Ensuring safety for vehicle parking tasks using Hamilton-Jacobi reachability analysis

Frank J. Jiang[1], Yulong Gao[1,2], Lihua Xie[2], Karl H. Johansson[1]

***Abstract*— In this paper, we propose an approach for ensuring the safety of a vehicle throughout a parking task, even when the vehicle is being operated at varying levels of automation. We start by specifying a vehicle parking task using linear temporal logic formulae that can be model checked for feasibility. The model-checking is facilitated by the construction of a temporal logic tree via Hamilton-Jacobi reachability analysis. Once we know the parking task is feasible for our vehicle model, we utilize the constructed temporal logic tree to directly synthesize control sets. Our approach synthesizes control sets that are least-restrictive in the context of the specification, since they permit any control inputs that are guaranteed not to violate the specification. This least-restrictive characteristic allows for the application of our approach to vehicles under different modes of operation (e.g., human-in-the-loop shared autonomy or fully-automated schemes). Implementing in both simulation and on hardware, we demonstrate the approach's potential for ensuring the safety of vehicles throughout parking tasks, whether they are operated by humans or automated driving systems.**

I. Introduction

Vehicle parking is one of the most time-consuming and safety-critical tasks that drivers perform daily. According to statistics reported in [1], drivers in the U.S., U.K. and Germany wasted 17, 44, and 41 hours a year, respectively; in total costing 72.7 billion dollars, 23.3 billion pounds, and 40.4 billion euros a year. In addition to being an unsustainable (due to wasted fuel) and economically wasteful task, vehicle parking is also often a dangerous and straining task for drivers that involves complex maneuvering into tight spaces with many chaotic safety constraints. Motivated by these issues, researchers have recently spent significant effort in advancing parking guidance and management technology.

In the literature, researchers devote most of their effort into solving the problem of generating obstacle-free trajectories for vehicles in tight parking environments. Nonlinear vehicle dynamics and non-convex environments make this problem difficult to solve in real time. For example, a distributed model predictive control formulation improves parking efficiency and helps ensure collision avoidance by taking into account human behavior prediction and vehicle coordination in [2]. In [3], authors decouple an automated parking problem into a centralized parking spot allocation and path generation problem, and a decentralized collision avoidance control problem. Although these approaches yield important results, the formulations are not suitable for checking whether the parking task is feasible in the first place, since checking the feasibility of a non-convex optimization problem is challenging.

Additionally, to the extent of the authors' knowledge, shared autonomy-based parking receives little attention in the research community, despite the fact that many of the current parking technologies rely on the presence of a human supervisor to perform the parking maneuver [4], [5]. Many have proposed the use of remote human operators as a layer of operations management and exception-handling for connected vehicles in general [6]. In the well-known report by the Society of Automative Engineers (SAE) that describes the "levels of driving automation", the authors even designate levels $\leq$3 as the levels that require the availability of a remote driver (if an on-board driver is not present) and level 4 as the level where a remote driver is an optional part of the operation of a vehicle [7]. These proposals and designations motivate the need for vehicle parking solutions that safely allow for varying levels of automation, be it full automation or partial, human-supported automation.

The main contribution of this paper is a unified solution for handling vehicle parking which can be used for varying levels of automation, ranging from Level $\leq$4, shared autonomy setups to Level 5, fully automated setups. Specifically, the contributions of this paper can be summarized as follows: (1) we specify the parking task for an ego vehicle with a linear temporal logic (LTL) formula and use Hamilton-Jacobi (HJ) reachability analysis to construct a temporal logic tree (TLT) from this LTL formula, which enables the real-time synthesis of feasible, least-restrictive control sets; (2) based on this formulation, we utilize an optimal controller in conjunction with the least-restrictive control sets that can be used for semi-automated or fully-automated parking and demonstrate their effectiveness in simulation and hardware-based experiments.

II. Preliminaries

In this section, we will briefly introduce the required background material for this work.

A. Plant model

Consider a continuous-time dynamic control system

$$\frac{dx}{dt} \triangleq \dot{x} = f(x, u, w), \tag{1}$$

This work is supported by the Swedish Strategic Research Foundation, the Swedish Research Council, and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

[1]F. J. Jiang, Y. Gao, and K. H. Johansson are with the Division of Decision and Control Systems, EECS, KTH Royal Institute of Technology, Malvinas väg 10, 10044 Stockholm, Sweden {frankji, yulongg@kth.se, kallej}@kth.se

[2]Y. Gao and L. Xie are with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore ygao009@ntu.edu.sg, elhxie@ntu.edu.sg

