MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Cooperating Modular Goal Selection and Motion Planning for Autonomous Driving

Ahn, Heejin; Berntorp, Karl; Di Cairano, Stefano

TR2020-168 December 16, 2020

Abstract

We present a decision making approach for autonomous driving that concurrently determines the driving mode and the motion plan that achieves the driving mode goal. To do this, we develop two cooperating modules: a mode activator and a motion planner. Based on the current mode in a non-deterministic automaton, the mode activator determines all the feasible next modes, i.e., the modes for which there exists a trajectory that reaches the associated goal. Then, the motion planner generates trajectories achieving the goals of such feasible modes, selects the next mode and trajectory that result in the best performance, and updates the current mode in the automaton. To determine the feasibility, the mode activator uses robust forward and backward reachability that accounts for the discrepancy between the simplified model used in the reachability computation and the more precise model used by the motion planner. We prove that, under normal operation, the mode activator always returns a nonempty set of feasible modes, so that the decision making algorithm is recursively feasible. We validate the algorithm in simulations and experiments using car-like laboratory-scale robots.

IEEE Conference on Decision and Control (CDC)

© 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Cooperating Modular Goal Selection and Motion Planning for Autonomous Driving

Heejin Ahn[1], Karl Berntorp[2], and Stefano Di Cairano[2]

Abstract— **We present a decision making approach for autonomous driving that concurrently determines the driving mode and the motion plan that achieves the driving mode goal. To do this, we develop two cooperating modules: a mode activator and a motion planner. Based on the current mode in a non-deterministic automaton, the mode activator determines all the feasible next modes, i.e., the modes for which there exists a trajectory that reaches the associated goal. Then, the motion planner generates trajectories achieving the goals of such feasible modes, selects the next mode and trajectory that result in the best performance, and updates the current mode in the automaton. To determine the feasibility, the mode activator uses robust forward and backward reachability that accounts for the discrepancy between the simplified model used in the reachability computation and the more precise model used by the motion planner. We prove that, under normal operation, the mode activator always returns a nonempty set of feasible modes, so that the decision making algorithm is recursively feasible. We validate the algorithm in simulations and experiments using car-like laboratory-scale robots.**

I. INTRODUCTION

The guidance system of an autonomous vehicle must determine the discrete driving mode such as lane changing, lane following, intersection crossing, and the continuous trajectory to achieve the goal of such mode [1]. Thus, the overall guidance problem requires mixed continuous-discrete decision-making algorithms that are notoriously challenging to implement in real-time in platforms with limited computational capabilities [2]. As a result, the mode determination and trajectory planning problems are often decoupled into separated modules, see Fig. 1a.

In this paper, we propose an architecture where the mode determination is performed in coordination with the trajectory generation. Specifically, the decision making and motion planning algorithm, which is from now on referred to simply as *decision making* for shortness, is based on the interaction between two cooperating modules: a *mode activator* and a *motion planner* (see Fig. 1b). Based on the currently active mode and a non-deterministic automaton providing the admissible next modes, the mode activator determines all *feasible modes*, i.e., the modes for which the motion planner can produce trajectories that achieve the corresponding goals, while safely behaving in traffic. For each feasible mode, the motion planner generates trajectories, and then selects the mode and trajectory that yield the best performance, providing the former to the mode activator as next mode.

[1]Heejin Ahn worked on the project when she was with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA. heejin.ahn@alum.mit.edu

[2]Karl Berntorp and Stefano Di Cairano are with MERL. karl.o.berntorp@ieee.org; dicairano@ieee.org

The proposed structure retains the reduced complexity and flexibility of a modular architecture, where different models and algorithms are used in different modules. For instance, unnecessary computations for a motion attempting to achieve an infeasible goal are avoided by simple calculations in the mode activator based on a simplified vehicle model. However, it can also achieve high performance by selecting the best mode, based on the corresponding trajectory computed with a higher precision model.

In our previous work [3], we developed a reachability-based decision making module for the architecture in Fig. 1a. The decision making approach in [3] requires the sequence of modes to be generated *a priori*, and selects the mode based only on its feasibility, without accounting for the performance achieved by the motion planner in such mode.

Other decision-making approaches in autonomous driving are often based on rules or forward simulations. Rule-based approaches were used in the DARPA Urban Challenge [1], where for example, lane changing is allowed if there is a free space on the next lane that is larger than some threshold. In forward simulation approaches, [4], [5] compute multiple trajectories, classify them depending on different modes, and choose one mode associated with an optimal trajectory. In [6], [7] other vehicles' (OVs) reactions are considered in the ego vehicle's (EV) planning process based on the forward simulation of all vehicles to predict outcomes of each decision. For other related works, readers are referred to [8]. The main drawbacks of rule-based approaches are the need for large amount of hand-tuning and the lack of guarantees, while for many forward simulation approaches the main issue is that the finite simulation horizon may not be guaranteed to retain feasibility at future time steps.

