Human-Centered Design for Safe Teleoperation of Connected Vehicles ⋆

Frank J. Jiang * **Yulong Gao** * **Lihua Xie** ** **Karl H. Johansson** *

* *Division of Decision and Control Systems, EECS, KTH Royal Institute of Technology, Malvinas väg 10, 10044 Stockholm, Sweden* {frankji, yulongg, kallej}@kth.se
** *School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore* elhxie@ntu.edu.sg

Abstract: In this paper, we propose a framework for the safe teleoperation of connected vehicles by remote human operators. The proposed framework combines model-checking, reachability analysis, and human factor design into a unified approach that checks the feasibility of linear temporal logic (LTL) specified teleoperation tasks, guarantees safety throughout the execution of the tasks, and keeps the remote operator as the primary decision maker in the control loop. We apply the general approach to a remote parking example to illustrate the framework. Then, to evaluate the framework's benefits and usability, we conduct a user study on an experimental remote control operator's room and a 1/10th scale vehicle. In the user study, we validate the approach's ability to ensure the completion of LTL-specified tasks and gain initial insight into whether human users find the teleoperation system intuitive.

Keywords: teleoperation, human-centered design, linear temporal logic, reachability analysis, safety, automated vehicles

1. INTRODUCTION

Ever since the DARPA Grand and Urban challenges, researchers and engineers have rapidly developed the automated capabilities of road vehicles. Nowadays, it's even possible to see several initial deployments of automated vehicles on public roads, e.g., Nobina (2019) and Einride (2019). However, as outlined in Koopman and Wagner (2017), even though the automation capabilities of vehicles continue to improve, there are several safety challenges lying ahead of us that are complex and nontrivial.

One solution that several automated vehicle companies are currently pursuing to address the complexities of deploying automated vehicles into society is remote teleoperation, e.g. Davies (2017) and Einride (2020). In most cases, remote teleoperation serves as a semi-automated mode that is used in situations where it is decided, either *a priori* or during operation, that a human operator should carry out a task instead of an automated driving system. As is prescribed in SAE On-Road Automated Vehicle Standards Committee (2018), remote teleoperation is used in situations that are outside of the operating design domain of an automated driving system.

There is a variety of literature that contributes to improving the design of remote teleoperation systems. In particular, Gnatzig et al. (2013) provides an early overview of the requirements on different components of remote teleoperation for road vehicles. In Neumeier and Facchi (2019) and Hosseini and Lienkamp (2016), authors propose different approaches for improving the safety of remote teleoperation in the presence of communication delay and network uncertainty. In the robotics community several approaches for shared autonomy are proposed by, for example, Dragan and Srinivasa (2013), Javdani et al. (2018), and Jeon et al. (2020), which employ different algorithms for goal inference and policy blending for improving teleoperation safety, efficiency, and experience. Moreover, in Ghasemi et al. (2019) and Muslim and Itoh (2019), authors design shared control schemes for vehicles that decrease the cognitive load on human users and encourage collaboration between human operators and automated driving systems. These approaches provide important solutions to different aspects of remote teleoperation. However, to the extent of the authors' knowledge, most approaches for remote teleoperation focus on handling communication delay and human intent prediction, and do not focus on providing strong feasibility and safety guarantees.

The main contribution of this paper is the design and empirical evaluation of a unified framework for safe, remote teleoperation of connected vehicles. Specifically, we will take a set of linear temporal logic (LTL) specified teleoperation tasks and use reachable sets to construct sets of control inputs that ensure one of the tasks is completed as specified. We allow for a set of tasks, as opposed to just a single task, to allow for multi-objective missions. Furthermore, this framework is designed to keep the human operator as the primary decision maker; this is important in real world deployments of connected vehicles, because human operators will typically be asked to teleoperate a vehicle in exceptional situations that require human situational awareness and decision-making. In this work, we extend the work in Gao et al. (2020b) with the use of temporal logic trees and Hamilton-Jacobi reachability analysis for both model-checking and control synthesis. Since we use Hamilton-Jacobi reachability analysis, our framework is able to handle nonlinear vehicle dynamics and non-convex safety-constraints. By using temporal logic trees, as opposed to automaton, we are able to

⋆ 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.

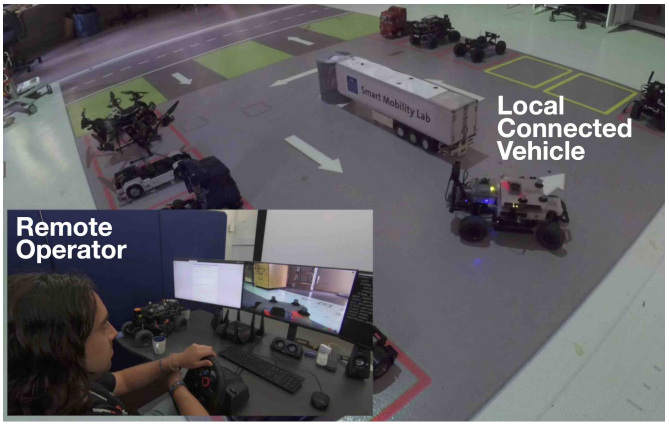

Fig. 1. Experimental setup in the Smart Mobility Lab at KTH Royal Institute of Technology.

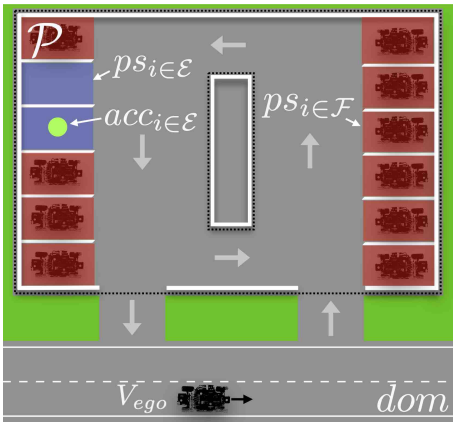

Fig. 2. Illustration of our parking scenario.

perform verification over any LTL formula (in this work we do not use the "next" operator, since we have a continuous-time system) for infinite, uncertain systems (e.g. with bounded disturbance) and can update the control policy in real-time when the LTL-specified objective changes; these characteristics are presented in more detail in Gao et al. (2020a). Specifically, the contributions of this paper are as follows:

(1) We introduce a safe teleoperation framework for connected vehicles using Hamilton-Jacobi reachability analysis and temporal logic trees.
(2) We develop and illustrate an application of our framework using multi-objective remote parking tasks.
(3) We evaluate the validity and benefits of our approach by performing user studies in an experimental operator room and a 1/10th scale vehicle.

