

On Infusing Reachability-Based Safety Assurance within Probabilistic Planning Frameworks for Human-Robot Vehicle Interactions

Journal Title
XX(X):1–16
©The Author(s) 2019
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/



Karen Leung¹, Edward Schmerling¹, Mengxuan Zhang¹, Mo Chen², John Talbot¹, J. Christian Gerdes¹, and Marco Pavone¹

Abstract

Action anticipation, intent prediction, and proactive behavior are all desirable characteristics for autonomous driving policies in interactive scenarios. Paramount, however, is ensuring safety on the road—a key challenge in doing so is accounting for uncertainty in human driver actions without unduly impacting planner performance. This paper introduces a minimally-interventional safety controller operating within an autonomous vehicle control stack with the role of ensuring collision-free interaction with an externally controlled (e.g., human-driven) counterpart while respecting static obstacles such as a road boundary wall. We leverage reachability analysis to construct a real-time (100Hz) controller that serves the dual role of (1) tracking an input trajectory from a higher-level planning algorithm using model predictive control, and (2) assuring safety through maintaining the availability of a collision-free escape maneuver as a persistent constraint regardless of whatever future actions the other car takes. A full-scale steer-by-wire platform is used to conduct traffic weaving experiments wherein two cars, initially side-by-side, must swap lanes in a limited amount of time and distance, emulating cars merging onto/off of a highway. We demonstrate that, with our control stack, the autonomous vehicle is able to avoid collision even when the other car defies the planner’s expectations and takes dangerous actions, either carelessly or with the intent to collide, and otherwise deviates minimally from the planned trajectory to the extent required to maintain safety.

Keywords

Probabilistic planning, safety-preserving controller, backward reachability analysis, vehicle model predictive control, human-robot interaction.

1 Introduction

Decision-making and control for mobile robots is typically stratified into levels. A high-level planner, informed by representative yet simplified dynamics of a robot and its environment, might be responsible for selecting an optimal, yet coarse trajectory plan, which is then implemented through a low-level controller that respects more accurate models of the robot’s dynamics and control constraints. While additional components may be required to flesh out a robot’s full control stack from model to motor commands, selecting the right “division of responsibilities” is fundamental to system design.

One consideration that defies clear classification, however, is how to ensure a mobile robot’s safety when operating in close proximity with a rapidly evolving and stochastic environment. Safety is a function of uncertainty in both the robot’s dynamics and those of its surroundings; high-level planners typically do not replan sufficiently rapidly to ensure split-second reactivity to threats, yet low-level controllers are typically too short-sighted to ensure safety beyond their local horizon.

Human-robot interactions are an unavoidable aspect of many modern robotic applications and ensuring safety for these interactions is critical, especially in applications such

as autonomous driving where collisions may lead to life-threatening injury. However, ensuring safety within the planning and control framework can be very challenging due to the uncertainty in how humans may behave. To quantify this uncertainty, robots often rely on generative models of human behavior in order to inform their planning algorithms Schmerling et al. (2018); Sadigh et al. (2016). This generally results in more efficient and communicative interactions since a robot can reason about the uncertainty to anticipate how a human may behave and leverage this to its advantage. In general, under nominal operating conditions that reflect the modeling assumptions, these model-based probabilistic planners can offer high performance (e.g., minimizing time and control effort, for both the robot and its counterparts). However, these planners alone are typically insufficient for ensuring absolute safety because (1) they depend on *probabilistic* models of human behavior and thus safety is not enforced deterministically, (2) dangerous but low-likelihood events may not be adequately captured in

¹Stanford University, Stanford, CA 94305, USA

²Simon Fraser University, Burnaby, BC V5A 1S6, Canada

Corresponding author:

Marco Pavone, 496 Lomita Mall, Rm. 261 Stanford, CA 94305

Email: pavone@stanford.edu

the human behavior prediction model, and (3) reasoning about these probabilistic behavior models is typically too computationally expensive for the planners to react in real time when humans strongly defy expectations and/or diverge from modeling assumptions.

In this work we implement a control stack for a full-scale autonomous car (the “robot”) engaging in close proximity interactions with a human-controlled vehicle (the “human”). Our control stack aims to stay true to planned trajectories from a high-level planner since freedom of motion is essential for the planner to carry out the driving task while conveying future intent to the other vehicle. At the same time, we allow the robot to deviate from the desired trajectory to the point that is necessary to maintain safety. Our primary tool for designing a controller that does not needlessly impinge upon the planner’s choices is *Hamilton-Jacobi (HJ) backward reachability*. We provide a brief overview of the reachability analysis literature relevant to our work in Section 3.

2 Related Work

For high-level planners, safety is often incentivized, but not strictly enforced as a hard constraint. For example, safety is often part of the objective function when selecting optimal plans, or represented via artificial potential fields Schmerling et al. (2018); Sadigh et al. (2016); Wolf and Burdick (2008) to encourage the selection of safer nominal trajectories. Although these approaches are designed to account for interactive scenarios in which another sentient agent is a key environmental consideration, they contain competing objectives (i.e., collision avoidance vs. goal-oriented performance), often do not plan at a sufficiently high rate to account for rapidly-changing environments, and ultimately cannot offer high confidence when it comes to ensuring safety.

Another common approach to finding collision-free plans is to use *forward* reachability. The idea is to fix a time horizon and compute the set of states where the other agents could possibly be in the future and plan trajectories for the robot that avoid this set Liu et al. (2017); Althoff and Dolan (2011). Although this gives a stronger sense of safety, this is only practical for short time horizons otherwise it will lead to overly conservative robot behaviors or even planning problem infeasibility as the set to avoid grows. In general, approaches that use forward reachability results in very conservative results, especially for interactive scenarios where freedom of motion (e.g., nudging) is necessary to convey intent. These methods, as well as methods that enforce safety as an objective, are often subjected to model simplification, such as using a linear dynamics model instead of a nonlinear one, or making simplifying assumptions for computational tractability. However, model simplifications can lead to overly conservative or imprecise results which may impede performance.

Safety can be introduced at the low-level which often obeys higher fidelity dynamics models. A common approach in ensuring safe low-level controls is to use reactive collision avoidance techniques—the robot is normally allowed to apply any control, but switches to an avoidance controller when near safety violation. Examples of this

general approach include HJ backward reachability-based controllers which have been applied in such a switched fashion Bajcsy et al. (2019); Chen and Tomlin (2018); Fisac et al. (2017) and have proven to be effective for avoiding other interacting agents and static obstacles. HJ reachability has been studied extensively and applied successfully in a variety of safety-critical interactive settings Bokanowski et al. (2010); Chen et al. (2017); Dhinakaran et al. (2017); Gattami et al. (2011); Margellos and Lygeros (2011); Mitchell et al. (2005) due to its flexibility with respect to system dynamics, and its optimal (i.e., non-overly-conservative) avoidance maneuvers stemming from its equivalence to an exhaustive search over joint system dynamics. Other approaches include using a precomputed emergency maneuver library Arora et al. (2015). A key drawback to switching controllers, however, is that the performance goals considered by the high-level planner are completely ignored when the reactive controller steps in. For some cases, this may be acceptable and necessary, but in general, it is desirable to ensure safety without unduly impacting planner performance where possible. Instead of switching to a different controller, Funke et al. (2017), Brown et al. (2017), and Gray et al. (2013) adapt the low-level online optimal controller to incorporate constraints that avoid static obstacles. The approach taken in Gray et al. (2013) aims to be minimally interventional, however, they only consider cases with disturbance uncertainty rather than environmental uncertainty stemming from the system interacting with a human. These approaches are able to strike the optimal balance between tracking performance and safety (i.e., optimizing performance subject to safety as a hard constraint), but these approaches as presented are effective for avoiding static obstacles only.

Inspired by the non-conservative nature and optimality of HJ reachability, and the performance of online optimal control for trajectory tracking, this work aims to infuse reachability-based safety assurance into the low-level controller such that when the robot is near safety violation, the robot is able to simultaneously maintain safety and follow the high-level plans without unduly impacting performance. To the best of our knowledge there has not been any work that explicitly addresses the integration of reachability-based safety controllers as a component within a robot’s control stack, i.e., with safety as a constraint upon a primary planning objective.

Statement of Contributions: The contributions of this paper are twofold. First, we propose a method for formally incorporating reachability-based safety within an existing optimization-based control framework. The main insight that enables our approach is the recognition that, near safety violation, the set of safety-preserving controls often contains more than just the optimal avoidance control. Instead of directly applying this optimal avoidance control when prompted by reachability considerations, as in a switching control approach, we quantify the set of safety-preserving controls and pass it to the broader control framework as a constraint. Our intent is to enable truly minimal intervention against the direction of a higher-level planner when evasive action is required. Second, we evaluate the benefits, performance, and trade-offs of this safe control methodology in the context of a probabilistic