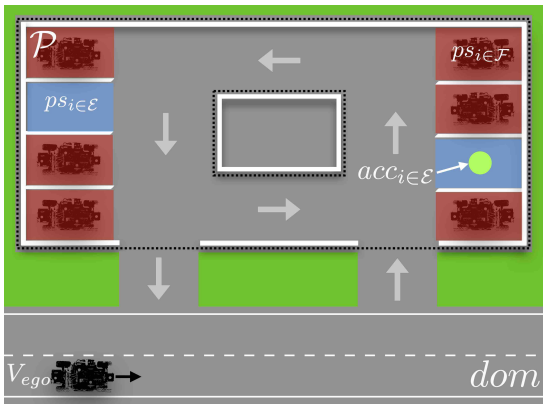

Fig. 1. Illustration of our motivating parking example, where the ego vehicle is drawn on the bottom left corner. Furthermore, we annotate the parking scenario with the XY projections of the important state sets in our problem formulation.

where $x \in \mathbb{R}^{n_x}$ is the system state, $u \in \mathbb{R}^{n_u}$ is the control input, $w \in \mathbb{R}^{n_w}$ is the disturbance, and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ are the dynamics. At each time instant t, the control input $u(t)$ is constrained by a set $\mathbb{U} \subset \mathbb{R}^{n_u}$ and the disturbance $w(t)$ belongs to a compact set $\mathbb{W} \subset \mathbb{R}^{n_w}$. We assume that the control function $u(\cdot)$ is measurable and if for all t, $u(t) \in \mathbb{U}$, we write $u(\cdot) \in \mathcal{U}$ where $\mathcal{U}$ is the function space containing all admissible control functions. We also write $w(\cdot) \in \mathcal{W}$ in the same fashion.

We assume that the system function f is uniformly continuous, bounded, and Lipschitz continuous in x for fixed u and w. As shown in [8], given measurable control and disturbance functions $u(\cdot)$ and $w(\cdot)$, there exists a unique trajectory solving (1). With $\zeta(\cdot; x_0, t_0, u(\cdot), w(\cdot))$, we denote the trajectory starting from an initial state $\zeta(t_0; x_0, t_0, u(\cdot), w(\cdot)) = x_0$ under $u(\cdot)$ and $w(\cdot)$. For the rest of the paper, we will sometimes write $\zeta(\cdot)$ to denote a trajectory for notational simplicity.

We will utilize atomic propositions to encode spatial invariances for system (1). Each proposition p_i is defined as an inequality in $\mathbb{R}^{n_x}$:

$$[p_i] \triangleq \{x \in \mathbb{R}^{n_x} \mid G_i(x) \leq g_i\}, G_i : \mathbb{R}^{n_x} \to \mathbb{R}^{n_i}, g_i \in \mathbb{R}^{n_i},$$

where n_i is the number of inequalities in the ith atomic proposition. Let $\mathcal{AP}$ be a finite set of atomic propositions, i.e., $\mathcal{AP} = \{p_i\}_{i=1}^{N_{\mathcal{A}}}$.

B. Reachability Analysis

Reachability analysis-based approaches are used in many applications in order to provide formal guarantees on the safety of vehicles [9], [10]. For example, a reachability analysis-based approach is presented in [11], where authors use reachability analysis to compute least-restrictive control sets from signal temporal logic formula, which inspired the development the work in [12]. In this subsection, we introduce the definitions of reachable sets and control invariant sets used in our approach.

Definition 2.1: For the system (1), the reachable set from $\Omega_1 \subseteq \mathbb{R}^{n_x}$ to $\Omega_2 \subseteq \mathbb{R}^{n_x}$ is defined as

$$\mathcal{R}(\Omega_1, \Omega_2) = \{x : \exists u(\cdot) \in \mathcal{U}, \forall w(\cdot) \in \mathcal{W}, \exists s > 0, \zeta(s; x, 0, u(\cdot), w(\cdot)) \in \Omega_2, x \in \Omega_1\}. \quad (2)$$

Definition 2.2: For the system (1), $\Omega_f \subseteq \mathbb{R}^{n_x}$ is said to be a robust control invariant set (RCIS) if for any $x \in \Omega_f$ at time t there exists a $u(\cdot) \in \mathcal{U}$ such that $\forall w(\cdot) \in \mathcal{W}$ and $\forall s > t$, $\zeta(s; x, t, u(\cdot), w(\cdot)) \in \Omega_f$.

Definition 2.3: For a set $\Omega \subseteq \mathbb{R}^{n_x}, \mathcal{RCI}(\Omega) \subseteq \mathbb{R}^{n_x}$ is the largest RCIS in Ω if every RCIS $\Omega_f \subseteq \Omega$ satisfies $\Omega_f \subseteq \mathcal{RCI}(\Omega)$.

C. Linear temporal logic

LTL is a convenient formalism for expressing time-related invariances for automated systems [13], [14] and, specifically, for automated driving systems [15], [16]. An LTL formula is defined over a finite set of atomic propositions $\mathcal{AP}$ with both logic and temporal operators. In order to apply our method to continuous-time systems, we consider the fragment "LTL minus next step" where the operator $\bigcirc$ is not included. Its syntax can be described with:

$$\varphi ::= \text{true} \mid p \in \mathcal{AP} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathsf{U} \varphi_2,$$

where U denotes the "until" operators. By using the negation operator and the conjunction operator, we can define disjunction, $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$. Then, by employing the until operator, we can define: (1) eventually, $\Diamond\varphi = \text{true} \cup \varphi$ and (2) always, $\Box\varphi = \neg\Diamond\neg\varphi$.