The remainder of the paper is organized as follows. In Section 2, we introduce an illustrative remote parking example to contextualize our problem. In Section 3, we outline our design approach to safe, remote teleoperation and apply our framework to the remote parking example. In Section 4, we detail our experimental setup and report results from our user study on the teleoperation framework. In Section 5, we conclude the paper with a discussion about the possible implications of our work and future directions.

2. MOTIVATING EXAMPLE AND PROBLEM STATEMENT

As is reported in Davies (2017), the need for remote teleoperation became more apparent when companies started to realize that there are scenarios (especially in environments shared by humans and human-driven vehicles) that are hard to handle with automation. In experimental deployments of automated vehicles, this issue is addressed by having a fallback human driver in the vehicle during operation. However, since having a human driver in each vehicle is not a scalable solution, remote teleoperation has gradually become an attractive alternative.

The motivating example we will use to study our safe teleoperation framework is remote parking. An experimental setup for remote parking is shown in Fig. 1. INRIX (2017) reports that parking vehicles costs drivers significant time and money annually, which is why the automation of vehicle parking is well-motivated. Moreover, since many parking lots are difficult to maneuver within and involve both pedestrians and human-driven vehicles, parking in a parking lot is not a task that is easy to fully automate from start to finish without high-fidelity models of human behavior. Despite this automation challenge, parking in a parking lot is a task that many ordinary human drivers are able to handle on a daily basis, further motivating the remote connection of a human operator throughout a parking task.

In a remote parking scenario, a remote operator needs to safely guide a vehicle to a parking spot of her choice. For example, in Fig. 2, an operator will remotely guide the vehicle outside of the parking lot into one of the free blue parking spots, while avoiding collision with the walls of the parking lot and the parked vehicles. Often, this parking task is both a safety-critical task and a dynamically challenging task (the maneuvering space is tight and there is a requirement to park accurately). To make matters worse, even though the remote operation of vehicles allows for scalable human support for automated vehicles, remote parking is more dangerous and difficult than in-person parking due to latency in the communication channel and the possible lack of embodiment.

Thus, to safely take advantage of the scalability benefits of a remote parking setup, we need to somehow ensure the feasibility and safety of remote parking, even when the human operator is in control, which leads us the following problem statement:

Problem 2.1. Given a vehicle and a multi-objective plan as a set of LTL specifications, design a teleoperation framework that checks whether the plan itself is feasible and guarantees the vehicle satisfies the feasible plan during operation, even under the control of a human operator.

When the human operator is in control of the remote vehicle, we can expect that incorrect control inputs will be given to the system due to issues such as latency, lack of embodiment, and human error; in the next section, we will outline how we have designed our framework to be robust to these issues.

3. SAFE TELEOPERATION FRAMEWORK

To address Problem 2.1, we first overview our safe remote teleoperation framework and then detail how the framework guarantees feasibility and safety under a set of LTL-specified teleoperation tasks. In particular, we will exemplify the framework by using the remote parking example.

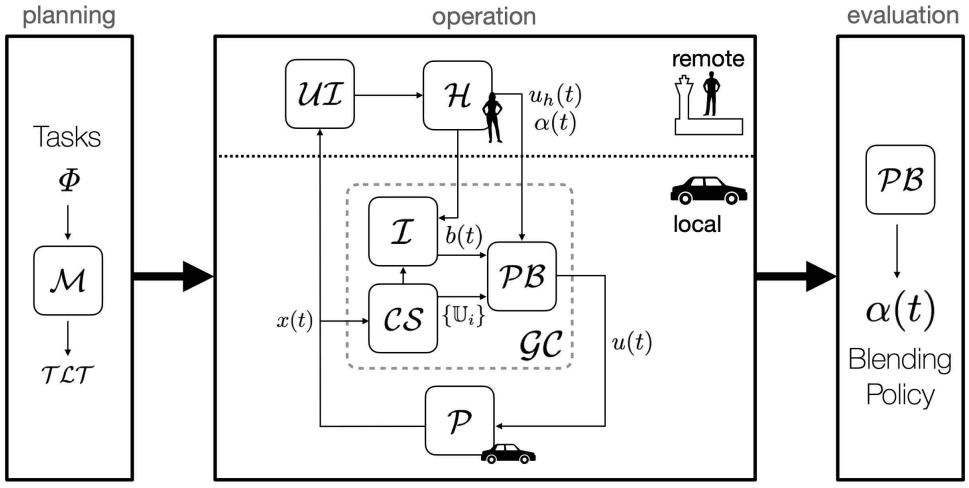

Fig. 3. A diagram overviewing the full deployment of remote teleoperation starting from the initial proposal Φ to the evaluation of the resulting blending policy.

3.1 Framework Overview

We propose a remote teleoperation framework (depicted in Fig. 3) that combines model-checking and reachability analysis to check the feasibility of tasks before operation and filter control inputs to guarantee recursive feasibility of the task during operation.

The framework in Fig. 3 is split into three stages: planning, operation, and evaluation. In the planning stage, the teleoperation task is formalised and checked for feasibility. If the formalised plan is valid, then the operation begins. The remote operator (block $\mathcal{H}$) begins communicating her decisions about how the vehicle (block $\mathcal{P}$) should be operated to the local automated driving system's guiding controller (block $\mathcal{GC}$). Upon receiving the remote operator's input and the vehicle's state information, the guiding controller implements the remote operator's input as long as the input will not violate the plan made in the planning stage. If the operator's input violates the plan, then the guiding controller will correct the input to guarantee the plan remains feasible. Finally, after operation, the deployment is reviewed in the evaluation stage.