The mode activator exploits well researched reachability tools [9]–[11] to determine whether a mode is feasible. We label as infeasible the modes for which the current state is outside the backward reachable set of the associated goal, and those for which achieving the goal necessarily causes a collision with the OVs. We handle the discrepancy between the simplified model of the mode activator and the more accurate model of the motion planner by shrinking or inflating the reachable sets. Moreover, by imposing an invariant property in the forward trajectories, we guarantee that there is always at least one feasible mode, which results in the algorithm to be recursively feasible. Reachability tools have been exploited in autonomous driving systems [12], [13], but the objective of the mode activator is different in that it determines the existence of trajectories that safely reach each mode goal prior to actually computing the trajectory itself.

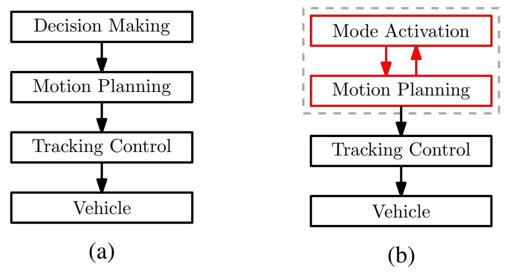

Fig. 1: Modular architectures for autonomous driving: (a) Sequential determination of driving mode and trajectory, e.g., [3]. (b) Concurrent determination of driving mode and trajectory by cooperating mode activation and motion planning (dashed box).

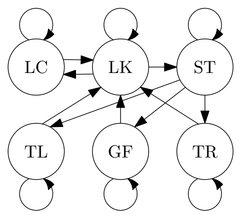

Fig. 2: Mode transition model as a (non-deterministic) automaton. The EV modes are lane keeping (LK), lane changing (LC), stopping (ST) at stop lines, turn left (TL), go forwards (GF), and turn right (TR) through intersections.

The rest of the paper is structured as follows. In Section II, we explain the models used in the mode activator and the model used in the motion planner, along with the dynamical model of OVs. In Section III, we present the design of the decision-making algorithm and prove that there is always at least one feasible mode. We provide the validation results via computer simulations and laboratory experiments in Section IV, and conclude the paper in Section V.

Notation: In the paper, $\mathbb{Z}, \mathbb{R}$ and $\mathbb{R}^n$ denote the sets of integers, real numbers, and real vectors of dimension n, respectively. Also, $\mathbb{Z}_+ = \{z \in \mathbb{Z} : z > 0\}$ and $\mathbb{Z}_{a:b} = \{z \in \mathbb{Z} : a \leq z \leq b\}$. With a discrete-time signal x_k, we use $\mathbf{x}_{a:b}$ to denote a sequence $(x_a, \ldots, x_b)$ with $a, b \in \mathbb{Z}$, or simply write $\mathbf{x}$ when it is not necessary to specify the interval. According to a dynamical model, $x_k(\mathbf{u}_{0:k-1}, x_0)$ denotes the state reached at time step k starting from the state x_0 with the input sequence $\mathbf{u}_{0:k-1} = (u_0, \ldots, u_{k-1})$. Given two sets A and B, the Minkowski set addition is $A \oplus B = \{a + b : a \in A, b \in B\}$ and the Pontryagin set difference is $A \ominus B = \{a : a + b \in A, \forall b \in B\}$.

II. Models in the Decision Making System

In this section, we introduce the models used in the mode activator and motion planner, and the dynamical model of OVs used to predict the behavior over a finite time horizon.

A. Models used in the Mode Activator

In the mode activator, for mode transition model we consider the automaton in Fig. 2. The discrete mode $q \in Q$ corresponds to a request for the EV to perform different tasks, such as lane changing (LC), lane keeping (LK), stopping (ST), and at intersections, turning left (TL), going forwards (GF), or turning right (TR). Here, $Q = \{\text{LC, LK, ST, TL, GF, TR}\}$ although other modes, such as U-turn, creeping in an intersection and merging, can also be added. Based on our target application, and for the sake of simplifying the description in this paper, we make the following two assumptions.

Assumption 1: From each mode in the automaton that admits a feasible transition, there is a way for the vehicle to eventually reach its target destination. ■

Assumption 2: The intersections are all-way stop, i.e., each vehicle must stop before crossing the intersection. ■

Assumption 1 allows to focus on selecting a mode based only on a medium term objective, with the guarantee that the overall destination will be achieved. This is normally guaranteed by a routing module like a car navigation system that provides directions that translate into desired modes. The information about desired modes will be included in the cost function to determine the next mode. Assumption 2 is mainly introduced to keep the transition model simple for the sake of exposition, since then transitioning to intersection crossing modes (TL, GF, TR) is only possible from the stop (ST) mode. Other priority rules or traffic lights can be incorporated with some modifications to the transition model.