Definition 2.4: (LTL semantics) For an LTL formula φ, a trajectory $\zeta(\cdot)$, and a time instant $t \geq t_0$, the satisfaction relation $(\zeta(\cdot), t) \vDash \varphi$ is defined as

$$\begin{aligned}
&(\zeta(\cdot), t) \vDash p \in \mathcal{AP} \Leftrightarrow \zeta(t) \in p,\\
&(\zeta(\cdot), t) \vDash \neg\varphi \Leftrightarrow (\zeta(\cdot), t) \nvDash \varphi,\\
&(\zeta(\cdot), t) \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow (\zeta(\cdot), t) \vDash \varphi_1 \wedge (\zeta(\cdot), t) \vDash \varphi_2,\\
&(\zeta(\cdot), t) \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow (\zeta(\cdot), t) \vDash \varphi_1 \vee (\zeta(\cdot), t) \vDash \varphi_2,\\
&(\zeta(\cdot), t) \vDash \varphi_1 \mathsf{U} \varphi_2 \Leftrightarrow \exists t_1 \in [t, \infty) \text{ s.t. } \begin{cases} (\zeta(\cdot), t_1) \vDash \varphi_2, \\ \forall t_2 \in [t, t_1), (\zeta(\cdot), t_2) \vDash \varphi_1, \end{cases}\\
&(\zeta(\cdot), t) \vDash \Diamond\varphi \Leftrightarrow \exists t_1 \in [t, \infty), \text{ s.t. } (\zeta(\cdot), t_1) \vDash \varphi,\\
&(\zeta(\cdot), t) \vDash \Box\varphi \Leftrightarrow \forall t_1 \in [t, \infty), \text{ s.t. } (\zeta(\cdot), t_1) \vDash \varphi.
\end{aligned}$$

D. Temporal Logic Tree

A typical approach to model-checking and synthesizing control from LTL formulae utilizes automata [17], [18]. While automaton-based approaches are useful and important, there are several limitations in their applications to vehicles. First, automaton-based approaches are difficult to apply to infinite systems with uncertainty without using LTL fragments such as bounded-time LTL [19] or co-safe LTL [20]. Second, authors show in [21]–[23] that performing control synthesis from LTL formulae using such approaches is, in general, nontrivial. Lastly, automaton-based approaches are usually designed to be offline approaches, meaning that they

cannot easily handle changes to LTL specifications in real-time.

In this work, we utilize a novel computational structure called TLTs that can be applied to infinite systems with uncertainty (e.g. bounded disturbance or additive noise) and to scenarios where our LTL specification may change over time (e.g. vehicle parking). For extensive details regarding TLTs, we refer readers to [12]. In this subsection, we introduce basic definitions and propositions for temporal logic trees that we will need later on in this work.

Definition 2.5: A TLT is a tree for which

- each node is either a *set* node, *a subset of* $\mathbb{R}^{n_x}$, or an *operator* node, *from* $\{\wedge, \vee, \mathsf{U}, \Box\}$;
- the root node and the leaf nodes are *set* nodes;
- if a set node is not a leaf node, its unique child is an operator node;
- the children of any operator node are set nodes.

Proposition 2.1: As shown in [12], for system (1), a TLT can be constructed from any LTL formula φ via reachability analysis.

In this work, we will use HJ reachability analysis to construct TLTs from LTL formulae, allowing us to apply our approach to nonlinear systems. We call this constructed TLT the controlled TLT. A useful result in [12] is that the control synthesis can be performed using the controlled TLT, instead of the LTL formula, and the resulting trajectory still satisfies the LTL formula since the controlled TLT under-approximates the LTL formula. For an illustrative example of a controlled TLT constructed for the parking scenario we will study in this paper, see Fig. 2.

III. Safe Parking Formulation

In most cases, road vehicles are considered safety-critical systems, since they often share space with humans and carry humans while carrying out tasks. In order to prove and derive safety guarantees for automated road vehicles, the tasks they carry out need to be formalized. For example, we will formalize the parking task shown in Fig. 1.

In Fig. 1, V_{ego} is the vehicle that needs to park in the parking lot. Let $x = [p_x,\ p_y,\ \theta,\ v]$ be the state, where p_x, p_y, θ, and v are V_{ego}'s x-position, y-position, heading, and velocity, respectively. Then, let $u = [\delta,\ a]$ be the input, where δ and a are the steering and acceleration inputs into V_{ego}, respectively. Explicitly, we write the dynamics as

$$f(x,u,w) = \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \frac{v\tan\delta}{L} \\ a \end{bmatrix} + w, \tag{3}$$

where L is the wheel-base length of V_{ego}. Here, $u \in \mathbb{U} \subset \mathbb{R}^2$ and $w \in \mathbb{W} \subset \mathbb{R}^4$.