Before we further detail our framework, we first introduce the vehicle model and task plans we use in our work.

Vehicle Model ($\mathcal{P}$) 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

$$f(x,u,w) = \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \frac{v\tan\delta}{L} \\ a \end{bmatrix} + w, \tag{1}$$

where L is the wheel-base length of V_{ego}. Here, $x \in \mathbb{R}^4$, $u \in \mathbb{U} \subset \mathbb{R}^2$, and $w \in \mathbb{W} \subset \mathbb{R}^4$. f is uniformly continuous, bounded, and Lipschitz continuous in x for fixed u and w. As shown in Coddington and Levinson (1955), given a measurable control and disturbance functions $u(\cdot)$ and $w(\cdot)$, there exists a unique trajectory solving (1). We denote by $\zeta(\cdot; x_0, t_0, u(\cdot), w(\cdot))$ 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.

Task Planning (Φ) When planning the teleoperation, the first step is to define the plan as a set of LTL formulae. LTL is a convenient way to express tasks with spatial and temporal requirements, such as obstacle avoidance or stability. For details about LTL, we refer the readers to Huth and Ryan (2004), Baier and Katoen (2008), and Belta et al. (2017). Moreover, in Gao et al. (2020b) and Guo et al. (2018), LTL has been shown to be a particularly useful formalism for shared-autonomy tasks. Next, let us show how to use a set of LTL formulae to define the parking task.

In the remote parking example, we consider a similar parking scenario used in Jiang et al. (2020). We ask users to remotely park the vehicles into a parking lot, as is depicted by Fig. 1. On a high-level, the teleoperation task we want the users to complete is to (1) enter the parking lot, (2) do not collide with the parking lot walls or any full parking spots (3) eventually park in any empty parking spot accurately. To define the parking task of the vehicle, we introduce notation for the domain and subdomains of the parking scenario, as shown in Fig. 2. Let the full set of possible states in our problem domain be $dom \subset \mathbb{R}^4$. 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. We denote the parking spots as $ps \subset \mathbb{R}^4$. Denote full parking spots as $ps_{j\in\mathcal{F}}$ and empty parking spots as $ps_{i\in\mathcal{E}}$, and the set of states within every $ps_{i\in\mathcal{E}}$ that correspond to an accurate and correct parking job as $acc_{i\in\mathcal{E}} \subset \mathbb{R}^4$.

Now, we can use LTL operators to formalize the high-level teleoperation task we described earlier. Namely, we can write the "enter the parking lot" part of the task as the stability formula: $\diamondsuit\square\mathcal{P}$. Note, this also encodes collision avoidance with the walls or boundary of the parking lot. Then, we can write the "do not collide with any full parking spots" part of the task as the safety formula: $\wedge_{j\in\mathcal{F}}\ \square\neg ps_j$. Finally, we can write the "eventually park in any empty parking spot accurately" part of the task as: $dom\ \mathsf{U}\ ps_i\ \mathsf{U}\ \square\ acc_i$ Explicitly, we can formalize the full teleoperation parking task for an empty parking spot $i \in \mathcal{E}$ as

$$\varphi_i = \diamondsuit\square\mathcal{P} \wedge (\wedge_{j\in\mathcal{F}}\ \square\neg ps_j) \wedge (dom\ \mathsf{U}\ ps_i\ \mathsf{U}\ \square\ acc_i). \tag{2}$$

Then, we use (2) to write the set of specified tasks as $\Phi = \{\varphi_i\}_{\forall i \in \mathcal{E}}$.

Framework Design After modeling the dynamics of the vehicle and planning the LTL tasks, we will show the function of each module in the safe teleoperation framework, as shown in Fig. 3. With the inputs of a set of LTL formulae Φ, a model-checking module ($\mathcal{M}$) is used to check whether the planned tasks are valid and possible for the teleoperated vehicle to satisfy by transforming the LTL formulae into a set of temporal logic trees (TLT), denoted by $\mathcal{TLT}$. The TLT is a useful tool proposed in our recent work, Gao et al. (2020a), for LTL model checking and control synthesis.

If the tasks are found to be possible for the vehicle to satisfy, then we begin the teleoperation. At each time instant t, the human operator makes a decision $u_h(t)$ based on the output of a user interface module ($\mathcal{UI}$), which visualizes the status of the vehicle to the human operator.

The human's decision $u_h(t)$ is then communicated to the guiding controller module ($\mathcal{GC}$) running locally on the vehicle. Within $\mathcal{GC}$ the control set synthesis module ($\mathcal{CS}$ in Fig. 3) takes the state of the vehicle and uses the TLTs computed in the planning phase to synthesize the set of feasible control inputs for each LTL task $\varphi_i \in \Phi$. These control sets are passed, along with the human operator's decided control input, to the inference module ($\mathcal{I}$ in Fig. 3), which then infers a belief about which task the human operator is trying to complete. The control sets from $\mathcal{CS}$, the belief from $\mathcal{I}$, and the control input from the human operator are passed to the policy blending module ($\mathcal{PB}$ in Fig. 3). $\mathcal{PB}$ takes in all the available information and decides a final control input to implement on the vehicle that best reflects the human operator's decided control input, while guaranteeing the satisfaction of the inferred LTL-specified task.

In the framework, $\mathcal{M}$ and $\mathcal{CS}$ are the two modules that guarantee the feasibility of the teleoperation tasks and the safety of the vehicle. Since module $\mathcal{CS}$ runs on-board the vehicle, the feasibility and safety guarantees it provides is unaffected by communication latency. The other two modules, $\mathcal{I}$ and $\mathcal{PB}$, handle the interaction between the human operator and the automated driving system via a shared autonomy approach. In the rest of this section, we will walk through each module and apply the safe teleoperation framework to the parking scenario used in Jiang et al. (2020).