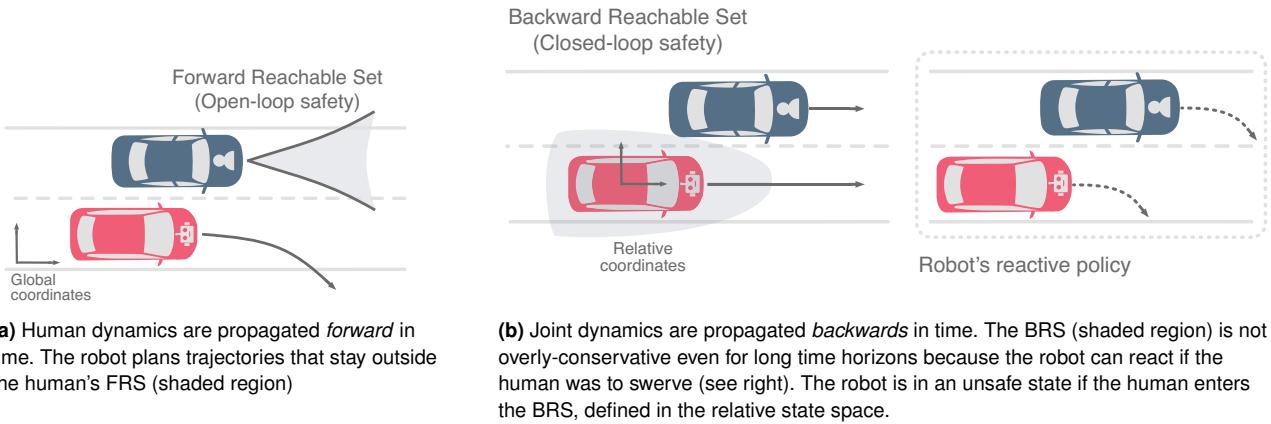


Figure 1. Illustration of forward and backward reachability, and how it is commonly used to ensure safe planning and controls.

planning framework for the traffic weaving scenario studied at a high level in Schmerling et al. (2018), wherein two cars, initially side-by-side, must swap lanes in a limited amount of time and distance. Experiments with a full-scale steer-by-wire vehicle reveal that our combined control stack achieves better safety than applying a tracking controller alone to the planner output, and smoother operation (with similar safety) compared with a switching control scheme; in our discussion we provide a roadmap towards improving the level of safety assurance in the face of practical considerations such as unmodeled dynamics, as well as towards generalizations of the basic traffic weaving scenario.

A preliminary version of this work appeared at the 2018 International Symposium on Experimental Robotics (ISER). In this revised and extended version, we provide the following additional contributions: (i) an extension to the control framework to account for static obstacles, (ii) more exposition on the vehicle dynamics models used and a deeper discussion on the underlying modeling assumptions, (iii) an empirical study of the safety-efficiency trade-off for our approach, and (iv) additional experimental results including trials with a static road boundary wall.

3 Background: Hamilton-Jacobi Reachability

In this section, we provide a brief overview of reachability analysis as a tool for constructing safe trajectory plans, and introduce HJ backward reachability concepts relevant to this work.

3.1 Overview: Reachability Analysis

Given dynamics governing a robotic system incorporating control and disturbance inputs, reachability analysis is the study of the set of states that the system can reach from its initial conditions. It is often used for formal verification as it can give guarantees on whether or not the evolution of the system will be safe, i.e., whether the reachable set includes undesirable outcomes. Reachability analysis can be divided into two main paradigms: (1) forward reachability and (2) backward reachability.

Forward Reachability Analysis: The *forward reachable set* (FRS) is the set of states that the system could potentially

be at after some time horizon T . This is computed by propagating the dynamics combined with all feasible control sequences and disturbances forward in time. When considering the interaction between two agents, for example a human and a robot, the forward reachable set is computed for the human (see Figure 1a) and the robot plans to avoid this set to ensure collision-free trajectories. This *open-loop* mentality however leads to an overly-conservative robot outlook. That is, while considering actions in the present, the robot does not incorporate the possibility that its future observations of where the human goes might influence how much it actually needs to take avoidance actions; in short the robot's plan to avoid the FRS does not incorporate closed-loop feedback. To reduce the over-conservative nature of forward reachability, the time horizon T for which the FRS is computed over is typically kept small and is recomputed frequently. This approach has been found to be effective in finding collision-free trajectories Althoff and Dolan (2014), but it is difficult to extend to interactive scenarios where there are more uncertainties in the rapidly changing environment. Aside from the overly-conservative nature of FRS, the key drawback with using forward reachability is that safety (i.e., avoiding the FRS) hinges on the planner's capabilities and operating frequency. Even if computing the FRS is instant, the planner may still be unable to react to split-second threats.

Backward Reachability Analysis: The *backward reachable set* (BRS) is the set of states that could result in the system being in the target set, assuming worst-case disturbances, after some time horizon T . In this work, the target set represents a set of undesirable states (e.g., collision states) and the BRS represents the set of states from which the controller cannot prevent the dynamics from being driven into the target set under the worst-case disturbance. As such, to rule out such an eventuality, the BRS is treated as the “*avoid set*”. Critical differences between the FRS and BRS are that (1) the BRS is computed *backwards* in time, and (2) the BRS is computed assuming *closed-loop* reactions to the disturbances. Illustrated in Figure 1b, for the case of relative dynamics between human- and robot-controlled vehicles, computation of the BRS takes into account the fact that the robot is able to react to the human at any time and in any state configuration (if the human was to swerve into the robot, the robot can swerve too to avoid a collision).

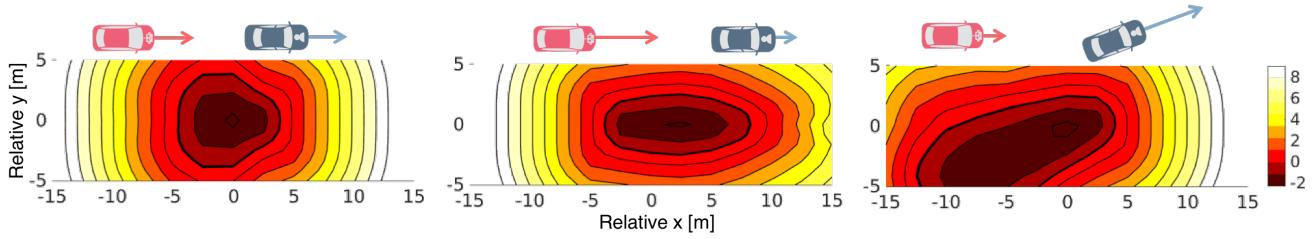


Figure 2. Contour plots of the HJI value function for the relative dynamics (8) projected on the relative position coordinates; slices show V as a function of relative x and y position with all other (velocity) states held fixed. The thick black line represents the zero-level set (i.e., the avoid set). Left: The pear-shaped BRS stems from the fact the robot car can swerve its front more rapidly than its rear. Middle: If the robot car is traveling faster, it is unsafe for the robot car to be in the region behind the human car because a collision may be unavoidable if the human brakes abruptly. Right: If the human car is traveling faster (in this case at an angle), it is unsafe for the robot to be in the region in front of the human car because the human can speed up and catch up to the robot car.

This leads to the BRS being less overly-conservative than the FRS. The conservatism in this case is in assuming the worst-case actions by the human but knowing that the robot can react to the human. The results of the BRS computation are cached, and at run-time, the optimal robot policy can be computed via a near-instant lookup of the reachability cache. As such, we can always compute optimal actions for the robot at any state configuration and *regardless* of the high-level planner used.

We elect to use backward reachability because (1) its non-overly conservative nature stemming from the closed-loop computation ensures that safe controls will be used only when necessary and not unduly impact planner performance, (2) safety is defined intrinsically in the BRS computation unlike FRS-based approaches where safety depends on the planned trajectory, and (3) it provides a computational handle on safe controls which can be evaluated at high operating frequencies to react to split-second threats.

Moreover, we will compute the BRS using Hamilton-Jacobi (HJ) reachability analysis, a particular approach to computing reachable sets. There are many existing approaches Greenstreet and Mitchell (1998); Frehse et al. (2011); Kurzhanski and Varaiya (2000); Althoff and Krogh (2014); Majumdar et al. (2014), but there is always a trade-off between modeling assumptions, scalability, and representation fidelity (i.e., whether the method computes over- or under-approximations of the reachable set). Compared to alternatives approaches, HJ reachability is the most computationally expensive, but it is able to compute the BRS exactly for any general nonlinear dynamics with control and disturbance inputs because it essentially uses a brute force computation via dynamic programming. Despite the apparent computational drawbacks, we note that the BRS may be computed off-line, and requires only a near-instant lookup during runtime, allowing it to be used in controllers or planners that run at a very high operational frequency.

3.2 Hamilton-Jacobi Backward Reachability Analysis

We briefly review relevant HJ backward reachability definitions for the remainder of this section; see Chen and Tomlin (2018) for a more in-depth treatment. HJ reachability casts the reachability problem as an optimal control problem and thus computing the reachable set is equivalent to

solving the Hamilton-Jacobi-Isaacs (HJI) partial differential equation.

The general HJ reachability formulation is as follows. Let the system dynamics be given by $\dot{x} = f(x, u, d)$ where $x \in \mathbb{R}^n$ is the state, $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control, and $d \in \mathcal{D} \subset \mathbb{R}^p$ is the disturbance. The system dynamics $f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \rightarrow \mathbb{R}^n$ are assumed to be uniformly continuous, bounded, and Lipschitz continuous in x for a fixed u and d . Let $\mathcal{T} \subseteq \mathbb{R}^n$ be the target set that the system wants to avoid at the end of a time horizon $|t|$ (note that $t < 0$ when propagating backwards in time). For collision avoidance, \mathcal{T} typically represents the set of states that are in collision with an obstacle.