To define the parking task of the vehicle, we introduce notation for the domain and subdomains of the parking scenario. Let the full set of possible states in our problem domain be $dom \subseteq \mathbb{R}^4$. We call the set of possible states in the parking lot $\mathcal{P} \subset \mathbb{R}^4$, which excludes the block in the middle of the parking lot. Denote the sets of indices corresponding to full and empty parking spots as $\mathcal{F}$ and $\mathcal{E}$, respectively. Then, we let $ps \subset \mathbb{R}^4$ be the parking spots and further denote the full spots as $ps_{i\in\mathcal{F}}$ and the empty spots as $ps_{i\in\mathcal{E}}$. Finally, we introduce the set of states within every $ps_{i\in\mathcal{E}}$ that correspond to an accurate and correct parking job as $acc_{i\in\mathcal{E}} \subset \mathbb{R}^4$. In our example, we define accurate parking as parking in a precise location and with a precise heading. Informally, we can describe the overall parking task as: if there is one or more $ps_{i\in\mathcal{E}}$, then park in one of them while staying safe.

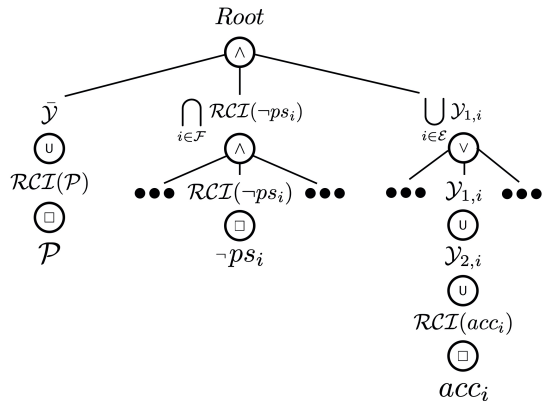

Fig. 2. An illustration of the constructed TLT for our LTL-specified parking task (4).

IV. Our Approach

For our approach, we start by defining an LTL formula that formalizes the parking task outlined in the previous section. Then, using the LTL specification, we use HJ reachability analysis to construct a controlled TLT. By constructing the TLT, we can model-check the LTL specification and design feasible feedback control sets that guarantee the specification is respected.

A. LTL specification for Safe Parking

For the example parking task, the controller needs to first guide V_{ego} from the road into the parking lot and keep V_{ego} in the parking lot. Using LTL, we can write this objective as a stability formula: $\varphi_{\mathcal{P}} = \Diamond\Box\mathcal{P}$. In other words, x_{ego} should eventually always be in $\mathcal{P}$. Next, the controller should ensure that V_{ego} never collides with a full parking spot. We can write this objective using LTL as a safety formula: $\varphi_{\mathcal{S}} = \wedge_{i\in\mathcal{F}}\ \Box\neg ps_i$. Finally, the controller needs to perform the final parking job that should eventually guide V_{ego} into one of the accurate parking spots. We can write this specification in a way that accounts for all available empty parking spots: $\varphi_{\mathcal{R}} = \vee_{i\in\mathcal{E}}\ (dom\ \mathsf{U}\ ps_i\ \mathsf{U}\ \Box\ acc_i)$. In other words, x_{ego} should, sequentially, be: (1) in dom until it has entered into a ps_i where $i \in \mathcal{E}$, (2) in ps_i until it is in $acc_{i\in\mathcal{E}}$, (3) always in $acc_{i\in\mathcal{E}}$. We define the full parking task with the following conjunction:

$$\varphi = \varphi_{\mathcal{P}} \wedge \varphi_{\mathcal{S}} \wedge \varphi_{\mathcal{R}}, \tag{4}$$

or explicitly,

$$\varphi = \Diamond\Box\mathcal{P} \wedge (\wedge_{i\in\mathcal{F}}\ \Box\neg ps_i) \wedge (\vee_{i\in\mathcal{E}}\ (dom\ \mathsf{U}\ ps_i\ \mathsf{U}\ \Box\ acc_i)).$$

B. Constructing TLTs using HJ Reachability Analysis

In [12], we construct the TLT corresponding to an LTL formula using reachable sets. In this work, we will use HJ reachability analysis to compute the required reachable sets for our TLT construction in order to handle the nonlinear dynamics of V_{ego}. Upon constructing the TLT, we use results from [12] to model-check and synthesize control sets from the LTL formula.