3.2 Model Checking ($\mathcal{M}$)

Given a set of desired LTL-specified tasks Φ, the model checking module checks if V_{ego}, starting at an initial state $x(0)$, is able to satisfy $\varphi_i, \forall \varphi_i \in \Phi$. To do this, we first use reachable sets to construct a TLT for each $\varphi_i \in \Phi$. Specifically, a TLT is a tree structure that consists of state set nodes and operator nodes. TLTs provide a work flow for tracking state trajectories satisfying LTL formulae. The construction procedure of such TLTs is recursive in the definition of the LTL formulae. Note that the constructed TLT under-approximates the corresponding LTL formula φ_i, which is shown in Gao et al. (2020a). We denote the set of constructed TLTs for Φ as $\mathcal{TLT}$. The model checking can be performed by using these TLTs. A sufficient condition on the feasibility of $\varphi_i \in \Phi$ for V_{ego} is if the initial state $x(0)$ belongs to the root node of the TLT for φ_i. For more details on construction of TLTs and the approximation relation

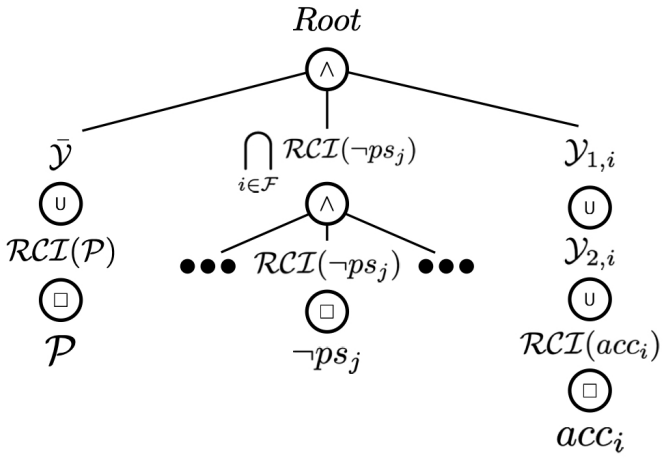

Fig. 4. An example TLT corresponding to task φ_i, where task φ_i is to safely park in an available empty parking spot $i \in \mathcal{E}$.

between TLT and LTL formulae, we refer readers to Gao et al. (2020a).

Next, we will show how to build the $\mathcal{TLT}$ for the remote parking example. We first introduce some definitions for reachable sets for V_{ego}.

Definition 3.1. For V_{ego}, the reachable set from $\Omega_1 \in \mathbb{R}^4$ to $\Omega_2 \in \mathbb{R}^4$ is defined as

$$\mathcal{R}(\Omega_1, \Omega_2) = \{x : \exists u(\cdot) \in \mathcal{U}, \forall w(\cdot) \in \mathcal{W}, \exists s > 0, \zeta(s; x, 0, u(\cdot), w(\cdot)) \in \Omega_2, x \in \Omega_1\}. \quad (3)$$

For V_{ego}, $\Omega_f \subseteq \mathbb{R}^4$ 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 $\forall s > t$, $\zeta(s; x, t, u(\cdot), w(\cdot)) \in \Omega_f$. For a set $\Omega \subseteq \mathbb{R}^4, \mathcal{RCI}(\Omega) \subseteq \mathbb{R}^4$ is the largest RCIS in Ω if every RCIS $\Omega_f \subseteq \Omega$ satisfies $\Omega_f \subseteq \mathcal{RCI}(\Omega)$.

We can compute reachable sets and RCIS's with Hamilton-Jacobi reachability analysis. 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):

$$\min\{\frac{\partial V_{\Omega_1,\Omega_2}(x,\tau)}{\partial \tau} + H(V_{\Omega_1,\Omega_2}(x,\tau), f(x,u,w)), h_{\Omega_1}(x) - V_{\Omega_1,\Omega_2}(x,\tau)\} = 0, \quad (4)$$

$$V_{\Omega_1,\Omega_2}(x, 0) = h_{\Omega_2}(x), \ \tau \leq 0, \quad (5)$$

where the Hamiltonian is given by

$$H(V_{\Omega_1,\Omega_2}(x,\tau), f(x,u,w)) = \max_{u\in\mathbb{U}} \min_{w\in\mathbb{W}} \nabla V_{\Omega_1,\Omega_2}(x,\tau) \cdot f(x,u,w). \quad (6)$$

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 3.1, we can compute the $\mathcal{R}(\Omega_1, \Omega_2)$ as

$$\mathcal{R}(\Omega_1, \Omega_2) = \{x \mid V^*_{\Omega_1,\Omega_2}(x) \geq 0\}. \quad (7)$$

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 Fernandez Fisac (2019).

Recall the parking scenario in Fig. 2. For the LTL parking task $\varphi_i = \diamondsuit\square\mathcal{P} \wedge (\wedge_{j\in\mathcal{F}} \square\neg ps_j) \wedge (dom \,\mathsf{U}\, ps_i \,\mathsf{U}\, \square\, acc_i)$ in (2), we can compute $\mathcal{RCI}(\mathcal{P})$, $\bar{\mathcal{Y}} = \mathcal{R}(dom, \mathcal{RCI}(\mathcal{P}))$, $\mathcal{RCI}(\neg ps_j)$, $\forall j \in \mathcal{F}$, $\mathcal{Y}_{2,i} = \mathcal{R}(ps_i, \mathcal{RCI}(acc_i))$, $\mathcal{Y}_{1,i} = \mathcal{R}(dom, \mathcal{Y}_{2,i})$, and $\mathcal{RCI}(acc_i)$, $\forall i \in \mathcal{E}$. Following Gao et al. (2020a), we can construct $\mathcal{TLT}$ for Φ (we show an example TLT for a task $\varphi_i \in \Phi$ in Fig. 4). If the initial state $x(0)$ belongs the root node of the corresponding TLT, then the parking task φ_i is feasible.