Remark 1: We call the automaton in Fig. 2 non-deterministic because multiple transitions may be allowed from a state at any time. Given q, the allowed modes to transition to are denoted by $Q(q)$, e.g., $Q(\text{LK}) = \{\text{LC, LK, ST}\}$, $Q(\text{ST}) = \{\text{ST, TL, GF, TR}\}$. While in rule-based decision making the transition to be executed is uniquely determined by guards within the automaton, here the transition to be executed is ultimately determined outside of the automaton by the motion planner. ■

The mode activator uses the discrete-time vehicle model

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

where $\hat{x}_k \in \hat{X} \subset \mathbb{R}^{\hat{n}_x}$ is the EV state, $\hat{u}_k \in \hat{U} \subset \mathbb{R}^{\hat{n}_u}$ is the EV input, and both $\hat{X}$ and $\hat{U}$ are bounded. Model (1) is a simplified vehicle motion model to enable rapid computations within the mode activator. In this paper, we use the unicycle model where $\hat{x} = (p_x, p_y, v, \theta)^\intercal$ and $\hat{u} = (\hat{u}_v, \hat{u}_\theta)^\intercal$. Here, (p_x, p_y) are the x and y positions in the global frame, v is the longitudinal velocity, θ is the heading angle, $\hat{u}_v$ is the longitudinal acceleration and $\hat{u}_\theta$ is the heading angular rate.

B. Model used in the Motion Planner

The motion planner uses the following vehicle model

$$x_{k+1} = f(x_k, u_k), \tag{2}$$

where $x_k \in X \subseteq \mathbb{R}^{n_x}$ is the EV state, $u_k \in U \subseteq \mathbb{R}^{n_u}$ is the input, and both X and U are bounded. Usually, (2) is a higher-fidelity model with respect to (1), where $\hat{n}_x \leq n_x$, $\hat{n}_u = n_u$. This is due to the different uses of (1) and (2), as (2) has to be sufficiently accurate to generate drivable trajectories, while (1) is used only for determining the existence of drivable trajectories. For (2), we use the kinematic bicycle model where $x = (p_x, p_y, v, \theta, \delta)^\intercal$ and $u = (u_v, u_\delta)^\intercal$. The additional state variable δ, compared with the unicycle model, is the steering angle of the front wheel, and u_δ is the steering rate.

In order to use (1) and obtain valid existence results for (2), we model the discrepancy between (1) and (2) as an additive disturbance $w = \Psi(x) - \hat{x} \in W$, where $\Psi : X \to \hat{X}$ is a given function (e.g., the projection onto $\hat{X}$). Specifically, we make the following assumption.

Assumption 3: There exist a bounded set $W \subset \mathbb{R}^{n_x}$ and a 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 $u \in U$ satisfying

$$\Psi(f(x, u)) - \hat{f}(\hat{x}, \hat{u}) \in W. \qquad (3)$$

■

As a result of Assumption 3, given a reference trajectory $\hat{\mathbf{x}}$, there exists a trajectory $\mathbf{x}$ such that $\Psi(\mathbf{x})$ lies in a tube surrounding $\hat{\mathbf{x}}$. While Ψ is generally based on model structure and design experience, the disturbance set W can be constructed from the models or by extensive numerical simulation.

C. Model of OVs

In the mode activator, to predict the OV behavior over the planning horizon, we use a dynamical model similar to (1),

$$\hat{x}^{OV}_{k+1} = \hat{f}(\hat{x}^{OV}_k, \hat{u}^{OV}_k) \qquad (4)$$

where $\hat{x}^{OV} \in \hat{X}^{OV} \subset \mathbb{R}^{\hat{n}_x}$ and $\hat{u}^{OV} \in \hat{U}^{OV} \subset \mathbb{R}^{\hat{n}_u}$ are the OV state and input, respectively, and $\hat{X}^{OV}$, $\hat{U}^{OV}$ are bounded. Centered at the state $\hat{x}^{OV}_{k,o}$ of the o-th OV, the occupancy set is denoted by $\mathcal{O}_{k,o} \subset \mathbb{R}^{\hat{n}_x}$, and the occupancy set at time k is $\mathcal{O}_k = \bigcup_{o=1}^{n_o} \mathcal{O}_{k,o}$, where n_o is the total number of OVs. To avoid collisions, the EV state must not overlap with $\mathcal{O}_k$ at any time step $k \in \mathbb{Z}_{0:N}$.

III. Decision-Making Algorithm

In the proposed cooperating decision-making algorithm, the mode activator identifies all modes for which the EV can achieve the corresponding goal, and provides to the motion planner only those feasible modes. The motion planner generates trajectories for the feasible modes and selects one mode and corresponding trajectory that yield the lowest cost.

The overall decision-making algorithm is presented in Algorithm 1. Given the current state x_0, mode q_0, and the predicted occupancy sets of OVs $\mathcal{O}_{0:N} = (\mathcal{O}_0, \ldots, \mathcal{O}_N)$, the mode activator returns the set of feasible modes $Q_F \subseteq Q(q_0)$, and the motion planner selects the mode $q_{\text{best}} \in Q_F$ resulting in the lowest-cost trajectory $\mathbf{x}_{\text{best}}$.

Algorithm 1 Cooperating Decision Making and Planning

```
• Input: State $x_0$, mode $q_0$, occupancy set $\mathcal{O}_{0:N}$.
• $Q_F = \texttt{ModeActivator}(x_0, q_0, \mathcal{O}_{0:N})$
• $(\mathbf{x}_{\text{best}}, q_{\text{best}}) = \texttt{MotionPlanner}(x_0, Q_F, \mathcal{O}_{0:N})$
• Return Next mode: $q_{\text{best}}$;   EV trajectory: $\mathbf{x}_{\text{best}}$.
```

Next, we detail the algorithms `ModeActivator` and `MotionPlanner`.