In the context of human-robot interactions, $f(\cdot)$ describes the *relative dynamics* between the human and the robot (denoted by $f_{\mathcal{R}}(\cdot)$ where the subscript \mathcal{R} indicates the relative human-robot system), u corresponds to the robot's controls, and d corresponds to the human's controls since the human actions are treated as disturbance inputs. More concretely, let $(x_{\mathcal{R}}, u_{\mathcal{R}})$ represent the robot state and control, $(x_{\mathcal{H}}, u_{\mathcal{H}})$ represent the human state and control, and $x_{\mathcal{R}}$ be the relative state between the human and the robot. Thus the relative dynamics of the robot and human are given by $\dot{x}_{\mathcal{R}} = f_{\mathcal{R}}(x_{\mathcal{R}}, u_{\mathcal{R}}, u_{\mathcal{H}})$, and \mathcal{T} represents the set of relative states corresponding to when the human and robot are in collision. The formal definition of the BRS, denoted by $\mathcal{A}(t)$, for the human-robot relative system is

$$\begin{aligned} \mathcal{A}(t) := \{ \bar{x}_{\mathcal{R}} \in \mathbb{R}^n : \exists u_{\mathcal{H}}(\cdot), \forall u_{\mathcal{R}}(\cdot), \exists s \in [t, 0], \\ x_{\mathcal{R}}(t) = \bar{x}_{\mathcal{R}} \wedge \dot{x}_{\mathcal{R}} = f_{\mathcal{R}}(x_{\mathcal{R}}, u_{\mathcal{R}}, u_{\mathcal{H}}) \wedge x_{\mathcal{R}}(s) \in \mathcal{T} \}. \end{aligned} \quad (1)$$

$\mathcal{A}(t)$ represents the set of “avoid states” at time t from which if the human followed an adversarial *policy* $u_{\mathcal{H}}(\cdot)$, any robot *policy* $u_{\mathcal{R}}(\cdot)$ would lead to the relative state trajectory $x_{\mathcal{R}}(\cdot)$ being inside \mathcal{T} within a time horizon $|t|$. Assuming optimal human actions, $\mathcal{A}(t)$ can be computed by defining a value function $V(t, x_{\mathcal{R}})$ which obeys the HJI partial differential equation Mitchell et al. (2005); Fisac et al. (2015); the solution $V(t, x_{\mathcal{R}})$ gives the BRS as its zero sublevel set:

$$\mathcal{A}(t) = \{ x_{\mathcal{R}} : V(t, x_{\mathcal{R}}) \leq 0 \}.$$

The HJI PDE is solved starting from the boundary condition $V(0, x_{\mathcal{R}})$, the sign of which reflects set membership of $x_{\mathcal{R}}$ in \mathcal{T} .*

*See Section 8 for discussion of specific choices of $V(0, x_{\mathcal{R}})$.

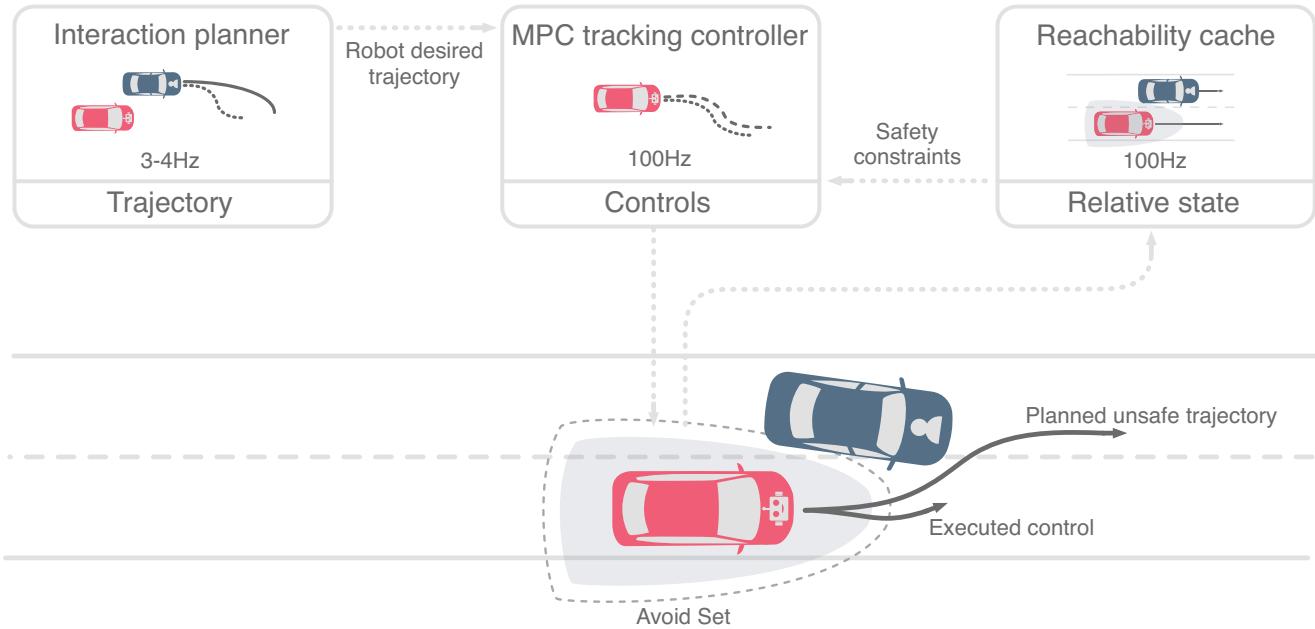


Figure 3. Decision-making and control stack for human-robot pairwise vehicle interactions. Our contribution in this work is the integration of safety-ensuring control constraints, derived from a HJ backward reachable set computed and cached offline, into a model predictive controller’s tracking optimization problem. A high-level interaction planner produces nominal trajectories for the robot car and the low-level safe tracking controller executes controls that minimally deviate from the planner’s choice if the vehicles approach the set of unsafe relative states.

For the case of the vehicle-vehicle interactions investigated in this work, when the control and maximum velocity capabilities of the human car are no greater than those of the robot car, one can take the limit $t \rightarrow -\infty$ and obtain the infinite time horizon BRS \mathcal{A}_∞ with corresponding value function $V_\infty(x_r)$.[†] Intuitively, this prescribed equal control authority ensures that if the human and robot start sufficiently far apart, then the human will never be able to “catch” the robot. This holds even if the human car may have transient maneuverability advantages over the robot as we assume later in this work. That is, we expect that the BRS will not encompass the entire state space as $t \rightarrow -\infty$, and in practice we compute the BRS over a sufficiently large finite time horizon to the point where it appears that the BRS has converged. We recognize that we make a strong modeling assumption in enabling this computation. In reality, the human and robot car may have very different control authority (e.g., different engines resulting in different acceleration and velocity capabilities) and the infinite BRS may not be bounded. In such cases, practitioners may compute the BRS over a horizon suitable for the interaction where guarantees afforded by HJ reachability only hold over that time horizon. Illustrative slices of the value function and the BRS for the vehicle-vehicle relative dynamics with equal control and velocity capabilities considered in this work are shown in Figure 2.

We can compute the optimal robot avoidance control

$$u_R^* = \arg \max_{u_R} \min_{u_H} \nabla V(x_r) \cdot f_r(x_r, u_R, u_H) \quad (2)$$

which offers the greatest increase in $V(x_r)$ assuming optimal (worst-case) actions by the human. Typical applications of HJI solutions have switched to this optimal control when near the boundary of the BRS Bajcsy et al.

(2019); Fisac et al. (2017), i.e., when safety is nearly violated.

In an interactive scenario where, for example, we may want to let a robot planner convey intent by nudging towards the human car to the extent that is safe, we prefer a less extreme control strategy. In the next section, we describe in detail how to infuse reachability-based safety assurance within a multi-tiered control framework that consists of different planning objectives.

4 Control Stack Architecture

In this section, we propose using a safety-preserving HJI controller, rather than switching to the optimal HJI controller (2), and describe how to incorporate it within an existing control stack—a high-level planner feeding desired trajectories to a low-level tracking controller—to enable safe human-robot interactions that minimally impinges on the high-level planning performance objective. The control stack architecture is illustrated in Figure 3. The proposed control stack is applicable to general human-robot interactions (e.g., an autonomous car interacting with a human-driven car, an autonomous manipulator arm working alongside a human, wheeled mobile robots navigating a crowded sidewalk). However, in this work, we focus on the traffic-weaving scenario, wherein two cars initially side-by-side must swap lanes in a limited amount of time and distance, because it is a representative interactive scenario that encapsulates many challenging characteristics inherent to human-robot interaction. Successful and smooth traffic-weaves rely on action anticipation, intent prediction, and proactive behavior

[†]For ease of notation going forward we will often write $V := V_\infty$.

from each vehicle, and ensuring safety is critical because collisions may lead to life-threatening injury.

4.1 Interaction Planner

Our proposed control framework is agnostic to the interaction planner used. The only assumption for the planner is that it outputs desired trajectories (which presumably reflect high-performance goal-oriented nominal behavior) for the robot car to track. In general, high-level planners often optimize objectives that weigh safety considerations (e.g., distance between cars) relative to other concerns (e.g., control effort), and typical to human-robot interactive scenarios, they may reason anticipatively with respect to a *probabilistic* interaction dynamics model. That is, although the planner is encouraged to select safer plans, safety is not enforced as a deterministic constraint at the planning level.

In this work, we use the traffic-weaving interaction planner from Schmerling et al. (2018). It uses a predictive model of future human behavior to select desired trajectories for the robot car to follow, updated at $\sim 3\text{Hz}$. We extend this work by using a hindsight optimization policy Yoon et al. (2008) instead of the limited-lookahead action policy in order to encourage more information-seeking actions from the robot.

4.2 MPC Tracking Controller

Given a desired trajectory from the interaction planner, the low-level tracking controller uses model predictive control (MPC) for computing optimal controls to track the desired trajectory. We assume that the outputs of the low-level tracking controller are used directly by the robot's actuators (e.g., steering and longitudinal force commands for a vehicle, torque commands for each joint on a manipulator arm). As such, the MPC tracking controller typically uses a more accurate dynamics model and operates at a much higher frequency ($\sim 100\text{Hz}$) than the interaction planner. To ensure safety with respect to a dynamic obstacle (i.e., a human-driven vehicle) we incorporate additional constraints computed from HJ reachability analysis into the MPC problem. These constraints are designed to ensure that at each control step the robot car does not enter an unsafe set of relative states that may lead to collision.