3.3 Control Set Synthesis ($\mathcal{CS}$)

Given the state $x(t)$ of V_{ego}, $\mathcal{CS}$ utilizes the $\mathcal{TLT}$ computed in $\mathcal{M}$ to synthesize least-restrictive control sets for V_{ego} that ensures the recursive feasibility (and safety) from $x(t)$ for each $\varphi_i \in \Phi$. The control sets are least-restrictive in the sense that they permit any control inputs that do not violate Φ. We denote the least-restrictive control sets as $\{\mathbb{U}_i(x)\}_{i=1,\dots,n_s}$. For simplicity, we refer to these control sets as $\{\mathbb{U}_i\}$. The algorithm performed for $\mathcal{CS}$ is as follows: (1) check if $x(t)$ belongs to the state set node and replace each state set node in the TLT with a corresponding control set according to the semantics of the operators; (2) compress this new tree and backtrack starting from a single control set through a bottom-up traversal. Note, this control set collects all the feasible control inputs such that the LTL specification is recursively feasible. This control set is useful in the policy blender for filtering the human operator's decision. For more details, we refer the readers to Gao et al. (2020a).

Next, we show how to compute the control set in the remote parking example. Denote our set nodes as $\mathbb{X} \in \{\mathcal{RCI}(\mathcal{P}), \bar{\mathcal{Y}}, \mathcal{RCI}(\neg ps_j), \forall j \in \mathcal{F}, \mathcal{Y}_{2,i}, \mathcal{Y}_{1,i}, \mathcal{RCI}(acc_i), \forall i \in \mathcal{E}\}$. Now, let the corresponding value functions to the set nodes be $G^*_{\bar{\mathcal{Y}}}(x) = V^*_{\bar{\mathcal{Y}},\mathcal{RCI}(\mathcal{P})}(x)$, $G^*_{\mathcal{RCI}(\mathcal{P})}(x) = V^*_{\mathcal{P},\mathcal{P}}(x)$, $G^*_{\mathcal{RCI}(\neg ps_i)}(x) = V_{\neg ps_i,\neg ps_i}(x,\tau)$, $G^*_{\mathcal{Y}_{1,i}}(x) = V^*_{\mathcal{Y}_{1,i},\mathcal{Y}_{2,i}}(x)$, $G^*_{\mathcal{Y}_{2,i}}(x,\tau) = V^*_{\mathcal{Y}_{2,i},\mathcal{RCI}(acc_i)}(x)$, and $G^*_{\mathcal{RCI}(acc_i)}(x) = V^*_{acc_i,acc_i}(x)$. Given the state x at the time instant t, for each set node $\mathbb{X}$, the corresponding control set is as follows

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

where δ is a small sampling period. Instead of computing the control set based on the current time step, we compute it based on the next sampled time step. This ensures that the chosen control input will guarantee the specification set is not violated in consequence of that input.

Following the control synthesis algorithm in Gao et al. (2020a), we can compute the feasible feedback control sets $\forall i \in \mathcal{E}$:

$$\begin{aligned}\mathbb{U}_i(x) = &\left(\mathbb{U}_{\mathcal{RCI}(\mathcal{P})}(x) \cup \mathbb{U}_{\bar{\mathcal{Y}}}(x)\right) \cap \left(\cap_{j\in\mathcal{F}} \mathbb{U}_{\mathcal{RCI}(\neg ps_j)}(x)\right) \\ &\cap \Big(\mathbb{U}_{\mathcal{Y}_{1,i}}(x) \cup \mathbb{U}_{\mathcal{Y}_{2,i}}(x) \cup \mathbb{U}_{\mathcal{RCI}(acc_i)}(x)\Big). \end{aligned} \tag{8}$$

In a slight abuse of notation, we will sometimes write $\mathbb{U}_i(x)$ as $\mathbb{U}_i$. As mentioned previously, (8) is a least restrictive control set that guarantees V_{ego} satisfies (2).

3.4 Inference ($\mathcal{I}$)

Given the control sets $\{\mathbb{U}_i\}$ computed in $\mathcal{CS}$, $\mathcal{I}$ infers beliefs about which $\varphi_i \in \Phi$ the operator is intending to complete. We denote the inferred beliefs as $b(t)$. There are many inference strategies that can be used in the Inference module $\mathcal{I}$ in the existing literature. We refer readers to the extensive survey presented by Brown et al. (2020).

In our parking example, we consider an inferred belief based on time optimality. For each $i \in \mathcal{E}$, let $\Pi_i = \{(\bar{\mathcal{Y}}, \mathcal{RCI}(\mathcal{P})), (\mathcal{Y}_{1,i}, \mathcal{Y}_{2,i}), (\mathcal{Y}_{2,i}, \mathcal{RCI}(acc_i))\}$ be state set pairs we will use to define our time-optimal controller. Assuming the vehicle has not parked yet (i.e. $x(t) \notin \mathcal{RCI}(acc_i)$), then, the minimal time-to-reach (TTR) for parking spot $i \in \mathcal{E}$ is

$$\tau^*_i(x) = \max_{(\Omega_1,\Omega_2)\in\Pi_i} \{\tau \mid \tau < 0, V_{\Omega_1,\Omega_2}(x,\tau) \geq 0\}. \tag{9}$$

Let the minimal TTR for all parking spots be $\tau^*(x) = \min_{i\in\mathcal{E}} \tau^*_i(x)$ and the corresponding optimal state set pair to $\tau^*(x)$ be (Ω^*_1, Ω^*_2). Then, a time-optimal control policy that the automated vehicle could use to park can be computed as

$$u^*(x) = \arg \max_{u\in\mathbb{U}_f(x)} \min_{w\in\mathbb{W}} \nabla V_{\Omega^*_1,\Omega^*_2}(x, \tau^*(x)) \cdot f(x,u,w). \tag{10}$$

Then, we define the inferred belief of the human operator's preference over the specifications $i \in \mathcal{E}$ as

$$b_i(t) = \begin{cases} 1, \text{ if } i = \text{argmin}_{i\in\mathcal{E}}\, \tau^*_i(x(t)), \\ 0, \text{ otherwise.} \end{cases} \tag{11}$$

Let $b(t)$ be the set of beliefs for every empty parking spot, in other words, $b(t) = \{b_i(t)\}_{\forall i\in\mathcal{E}}$.