A. The Mode Activator

The mode activator computes a set of feasible modes Q_F based on the following principles.

- If the current EV state x_0 has not reached yet the current goal $\mathcal{G}(q_0)$, the mode activator maintains the current mode q_0 by letting $Q_F = \{q_0\}$.
- If the EV state can reach the goal $\mathcal{G}(q)$ within the N-step horizon safely, i.e., without collisions, $q \in Q_F$.

Here, the goal set $\mathcal{G}(q) \subset \hat{X}$ is defined for each mode $q \in Q$. For LC, the goal is to reach the center line of the next lane; for LK, the goal is to remain in the vicinity of the center line of the current lane; for ST, the goal is to reach an area immediately preceding the stop line at zero speed; and for TL, GF, and TR, the goal is to reach the desired lane after the intersection. These allow us to more precisely define feasible modes: a mode $q \in Q$ is said to be *feasible* if there exists a trajectory $\mathbf{x}_{0:N}$ of model (2) such that $\Psi(x_k) \in \mathcal{G}(q)$ for some $k \in \mathbb{Z}_{0:N}$ and $\Psi(x_k) \notin \mathcal{O}_k$ for all $k \in \mathbb{Z}_{0:N}$.

The mode activator checks several conditions to determine a set of feasible modes Q_F: for some $\hat{x}_0$ such that $\Psi(x_0) - \hat{x}_0 \in W$,

(a) $\hat{x}_0$ is outside the set $\tilde{\mathcal{G}}(q_0)$;
(b) $\hat{x}_0$ is within the backward reachable set of $\tilde{\mathcal{G}}(q)$; and
(c) there exists a trajectory that safely reaches $\tilde{\mathcal{G}}(q)$ within N planning steps.

Here, $\tilde{\mathcal{G}}(q)$ is a subset of the goal set $\mathcal{G}(q)$, to ensure robustness as explained later in this section. The mode activator checks whether the above conditions hold "for some $\hat{x}_0$", which means that it uses the most favorable state. This is possible because Assumption 3 ensures that the EV state can be kept in the range of the uncertainty for any initial state chosen within the range of the uncertainty.

Next, we elaborate on each condition and then provide the mode activator algorithm that implements such conditions.

Condition (a): This condition means that the EV has not reached the current goal. In this case, we force the mode activator to return q_0 until the EV state reaches the goal.

Condition (b): With this condition, the mode activator can discard some infeasible mode q before explicitly seeking a trajectory that reaches the set $\tilde{\mathcal{G}}(q)$. The backward reachable set of $\tilde{\mathcal{G}}(q)$ is defined as

$$\begin{aligned}\texttt{BRS}(\tilde{\mathcal{G}}(q)) = \{\hat{x}_0 \in \hat{X} : \exists k \in \mathbb{Z}_+, \exists \hat{\mathbf{u}}_{0:k-1} \in \hat{U}^k,& \\ \hat{x}_k(\hat{\mathbf{u}}_{0:k-1}, \hat{x}_0) \in \tilde{\mathcal{G}}(q)\}.& \qquad (5)\end{aligned}$$

This is the set of states from which there is an admissible input sequence to reach the goal $\tilde{\mathcal{G}}(q)$ at some future time.

In the above conditions, the set $\tilde{\mathcal{G}}(q)$ is defined as

$$\tilde{\mathcal{G}}(q) := \mathcal{G}(q) \cap \texttt{BRS}(\mathcal{G}(\text{ST})),$$

because then $\hat{x}_0 \in \texttt{BRS}(\tilde{\mathcal{G}}(q))$ ensures that the EV will be able to stop at the end of each lane, following the assumption of all-way stops. For example, LC should be feasible when it is possible for the EV to change lane and then stop at the subsequent stop line. If $\hat{x}_0 \notin \texttt{BRS}(\tilde{\mathcal{G}}(q))$, then the EV will not be able to reach the goal sets $\mathcal{G}(q)$ and $\mathcal{G}(\text{ST})$ at any future time, and thus q is infeasible.

Condition (c): The mode q is feasible if there exists a finite trajectory, called *a safe state trajectory*, that reaches the goal set $\mathcal{G}(q)$ without collisions. In addition, we impose an invariant property on such a trajectory to ensure that the set Q_F returned by the mode activator is always nonempty.

In the following, we introduce the concept of a control invariant set [10] and define a safe state trajectory. A control invariant set is the set of states in which there exists an admissible input that keeps the state inside the set.

Definition 1: A set $K \subseteq \hat{X}$ is *control invariant* for (1), if for all $\hat{x} \in K$, there exists $\hat{u} \in \hat{U}$ such that $\hat{f}(\hat{x}, \hat{u}) \in K$. ■
The goal set of stopping, $\mathcal{G}(\text{ST})$, is control invariant because within the set the EV is stationary at a stop line and can remain stationary at any future time.

