner should satisfy the following three conditions:

(a) no EV state in the sequence overlaps with any of the OV occupancy sets;
(b) some EV states in the sequence reach the goal set $\mathcal{G}^*(q_{i+1}, q_{i+2})$ at some time step;
(c) the last state in the sequence resides in a control invariant set, either $\mathcal{G}(S, \cdot)$ or $\left(\bigcup_j \mathcal{C}^j(B)\right)^c$.

These conditions are the same as (ii)–(iv) in Definition 4 of the safe reference state sequence $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$, except (a)–(c) do not consider the disturbance $w \in W$ because the motion planner considers the more detailed dynamical model (7).

For achieving (a)–(c), the decision maker propagates information necessary for the motion planner to achieve such a state sequence; see Fig. 8 that depicts the integration of decision making with motion planning. Specifically, the decision maker propagates a profile for the velocity range $\mathbf{v}_{\text{bound}} = (\mathbf{v}_l, \mathbf{v}_h)$ over the planning horizon as a function of the longitudinal distance to bound the velocity of a motion planner state sequence $\mathbf{x_{0:N}}$, where $\mathbf{v}_l$ and $\mathbf{v}_h$ are the velocity lower and upper bound profiles, respectively. This transmits all necessary information on $\mathcal{G}^*(q_i, q_{i+1})$ and $\left(\bigcup_j \mathcal{C}^j(B)\right)^c$, since their boundaries are fully described by the velocity profiles as shown in Figs. 6 and 7a. For instance, if $q_i = S$, the velocity profile $\mathbf{v}_h$ decreases to zero at the stop location, to ensure that stopping can be performed. Similarly, the constraints $\Psi(x_1) \in \mathcal{G}^*(q_i, q_{i+1})$ and $\Psi(x_1) \in \mathcal{G}^*(q_i, q'_{i+1})$ in Algorithm 1 are imposed through the velocity profile $\mathbf{v}_{\text{bound}}$. Then, the motion planner only needs to satisfy velocity limits (see for example [1]). The motion planner ensures avoidance of collision with OVs by Condition (a) above, since by construction any trajectory that does not satisfy Condition (a) will have a zero fitness or an infinite cost.

VI. Validation in Simulation and Experiments

We integrate the decision maker given in Algorithm 1 with the motion planner and a trajectory tracking vehicle controller [20] and implement it on the test scenario consisting of the EV and two OVs on an ∞-shaped loop with two one-direction lanes, as shown in Fig. 9. On this scenario, we perform both software simulation and hardware experiments. For experiments, we use Hamster v5 robots [21] that are small-scale car-like robots with Ackerman steering geometry similar to regular cars, equipped with an inertial measurement unit, lidar, camera, and motor encoders and with Raspberry PI3 processors. For robot localization, we use a camera-based Optitrack [22] motion-capture system that fuses data from 10 cameras to determine the pose of the rigid bodies of EV and OVs, as barycenter position and orientation.

Algorithm 1 and the motion planner are implemented in MATLAB, while the trajectory tracking vehicle controller is a Nonlinear Model Predictive Control (NMPC) [23] implemented in C. All these modules are implemented as nodes in a ROS network and executed on a desktop workstation[1]. The EV and OVs motion simulators and hardware drivers are also implemented as ROS network nodes in simulations and experiments, respectively, while the first ones are executed in the same computer as the controller, and the second ones are executed on the Hamsters' Raspberry PIs. The decision making and motion planning algorithms are executed every $0.85\,\text{s}$, while the NMPC is executed every $0.04\,\text{s}$. According to the discrete mode determined by the decision maker, the motion planner generates a trajectory and the NMPC computes the commands for steering angle and velocity to track the trajectory. These control commands are transmitted to the EV simulator or hardware drivers, for simulations and experiments, respectively. Instead, the OVs are controlled by fairly standard PI tracking controllers commanding steering position and velocity, with inner-loop PD actuation controllers. Also, the OVs are equipped with a simple logic to stop at the intersections for a fixed amount of time, and proceed when the intersection is free. In addition, an operator can command the OVs to change lane and target velocity.

As for prediction models, the decision maker uses the kinematic unicycle model (4), the motion planner uses a kinematic bicycle model, and the NMPC uses a kinematic bicycle model with actuation dynamics, delays, and a time varying input offset, which is estimated in real-time by an extended Kalman filter to provide integral action [20]. The set W is obtained offline by extensive simulation, and slightly enlarged to account for additional robustness. As discussed in Section II-E, for the OVs we use the the same model as for the EVs where the heading angle is considered to be aligned with the centerlane, and we compute W_E similarly to W.

The decision maker operates in eight discrete modes

$$Q = \{LF_1, LF_2, LF_3, LF_4, S_1, S_2, S_3, S_4\},$$

where LF_i represents the mode of following lane i and S_i represents the mode of stopping at the end of lane i; see Fig. 9 for numbering. For compactly showing the results, in Figs. 10-12 we show the actions that the decision maker is commanding rather than the internal modes. Such actions are lane changing (LC), lane keeping (LK), stopping (ST), turning left (TL), going forward (GF), and turning right (TR). In simulation and experiments, we do not specify the entire route ahead of time; rather, the decision maker receives randomly generated candidate modes and accepted or rejected them according to the conditions described in Section IV to run continuously and

[1]We use a computer with Intel Core i7 processor at 4.20 GHz and 64 GB RAM for simulation and a computer with Intel Core i7 processor at 3.20 GHz and 64 GB RAM for experiments.

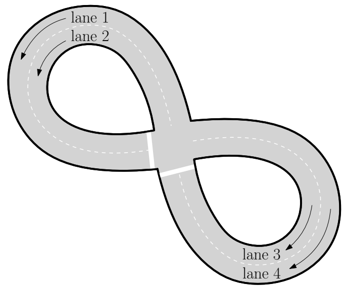


Fig. 9. Test circuit for simulation and experimental validation. The lanes and the stop lines at their end are also shown, LF_i, S_i, respectively, $i \in \{1,2,3,4\}$.