3.5 Policy Blender ($\mathcal{PB}$)

Our proposed policy blender module is depicted in Fig. 5. $\mathcal{PB}$ takes in beliefs $b(t)$ computed by $\mathcal{I}$, least-restrictive control sets $\{\mathbb{U}_i\}$ from $\mathcal{CS}$ and a decision from the remote human operator in the form of $u_h(t)$ and $\alpha(t)$, then outputs a control input $u(t)$ that will be implemented on V_{ego}.

The first step in $\mathcal{PB}$ is to compute an automated driving input. In Fig. 5, we denote this step as $\mathcal{C}$. $\mathcal{C}$ corresponds to the input an automated driving system would compute to complete the inferred task $\varphi_i \in \Phi$. Let the control input computed by the automated driving system be $u_a(t)$. Then, we blend the automated driving input with the human input in $\mathcal{B}$. We take the weighted sum of $u_a(t)$ and the control input from the human, $u_h(t)$ using the weight $\alpha(t)$, as follows:

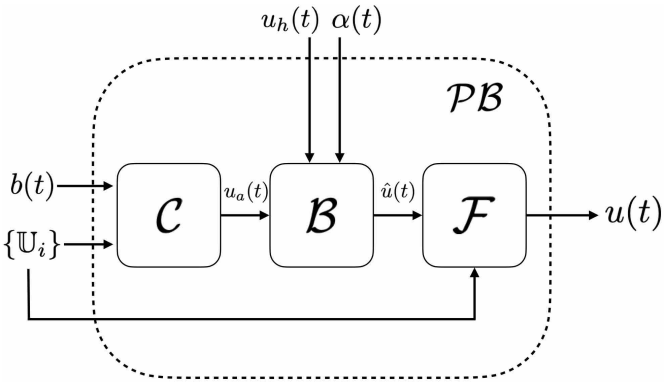

Fig. 5. Breakdown of policy blending module. The remote human operator inputs their control input and blending policy into $\mathcal{B}$ to be combined with the automated driving system's control input.

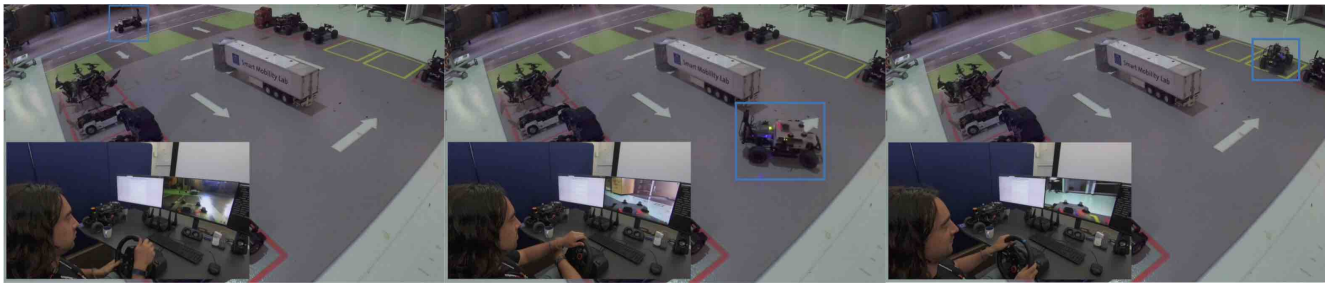

Fig. 6. Snapshots of an example trial of the proposed remote teleoperation system, starting from the start of the trial (on the left) until the satisfaction of one of the LTL-specified parking tasks (on the right).

$$\hat{u}(t) = (1 - \alpha(t)) \cdot u_h(t) + \alpha(t) \cdot u_a(t). \tag{12}$$

Changing $\alpha(t)$ gives the operator the ability to change the balance between the automated driving system and her control over V_{ego}. In other words, $\alpha(t)$ allows the human operator to decide the level of automation the task is completed at.

Finally, we filter the control input in module $\mathcal{F}$ to enforce the satisfaction of the inferred task φ_i.

$$u(t) = \begin{cases} \hat{u}(t) & \text{if } \exists i, \hat{u}(t) \in \mathbb{U}_i(x(t)) \\ \underset{u \in \mathbb{U}_i(x(t)), \forall i}{\operatorname{argmin}} \frac{\|u - u_h(t)\|}{b_i(t)}, & \text{otherwise.} \end{cases} \tag{13}$$

Using (13), we can ensure that the resultant control input closest reflects the inferred belief of the human operator's objective while guaranteeing Φ is still satisfied.

Remark 3.1. Automated control module $\mathcal{C}$ can be replaced by any automated control policy of interest. We will choose an time-optimal controller in the remote parking scenario, but, for example, this could easily be replaced by a controller that compensates for communication delay or optimizes other objectives such as passenger comfort or fuel efficiency.

For the remote parking example, we can assist the human operator using the control set given by (8). As an example, we assign the automated driving policy that we use for assisting the human operator to be the time optimal control policy (10):

$$u_a(t) = u^*(x(t)). \tag{14}$$

We emphasize that our framework gives the remote human operator the freedom to decide on both the control of the vehicle (using $u_h(t)$) and how much control they have over the automated driving system (using $\alpha(t)$), as long as the vehicle is still guaranteed to satisfy the teleoperation task. This level of freedom is designed to give the operator both the freedom and the automation she needs to efficiently complete the teleoperation task. In the next section, we will perform the experiments to show the effectiveness of the remote parking in the safe framework and to evaluate the interaction between the human operator and the automated driving system.

4. EXPERIMENTAL EVALUATIONS

In Section 3, we have defined a safe teleoperation framework that allows the deployment of a parking teleoperation task with guarantees on its feasibility and safety. Moreover, we defined an automated driving policy (14) that the remote operator can use for assistance by controlling their desired blending policy. In this section, we validate an experimental implementation of this remote parking system and evaluate how real human operators interact with the system through user studies.