For performing HJ reachability analysis, we utilize a similar formulation and numerical method to the ones described in [24]; thus for more details, we refer readers to this work. We start by defining two state sets Ω_1 and Ω_2 as the zero superlevel sets of bounded, Lipschitz continuous functions $h_{\Omega_1} : \mathbb{R}^4 \to \mathbb{R}$ and $h_{\Omega_1} : \mathbb{R}^4 \to \mathbb{R}$. Namely, $\Omega_1 = \{x \mid h_{\Omega_1}(x) \geq 0\}$ and $\Omega_2 = \{x \mid h_{\Omega_2}(x) \geq 0\}$. Then, we solve for the value function $V_{\Omega_1,\Omega_2}(x,\tau)$ that satisfies the following Hamilton-Jacobi-Isaacs variational inequality (HJI VI):

$$\min\{\frac{\partial V_{\Omega_1,\Omega_2}(x,\tau)}{\partial \tau} + H(V_{\Omega_1,\Omega_2}(x,\tau), f(x,u,w)), \\ h_{\Omega_1}(x) - V_{\Omega_1,\Omega_2}(x,\tau)\} = 0, \quad (5)$$

$$V_{\Omega_1,\Omega_2}(x,0) = h_{\Omega_2}(x),\ \tau \leq 0, \quad (6)$$

where the Hamiltonian is given by

$$H(V_{\Omega_1,\Omega_2}(x,\tau), f(x,u,w)) = \\ \max_{u\in\mathbb{U}} \min_{w\in\mathbb{W}} \nabla V_{\Omega_1,\Omega_2}(x,\tau)\cdot f(x,u,w). \quad (7)$$

Once the value function $V_{\Omega_1,\Omega_2}(x,\tau)$ is computed, let $V^*_{\Omega_1,\Omega_2}(x) = \lim_{\tau\to-\infty} V_{\Omega_1,\Omega_2}(x,\tau)$. After recalling Definition 2.1, we can compute the $\mathcal{R}(\Omega_1,\Omega_2)$ as

$$\mathcal{R}(\Omega_1,\Omega_2) = \{x \mid V^*_{\Omega_1,\Omega_2}(x) \geq 0\}. \quad (8)$$

As a special case where $\Omega_1 = \Omega_2 = \Omega$, the reachable set $\mathcal{R}(\Omega,\Omega)$ is the largest RCIS within Ω, i.e., $\mathcal{RCI}(\Omega) = \mathcal{R}(\Omega,\Omega)$; to see this, please refer to Proposition 2.5 in [25].

Recall the parking scenario in Section III. Define the sets $\bar{\mathcal{Y}} = \mathcal{R}(dom, \mathcal{RCI}(\mathcal{P}))$, $\mathcal{Y}_{2,i} = \mathcal{R}(ps_i, \mathcal{RCI}(acc_i))$, and $\mathcal{Y}_{1,i} = \mathcal{R}(dom, \mathcal{Y}_{2,i})$. Now, to construct the TLT corresponding to our LTL formula, we start with the atomic propositions that will be the leaves of the TLT, and progressively construct the TLT bottom-up, operator-by-operator. By following Theorem 5.1 in [12], we can construct a controlled TLT corresponding to (4), as shown in Fig. 2. We denote the root node as

$$Root = \bar{\mathcal{Y}} \cap (\cap_{i\in\mathcal{F}} \mathcal{RCI}(\neg ps_i) \cap (\cup_{i\in\mathcal{E}} \mathcal{Y}_{1,i}).$$

According to the HJ reachability analysis, we can represent each set node in the controlled TLT in the form of value functions. In particularly, we denote by $G^*_{\bar{\mathcal{Y}}}(x) = V^*_{\bar{\mathcal{Y}},\mathcal{RCI}(\mathcal{P})}(x)$, $G^*_{\mathcal{RCI}(\mathcal{P})}(x) = V^*_{\mathcal{P},\mathcal{P}}(x)$, $G^*_{\mathcal{RCI}(\neg ps_i)}(x) = V_{\neg ps_i,\neg ps_i}(x,\tau)$, $G^*_{\mathcal{Y}_{1,i}}(x) = V^*_{\mathcal{Y}_{1,i},\mathcal{Y}_{2,i}}(x)$, $G^*_{\mathcal{Y}_{2,i}}(x,\tau) = V^*_{\mathcal{Y}_{2,i},\mathcal{RCI}(acc_i)}(x)$, and $G^*_{\mathcal{RCI}(acc_i)}(x) = V^*_{acc_i,acc_i}(x)$ the corresponding value functions of the set nodes $\bar{\mathcal{Y}}$, $\mathcal{RCI}(\mathcal{P})$, $\mathcal{RCI}(\neg ps_i)$, $\mathcal{Y}_{1,i}$, $\mathcal{Y}_{2,i}$, and $\mathcal{RCI}(acc_i)$, respectively.

C. Controller Synthesis

In this subsection, we detail the feedback control design we use for our parking task. Given the state x, for each set node $\mathbb{X} \in \{\mathcal{RCI}(\mathcal{P}), \bar{\mathcal{Y}}, \mathcal{RCI}(\neg ps_i), \forall i \in \mathcal{F}, \mathcal{Y}_{2,j}, \mathcal{Y}_{1,j}, \mathcal{RCI}(acc_j), \forall j \in \mathcal{E}\}$, we compute the corresponding control set. Instead of computing the control set based on the current time step, we compute it based on the next sampled time step:

$$\mathbb{U}_{\mathbb{X}}(x) = \{u \in \mathbb{U} \mid G^*_{\mathbb{X}}(x) - \min_{w\in\mathbb{W}} \delta \nabla G^*_{\mathbb{X}}(x)\cdot f(x,u,w) \geq 0\},$$

where δ is a small sampling period. We design the control set computation around the near future to ensure that the chosen control input will guarantee the specification is not violated in consequence of that input.