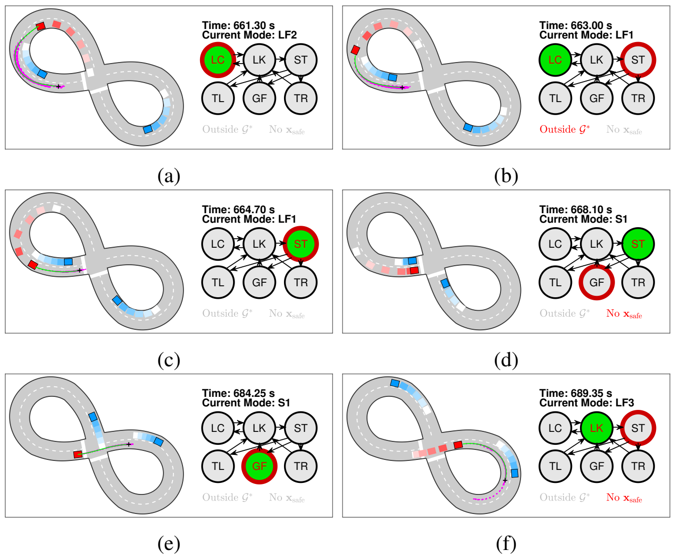


Fig. 10. Simulation results. The EV (the red box) starts changing lanes in (a), and rejects the request of stopping because it has not reached lane 1 yet in (b). It starts braking in (c), waits for the OVs (the blue boxes) to cross the intersection in (d), starts crossing in (e), and completes the crossing in (f).

stress-test the system. Similarly, the decision maker determines if it is necessary to transition to the mode sequence in the alternative route, which always amounts to remain in the same lane and stop in the corresponding stop area.

At the intersection, each vehicle must stop for at least three seconds and start moving when there is no other vehicle within the intersection area. While any intersection coordination rule can be implemented to the EV and OVs, we programmed the EV to always yield to the OVs. That is, the EV waits as long as there is an OV that is stopped at or crossing the intersection.

A. Simulation Results

Fig. 10 shows the simulation results. The red and blue rectangles represent the EV and OVs, respectively. The green line is the trajectory that the motion planner generates using particles (purple dots) according to discrete inputs of the decision maker, and the plus marker indicates a reached point in goal sets. In the directed graph, a green node is the current action of the decision maker, and a node surrounded by the red thick circle is the randomly generated requested action. When the decision maker accepts the requested action, this is shown as a green circle with red border.

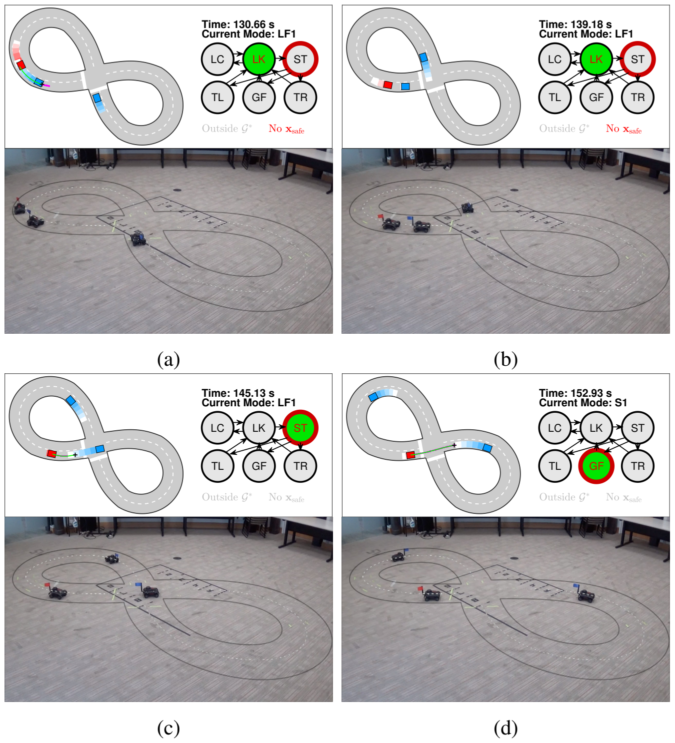


Fig. 11. Experimental results where the EV (car with the red flag) follows an OV (car with the blue flag) and slows down as the OV stops. After the OV leaves, the EV reaches the stop line and crosses the intersection.

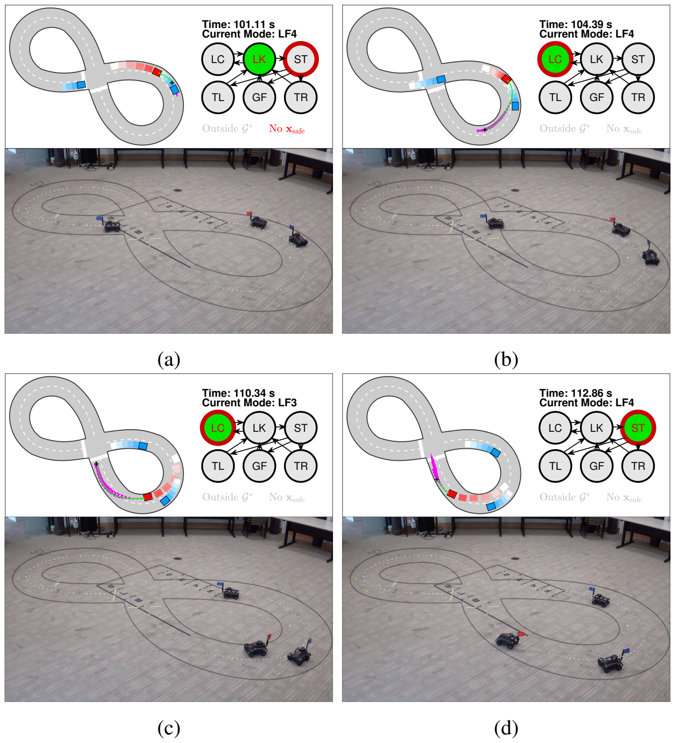


Fig. 12. Experimental results of the EV overtaking an OV. The decision maker accepts the request of lane changing in (c) because the EV is far enough from the intersection to sequentially change lane and stop at the stop line.