For vehicle trajectory tracking, we adapt the real-time MPC tracking controller from Brown et al. (2017) by modifying it to include an additional invariant set constraint detailed in the next section. This MPC tracking controller, operating at 100Hz, computes optimal controls to track a desired trajectory by solving an optimization problem at each iteration. The optimization problem is based on a single track vehicle model (also known as the bicycle model) and incorporates friction and stability control constraints (in addition to control and state constraints) while minimizing a combination of tracking error and control derivatives. A more in-depth treatment of this combined MPC and HJI controller is given in Section 6.

4.3 Safety-Preserving HJI Control

Rather than switching to the optimal avoidance controller (2) when nearing safety violation, we propose adding containment in the *set of safety-preserving controls* as

an additional constraint to the low-level MPC tracking controller. The set of safety-preserving controls

$$\mathcal{U}_R(x_R) = \{u_R : \min_{u_H} \nabla V(x_R) \cdot f_R(x_R, u_R, u_H) \geq 0\} \quad (3)$$

represent the set of robot controls that ensure the value function is nondecreasing. This constraint can be computed online since the BRS is computed offline and the value function V and its gradient ∇V are cached. Online we employ a safety buffer $\epsilon > 0$ so that when the condition $V(x_R) \leq \epsilon$ holds, indicating that the robot is nearing safety violation, we add the constraint $u_R \in \mathcal{U}_R(x_R)$ to the list of tracking controller constraints. By adding this additional safety-preserving constraint when near safety violation, the MPC tracking controller selects control actions that prevent the robot from further violating the safety threshold while simultaneously optimizing for tracking performance. This results in a *minimally interventional safety controller*—the MPC tracking controller will only minimally deviate from the desired trajectory to the extent necessary to maintain safety for the robot car.

For the traffic-weaving scenario investigated in this paper, the MPC problem for vehicle trajectory tracking Brown et al. (2017) is formulated as a quadratic program (QP) (to enable fast solve time amenable to a 100Hz operating frequency) and hence requires the constraints to be linear. As such, we instead apply the constraint $u_R \in \tilde{\mathcal{U}}_R(x_R)$ where

$$\tilde{\mathcal{U}}_R(x_R) = \{u_R : M_{HJI} \cdot u_R + b_{HJI} \geq 0\} \quad (4)$$

is a linearized approximation of $\mathcal{U}_R(x_R)$. Specifically, for the current relative state \tilde{x}_R , current robot control \tilde{u}_R , and optimal, i.e., worst-case, human action defined analogously to (2) u_H^* , the terms in the linearization are

$$M_{HJI} = \nabla V \cdot \frac{\partial f_R}{\partial u_R}(\tilde{x}_R, u_H^*, \tilde{u}_R),$$

$$b_{HJI} = \nabla V \cdot f_R(\tilde{x}_R, u_H^*, \tilde{u}_R) - M_{HJI} \cdot \tilde{u}_R.$$

In general, $\mathcal{U}_R(x_R)$ may not be a half-plane, leading to the linear approximation $\tilde{\mathcal{U}}_R(x_R)$ including controls outside of $\mathcal{U}_R(x_R)$. However, since we bound the change in the control inputs across each time step, feasible controls remain close to the linearization point where the approximation error is small.

5 Dynamics

In this section, we detail the vehicle dynamics model used to model the robot and human car, the relative dynamics model between the human and the robot necessary for computing the BRS, and the tracking dynamics used for the MPC tracking controller. We use a six-state nonlinear single track model to describe the robot car's dynamics and assume the human car obeys a four-state dynamically extended unicycle model with longitudinal acceleration and yaw rate as control inputs. As a result, the relative dynamics model has seven states. This represents a compromise between model fidelity and the number of state dimensions in the relative dynamics; reducing the latter is essential since solving the HJI PDE suffers greatly from the curse of dimensionality. The computation becomes notoriously expensive past five

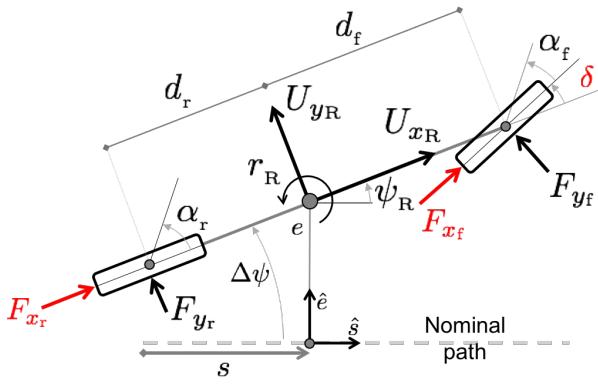


Figure 4. Schematic of the single track model (bicycle model) tracking a path. This is the dynamics model used to describe the robot vehicle.

or more state dimensions without compromising on grid discretization or employing some decoupling strategy.

5.1 Robot Vehicle Dynamics

The robot car (denoted by subscript R) will be modeled using the single track vehicle model illustrated in Figure 4. Let (p_{x_R}, p_{y_R}) be the position of the robot car's center of mass defined in an inertial reference frame and ψ_R be the yaw angle (heading) of the robot car relative to the horizontal axis. U_{x_R} and U_{y_R} are the velocities in the robot car's body frame, and r_R is the yaw rate. The state for the single track model is $x_R = [p_{x_R} \ p_{y_R} \ \psi_R \ U_{x_R} \ U_{y_R} \ r_R]^T$. The control input $u_R = [\delta \ F_x]^T$ consists of the steering command δ and longitudinal tire force F_x which is distributed between the front and rear tires $F_x = F_{x_f} + F_{x_r}$ via a fixed mapping. Assuming a quadratic model of longitudinal drag force ($F_{x_{\text{drag}}} = -C_{d_0} - C_{d_1}U_{x_R} - C_{d_2}U_{x_R}^2$) and for vehicle mass (m) and moment of inertia (I_{zz}), and the distances from the center of mass to the front and rear axles (d_f, d_r), the equations of motion for the robot car are

$$\dot{x}_R = \begin{bmatrix} U_{x_R} \cos \psi_R - U_{y_R} \sin \psi_R \\ U_{x_R} \sin \psi_R + U_{y_R} \cos \psi_R \\ r \\ \frac{1}{m}(F_{x_f} \cos \delta - F_{y_f} \sin \delta + F_{x_r} + F_{x_{\text{drag}}}) + r_R U_{y_R} \\ \frac{1}{m}(F_{y_f} \cos \delta + F_{y_r} + F_{x_f} \sin \delta) - r_R U_{x_R} \\ \frac{1}{I_{zz}}(d_f F_{y_f} \cos \delta + d_f F_{x_f} \sin \delta - d_r F_{y_r}) \end{bmatrix} \quad (5)$$

The controls are assumed to be limited by the steering system, friction limits, and power capacity of the vehicle. Using the brush coupled tire model by Pacejka (2002), the lateral tire force F_{y_i} and F_{y_r} at the front and rear tires is a function of slip angle (α_f, α_r), tire cornering stiffness ($C_{\alpha_f}, C_{\alpha_r}$), longitudinal tire forces (F_{x_f}, F_{x_r}), coefficient of friction (μ), and normal tire forces (F_{x_f}, F_{z_i}). As such, the lateral tire force for either the front or rear tires (denoted by i) is

$$F_{y_i} = \begin{cases} 0 & \text{if } F_{x_i} > \mu F_{z_i} \\ -C_{\alpha_i} \tan \alpha_i (1 - \gamma + \frac{1}{3}\gamma^2) & \text{if } \gamma < 1 \\ -F_{y_{\max}} \text{sign}(\tan \alpha_i) & \text{if } \gamma \geq 1 \end{cases} \quad (6)$$

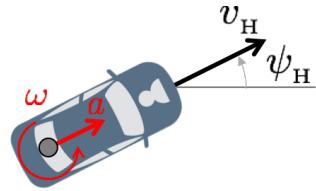


Figure 5. Schematic of the dynamically-extended unicycle model. This is the dynamics model used to describe the human vehicle.

where $\gamma = \left| \frac{C_{\alpha_i} \tan \alpha_i}{3F_{y_{\max}}} \right|$ and $F_{y_{\max}} = \sqrt{\mu^2 F_{z_i}^2 - F_{x_i}^2}$. The slip angles and normal forces (accounting for weight transfer due to F_{x_i}) for the front and rear tires can be computed using the following equations

$$\alpha_f = \tan^{-1} \frac{U_{y_R} + d_f r_R}{U_{x_R}} - \delta, \quad \alpha_r = \tan^{-1} \frac{U_{y_R} - d_r r_R}{U_{x_R}},$$

$$F_{z_f} = \frac{mgd_r - h\tilde{F}_x}{L}, \quad F_{z_r} = \frac{mgd_f + h\tilde{F}_x}{L},$$

where h is the distance from the center of mass to the ground and $\tilde{F}_x = F_{x_f} \cos \delta - F_{y_f} \sin \delta + F_{x_r}$ is the total longitudinal force in the vehicle's body frame.

5.2 Human Vehicle Dynamics

The human car (denoted by subscript H) will be modeled using the dynamically extended unicycle model illustrated in Figure 5. Let (p_{x_H}, p_{y_H}) be the position of the center of the human car's rear axle defined in an inertial reference frame and ψ_H be the yaw angle (heading) of the human car relative to the horizontal axis. The velocity of the human car in the vehicle frame is v_H . The state for the dynamically extended unicycle model is $x_H = [p_{x_H} \ p_{y_H} \ \psi_H \ v_H]^T$. The control input $u_H = [\omega \ a]^T$ consists of the yaw rate ω and longitudinal acceleration a . The control limits of the human car are chosen such that the robot and human car share the same power, steering, and friction limits. The equations of motion for the human car are

$$\dot{x}_H = \begin{bmatrix} v_H \cos \psi_H \\ v_H \sin \psi_H \\ \omega \\ a \end{bmatrix}. \quad (7)$$

Due to its simpler dynamics representation, the human car has a transient advantage in control authority over the robot car (it may change its path curvature discontinuously, while the robot may not), but by equating the steady-state control limits we ensure that the infinite time horizon BRS computation converges.