Given the preceding vehicle state $\hat{x}_0^{OV}$, the *safe set* is

$$\begin{aligned}\mathcal{S}(\hat{x}^{OV}) = \{\hat{x} \in \hat{X} :\ \exists \hat{u} \in \hat{U}, d(\hat{x}, \hat{x}^{OV}) > d_{\min},\\ \hat{f}(\hat{x}, \hat{u}) \in \mathcal{S}(\hat{f}(\hat{x}_k^{OV}, \hat{u}_k^{OV})),\ \forall \hat{u}^{OV} \in \hat{U}^{OV}\}. \quad (6)\end{aligned}$$

Here, $d(\hat{x}, \hat{x}^{OV})$ is the longitudinal distance between $\hat{x}$ and $\hat{x}^{OV}$ along the lane's center line, and $d_{\min}$ is the minimum safety distance. By (6), the safe set is a control invariant set and if the EV state is inside the safe set, there exists an input $\hat{u} \in \hat{U}$ that keeps the longitudinal distance between the EV and the preceding OV greater than $d_{\min}$ regardless of the OV input $\hat{u}^{OV} \in \hat{U}^{OV}$ (even in case of sudden braking of the OV). By maintaining the EV state in the safe set, any rear-end collision between the EV and the preceding OV can be avoided for all future times.

Definition 2: Given the current state x_0, mode q, and occupancy set $\mathcal{O}_{0:N}$, a state sequence $(\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N)$ is called a *safe state sequence* $\hat{\mathbf{x}}_{\text{safe}}(x_0, q)$ if

- $\Psi(x_0) - \hat{x}_0 \in W$;
- $\hat{x}_k \notin \mathcal{O}_k \oplus W$ for all $k \in \mathbb{Z}_{0:N}$;
- $\hat{x}_{N_{\text{reach}}} \in \tilde{\mathcal{G}}(q) \ominus W$ for some $N_{\text{reach}} \in \mathbb{Z}_{0:N}$;
- If $q = \text{ST}$, then $\hat{x}_k \in \mathcal{G}(\text{ST}) \ominus W$ for all $k \in \mathbb{Z}_{N_{\text{reach}}:N}$. Otherwise, $\hat{x}_k \in \mathcal{S}(\hat{x}_k^{OV}) \ominus W$ for all $k \in \mathbb{Z}_{0:N}$. ■

According to Definition 2, starting from $\hat{x}_0$, the state never overlaps with the occupancy set $\mathcal{O}_k$ inflated by the disturbance set W. Also, one of the states reaches $\tilde{\mathcal{G}}(q)$ shrunk by W. Lastly, the safe state sequence enters a control invariant set to guarantee the existence of an input u_N at the subsequent time step.

In defining $\hat{\mathbf{x}}_{\text{safe}}(x_0, q)$, we use sets that are inflated or shrunk by the disturbance set W to account for the discrepancy between models (1) and (2). This ensures that if $\hat{\mathbf{x}}_{\text{safe}}(x_0, q)$ exists for (1), then there exists a sequence $\mathbf{x}_{0:N}$ for (2) that satisfies $\Psi(x_k) \notin \mathcal{O}_k$ for all k, $\Psi(x_{N_{\text{reach}}}) \in \tilde{\mathcal{G}}(q)$, and $\Psi(x_k) \in \mathcal{G}(\text{ST})$ for all $k \geq N_{\text{reach}}$ or $\Psi(x_k) \in \mathcal{S}(\hat{x}_k^{OV})$ for all k. Therefore, we can determine the existence of drivable trajectories by using only the simplified model (1).

Algorithm: In summary, we outline `ModeActivator`. If the the current goal set has not been reached, $\Psi(x_0) \notin \tilde{\mathcal{G}}(q_0)$, (Condition (a)), `ModeActivator` returns the current mode q_0, which is still feasible by the invariance condition. Otherwise, it initializes $Q_F = \emptyset$ and evaluates each $q \in Q(q_0)$. A candidate mode $q \in Q(q_0)$ is infeasible if the EV state is outside the backward reachable set of $\tilde{\mathcal{G}}(q)$ (Condition (b)), or if a safe state sequence $\hat{\mathbf{x}}_{\text{safe}}(x_0, q)$ does not exist (Condition (c)). Additional modes can be removed from Q_F to enforce desired behaviors. For instance, if both LK and ST are in Q_F, LK can be removed to enforce conservative stopping. Similarly, if any among TL, GF, or TR is inside Q_F, ST can be removed to enforce the EV to start crossing immediately. If these are not removed, the decision can be ultimately resolved by the motion planner.

Algorithm 2 `ModeActivator`$(x_0, q_0, \mathcal{O}_{0:N})$

```
• Input: State $x_0$, mode $q_0$, occupancy set $\mathcal{O}_{0:N}$.
• If $\Psi(x_0) \notin \tilde{\mathcal{G}}(q_0)$, then return $Q_F \leftarrow \{q_0\}$.
• Otherwise, initialize $Q_F \leftarrow \emptyset$.
• For all $q \in Q(q_0)$:
    – if $\Psi(x_0) \notin \mathtt{BRS}(\tilde{\mathcal{G}}(q))$, reject $q$.
    – else if $\nexists \hat{\mathbf{x}}_{\text{safe}}(x_0, q)$, reject $q$.
    – otherwise, $Q_F \leftarrow Q_F \cup \{q\}$.
    – (Optional) if $\{\text{ST,LF}\} \subseteq Q_F$, then $Q_F \leftarrow Q_F \backslash \text{LK}$.
    – (Optional) if $\{\text{TL,GF,TR}\} \cap Q_F \neq \emptyset$, then $Q_F \leftarrow Q_F \setminus \text{ST}$.
• Return $Q_F$.
```