4.1 Experimental Setup

Shown in Fig. 1, we evaluate the human users in an experimental control operator room, where they are able to teleoperate a remote 1/10th scale vehicle. The 1/10th scale vehicles used in the experiment are the Small Vehicles for Autonomy (SVEA) platform developed in the Smart Mobility Lab at the KTH Royal Institute of Technology. All of the computations in $\mathcal{GC}$ in Fig. 3 are performed on the SVEA vehicle's on-board NVIDIA TX2. The communication link between the vehicle and the control operator room is performed by a WebRTC-based protocol over a 5G link. To visualize the status of the vehicle to the operator, we simply stream video from a forward-facing 4K camera. Additionally, we have integrated Logitech's G29 steering wheel and pedals into the system for the human operator to give both $u_h(t)$ and $\alpha(t)$. By communicating high resolution video and control inputs over a low-latency wireless channel, we are able to experiment with similar communication conditions utilized in deployments such as Einride (2020).

4.2 Experimental Design

We set up a parking lot with a similar layout to Fig. 2 and ask human users to park the SVEA into an empty parking spot of their choice from the control operator room. We evaluate the subjects under two conditions: (1) without the optimal control policy, (2) with assistance from the optimal control policy. While validating the safe teleoperation framework, this will also allow us to evaluate how human users utilize the optimal control policy when it is available. Before the trials begin, we introduce the test users to the controls and the task and allow the users to operate the system until they state they feel comfortable with the system.

Throughout the trials, we collect three types of objective measures on the test users. First, we count the number of times V_{ego} needs to be corrected by $\mathcal{F}$. Second, we compute the sub-optimality of the users compared to the globally optimal control policy. Third, we record the blending policy of the users. With these measures we will be able to evaluate both the ensuring and the assistive capabilities of the framework.

In addition to the objective data, we also collect subjective data on the test users. Using a survey, we ask the users whether they think the system is **intuitive**, **felt safe**, and whether they **felt in control**.

4.3 Results

In this section, we show the results of evaluating our framework on two human users. The first human user (referred to as

”Human 1”) is an inexperienced user who has not previously used our teleoperation system. The second human user (referred to as ”Human 2” is an experienced user who has used our teleoperation system before. Both users are experienced drivers with driving licenses. A video of an example trial can be seen at [https://youtu.be/TJhnQ3URrho].

We illustrate the trajectories of the two human users in each trial in Fig. 7. As a baseline, we run the system with full automation and visualize the trajectory alongside the the trajectories of the human users. Furthermore, we also visualize the number of times module $\mathcal{F}$ corrects the human users when they are about to violate Φ. We can see that both users are able to safely park into their chosen parking spot.

In Fig. 8 and Fig. 9, we visualize a comparison between the time optimality of the automated system and the human users with and without assistance. In Fig. 8, since the user starts outside of the $\mathcal{RCI}$ of the parking lot, the time-to-reach reaches zero twice in each trial: once for reaching the $\mathcal{RCI}$ of the parking lot and again for reaching the parking spot. When compared to the fully automated optimal controller, we see that the human users are less optimal. Among all of the driving, the least optimal driving was by Human 1 without any assistance. Furthermore, both Human 1 and Human 2 exhibit a blending policy that favors automation in the middle of the parking task and at the end of the parking task. Human 2 additionally opted to start the task using the full automation.

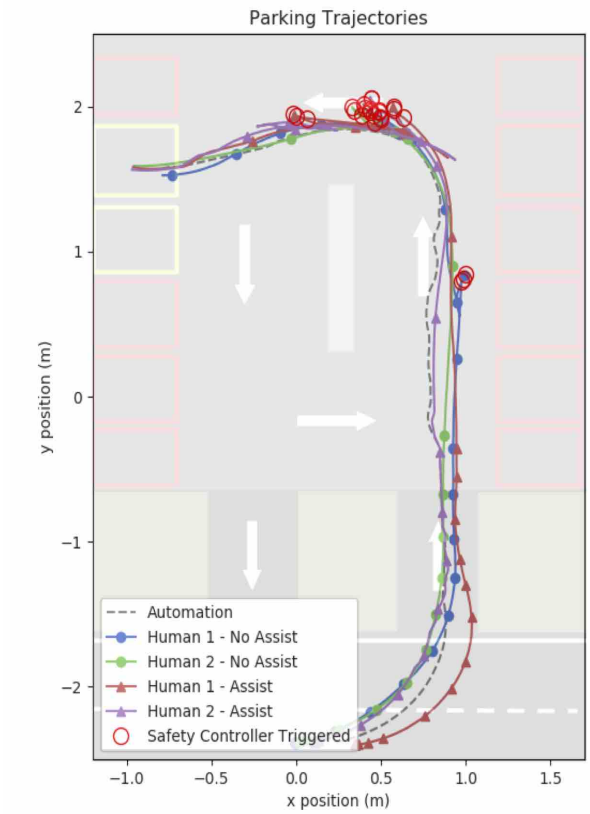

Fig. 7. The collective trajectories from our preliminary user study. We have included the trajectory of a fully automated vehicle completing the task to serve as a baseline for evaluating the human performance. Note, when looking at the trajectories, that the state of the vehicle corresponds to the center point of the rear axle of the vehicle.

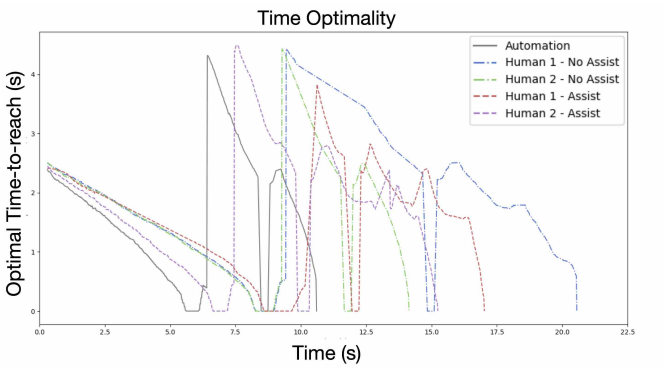

Fig. 8. We illustrate the change of the minimal TTR over all empty parking spots over time for each trial.