5.3 Relative Dynamics

The relative state (denoted by subscript \mathcal{R}) between the robot car and human car is defined with respect to a coordinate system centered on and aligned with the robot car's vehicle frame. We define the relative position (p_{x_R}, p_{y_R}) as

$$\begin{bmatrix} p_{x_R} \\ p_{y_R} \end{bmatrix} = \begin{bmatrix} \cos \psi_R & \sin \psi_R \\ -\sin \psi_R & \cos \psi_R \end{bmatrix} \begin{bmatrix} p_{x_H} - p_{x_R} \\ p_{y_H} - p_{y_R} \end{bmatrix},$$

and the relative heading $\psi_{\mathcal{R}}$ as $\psi_{\mathcal{R}} = \psi_{\text{H}} - \psi_{\text{R}}$.

Since the velocity states are defined with respect to the vehicle frame, we cannot define analogous relative velocity states and must include the individual velocity states of each vehicle. As such, the relative state for the human-robot vehicle system is $x_{\mathcal{R}} = [p_{x_{\mathcal{R}}} \ p_{y_{\mathcal{R}}} \ \psi_{\mathcal{R}} \ U_{x_{\mathcal{R}}} \ U_{y_{\mathcal{R}}} \ v_{\text{H}} \ r_{\text{R}}]^T$. In the language of HJ reachability, the disturbance input of the system is the human car's control $d = u_{\text{H}} = [\omega \ a]^T$ and the control input is the robot car's control $u = u_{\text{R}} = [\delta \ F_x]^T$. Combining (5) and (7), the equations of motion for the relative system are

$$\dot{x}_{\mathcal{R}} = \begin{bmatrix} v_{\text{H}} \cos \psi_{\mathcal{R}} - U_{x_{\mathcal{R}}} + p_{y_{\mathcal{R}}} r_{\text{R}} \\ v_{\text{H}} \sin \psi_{\mathcal{R}} - U_{y_{\mathcal{R}}} - p_{x_{\mathcal{R}}} r_{\text{R}} \\ \omega - r_{\text{R}} \\ \frac{1}{m}(F_{x_f} \cos \delta - F_{y_f} \sin \delta + F_{x_r} + F_{x_{\text{drag}}}) + r_{\text{R}} U_{y_{\mathcal{R}}} \\ \frac{1}{m}(F_{y_f} \cos \delta + F_{y_r} + F_{x_f} \sin \delta) - r_{\text{R}} U_{x_{\mathcal{R}}} \\ a \\ \frac{1}{I_{zz}}(d_f F_{y_f} \cos \delta + d_f F_{x_f} \sin \delta - d_r F_{y_r}) \end{bmatrix}. \quad (8)$$

5.4 Tracking Dynamics

The tracking MPC controller relies on an error dynamics model. Define a path (see Figure 4) through space where s is the distance along the path and at any distance s , we know the path heading $\psi(s)$, the curvature $\kappa(s)$, and a coordinate system (\hat{s}, \hat{e}) tangential and normal to the path. Given the position and heading of the robot car, we can project the car to the closest point on the path. Let the lateral error e be the lateral distance along direction \hat{e} and $\Delta\psi = \psi_{\text{R}} - \psi_{\text{path}}$ be the robot car's heading relative to the path computed from this closest point. Using this projection and the robot car's dynamics from (5), we can compute the error dynamics relative to the desired path. Using the same notation defined previously in (5), the state for the robot car tracking a desired path is $\hat{x} = [s \ U_{x_{\mathcal{R}}} \ U_{y_{\mathcal{R}}} \ r_{\text{R}} \ \Delta\psi \ e]^T$. The tracking dynamics are

$$\dot{\hat{x}} = \begin{bmatrix} U_{x_{\mathcal{R}}} \cos \Delta\psi - U_{y_{\mathcal{R}}} \sin \Delta\psi \\ \frac{1}{m}(F_{x_f} \cos \delta - F_{y_f} \sin \delta + F_{x_r} + F_{x_d}) + r_{\text{R}} U_{y_{\mathcal{R}}} \\ \frac{1}{m}(F_{y_f} \cos \delta + F_{y_r} + F_{x_f} \sin \delta) - r_{\text{R}} U_{x_{\mathcal{R}}} \\ \frac{1}{I_{zz}}(\bar{a}F_{y_f} \cos \delta + \bar{a}F_{x_f} \sin \delta - \bar{b}F_{y_r}) \\ r - (U_{x_{\mathcal{R}}} \cos \Delta\psi - U_{y_{\mathcal{R}}} \sin \Delta\psi) \kappa(s) \\ U_{x_{\mathcal{R}}} \sin \Delta\psi + U_{y_{\mathcal{R}}} \cos \Delta\psi \end{bmatrix}. \quad (9)$$

6 The MPC+HJI Trajectory Tracking Controller

In this section, we detail the optimization problem used in the MPC+HJI tracking controller, describe how this problem can be adapted to accommodate collision avoidance for static obstacles, and provide numerical details about the BRS computation used in this formulation. Central to an MPC controller is an optimization problem; at each time step, the controller solves an optimization problem to find an optimal control sequence, passes the first control input to the actuator, and then repeats this process. To be amenable to real-time

applications, the optimization problem requires a fast solve time (~ 0.01 s). In this work, the MPC tracking problem is formulated as a convex optimization problem, namely a QP, enabling the use of efficient solvers Stellato et al. (2017); Mattingley and Boyd (2012) which are capable of solving the QP within the tight operating frequency.

6.1 Optimization Problem

Both the trajectory tracking objective and safety-preserving control constraint rely on optimizing over the robot steering and longitudinal force inputs simultaneously. Let $q_k = [\Delta s_k \ U_{x_{R,k}} \ U_{y_{R,k}} \ r_{R,k} \ \Delta\psi_k \ e_k]^T$ be the state of the robot car with respect to a nominal trajectory at discrete time step k . Δs_k , e_k and $\Delta\psi_k$ denote longitudinal, lateral, and heading error; $U_{x_{R,k}}$, $U_{y_{R,k}}$, and $r_{R,k}$ are body-frame longitudinal and lateral velocity, and yaw rate respectively as defined in Figure (4). Let $u_k = [\delta_k \ F_{x,k}]^T$ be the controls at step k and let $A_k q_k + B_k^- u_k + B_k^+ u_{k+1} + c_k = q_{k+1}$ denote linearized first-order-hold dynamics of (9). We adopt the varying time steps method and stable handling envelope constraint from Brown et al. (2017). To ensure the existence of a feasible solution, we use slack variables $\sigma_{\beta,k}$, $\sigma_{r,k}$, and $\sigma_{\text{HJI},k}$ on the stability and HJI constraints. The HJI reachability constraint $M_{\text{HJI}} u_k + b_{\text{HJI}} \geq -\sigma_{\text{HJI}}$ is activated only when $V(x_{\mathcal{R}}) \leq \epsilon$. Although HJI theory suggests that applying this constraint on the next action alone is sufficient, we apply it over the next $T_{\text{HJI}} = 3$ timesteps (30ms lookahead) to account for the approximations inherent in our QP formulation. The MPC tracking problem is a quadratic program (QP) of the form

$$\begin{aligned} & \underset{q, u, \sigma, \sigma_{\text{HJI}}, \Delta\delta, \Delta F_x}{\text{minimize}} \sum_{k=1}^T \Delta s_k^T Q_{\Delta s} \Delta s_k + \Delta\psi_k^T Q_{\Delta\psi} \Delta\psi_k + \\ & \quad e_k^T Q_e e_k + \Delta\delta_k^T R_{\Delta\delta} \Delta\delta_k + \\ & \quad \Delta F_{x,k}^T R_{\Delta F_x} \Delta F_{x,k} + \\ & \quad W_{\beta} \sigma_{\beta,k} + W_r \sigma_{r,k} + W_{\text{HJI}} \sigma_{\text{HJI},k} \\ & \text{subject to } q_1 = q_{\text{curr}}, \quad u_1 = u_{\text{curr}}, \\ & \quad \delta_{k+1} - \delta_k = \Delta\delta_k, \\ & \quad \Delta\delta_{\min} \leq \Delta\delta_k \leq \Delta\delta_{\max}, \\ & \quad \delta_{\min} \leq \delta_k \leq \delta_{\max}, \\ & \quad F_{x,k+1} - F_{x,k} = \Delta F_{x,k}, \\ & \quad F_{x,\min} \leq F_{x,k} \leq F_{x,\max}, \\ & \quad V_{\min} \leq U_{x_{R,k}} \leq V_{\max}, \\ & \quad \sigma_{1,k} \geq 0, \quad \sigma_{2,k} \geq 0, \\ & \quad \sigma_{\text{HJI},j} \geq 0 \quad (\text{if } V(x_{\mathcal{R}}) \leq \epsilon), \\ & \quad A_k q_k + B_k^- u_k + B_k^+ u_{k+1} + c_k = q_{k+1}, \\ & \quad H_k \begin{bmatrix} U_{y,k} \\ r_k \end{bmatrix} - G_k \leq \begin{bmatrix} \sigma_{\beta,k} \\ \sigma_{r,k} \end{bmatrix}, \\ & \quad M_{\text{HJI}} u_j + b_{\text{HJI}} \geq -\sigma_{\text{HJI}} \quad (\text{if } V(x_{\mathcal{R}}) \leq \epsilon), \\ & \quad \text{for } j = 1, \dots, T_{\text{HJI}}, \quad k = 1, \dots, T. \end{aligned} \quad (10)$$