Following the control synthesis algorithm in [12], we can compute the feasible feedback control set:

$$\mathbb{U}_f(x) = \big(\mathbb{U}_{\mathcal{RCI}(\mathcal{P})}(x) \cup \mathbb{U}_{\bar{\mathcal{Y}}}(x)\big) \cap \big(\cap_{i\in\mathcal{F}} \mathbb{U}_{\mathcal{RCI}(\neg ps_i)}(x)\big) \\ \cap \Big(\cup_{i\in\mathcal{E}} \big(\mathbb{U}_{\mathcal{Y}_{1,i}}(x) \cup \mathbb{U}_{\mathcal{Y}_{2,i}}(x) \cup \mathbb{U}_{\mathcal{RCI}(acc_i)}(x)\big)\Big).$$

The set $\mathbb{U}_f$ is a least-restrictive feedback control set that guarantees we satisfy (4). This means that during operation, we can choose any control policy that satisfies $\mathbb{U}_f$ to guarantee that the corresponding specification will be satisfied. In our parking scenario, we choose to utilize a time-optimal control policy as an example. We can compute the time-optimal control policy for reaching a target set at no extra computational cost. Define the set of the state set pairs as $\Pi = \{(\bar{\mathcal{Y}}, \mathcal{RCI}(\mathcal{P})), (\mathcal{Y}_{1,i}, \mathcal{Y}_{2,i}), (\mathcal{Y}_{2,i}, \mathcal{RCI}(acc_i)), i \in \mathcal{E}\}$. This set will be used to define the optimal controller derived from the minimum time-to-reach (TTR).

If the current state $x \in \mathcal{RCI}(acc_i)$ for some $i \in \mathcal{E}$, it implies that the parking specification is satisfied. In this case, the optimal controller is defined as

$$u^*(x) = \arg\max_{u\in\mathbb{U}_f(x)} \min_{w\in\mathbb{W}} \nabla G^*_{\mathcal{RCI}(acc_i)}(x)\cdot f(x,u,w). \quad (9)$$

Otherwise, we first define the minimum TTR as

$$\tau^* = \max_{(\Omega_1,\Omega_2)\in\Pi} \{\tau \mid \tau < 0, V_{\Omega_1,\Omega_2}(x,\tau) \geq 0\} \quad (10)$$

and let (Ω^*_1, Ω^*_2) be the optimal solution. The time-optimal controller is defined as

$$u^*(x) = \arg\max_{u\in\mathbb{U}_f(x)} \min_{w\in\mathbb{W}} \nabla V_{\Omega^*_1,\Omega^*_2}(x,\tau^*)\cdot f(x,u,w). \quad (11)$$

We will utilize (11) for our fully-automated vehicle that performs parking maneuvers in a time-optimal fashion. Additionally, we also make (11) available to the human operator in our shared-automation example. We further emphasize that any other control policy that satisfies $\mathbb{U}_f$ can be used instead of (11); we choose to use a time-optimal control policy as a useful example.

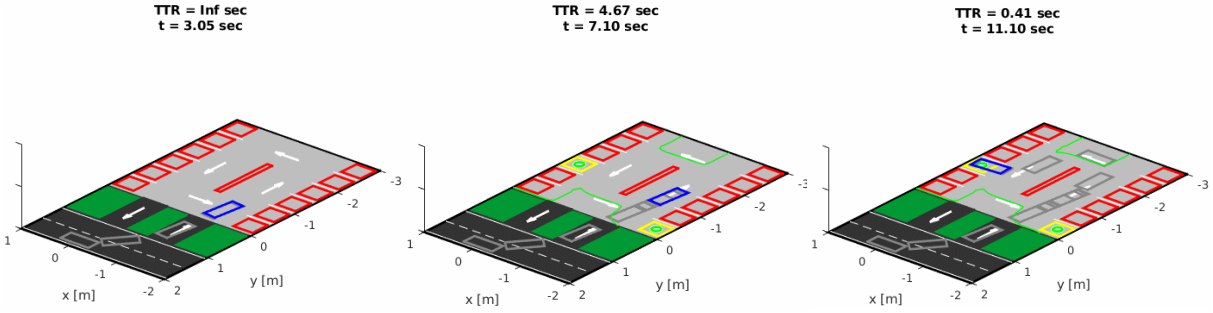

Fig. 3. Shown are three different snapshots in our "handling specification changes" example. The blue rectangle is V_{ego}, the gray rectangles sample the past trajectory of V_{ego}, parking spots with red rectangles have indices in $\mathcal{F}$, spots with yellow rectangles have indices in $\mathcal{E}$, and the small green circles are the accurate state sets. The green curves are the XY projections of the current reachable set. In this example, there are no empty spots initially, and only around $t = 7$ seconds do empty spots appear. When this happens, the vehicle stops waiting, and goes to the empty spot that is most time-optimal from it's position when the spots appear.