In Fig. 10a, the decision maker determines that the EV is able to change lane from 2 to 1, and the motion planner generates a trajectory that accomplishes such a lane change. In Fig. 10b, the mode has changed from LF_2 to LF_1. Because the EV is in the middle of lane change and hasn't reached lane 1 yet, the EV's state is outside the set $\mathcal{G}^*(LF_1, \cdot)$ and the decision maker rejects the request of stopping. Near the intersection in Fig. 10c, the decision maker returns stopping, and changes the discrete mode from LF_1 to S_1. Note that a lane change is allowed in Fig. 10a because the EV can reach the stopping mode even after such a lane change. In Fig. 10d, the EV successfully stops before the intersection and waits for the two OVs crossing the intersection. When the decision maker verifies that the intersection is not occupied by any of the OVs in Fig. 10e, it lets the motion planner generate a trajectory that goes through the intersection. After crossing in Fig. 10f, the EV reaches lane 3 and maintains the lane because it is still too far from the stop line to start the requested braking action. During this segment of simulation tests, the sequence of the executed driving modes is $LF_2 \rightarrow LF_1 \rightarrow S_1 \rightarrow LF_3$.

The computation time of Algorithm 1 is never longer than $0.0079\,\text{s}$. This confirms that the decision maker achieves fast computation, thus capable of reacting to rapidly changing environments and also suitable for implementation in platforms with limited computational capabilities.

B. Experimental Results

In the experiments, we use three car-like Hamster robots whose pose is measured by the OptiTrack system, while the on board IMU and wheel speed are used to estimate velocity and yaw rate. Shown in Figs. 11 and 12 are two scenarios. The EV is the Hamster with the red tail flag, represented by the red rectangle, and the OVs are the Hamsters with the blue tail flag, represented by blue rectangles. The first scenario, shown in Fig. 11, is the queuing and intersection crossing. Specifically, in Fig. 11a, the decision maker rejects the request of stopping because there is no safe reference state sequence $\hat{\mathbf{x}}_{\text{safe}}(S_1, x_0)$ that reaches the stop line within the planning horizon due to the presence of the preceding OV. In Fig. 11b, the EV stops behind the OV, while still maintaining the current mode of lane keeping. In Fig. 11c, as the OV no longer occupies the stop goal region, the decision maker allows the EV to resume moving and reaching the stop goal region. In Fig. 11d, after waiting for three seconds, the EV crosses the intersection.

The second scenario, shown in Fig. 12, is overtaking and stopping at the intersection. Specifically, the decision maker rejects the request of stopping action in Fig. 12a because the EV is too far from the intersection to fully stop at the stop line within the planning horizon. In Fig. 12b, it accepts the request of lane-change action because there exists a safe reference state sequence $\hat{\mathbf{x}}_{\text{safe}}(LF_3, x_0)$. In Fig. 12c, the EV has changed lane without conflict (so mode has changed to LF_3), and again, the decision maker accepts another request of lane-change action. In Fig. 12d, the decision maker allows the stopping action for the EV to stop before the intersection. The decision maker accepts the second lane change in Fig. 12c because at that point, it was possible for the EV to sequentially perform a lane change and a full stop. For various other scenarios, see the video of the experiments at [24].

In Fig. 13, we present a scenario when the EV starts to

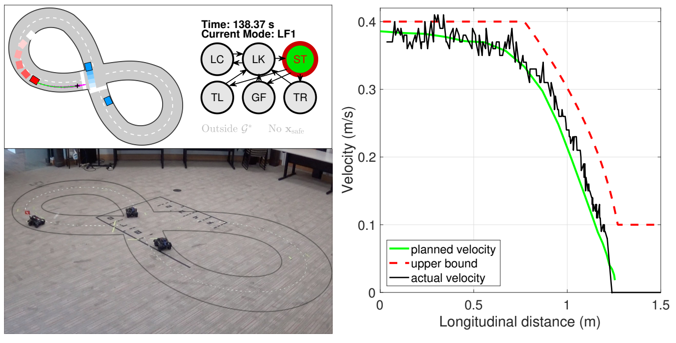


Fig. 13. In the condition shown in (a), the velocity upper bound profile $\mathbf{v}_h$ by the decision maker (red dotted line) and the velocity profile computed by the motion planner (green line) are shown in (b). The velocity upper bound guides the motion planner in computing a trajectory stopping in the stop region.

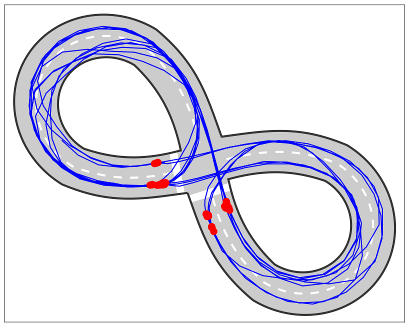


Fig. 14. Trace of the EV for 600 s. The red dots represent the positions at which the EV has stopped. During the entire experiment, the EV stopped before the stop lines, crossed intersections when free, and avoided all obstacles. The motion planner never returned failure, and hence could always achieve the goal received from the decision maker, validating the method developed in this paper.

brake and velocity profiles generated at that moment with respect to the longitudinal distance from its current position. The decision maker computes the velocity profile $\mathbf{v}_{\text{bound}}$ (red line in Fig. 13b) with $v_{\max} = 0.4$ m/s, based on the backward reachable set $\mathcal{G}^*(LF_1, S_1)$ (depicted in Fig. 6a). The motion planner generates a velocity profile (green line in Fig. 13b) that satisfies the velocity bound. By satisfying the bound constraint, the motion planner is able to generate a trajectory that reaches the stop goal region with zero speed[2]. In Fig. 13b, the actual velocity as the EV progresses is shown as the black line.