The objective strives to minimize a combination of tracking error (longitudinal, lateral and angular), control rates (steering and longitudinal tire forces), and magnitude of the slack variables. The constraints ensure (1) continuity

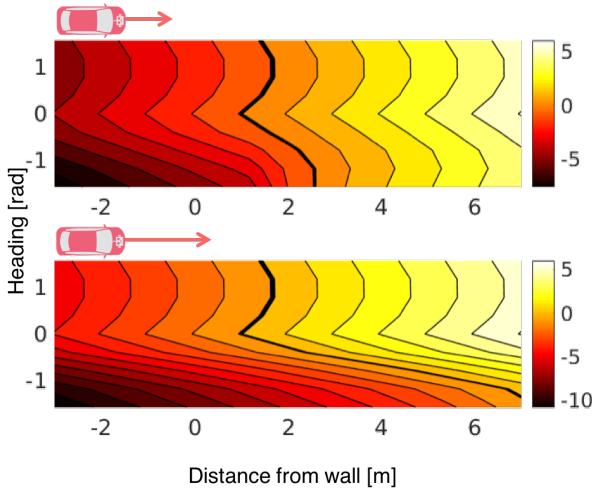


Figure 6. The BRS of the robot-wall system where zero heading corresponds to the robot car being parallel to the wall. In order to avoid a collision, the robot needs to be further away from the wall if it is traveling faster and headed towards the wall.

with the current and next state and control, (2) the change in controls across each time step is bounded, (3) the positivity of slack variables, (4) the (linearized) dynamics are satisfied, (5) the vehicle stability constraints are satisfied, and (6) the HJI safety-preserving half-plane constraint is satisfied when $V(x_{\mathcal{R}}) \leq \epsilon$.

The time-discretization and linearizations (dynamics and constraints) we apply amount to an approximate variant of sequential quadratic programming (SQP). In particular, however, we solve one QP at each MPC step rather than the usual iteration until convergence. Since the tracking problems are so similar from one MPC step to the next, we find that this approach yields sufficient performance for our purposes. We interpolate along each solution trajectory to compute the linearization nodes for the QP at the next MPC step.

We use the `ForwardDiff.jl` automatic differentiation (AD) package implemented in the Julia programming language [Revels et al. \(2016\)](#) to linearize the trajectory tracking dynamics as well as the HJI relative dynamics for the safety-preserving constraint. We call the Operator Splitting Quadratic Program (OSQP) solver [Stellato et al. \(2017\)](#) through the `Parametron.jl` modeling framework [Kooleen and contributors](#); this combination of software enables us to solve the following MPC optimization problem at 100Hz. The MPC code, including optimization parameters and vehicle parameters, can be found here: <https://github.com/StanfordASL/Pigeon.jl>.

6.2 Adding Static Obstacles

The current formulation does not prevent the robot from driving completely off the road in order to avoid the human. In more realistic road settings, there could be environmental constraints such as a concrete road boundary. In the same way that the HJ reachability-based safety is formulated as an additional constraint to the MPC problem, we can add another safety constraint to account for a static wall to prevent the robot from swerving completely off the road. We consider two approaches for deriving this constraint: the

first based on an additional HJI computation, and the second treating the wall as a static obstacle with associated state constraints.

We can compute a value function $V_{\text{WALL}}(x_{\mathcal{R}})$ describing the interaction between the robot and the wall by using (5) without the $p_{x_{\mathcal{R}}}$ state (we only care about the distance from the wall and not how far along the wall the robot is). Two slices of the robot-wall BRS projected on the $p_{y_{\mathcal{R}}} - \psi_{\mathcal{R}}$ plane are illustrated in Figure 6. Note that since the wall is static, there is no disturbance input. The HJI-wall constraint $M_{\text{WALL}} u_{\mathcal{R}} + b_{\text{WALL}} \geq 0$ becomes active when $V_{\text{WALL}}(x_{\mathcal{R}}) \leq \epsilon$. This means that it is possible for both HJI-safety constraints to be active simultaneously. We note that theoretically we could consider the robot, human, and wall simultaneously by computing the BRS for the joint system. However, naively increasing the state size without any decomposition would be computationally undesirable even offline. Thus we treat the human-robot and robot-wall systems separately, but will discuss later the impact of this design choice.

Alternatively, we can account for a static wall by adding lateral error bound constraints into the MPC tracking problem—this is the approach taken in [Brown et al. \(2017\)](#). This involves always adding a left and right lateral error constraint ($e_{\min,k} \leq e_k \leq e_{\max,k}$) on each node point along the MPC trajectory such that the lateral deviation from the desired trajectory does not exceed the lateral distance to the wall. This approach ensures no collision with the wall over the MPC time horizon only (in contrast to HJI which is over an infinite time horizon) and in general provides higher tracking performance because the MPC controller can optimize tracking states and controls over the entire tracking trajectory. We investigate both these approaches and provide more discussion in Section 8.

6.3 BRS Computation

We use the BEACLS toolkit [Tanabe and Chen](#) implemented in C++ to compute the BRS. Since HJ reachability suffers from the curse of dimensionality, this is, to the best of our knowledge, the first attempt to use HJ reachability to compute the BRS for a seven-state relative system, especially with such high modeling fidelity. We, however, do sacrifice on grid size and use a relatively coarse grid compared to other literature standards. We use a grid size of $13 \times 13 \times 9 \times 9 \times 9 \times 9 \times 9$ for our 7D system uniformly spaced over $(p_{x_{\mathcal{R}}}, p_{y_{\mathcal{R}}}, \psi_{\mathcal{R}}, U_{x_{\mathcal{R}}}, U_{y_{\mathcal{R}}}, v_{\mathcal{H}}, r_{\mathcal{R}}) \in [-15, 15] \times [-5, 5] \times [-\pi/2, \pi/2] \times [1, 12] \times [-2, 2] \times [1, 12] \times [-1, 1]$; computing the BRS with this system and discretization takes approximately 70 hours on a 3.0GHz octocore AMD Ryzen 1700 CPU.

Computing the BRS requires computing the optimal control and disturbance defined in (2). For the relative dynamics model, the optimal disturbance (i.e., human actions) is a bang-bang solution since the disturbance is affine. Due to the highly nonlinear nature of the dynamics, we use a uniform grid search across δ and F_x to calculate optimal actions (i.e., robot actions).

For the robot-wall system, we compute the optimal control actions in the same fashion. We use a grid size of $21 \times 9 \times 9 \times 9 \times 9$ uniformly spaced over $(p_{y_{\mathcal{R}}}, \psi_{\mathcal{R}}, U_{x_{\mathcal{R}}}, U_{y_{\mathcal{R}}}, r_{\mathcal{R}}) \in [-3, 7] \times [-\pi/2, \pi/2] \times$



Figure 7. X1: a steer-by-wire experimental vehicle platform. It is equipped with three LiDARs (one 32-beam and two 16-beam), a differential GPS/INS which provides pose estimates accurate to within a few centimeters as well as high fidelity velocity, acceleration, and yaw rate estimates. The 1/10-scale RC car has a mast that is visible to the LiDARs. ROS is used to interface the planning/control stack described in this work with a dSpace MicroAutoBox onboard X1 which handles sensing and control at the hardware level.

$[1, 12] \times [-2, 2] \times [-1, 1]$; computing the BRS with this system and discretization takes approximately 40 minutes.

7 Results

7.1 Experimental Vehicle Platform

X1 is a flexible steer-by-wire, drive-by-wire, and brake-by-wire experimental vehicle developed by the Stanford Dynamic Design Lab (see Figure 7). To control X1, desired steering (δ) and longitudinal tire force (F_{x_f}, F_{x_c}) commands are sent to the dSpace MicroAutoBox (MAB) which handles all sensor inputs except LiDAR (handled by the onboard PC) and implements all low level actuator controllers. Similarly, state information about the vehicle is obtained from the MAB. We use the Robot Operating System (ROS) to communicate with the MAB; the planning/control stack described in this work is running onboard X1 on a consumer desktop PC running Ubuntu 16.04 equipped with a quadcore Intel Core i7-6700K CPU and an NVIDIA GeForce GTX 1080 GPU. We also perform experiments using a LiDAR-visible 1/10-scale RC car (Figure 7 right) as the human-driven car to investigate the robustness of our proposed control stack with perception uncertainty.

The code used to run the experiments can be found here: https://github.com/StanfordASL/safe_traffic_weaving.

7.2 Experiments

To evaluate our proposed control stack—a synthesis of a high-level probabilistic interaction planner with the MPC+HJI tracking controller—we perform full-scale human-in-the-loop traffic-weaving trials with X1 taking on the role of the robot car. We scaled the highway traffic-weaving scenario (mean speed $\sim 28\text{m/s}$) in Schmerling et al. (2018) down to a mean speed of $\sim 8\text{m/s}$ by shortening the track (reducing longitudinal velocity by a constant) and scaling time by a factor of 4/3 (with the effect of scaling speeds by 3/4 and accelerations by 9/16).

We investigate and evaluate the effectiveness of our control stack by allowing the human car to act carelessly

(i.e., swerving blindly towards the robot car) during the experiments. We compare our proposed controller (MPC+HJI) against a tracking-only MPC controller (MPC) and a controller that switches to the HJI optimal avoidance controller when near safety violation (switching). To ramp up towards testing with two full-scale vehicles in the near future, we investigate two types of human car: (1) a virtual human-driven car and (2) a 1/10-scale LiDAR-visible human-driven RC car.