V. Results

We present an example in simulation and experimental results on the Small Vehicles for Autonomy (SVEA) platform in the Smart Mobility Lab at the KTH Royal Intitute of Technology. To compute the reachable sets required in our implementation, we use the MATLAB Level Set Method Toolbox [26].

A. Handling specification changes

In our simulated example (illustrated in Fig. 3), we investigate a scenario where V_{ego}, starts on the road, and then navigates into $\mathcal{P}$, where there are no available parking spots. Inside $\mathcal{RCI}(\mathcal{P})$, the vehicle just waits for a parking spot to open up. At time $t = 7$ seconds, two spots open up. By implementing (11), V_{ego} is guided to the spot that is more time-optimal. Here, we are able to observe one of the benefits of using TLTs for performing control synthesis from an LTL specification. Despite the changes in the LTL specification (the change of two full parking spots to empty parking spots), our controller is able to adapt online and utilize offline computations to recompute $\mathbb{U}_f$.

Remark 5.1: Note, we do not consider the presence of physical vehicles in the parking lot. Since our approach computes the least-restrictive control sets for the parking task, dynamic constraints like other physical vehicles can be handled with additional controllers (i.e. model predictive control) on top of our approach. In other words, similar to (11), more sophisticated approaches can be used for choosing the control inputs that satisfy $\mathbb{U}_f$.

B. Hardware Experiments

In our implementation on the SVEA vehicles, we overview both practical applications and considerations of our approach for two different levels of autonomy. We demonstrate how our approach ensures the safety for a Level 5 automated vehicle, which performs the parking maneuver optimally. Then, we show a shared-autonomy setup where we ensure the safety for a vehicle with level 2 driving automation that is being remotely parked by a human operator. The videos for both cases can be found online at [https://bit.ly/HJLTLpark].

Our experimental setup is as follows. All real-time computations happen on the SVEA's onboard computer, which is an NVIDIA Jetson TX2 embedded computer. The state of V_{ego} is given by a Qualisys motion capture system. The chassis of the SVEA vehicle has the control bounds of $\delta \in [-\pi/5, \pi/5]$ rad and $a \in [-1.5, 1.5]$ m/s^2. To handle model errors caused by phenomenon such as the friction between the ground and the SVEA's wheels, let $w_{\max} = [0.05, 0.05, \pi/60, 0.1]$ with units [m, m, rad, m/s^2], and we then set $w \in [-w_{\max}, w_{\max}]$. The required reachable sets are computed offline beforehand, and loaded onto the SVEA's TX2 for real-time computation of (9) and (11).

1) (Level 5) Full Automation: For the full automation case, we implement our control synthesis approach on the SVEA and leave the system to complete the task on it's own. As can be seen from the video provided, the maneuver can be completed accurately and satisfy the original specified task. Furthermore, at the end of the maneuver, we can see the advantage of using a richer nonlinear vehicle model, as the SVEA makes full utilization of its dynamics to maneuver into the parking spot accurately.

2) (Level 2) Shared Automation: An important benefit of our approach is that we compute the least-restrictive control sets for satisfying an LTL formula. Much like in [10], by generating least-restrictive control sets instead of single control policies, our approach is more suitable for human-in-the-loop control formulations, since the human is given the largest acceptable degree of control freedom. To illustrate this, we set up a shared-autonomy example where we enable the SVEA vehicle with level 2 automation and put a human operator in control. Specifically, we enable the SVEA vehicle with the same automation capabilities as the Level 5 automation example, but require an engaged, remote human supervisor for the vehicle's operation. As can be seen from the video link provided, the human operator drives the V_{ego} while V_{ego} continues to satisfy the parking specification, and arrives to just before the desired parking spot. In our particular implementation, we enable the driver to turn on and off the optimal controller used in previous examples. Since the parking spot can be hard to park in efficiently and accurately, the human operator uses the optimal controller

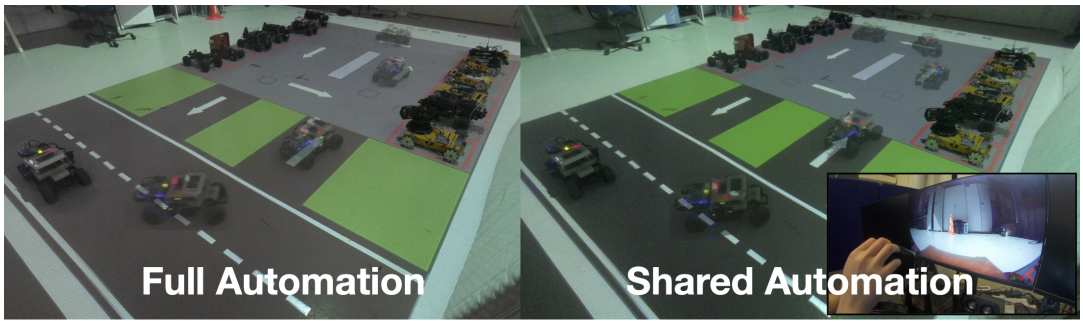

Fig. 4. Time-lapse images from our two experiment examples. Video link: https://bit.ly/HJLTLpark.

to do the final parking maneuver. This application provides flexibility for the human to fully utilize their situational awareness whenever is needed. Furthermore, the setup allows the operator to decide when she wants to use automation to reduce strain and perform parts of the specified task more efficiently.