In Fig. 14, we present the entire trace of the EV for 600 s of the experiment. The EV stopped at the positions indicated by the red dots, which are all in the goal sets for stopping modes. Fig. 14 shows that the EV executes lane changing, braking, and intersection crossing without conflicts with the OVs, and indicates that the decision maker always continues operating and never reaches a fault condition.

In the experiments, there are a number of imperfect conditions, such as: (i) ROS communication delays, ranging from 0.1 to 0.3 s between sensing and command execution, and lack of synchronization, due to all vehicles operating as stand-alone ROS nodes with the EV actually using different nodes for the different layers; (ii) imperfect tracking due to the entire control stack using approximated models; (iii) imprecise actuation due to low-cost servo-motors and traction motors; (iv) noise in the sensors, especially for velocity and IMU, but also for positioning; (v) uneven surfaces, due to carpet with power plug holes and deterioration; (vi) imperfect prediction of the OVs. All these conditions cause uncertainties in the prediction of the EV and OVs behavior. However, we observe that the decision maker is robust to such uncertainties. In particular, the decision maker is robust to the uncertainty in the OVs behavior due to the use of the capture set and since it operates in conjunction with the motion planning system and the vehicle control system that reject part of the disturbances.

[2]In the experiment, the EV stops if its speed is lower than 0.1 m/s.

VII. Conclusion

We presented the design of a decision maker that ensures safety and liveness. From a sequence of modes defining the route provided by a navigation system, the decision maker determines the timing of discrete mode transitions. Also, if the next mode becomes impossible to reach ever in the future, the decision maker determines the transition to a backup route. The underlying motion planner generates a trajectory that reaches the goal set of the next discrete mode. The key idea of the design is that the decision maker keeps the EV state inside the backward reachable sets of the subsequent modes' goal sets so that there always exists at least one input sequence that makes the EV reach all the modes at some time. To efficiently compute the sets, the decision maker uses a simplified vehicle dynamical model, leaving the motion planner to care for a more accurate and realistic behavior of the EV. The resulting discrepancy is modeled as a bounded set representing additive model errors, and handled by designing the decision maker to be robust to the disturbances in such set. We validated the decision maker integrated with the motion planner and vehicle controller in simulations and experiments using small-scale car-like robots. The results show that the decision maker enables the EV to travel in city scenarios, which include lane following, following a slower vehicle, overtaking, stopping, waiting and crossing an intersection, and queuing in traffic.

Appendix A
Proof of Theorem 1

In this section, we prove that Algorithm 1 solves Problem 1. The proof is based on the following ideas:

1) if a safe reference state sequence exists, there exists a state sequence of the motion planning model (7), called *a safe state sequence*, that reaches the goal set without collisions;
2) if there exists a safe state sequence at some time instant, there exists a safe state sequence at any future time instant.

The first idea connects the existence of a safe reference state sequence with that of an input sequence of the motion planning model (7) that satisfies liveness and safety. The second idea ensures the existence of a safe state sequence at all executions of Algorithm 1, thereby proving the liveness.

We first formalize the first idea, by defining a safe state sequence and relating it to the existence of a safe reference state sequence. Then, we use the second idea in the main proof of Theorem 1.

Definition 5. A *safe state sequence* $\mathbf{x}_{\text{safe}}(q_{i+1}, x_0)$ is a finite sequence of states $\mathbf{x}_{0:N}$ that satisfies

(i) $\Psi(x_k) \notin \mathcal{S}_k$ for all $k \in \{0, \ldots, N\}$;
(ii) $\Psi(x_{N_{\text{reach}}}) \in \mathcal{G}^*(q_{i+1}, q_{i+2})$ for some $N_{\text{reach}} \in \{0, \ldots, N\}$;
(iii) If $q_{i+1} = S$, then $\Psi(x_k) \in \mathcal{G}^*(S, \cdot)$ for all $k \in \{N_{\text{reach}}, \ldots, N\}$. If $q_{i+1} = LF$, then $h(\hat{x}_k^{OV}) - h(\Psi(x_k)) \notin \bigcup_{j=0}^{\infty} \mathcal{C}^j(B)$ for $k \in \{0, \ldots, N\}$.

Here, $\hat{x}_k^{OV}$ denotes the state of the preceding OV at time step k. The following lemma states the relation between the safe state sequences $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$ and $\mathbf{x}_{\text{safe}}(q_{i+1}, x_0)$.

Lemma 1. *Under Assumption 1, if there exists $\hat{\mathbf{x}}_{safe}(q_{i+1}, x_0)$ according to Definition 4, then there exists $\mathbf{x}_{safe}(q_{i+1}, x_0)$ according to Definition 5.*

Proof. Given $\hat{\mathbf{x}}_{0:N}$ that satisfies Definition 4, we construct $\mathbf{x}_{0:N}$ that satisfies Definition 5.

First, we show by induction on k that there exists a sequence of inputs $\mathbf{u}_{0:N-1}$ such that $u_k \in U$ and

$$\Psi(x_k(\mathbf{u}, x_0)) - \hat{x}_k \in W, \ \forall k \in \{0, 1, \ldots, N\}. \tag{17}$$

Due to Condition (i) of Definition 4, $\Psi(x_0) - \hat{x}_0 \in W$. Suppose $\Psi(x_k(\mathbf{u}, x_0)) - \hat{x}_k \in W$. By Assumption 1, for any $\hat{u} \in \hat{U}$, there exists $u \in U$ such that $\Psi(f(x_k(\mathbf{u}, x_0), u)) - \hat{f}(\hat{x}_k, \hat{u}) \in W$. Thus, there exists an input sequence $\mathbf{u}$ satisfying (17).