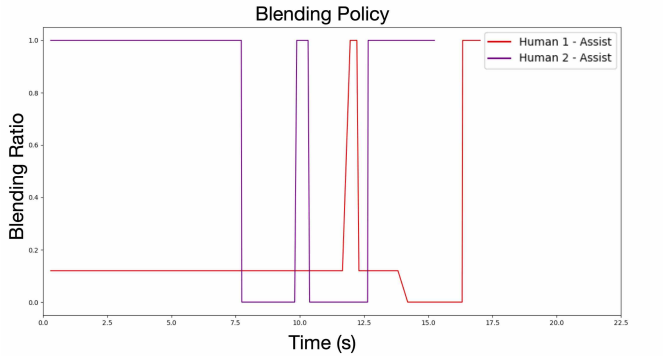

Fig. 9. We show the blending policy used by the two human users when the driving assistance is active.

Both humans were asked, on a scale from 1-7 whether the system is intuitive, feels safe, and whether the users felt in control. Human 1 responded with $\{4, 6, 5\}$ and Human 2 responded with $\{5, 6, 6\}$, respectively. These results are preliminary, but give initial indication that the system feels intuitive, safe, and controllable.

5. CONCLUSION

In order to allow the deployment of automated vehicles, we need to ensure they are performing tasks that are feasible and safe. In this paper, we present an approach for the planning and deployment of a safe remote teleoperation of a connected vehicle that may have varying levels of automation. We assume that the vehicle has the minimal capability of avoiding static obstacles and parking itself into a parking spot in a time-optimal fashion. Since, in the near future, vehicles will often require the situational awareness and exception-handling capabilities of human operators, we formulate our approach for remote teleoperation that gives the human operator decision-making freedom while guaranteeing the teleoperation task will still be satisfied. Furthermore, we present preliminary experiments validating the approach’s ability to ensure the completion of LTL-specified teleoperation tasks.

In future work, we will continue to conduct experiments on human users, to find a statistically significant blending policy preference or strategy. Furthermore, we will extend this framework to include more complex tasks and a richer inference approach for better estimating the human’s preference over the different driving tasks.

REFERENCES

Baier, C. and Katoen, J.P. (2008). *Principles of Model Checking*. MIT press.

Belta, C., Yordanov, B., and Gol, E.A. (2017). *Formal Methods for Discrete-Time Dynamical Systems*. Springer.

Brown, K., Driggs-Campbell, K., and Kochenderfer, M.J. (2020). Modeling and Prediction of Human Driver Behavior: A Survey.

Coddington, E.A. and Levinson, N. (1955). *Theory of ordinary differential equations*. McGraw-Hill.

Davies, A. (2017). Nissan's Path to Self-Driving Cars? Humans in Call Centers. *Wired Transportation*. URL https://www.wired.com/2017/01/nissans-self-driving-teleoperation/.

Dragan, A.D. and Srinivasa, S.S. (2013). A policy-blending formalism for shared control. In *International Journal of Robotics Research*. doi:10.1177/0278364913490324.

Einride (2019). World premiere: First cab-less and autonomous, fully electric truck in commercial operations on public road. URL https://www.einride.tech/news/world-premiere-first-cab-less-and-autonomous-fully-electric-truck-in-commercial-operations-on-public-road.

Einride (2020). Einride Will Hire Its First Remote Autonomous Truck Operator in 2020. URL https://www.einride.tech/news/einride-will-hire-its-first-remote-autonomous-truck-operator-in-2020.

Fernandez Fisac, J. (2019). *Game-Theoretic Safety Assurance for Human-Centered Robotic Systems*. Ph.D. thesis, UC Berkeley.

Gao, Y., Abate, A., Jiang, F.J., Giacobbe, M., Xie, L., and Johansson, K.H. (2020a). Temporal Logic Trees for Model Checking and Control Synthesis of Uncertain Discrete-time Systems. URL http://arxiv.org/abs/2007.02271.

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

Ghasemi, A.H., Jayakumar, P., and Gillespie, R.B. (2019). Shared control architectures for vehicle steering. *Cognition, Technology and Work*, 21(4), 699–709. doi:10.1007/s10111-019-00560-9. URL https://doi.org/10.1007/s10111-019-00560-9.

Gnatzig, S., Chucholowski, F., Tang, T., and Lienkamp, M. (2013). A system design for teleoperated road vehicles. In *ICINCO (2)*, 231–238.

Guo, M., Andersson, S., and Dimarogonas, D.V. (2018). Human-in-the-loop mixed-initiative control under temporal tasks. In *2018 IEEE International Conference on Robotics and Automation*, 6395–6400.

Hosseini, A. and Lienkamp, M. (2016). Predictive safety based on track-before-detect for teleoperated driving through communication time delay. In *2016 IEEE Intelligent Vehicles Symposium*, 165–172.

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

INRIX (2017). The impact of parking pain in the US, UK and Germany.

Javdani, S., Admoni, H., Pellegrinelli, S., Srinivasa, S.S., and Bagnell, J.A. (2018). Shared autonomy via hindsight optimization for teleoperation and teaming. *The International Journal of Robotics Research*, 37(7), 717–742.

Jeon, H.J., Losey, D.P., and Sadigh, D. (2020). Shared Autonomy with Learned Latent Actions. *arXiv preprint arXiv:2005.03210*.

Jiang, F.J., Gao, Y., Xie, L., and Johansson, K.H. (2020). Ensuring safety for linear temporal logic-specified vehicle parking tasks using Hamilton-Jacobi reachability analysis. In *IEEE 59th Conference on Decision and Control (to appear)*.

Koopman, P. and Wagner, M. (2017). Autonomous Vehicle Safety : An Interdisciplinary Challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1), 90–96.

Muslim, H. and Itoh, M. (2019). A theoretical framework for designing human-centered automotive automation systems. *Cognition, Technology and Work*, 21(4), 685–697.

Neumeier, S. and Facchi, C. (2019). Towards a driver support system for teleoperated driving. In *2019 IEEE Intelligent Transportation Systems Conference*, 4190–4196. IEEE.

Nobina (2019). AUTONOMOUS. URL https://www.nobina.com/en/nobina-technology/products/autonomous/.

SAE On-Road Automated Vehicle Standards Committee (2018). Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. Technical report, SAE International.