Theorem 1: Under Assumptions 1–3, let $Q_{F,k-1} = \mathtt{ModeActivator}(x_{k-1}, q_{k-1}, \mathcal{O}_{k-1:k+N-1}) \neq \emptyset$ at time step $k-1$, and let $q_k \in Q_{F,k-1}$ be the next mode with $\hat{\mathbf{x}}_{\text{safe}}(x_{k-1}, q_k) = (\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N)$. Then, at step k, with $\Psi(x_k) - \hat{x}_1 \in W$,

$$Q_{F,k} = \mathtt{ModeActivator}(x_k, q_k, \mathcal{O}_{k:k+N})$$

is nonempty.

Proof: (Sketch) If $\Psi(x_k) \notin \tilde{\mathcal{G}}(q_k)$, then $Q_F = \{q_k\}$, so it is not empty. If $\Psi(x_k) \in \tilde{\mathcal{G}}(q_k)$, we will prove that either LK or ST is always in the set Q_F except the case that we purposely remove it from the set. Because the goal sets are the center line of targeted lanes, $\Psi(x_k) \in \tilde{\mathcal{G}}(q_k) = \mathcal{G}(q_k) \cap \mathtt{BRS}(\mathcal{G}(\text{ST}))$ means that the EV state is on the center line and has an input to stop at stop lines at some future time. When $q_k = \text{ST}$, $\hat{\mathbf{x}}_{\text{safe}}(x_k, \text{ST})$ exists because $\Psi(x_k) \in \mathcal{G}(\text{ST})$ and $\mathcal{G}(\text{ST})$ is control invariant. For all other q_k, $\hat{\mathbf{x}}_{\text{safe}}(x_k, \text{LK})$ exists because there was a safe state sequence $\hat{\mathbf{x}}_{\text{safe}}(x_{k-1}, q_k)$ at a previous time step that reaches the center line of lane corresponding to q_k at some future time N_{reach} while entering a control invariant set. ■

The exact computation of $\mathtt{BRS}(\tilde{\mathcal{G}}(q))$ and $\hat{\mathbf{x}}_{\text{safe}}(x_0, q)$ may be challenging, especially for automotive computational platforms that have limited capabilities [2]. However, the computations can be approximated while maintaining safety, e.g., by evaluating motion primitives as discussed in [3].

B. The Particle-filter based Motion Planner

Here, we briefly describe the particle-filter based motion planner used to determine a trajectory given multiple feasible modes. For a detailed description, see [14].

The planner relies on *a priori* defined requirements c_k

$$c_k = r^{(q)}(x_k) + \nu_k \tag{7}$$

where $r^{(q)}$ is the known mode-dependent requirement function, and ν_k is the tolerated probabilistic deviation. Some common requirements are tracking the centerlane, maintaining a target velocity, keeping a safe distance from the OV in front [14]. The requirements are mode dependent because their importance may change based on the mode, e.g, in LC it is appropriate to drive far from the centerlane.

In a Bayesian framework, by interpreting u in (2) as process noise and ν in (7) as measurement noise, the motion-planning model can be written as $x_{k+1} \sim p(x_{k+1}|x_k)$, and $c_k \sim p(c_k|x_k, \mathcal{O}_{0:N}, q)$, where x_{k+1} and c_k are regarded as samples and $q \in Q_F$.

The motion planner approximates the density function $p(\mathbf{x}_{0:N}|\mathbf{c}_{0:N}, \mathcal{O}_{0:N}, Q_F)$ by a set of n_p particles $\mathbf{x}^i_{0:N}$,

$$p(\mathbf{x}_{0:N}|\mathbf{c}_{0:N}, \mathcal{O}_{0:N}, Q_F) \approx \sum_{i=1}^{n_p} \omega^i_N \delta(\mathbf{x}_{0:N} - \mathbf{x}^i_{0:N}), \tag{8}$$

where ω^i_N in (8) is the importance weight for the ith particle, $\delta(\cdot)$ is the Dirac delta mass, and $\mathbf{c}_{0:N}$ is the sequence of outputs according to (7). Due to the different modes q, the distribution (8) is generally multimodal. Based on (7), the motion planner extracts $\mathbf{x}_{0:N}$ from the density function that minimizes a compound cost of the trajectory and the mode,

$$J(\mathbf{x}_{0:N}, q) = J_1^{(q)}(\mathbf{x}_{0:N}) + J_2(q) \tag{9}$$

where $J_1^{(q)}$ is a normalization of the mode-dependent trajectory cost which represents the desirability of the computed trajectory and is related to its probability according to (8), e.g., its mean-square estimate, and J_2 is the mode selection cost which represents the desirability of the mode. In the validation in Section IV, $J_1^{(q)}$ in (9) penalizes the inverse of the distance from the preceding OVs, the deviation from centerlane, the deviation from target speed, and the control aggressiveness, among others. The relative weighting in $J_1^{(q)}$ depends on the mode q, and lane changes are penalized by J_2 to avoid them unless they bring significant driving benefits.