Next, we show that the sequence of $x_k(\mathbf{u}, x_0)$ for $k = 0, \ldots, N$ satisfies the conditions in Definition 5. Since $\hat{x}_k \notin \mathcal{S}_k \oplus W$, for all $w \in W$ we have $\hat{x}_k - w \notin \mathcal{S}_k$. Thus, $\Psi(x_k) \notin \mathcal{S}_k$ by the symmetry of W (Condition (i) of Definition 5). Regarding Condition (ii) of Definition 5, the condition $\hat{x}_{N_{\text{reach}}} \in \hat{\mathcal{G}}(q_{i+1}, q_{i+2})$ is equivalent to $\hat{x}_{N_{\text{reach}}} + w \in \mathcal{G}^*(q_{i+1}, q_{i+2})$ for all $w \in W$. Since $\Psi(x_{N_{\text{reach}}}) = \hat{x}_{N_{\text{reach}}} + w$ for some $w \in W$, we have $\Psi(x_{N_{\text{reach}}}) \in \mathcal{G}^*(q_{i+1}, q_{i+2})$. Similarly, $\hat{x}_N \in \hat{\mathcal{G}}(S, \cdot)$ implies $x_N \in \mathcal{G}^*(S, \cdot)$, and $e_N \in \hat{\mathcal{I}}$ implies that $h(\hat{x}_N^{OV}) - h(\Psi(x_N)) \in (\bigcup_{j=0} \mathcal{C}^j(B))^c$ because $h(\hat{x}_N) - h(\Psi(x_N)) \in W_E$. □

We prove one more lemma before delving into the proof of Theorem 1. Recall that the complement of the capture set, $\left(\bigcup_k \mathcal{C}^k(B)\right)^c$, is control invariant. The following lemma states that its tightened set is also control invariant.

Lemma 2. *If $e_N \in \hat{\mathcal{I}}$, there exists a longitudinal input $\hat{u}_v \in \hat{U}_v$ that satisfies $g_e(e_N, \hat{u}_v, \hat{u}_v^{OV}) \in \hat{\mathcal{I}}$ for all $\hat{u}_v^{OV} \in \hat{U}_v^{OV}$.*

Proof. The worst case action that the preceding OV can take is $\hat{u}_{v,\min}^{OV}$. For this case, $\hat{u}_{v,\min}$ keeps the error e_N inside the complement of the capture set. Moreover, since W_E has a box shape, the boundary of the tightened set $\hat{\mathcal{I}}$ maintains the same shape as the boundary of $\left(\bigcup_k \mathcal{C}^k(B)\right)^c$, shrunk by the range of W_E, see Fig. 7b. Therefore, if $e_N \in \hat{\mathcal{I}}$, then $g_e(e_N, \hat{u}_{v,\min}, \hat{u}_v^{OV}) \in \hat{\mathcal{I}}$ for all $\hat{u}_v^{OV} \in \hat{U}_v^{OV}$. □

Now, we are ready to prove Theorem 1.

Proof of Theorem 1. Due to the possible outputs of $\pi_\varepsilon(q_i, x_0, \mathcal{M})$, for liveness we need to prove that there always exists one among $\mathbf{x}_{\text{safe}}(q_{i+1}, x_0), \mathbf{x}_{\text{safe}}(q'_{i+1}, x_0)$, and $\mathbf{x}_{\text{safe}}(q_i, x_0)$, and that is not always $\mathbf{x}_{\text{safe}}(q_i, x_0)$ to avoid the system to be stuck in q_i forever.

Algorithm 1 returns $\pi_\varepsilon(q_i, x_0, \mathcal{M}) = \varepsilon_{i+1}$ or ε'_{i+1} if there exists $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$ or $\hat{\mathbf{x}}_{\text{safe}}(q'_{i+1}, x_0)$, respectively. This implies, by Lemma 1, the existence of $\mathbf{x}_{\text{safe}}(q_{i+1}, x_0)$ or $\mathbf{x}_{\text{safe}}(q'_{i+1}, x_0)$. Thus, in the rest of the proof, we focus on the case when Algorithm 1 returns $\pi_\varepsilon(q_i, x_0, \mathcal{M}) = \varepsilon_0$.

When $\Psi(x_0) \notin \mathcal{G}^*(q_i, q_{i+1})$, the mode has changed to q_i because there existed $\hat{\mathbf{x}}_{\text{safe}}(q_i, x_{-k})$ (and thus $\mathbf{x}_{\text{safe}}(q_i, x_{-k})$) at some previous time step $-k$. For each time step between $-k+1$ and -1, the algorithm returns ε_0 because the state is outside $\mathcal{G}^*(q_i, q_{i+1})$. Suppose there was a safe sequence $\mathbf{x}_{\text{safe}}(q_i, x_{-1})$ at the previous time step. Then, at the current time ($k = 0$), there exists a safe sequence $\mathbf{x}_{\text{safe}}(q_i, x_0)$ because it can take the last $N-1$ states in $\mathbf{x}_{\text{safe}}(q_i, x_{-1})$ and concatenate a last state x_N that satisfies $\Psi(x_N) \notin \mathcal{S}_N$ and $h(\hat{x}_N^{OV}) - h(\Psi(x_N)) \in \left(\bigcup \mathcal{C}^k(B)\right)^c$. Such x_N exists because the OVs do not seek collisions, and $\hat{\mathcal{I}}$ is control invariant by Lemma 2. More precisely, for the last state x_{N-1} of $\mathbf{x}_{\text{safe}}(q_i, x_{-1})$, we can find $\hat{x}_{N-1}$ within $x_{N-1} \oplus W$ that satisfies $\hat{x}_{N-1} \in \hat{\mathcal{I}}$. Because $\hat{\mathcal{I}}$ is control invariant, there exists $\hat{x}_N \in \hat{\mathcal{I}}$ and hence there exists $x_N \in \mathcal{I}$.