7.2.1 Virtual Human-Driven Vehicle To ensure a completely safe experimental environment, our first tier of experimentation uses a joystick-controlled virtual vehicle for the human car and allows the robot control stack to have perfect observation of the human car state. Experimental trials of the probabilistic planning framework using (1) our proposed approach (MPC+HJI) and (2) switching to the optimal HJI controller (switching) are shown in Figure 8, along with a simulated comparison between the two safety controllers and the tracking-only controller (MPC). For comparative purposes, the controllers were simulated with the displayed nominal trajectory held fixed, but in reality, the nominal trajectory in these experiments was updated at $\sim 3\text{Hz}$. As expected, we see that when safety violation occurs, the MPC+HJI controller represents a middle ground between the tracking-only MPC which does not react to the human car’s intrusion, and the switching controller which arguably overreacts with a large excursion outside the lane boundaries. Evidently, our proposed controller tries to be minimally interventional—the robot car swerves/brakes but only to an extent that is necessary.

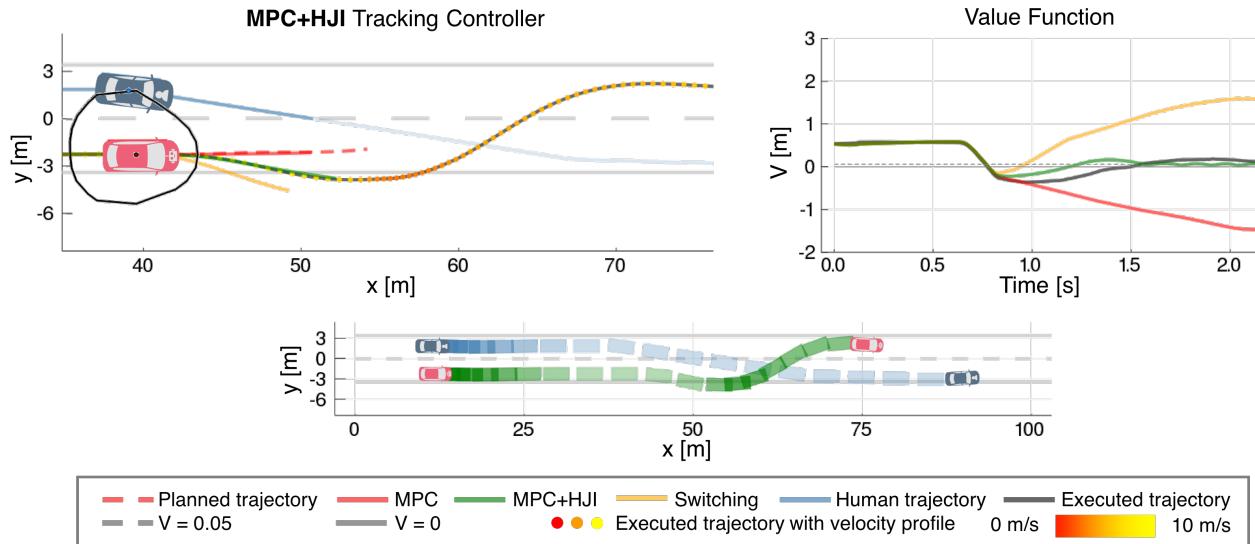
Looking at the value function, we see that, as expected, the MPC+HJI controller aims to keep the value positive, but does not necessarily strive to increase it, while the switching controller aims to increase the value as much as possible. The MPC (tracking only) controller fails to increase the value at all when safety violation occurs. All controllers however, experience a period where the value is negative, even the two HJI-based controllers which theoretically guarantee safety. We believe this is due to model mismatch; we will discuss this point in more depth in Section 8.

Additionally, in some trials when the robot car was traveling faster, the activation of the HJI constraint resulted in the robot car performing a large but smooth swerve that traversed completely outside the lane boundaries. We address this limitation by adding a wall constraint into the MPC formulation discussed in Section 7.2.3.

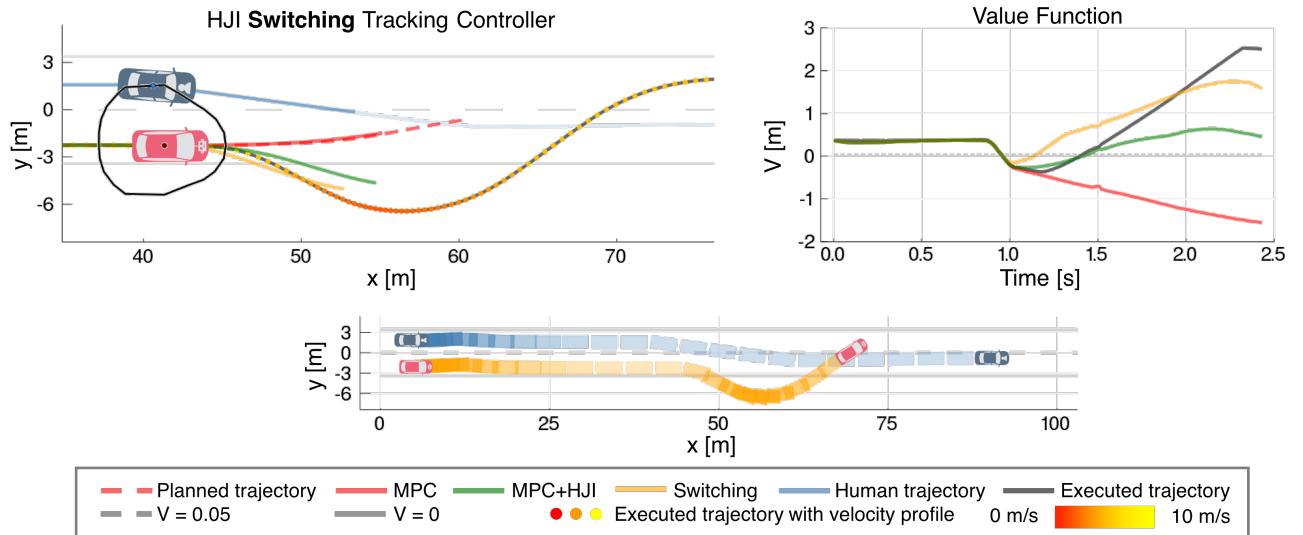
7.2.2 Safety and Efficiency Trade-off Our MPC+HJI controller considers the set of safety-preserving controls while optimizing its tracking performance when it is near safety violation. In contrast, the HJI switching controller uses the optimal avoidance control policy and as a result neglects to track the desired trajectory that was selected by the planner for interaction performance. As such, there is a trade off between safety, defined with respect to the value function, and efficiency.

For use as a comparison metric, we define the notion of *total safety* of an interaction of time length T as

$$S_{\text{total}} := \int_0^T \mathbf{1}[V(x_{\mathcal{R}}(t)) \leq 0] V(x_{\mathcal{R}}(t)) dt$$



(a) Controller comparison on experimental data where the robot car uses the MPC+HJI controller.



(b) Controller comparison on experimental data where the robot car uses the switching controller.

Figure 8. Controller comparison on planner trajectories from X1-virtual human-driven vehicle experiments. Left: Simulations of using the MPC, MPC+HJI, and switching controllers when $V(x_{\mathcal{R}}(t))$ first drops below $\epsilon = 0.05$ are shown, and are compared against the executed trajectory (from experiments) and the desired trajectory (from the interaction planner). Right: The corresponding evolutions of $V(x_{\mathcal{R}}(t))$. Bottom: Illustration of the traffic-weaving interaction. The human controlled car (blue car) carelessly drives into the path of the robot car (red car), causing the robot car to react by swerving. The transparency of the boxes (car) corresponds to the speed of the car (higher transparency corresponds to higher speed).

which is the integral of the value function when it is negative. Total safety considers not only the magnitude of the safety violation, but also the duration of the violation. We can also define the *worst-case safety* as

$$S_{\text{worst}} := \min_{t \in [0, T]} V(x_{\mathcal{R}}(t))$$

which does not consider the duration of safety violation, but rather the worst-case safety violation with respect to the value function over the interaction.

Efficiency of the interaction is more difficult to quantify. In this analysis, we define efficiency with respect to the *g-forces* experienced by the vehicle as this is a proxy for control effort and passenger comfort. Alternative metrics include time taken to complete the interaction, and friction available

in the tires. We define *average efficiency* as

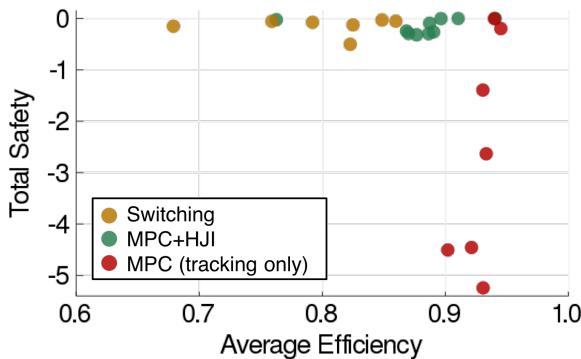
$$E_{\text{avg}} := 1 - \frac{1}{T} \int_0^T \frac{1}{g} \sqrt{a_x(t)^2 + a_y(t)^2} dt$$

where $a_x(t)$ and $a_y(t)$ are the x and y acceleration of the robot car; larger E_{avg} values correspond to better efficiency. *g-forces* should not exceed one as this is beyond the physical limits of a vehicle. We also define the *worst-case efficiency* as

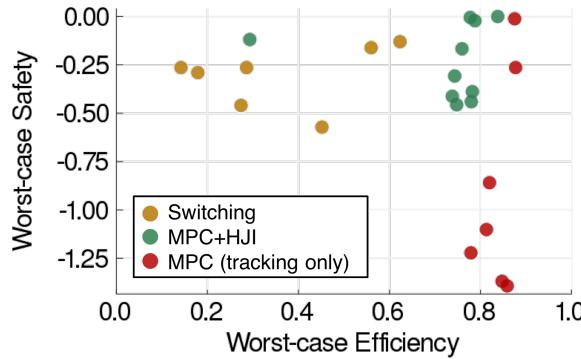
$$E_{\text{worst}} := \max_{t \in [0, T]} \left(1 - \frac{1}{g} \sqrt{a_x(t)^2 + a_y(t)^2} \right)$$

which considers the largest *g-force* experienced during the interaction.