Algorithm 3 outlines the `MotionPlanner` algorithm.

Algorithm 3 `MotionPlanner`$(x_0, Q_F, \mathcal{O}_{0:N})$

```
• Input: State $x_0$, set of modes $Q_F$, occupancy set $\mathcal{O}_{0:N}$.
• For $q \in Q_F$:
    – determine (8) and $\mathbf{x}_{0:N}$
    – $(\bar{\mathbf{x}}_{0:N}, \bar{q}) \leftarrow (\bar{\mathbf{x}}_{0:N}, \bar{q}) \cup \{(\mathbf{x}_{0:N}, q)\}$.
• Return lowest-cost safe state trajectory and corresponding mode $(\mathbf{x}_{\text{best}}, q_{\text{best}}) \in (\bar{\mathbf{x}}_{0:N}, \bar{q})$.
```

IV. Validation in Simulation and Experiment

We validate the results of Algorithm 1 in simulations and experiments. We consider an eight-shaped track, see Fig. 3, with the EV and two OVs, where the EV implements the stack in Fig 1b, while the OVs have a fairly simple control

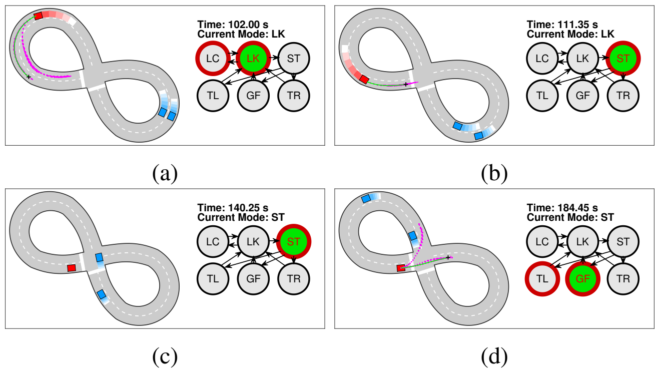

Fig. 3: Simulation results. Left: EV (red box), OVs (blue boxes), motion planner samples (magenta), trajectory (green) / Right: modes and transitions from Fig. 2 (self-loops not shown), feasible modes Q_F (red border), selected mode q_{best} (green fill). The EV (a) maintains the lane as unnecessary lane changes are to be avoided, (b) starts braking, (c) waits for the OVs to cross the intersection, and (d) travels through the intersection.

and drive slower than the EV target speed. The EV always yields to the OVs at intersections, primarily because the OVs have simple logics to handle intersections, although other intersection rules such as first-in, first-out can be easily implemented. In the validation, we do not use a routing module, so the turns at the intersection are chosen only by the quality of the computed trajectory.

A. Simulation

Fig. 3 illustrates the simulation results. In Fig. 3a, the mode activator outputs $Q_F = \{\text{LC, LK}\}$, and then, the motion planner evaluates the two modes and selects LK due to a penalty for non-required lane changes in J_2 in (9). In Fig. 3b, the EV starts to brake, and stops in Fig. 3c while yielding to the OVs. In Fig. 3d, the mode activator allows TL and GF through the intersection, and the motion planner selects GF. Here, TR is not allowed due to the map geometry.

B. Experiment

For experiments, we setup a ROS network with three small-scale car-like Hamster[1] robots, one as EV and two as OVs. The robot poses are estimated by an OptiTrack motion capture system, while we use Hamster on-board encoder and an inertial measurement unit to estimate the velocity, which is shared through ROS. As for the EV tracking control in Fig. 1b, we use a nonlinear model predictive control. More details on the experimental setup are in [15].

A segment of the experimental results is shown in Fig. 4. The EV is the robot with a red flag and represented by the red box. Similarly, the robots with blue flags and blue boxes represent the OVs. In Fig. 4a, the mode activator determines that LK and LC are feasible, and in fact the motion planner is able to generate trajectories (the particles represented by the magenta dots) that correspond to the two modes. The motion planner chooses LK because changing lane will make the EV follow the slow OV. In Fig. 4b, the mode activator determines that LC is not feasible anymore because it is not