Similarly, when $\Psi(x_0) \in \mathcal{G}^*(q_i, q_{i+1})$, $\mathbf{x}_{\text{safe}}(q_i, x_0)$ exists by induction on time step k. Suppose that in the previous step ($k = -1$), there existed a safe sequence $\mathbf{x}_{\text{safe}}(q_i, x_{-1})$. Again, $\mathbf{x}_{\text{safe}}(q_i, x_0)$ can take the last $N-1$ states in the sequence $\mathbf{x}_{\text{safe}}(q_i, x_{-1})$ and concatenate a last state x_N that satisfies $\Psi(x_N) \notin \mathcal{S}_N$ and $h(\hat{x}_N^{OV}) - h(\Psi(x_N)) \in \left(\bigcup \mathcal{C}^k(B)\right)^c$.

Algorithm 1 does not return ε_0 indefinitely because

- when the algorithm returns ε_0, it will reach the condition for the transition to the next mode at some future time step, unless this becomes unreachable;
- in case the next mode becomes or has become unreachable, a fallback state sequence $\mathbf{x}_{\text{safe}}(q'_{i+1}, x_0)$ will be eventually feasible at some future time step.

The first is due to the EV state not exiting the backward reachable set at the next time step, which is guaranteed by enforcing $\Psi(x_1) \in \mathcal{G}^*(q_i, q_{i+1})$ or $\Psi(x_1) \in \mathcal{G}^*(q_i, q'_{i+1})$ when the algorithm returns ε_0. Because the algorithm keeps the state inside the backward reachable set of the next mode, there exists an input sequence that enables the EV to reach the next mode at some future time, unless this becomes unreachable due to the OVs behavior. The second is true because we can consider a backup sequence that follows the current lane and stops at the next intersection. As the EV is following the current lane, stopping at the end of the same lane within the N-step planning horizon will eventually be feasible.

The safety proof follows by construction, because the decision maker returns a mode transition only when there exists a safe state sequence, and in that case the motion planner can compute a trajectory that satisfies Definition 5. □

Appendix B
Implementation of Conditions in Algorithm 1

In Section V-A, we discussed the computation of $\mathcal{G}^*(q_i, q_{i+1})$ and $\left(\bigcup \mathcal{C}^k(B)\right)^c$. In this section, we briefly discuss the computation of the other conditions in Algorithm 1, namely $\exists \hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$ and $\text{Reach}(\{\hat{x}_0\}) \cap \hat{\mathcal{G}}(q_i, q_{i+1}) = \emptyset$.

The input sequence $\hat{\mathbf{u}}_{0:N-1}^{LF_2,LF_1}$ achieving the sharpest lane change trajectory ($LF_2 \to LF_1$) was defined by (14). For stopping ($LF \to S$), given $k_1, k_2 \in \mathbb{Z}_{\geq 0}$, the input sequence for the sharpest trajectory are $\hat{\mathbf{u}}_{0:N-1}^{LF,S} = (\hat{u}_{v,k}, \hat{u}_{\theta,k})_{k=0}^{N-1}$,

$$\hat{u}_{v,k} = \begin{cases} \hat{u}_{v,\max} & \text{for } 0 \leq k \leq k_1 - 1, \\ 0 & \text{for } k_1 \leq k \leq k_2 - 1, \\ \hat{u}_{v,\min} & \text{for } k_2 \leq k \leq N - 1, \end{cases} \tag{18}$$
$$\hat{u}_{\theta,k} = \hat{u}_{\text{lane-following},k} \quad \text{for all } k$$

and $\hat{u}_{\text{lane-following},k}$ is the orientation rate for following the lane.

1) Existence of $\hat{\mathbf{x}}_{safe}(q_{i+1}, x_0)$: We seek a state sequence that satisfies Definition 4. For stopping ($LF \to S$), we apply the $\hat{\mathbf{u}}_{0:N-1}^{LF,S}$ and find $k_1, k_2 \in \mathbb{Z}_{\geq 0}$ in (18) that make $\hat{x}_{N_{\text{reach}}} \in \hat{\mathcal{G}}(S, \cdot)$ for $N_{\text{reach}} \leq N$. For lane changing, we apply the extreme input sequence $\hat{\mathbf{u}}_{0:N-1}^{LF_2,LF_1}$ and find integers $k_1, k_2 \in \mathbb{Z}_{\geq 0}$ in (14) that make $\hat{x}_{N_{\text{reach}}} \in \hat{\mathcal{G}}(LF_1, \cdot)$ for some $N_{\text{reach}} \leq N$. For intersection crossing ($S \to LF$), we also consider $k_1, k_2 \in \mathbb{Z}_{\geq 0}$, and apply an extreme input sequence, $\hat{u}_{v,k} = \hat{u}_{v,\max}$ until reaching $v_k = v_{\max}$ and $\hat{u}_{\theta,k}$ taking first 0 until $k = k_1$, then either $\hat{u}_{\theta,\min}$ or $\hat{u}_{\theta,\max}$ depending on turn directions for $k \in [k_1, k_2]$ and then $\hat{u}_{\text{lane-following},k}$ for $k > k_2$. Again, we find integers k_1, k_2 that make $\hat{x}_{N_{\text{reach}}} \in \hat{\mathcal{G}}(LF, \cdot)$ for some $N_{\text{reach}} \leq N$.