VI. Conclusion

In general, guaranteeing safety for automated vehicles over their entire operation is difficult, but remains as an important problem we need to solve. We show in this paper how one might specify parking missions formally and how to compute least-restrictive control sets from the specification. By leveraging HJ reachability, we can compute the control sets for nonlinear systems and provide strong guarantees that the specification will be satisfied by the vehicle throughout the parking maneuver. We show applications in both simulation and on hardware of the resulting least-restrictive control sets for ensuring safety for both fully-automated and partially automated vehicle parking. For future work, we will continue to apply our method to different vehicle tasks. By formalizing more vehicle tasks, we will be able to compose more complex missions and apply our unified control framework to guarantee the missions both *can* and *will* be completed.

Acknowledgment

The authors would like to thank Alessandro Abate for providing insightful comments about this work.

References

[1] INRIX, "The impact of parking pain in the us, uk and germany," Tech. Rep., 2017. [Online]. Available: http://www2.inrix.com/research-parking-2017

[2] Y. Li, K. H. Johansson, and J. Mårtensson, "A hierarchical control system for smart parking lots with automated vehicles: improve efficiency by leveraging prediction of human drivers," in *Proceedings of 18th European Control Conference*, 2019, pp. 2675–2681.

[3] X. Shen, X. Zhang, and F. Borrelli, "Autonomous parking of vehicle fleet in tight environments," 2019. [Online]. Available: https://arxiv.org/abs/1910.02349

[4] D. AG. (2020) Innovative parking solutions. convenient, customer-oriented and efficient. [Online]. Available: https://www.daimler.com/innovation/parking.html

[5] B. AG. (2018) Intelligent parking. [Online]. Available: https://www.bmw.co.uk/bmw-ownership/connecteddrive/driver-assistance/intelligent-parking#gref

[6] Wired. (2020) The war to remotely control self-driving cars heats up. [Online]. Available: https://www.wired.com/story/designated-driver-teleoperations-self-driving-cars/

[7] SAE On-Road Automated Vehicle Standards Committee, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," SAE International, Tech. Rep., 2018.

[8] E. A. Coddington and N. Levinson, *Theory of ordinary differential equations*. McGraw-Hill, 1955.

[9] E. Coelingh, L. Jakobsson, H. Lind, and M. Lindman, "Collision warning with auto brake: a real-life safety perspective," Innovations for Safety: Opportunities and Challenges, Tech. Rep., 2007.

[10] K. Leung, E. Schmerling, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, "On infusing reachability-based safety assurance within probabilistic planning frameworks for human-robot vehicle interactions," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 561–574.

[11] M. Chen, Q. Tam, S. Livingston, and M. Pavone., "Signal temporal logic meets Hamilton-Jacobi reachability: connections and applications," in *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, 2018.

[12] Y. Gao, A. Abate, F. J. Jiang, M. Giacobbe, L. Xie, and K. H. Johansson, "Temporal logic trees for model checking and control synthesis of uncertain discrete-time systems," *arXiv preprint arXiv:2007.02271*, 2020.

[13] M. Huth and M. Ryan, *Logic in computer science: modelling and reasoning about systems*. Cambridge University Press, 2004.

[14] G. Fainekos, H. Kress-Gazit, and G. Pappas., "Temporal logic motion planning for mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2005, pp. 2020–2025.

[15] Y. Gao, F. J. Jiang, X. Ren, L. Xie, and K. H. Johansson, "Reachability-based human-in-the-loop control with uncertain specifications," in *Proceedings of 21st IFAC World Congress*, 2020.

[16] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada, "Correct-by-construction adaptive cruise control: Two approaches," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1294–1307, 2016.

[17] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.

[18] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[19] P. G. Sessa, D. Frick, T. A. Wood, and M. Kamgarpour, "From uncertainty data to robust policies for temporal logic planning," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control*, 2018, p. 157–166.

[20] K. Hashimoto and D. Dimarogonas, "Resource-aware networked control systems under temporal logic specifications," *Discrete Event Dynamic Systems*, 09 2019.

[21] P. Tabuada and G. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[22] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer, 2009.

[23] C. Belta, B. Yordanov, and E. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017.

[24] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi Reachability : A Brief Overview and Recent Advances," in *IEEE 56th Annual Conference on Decision and Control*, 2017, pp. 2242–2253.

[25] J. Fernandez Fisac, "Game-theoretic safety assurance for human-centered robotic systems," Ph.D. dissertation, UC Berkeley, 2019.

[26] I. M. Mitchell, "A Toolbox of Level Set Methods," Tech. Rep., 2007.