[1] `https://www.hamster-robot.com`

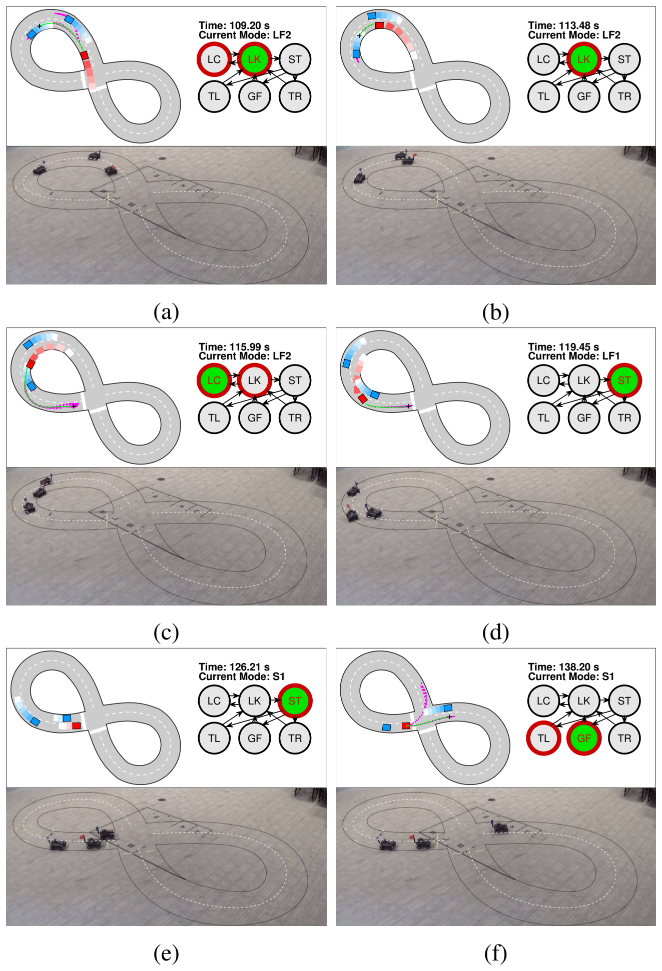

Fig. 4: Experimental results. Top: same coloring as Fig.3. / Bottom: live scene. In (a), while lane changing is possible, it is not desirable due to slow traffic. In (b), lane changing becomes impossible due to the presence of traffic, and in (c), the EV changes lane as it is now feasible again. The EV slows down in (d) and reaches the stop in (e). In (f), the EV starts crossing when the intersection is clear.

possible to change lane without colliding with the OV on the target lane. LC becomes feasible again in Fig. 4c, and the motion planner selects it because now the slow OV ahead causes a large cost for staying in the same lane. Approaching the stop line, the EV begins to slow down in Fig. 4d, and stops in Fig. 4e. In Fig. 4f, the motion planner starts crossing the intersection straight.

In the experiments, we observed that for any mode returned by the mode activator, the motion planner was able to find a feasible trajectory, as expected. The mode activator always returned a nonempty set Q_F as proved in Theorem 1.

In terms of computational load, the decision-making sampling period is $0.85\,\mathrm{s}$, and the worst-case execution time for a Matlab m-code implementation in a desktop with an Intel Core i7 $3.20\,\mathrm{GHz}$ CPU and $64\,\mathrm{GB}$ RAM is $0.15\,\mathrm{s}$ for the motion planner, and $0.03\,\mathrm{s}$ for the mode activator. In the same computer, also the EV tracking controller and the OVs controllers and logics are simultaneously executed, while the low level drivers and controllers run on the robots. Despite the non-optimized m-code implementation, the decision-making execution time is well below the sampling period and the mode activator is particularly quick, indicating that eliminating the infeasible modes before attempting to compute a trajectory for them brings computational advantages.

V. Conclusion

We have presented a cooperating modular decision-making and motion planning approach for autonomous driving. By using forward and backward reachability computations based on simplified vehicle dynamics and a disturbance set, the mode activator determines a set of feasible modes such that the motion planner is able to generate trajectories that achieve their goals. The motion planner selects the trajectory and mode that yields the best performance. By eliminating infeasible modes before the motion planner actually computes trajectories, the decision-making process achieves fast computation, resulting in reactivity to varying environments and low computational loads. We have proved the recursive feasibility of the approach and validated the decision-making algorithm through simulations and experiments.

References

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

[2] S. Di Cairano and I. V. Kolmanovsky, “Real-time optimization and model predictive control for aerospace and automotive applications,” in *Proc. American Control Conference*, 2018, pp. 2392–2409.

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

[4] 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. Intelligent Robots and Systems*, 2016, pp. 5474–5480.

[5] K. Esterle, P. Hart, J. Bernhard, and A. Knoll, “Spatiotemporal motion planning with combinatorial reasoning for autonomous driving,” in *Proc. Int. Conf. Intelligent Transportation Systems*, Nov. 2018, pp. 1053–1060.

[6] 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.

[7] 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. on Intell. Vehicles*, vol. 3, no. 1, pp. 5–17, Mar. 2018.

[8] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, Jan. 2018.

[9] C. Tomlin, G. J. Pappas, and S. Sastry, “Conflict resolution for air traffic management: a study in multiagent hybrid systems,” *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 509–521, Apr. 1998.

[10] F. Blanchini and S. Miani, *Set-theoretic methods in control*. Birkhäuser Boston, 2008.

[11] J.-P. Aubin, *Viability theory*. Springer Science & Business Media, 2009.

[12] M. Althoff and J. M. Dolan, “Online verification of automated road vehicles using reachability analysis,” *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Apr. 2014.

[13] S. Söntges and M. Althoff, “Computing the drivable area of autonomous road vehicles in dynamic road scenes,” *IEEE Trans. Intell. Transport. Syst.*, vol. 19, no. 6, pp. 1855–1866, Jun. 2018.

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

[15] K. Berntorp, T. Hoang, R. Quirynen, and S. Di Cairano, “Control architecture design for autonomous vehicles,” in *Proc. IEEE Conf. Control Technology and Applications*, 2018, pp. 404–411.