If the resulting state trajectory satisfies $\hat{x}_k \notin \mathcal{S}_k \oplus W$ for all k and $\hat{x}_N \in \hat{\mathcal{G}}(S, \cdot)$ or $e_k \in \hat{\mathcal{I}}$ for all k, then $\hat{\mathbf{x}}_{\text{safe}}(q_{i+1}, x_0)$ exists. These checks are explained next.

2) $\hat{x}_k \notin \mathcal{S}_k \oplus W$ and $\hat{x}_k \in \hat{\mathcal{G}}(q_i, q_{i+1})$: Let $\hat{x}_k = (p_{x,k}, p_{y,k}, v_k, \theta_k)$. To check whether $\hat{x}_k \in \hat{\mathcal{G}}(q_i, q_{i+1})$, suppose that $\mathcal{G}(q_i)$ is the goal set allowing position margins Δp_x and Δp_y around the x and y positions, respectively, and angle margin $\Delta\theta$ around the path direction angle. That is, $\hat{x}_k = (p_{x,k}, p_{y,k}, v_k, \theta_k) \in \mathcal{G}(q_i)$ if $p_{x,k} \in p_x^* + [-\Delta p_x, \Delta p_x], p_{y,k} \in p_y^* + [-\Delta p_y, \Delta p_y], v_k \in [0, v_{\max}]$, and $\theta_k \in \theta^* + [-\Delta\theta, \Delta\theta]$ where (p_x^*, p_y^*) is a point on the centerline and θ^* is the path direction at the point. We compute the positions $(p_{x,k}^*, p_{y,k}^*)$ on the centerline such that $\hat{x}_k$ satisfies $p_{x,k} \in p_{x,k}^* + [-\Delta p_x + w_{x,\max}, \Delta p_x + w_{x,\min}]$ and $p_{y,k} \in p_{y,k}^* + [-\Delta p_y + w_{y,\max}, \Delta p_y + w_{y,\min}]$. At these positions $(p_{x,k}^*, p_{y,k}^*)$, suppose the angle of the path direction is θ_k^*, and the lower and upper bounds of the velocity given by the backward reachable sets are $v_{l,k}^*$ and $v_{h,k}^*$, see Fig. 6. If there is a position $(p_{x,k}^*, p_{y,k}^*)$ such that $\theta_k \in \theta_k^* + [-\Delta\theta + w_{\theta,\max}, \Delta\theta + w_{\theta,\min}]$ and $v_k \in [v_{l,k}^* + w_{v,\max}, v_{h,k}^* + w_{v,\min}]$, then the state $\hat{x}_k$ is inside the set $\hat{\mathcal{G}}(q_i, q_{i+1})$, where we compute the Pontryagin set difference by shifting the set boundary by the disturbance bounds.

To check whether $\hat{x}_k \notin \mathcal{S}_k \oplus W$, we discretize the boundary of $\mathcal{S}_k$ and express its position as $[\bar{p}_{x,j}, \underline{p}_{x,j}] \times [\bar{p}_{y,j}, \underline{p}_{y,j}]$ for discretized point j. Then, we can determine the truth of $\hat{x}_k \notin \mathcal{S}_k \oplus W$ by checking that $(p_{x,k}, p_{y,k}) \notin [\bar{p}_{x,j} + w_{x,\min}, \underline{p}_{x,j} + w_{x,\max}] \times [\bar{p}_{y,j} + w_{y,\min}, \underline{p}_{y,j} + w_{y,\max}]$ for all j. As discussed in Section IV, this condition is effectively used only when OVs are changing lanes to avoid collisions with the corners of such OVs. The collision avoidance in a single lane is implied by the capture set condition $e_k \notin \hat{\mathcal{I}}$ and the collision avoidance at intersections is implied by uniqueness in intersection occupancy.

3) $\text{Reach}(\{\hat{x}_0\}) \cap \hat{\mathcal{G}}(q_i, q_{i+1}) = \emptyset$: The condition is checked in Algorithm 1 when $\Psi(x_0) \in \mathcal{G}^*(q_i, q_{i+1})$, for detecting that the state will exit $\hat{\mathcal{G}}(q_i, q_{i+1})$ at the next time step, regardless of the input. Since the boundary of $\hat{\mathcal{G}}(q_i, q_{i+1})$ is determined by $\hat{u}_{v,\min}$, the condition is equivalent to

$$\hat{f}(\hat{x}_0, (\hat{u}_{v,\min}, \hat{u}_{\text{lane-following}})) \notin \hat{\mathcal{G}}(q_i, q_{i+1}). \tag{19}$$

In practice, for computational reasons we determine that $\text{Reach}(\{\hat{x}_0\}) \cap \hat{\mathcal{G}}(q_i, q_{i+1}) = \emptyset$ when (19) holds for all the vertexes of $\hat{x}_0 \oplus W$. Since (19) is not convex as w appears nonlinearly in the unicycle model (4), this is a slight approximation that increases the conservativeness, but retains the property that the motion planner will never be asked to generate a trajectory for an infeasible goal. For our choice of models (3), (7), and the resulting W, the loss due to such an approximation is small.

References

[1] K. Berntorp, T. Hoang, and S. Di Cairano, "Motion planning of autonomous road vehicles by particle filtering," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 2, pp. 197–210, Jun. 2019.

[2] J. Hardy and M. Campbell, "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles," *IEEE Trans. Robotics*, vol. 29, no. 4, pp. 913–929, Aug. 2013.

[3] T. Gu, J. M. Dolan, and J. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Systems*, 2016, pp. 5474–5480.

[4] K. Esterle, P. Hart, J. Bernhard, and A. Knoll, "Spatiotemporal motion planning with combinatorial reasoning for autonomous driving," in *Proc. Int. Conf. Intell. Transp. Systems*, 2018, pp. 1053–1060.

[5] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[6] H. Ahn, K. Berntorp, and S. Di Cairano, "Reachability-based decision making for city driving," in *Proc. Amer. Control Conf.*, 2018, pp. 3203–3208.

[7] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

[8] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

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

[10] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transp. Systems*, vol. 19, no. 6, pp. 1855–1866, Jun. 2018.

[11] M. Bahram, A. Lawitzky, J. Friedrichs, M. Aeberhard, and D. Wollherr, "A game-theoretic approach to replanning-aware interactive scene prediction and planning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3981–3992, Jun. 2016.

[12] 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 Systems Tech.*, vol. 26, no. 5, pp. 1782–1797, Sep. 2018.

[13] 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," *Autonomous Robots*, vol. 41, no. 6, pp. 1367–1382, Aug. 2017.

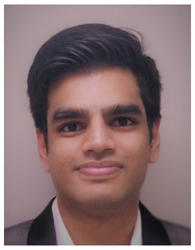
[14] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller, "Automated driving in uncertain environments: planning with interaction and uncertain maneuver prediction," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 1, pp. 5–17, Mar. 2018.

[15] S. Di Cairano, U. Kalabić, and K. Berntorp, "Vehicle tracking control on piecewise-clothoidal trajectories by mpc with guaranteed error bounds," in *Proc. IEEE Conf. Decision and Control*, Dec. 2016, pp. 709–714.

[16] S. Lefèvre, D. Vasquez, and C. Laugier, “A survey on motion prediction and risk assessment for intelligent vehicles,” *ROBOMECH journal*, vol. 1, no. 1, Jul. 2014.
[17] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, “Cooperative collision avoidance at intersections: Algorithms and experiments,” *IEEE Trans. Intell. Transp. Systems*, vol. 14, no. 3, pp. 1162–1175, Mar. 2013.
[18] S. Di Cairano and I. V. Kolmanovsky, “Real-time optimization and model predictive control for aerospace and automotive applications,” in *Proc. Amer. Control Conf.*, 2018, pp. 2392–2409.
[19] D. Hoehener, G. Huang, and D. Del Vecchio, “Design of a lane departure driver-assist system under safety specifications,” in *Proc. IEEE Conference Decision and Control*, Dec. 2016, pp. 2468–2474.
[20] K. Berntorp, T. Hoang, R. Quirynen, and S. Di Cairano, “Control architecture design for autonomous vehicles,” in *Proc. IEEE Conf. Control Tech. and Applications*, 2018, pp. 404–411.
[21] Cogniteam, “The hamster,” https://www.hamster-robot.com/, 2019, accessed on 2020/04/08.
[22] Optitrack, “Prime 13,” http://optitrack.com/products/prime-13/, 2019, accessed on 2020/04/08.
[23] R. Quirynen, A. Knyazev, and S. Di Cairano, “Block structured preconditioning within an active-set method for real-time optimal control,” in *Proc. Europ. Control Conference*, 2018, pp. 1154–1159.
[24] S. Di Cairano, H. Ahn, P. Inani, A. J. Ram, and K. Berntorp, “Decision making system experiments on Hamster,” https://www.youtube.com/watch?v=uLOsCZ4s03U, accessed on 2020/08/28.

Heejin Ahn received the S.M. and Ph.D. degrees in Mechanical Engineering in 2014 and 2018, respectively, from the Massachusetts Institute of Technology (MIT), MA, USA. During 2018-2019, she was a visiting research scientist at Mitsubishi Electric Research Laboratories, MA, USA. She is currently a postdoctoral research fellow at the University of British Columbia, BC, Canada. Her research interests include the verification and control of safety-critical dynamic systems with applications to transportation systems.

Karl Berntorp (SM’20) received the M.Sc. degree in Engineering Physics in 2009 and the Ph.D. degree in Automatic Control in 2014, from Lund University, Lund, Sweden. In 2008 he was a visiting researcher at Daimler AG in Sindelfingen, Germany. Since 2014 he has worked with Mitsubishi Electric Research Laboratories in Cambridge, MA. His research is on statistical signal processing, Bayesian inference, sensor fusion, and optimization-based control, with applications to automotive, transportation, navigation, and communication systems. His work includes design and implementation of estimation, constrained control, motion-planning, and learning algorithms. Dr. Berntorp is the author of more than 70 papers in journals and conferences and has more than 10 granted patents.



Pranav Inani was born in Indore, India, in 1994. He received the M.Eng. degree in Robotics in 2019 from the University of Maryland - College Park, US. He is currently an autonomous software engineer in the Behaviors and Planning team at Torc Robotics. His research interests include motion planning, controls and optimization.

Arjun Jagdish Ram received the MSc. degree in Robotics in 2019 from Worcester Polytechnic Institute, Massachusetts, USA. He has been working with Torc Robotics, Blacksburg, Virginia, USA since 2019 as a Motion Planning and Control Automotive Software Engineer. His research interests lie in the field of numerical control and planning for autonomous vehicles.

Stefano Di Cairano (SM’15) received the Master’s (Laurea) and the Ph.D. in Information Engineering in 2004 and 2008, respectively, from the University Siena, Italy. During 2008-2011, he was with Powertrain Control R&A, Ford Research and Advanced Engineering, Dearborn, MI. Since 2011, he is with Mitsubishi Electric Research Laboratories, Cambridge, MA, where he is currently a Senior Team Leader and a Distinguished Researcher. His research focuses on control strategies for complex mechatronic systems, in automotive, transportation systems, and aerospace. His interests include model predictive control, constrained control, path planning, hybrid systems, optimization. He has authored/coauthored more than 150 peer-reviewed papers and 40 patents.

Dr. Di Cairano was the Chair of the IEEE CSS Tech. Comm. on Automotive Controls, of the IEEE CSS Stand. Comm. on Standards, and was an Associate Editor of the IEEE Trans. Control Systems Tech.. He is the inaugural Chair of the IEEE Technology Conf. Editorial Board.