Given these quantities, we can compare the safety and efficiency trade-offs between the MPC, MPC+HJI, and



(a) Trade-off between total safety and average efficiency.

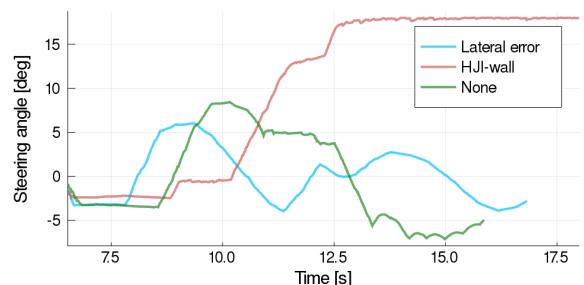


(b) Trade-off between worst-case safety and worst-case efficiency.

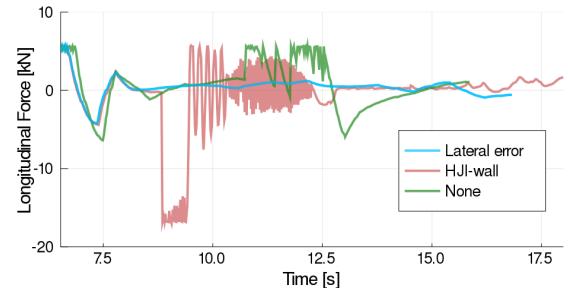
Figure 9. Safety and efficiency trade-offs for the different control strategies.

switching controllers. Multiple experimental trials using X1 were carried out using each controller, and Figure 9a compares the average/total metrics while Figure 9b compares the worst-case metrics. We see that in both cases, the MPC+HJI controller provides a good balance between safety and efficiency; the safety almost equaling that of the optimal switching controller, and efficiency almost equaling that of the MPC controller. The switching controller provides the highest level of safety (with respect to the value function) but experiences lower efficiency since it often results in heavy braking and sharp swerving. The MPC controller provides lower safety scores but with larger variations as the resulting safety score is scenario dependent rather than controller dependent.

7.2.3 Static wall We investigate two methods, (1) persistently adding lateral error state constraints into the MPC problem and (2) adding an additional HJI-wall constraint into the MPC problem when $V_{\text{WALL}}(x_R) \leq \epsilon$, for avoiding a static (virtual) road boundary wall when the robot car is swerving off the road to avoid a collision with the human car. Figure 10 compares the control sequences and Figure 11 compares the trajectory of both approaches. Here, the human car is commanded to stay in the lane and is initialized inside the BRS so that when the robot starts planning autonomously, the robot car will react immediately and swerve out of the way—we investigate the nature of the swerving. We do note that the human car is able to make the robot car swerve even though it is staying in its lane. This is due to the wide, pear-shape BRS when the two cars are parallel and at similar



(a) Comparison of steering angle. The plot begins when the robot car is planning autonomously.



(b) Comparison of longitudinal force. The plot begins when the robot car is planning autonomously.

Figure 10. Comparison of control sequences from using MPC lateral error state constraints, HJI-wall constraint, and no constraints to avoid a static road boundary.

speeds (see Figure 1b left). We discuss this in more detail in the next section.

Around $x = 20$ in Figure 10, the robot car begins to plan autonomously. Before then, the cars are following a straight line in order to speed up from $\approx 1m/s$ to the desired interaction speed. To provide the cleanest comparison all results in this subsection are derived from simulation. Starting in the right lane, we see in all cases that when safety is violated the robot car swerves to the right ($\delta < 0$). When using the lateral error constraints (Figure 10, top), the robot car essentially has a “look-ahead” capability because it is able to optimize its steering commands over the MPC tracking horizon, essentially distributing the responsibility of avoiding the wall across the entire tracking MPC horizon. As a result, the robot car is able to successfully and smoothly steer back onto the road and avoid the wall.

When using the HJI-wall constraint (Figure 10, center), the robot car does not preemptively swerve back onto the road and instead only reacts to the wall when $V_{\text{WALL}}(x_R) \leq \epsilon$ as designed. The robot car performs a hard brake to the point of almost stopping[‡] and then begins to eventually command a maximum steering angle (18 deg). The sharp and abrupt behavior stems from the HJI formulation assuming the robot can and will take extreme actions, including cases when using the safety-preserving control set. As a result, the responsibility for evasive action is triggered at the very latest

[‡]Since the dynamic bicycle model is ill-defined for low speeds (thus explaining the oscillations in F_x), the experiment (when using the HJI-wall constraint) is essentially over around $t = 9$. In general, the MPC problem can switch to the kinematic model at low speeds Patterson et al. (2018) which is well defined in that speed region.

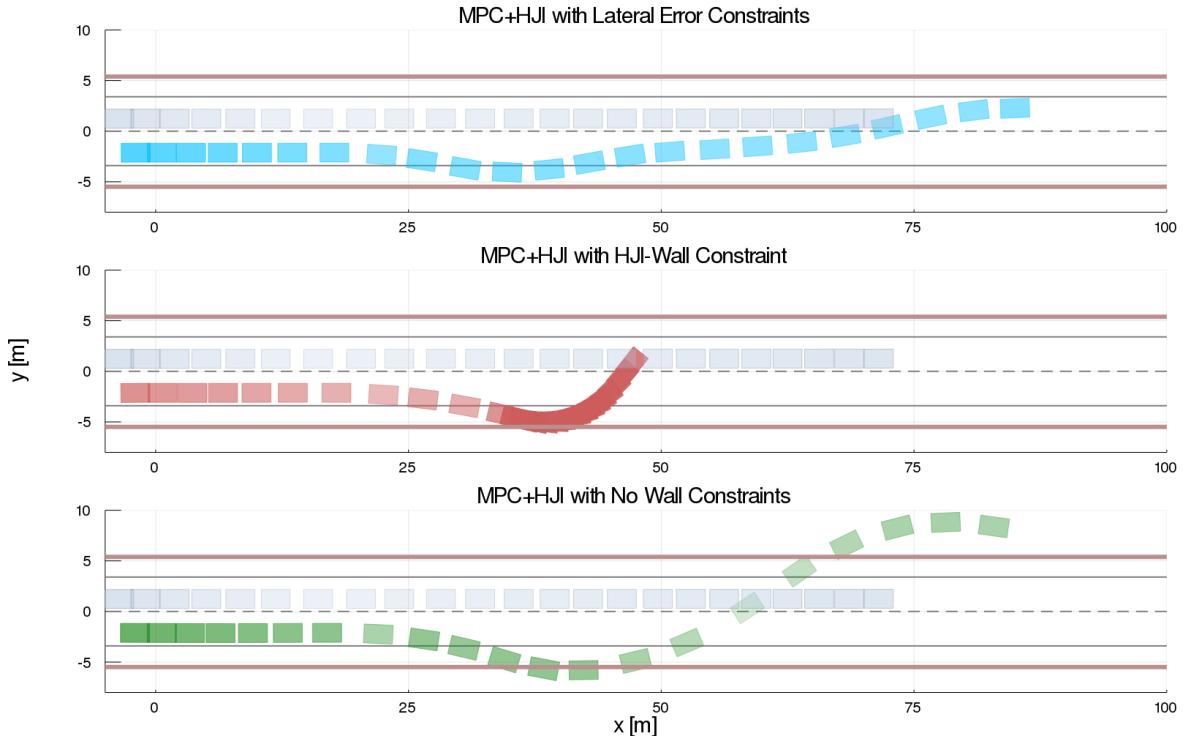


Figure 11. Trajectory comparison between using HJI-wall constraint (top), a lateral error constraints (middle) and using no wall constraints (bottom). The human car is in the left (top) lane and is constant across all three trials, and the robot car is in the right (bottom) lane. The transparency of the car corresponds to the speed of the car (higher transparency corresponds to higher speed).

possible instance and compressed into a control constraint over a single (in practice 3) MPC timestep. This is in contrast to the other approach of always having MPC lateral error state constraints over the entire MPC horizon. The MPC+HJI controller is effective in avoiding dynamic obstacles, but for static obstacles, HJI is not suitable because it is unnecessary to reason about the dynamics of something that is static. With no wall constraints (Figure 10, bottom), the robot car swerves completely off the road, and in order to quickly get to the left lane before the end of the road (inscribed as an objective in the high-level planner), it overshoots to the other side of the road and also drives beyond the road boundary on the other side.

7.2.4 1/10-Scale Human-Driven Vehicle To begin investigating the effects of perception uncertainty on our safety assurance framework, we use three LiDARs onboard X1 to track a human-driven RC car, and implement a Kalman filter for human car state estimation (position, velocity, and acceleration). Even with imperfect observations, we show some successful preliminary results (an example is shown in Figure 12) at mean speeds of 4m/s, close to the limits of the RC car + LiDAR-visible mast in crosswinds at the test track. We observe similar behavior as in the virtual human car experiments, including the fact that the value function dips briefly below zero before the MPC+HJI controller is able to arrest its fall; we discuss this behavior in the next section.

8 Discussion

Beyond the qualitative and quantitative confirmation of our design goals, our experimental results reveal three main insights.

Takeaway 1: The reachability cache is underly-conservative with respect to robot car dynamics and overly-conservative with respect to human car dynamics.

In all cases—hardware experiments as well as simulation results—the HJI value function V dips below zero, indicating that neither the HJI+MPC nor even the optimal avoidance switching controller are capable of guaranteeing safety in the strictest sense. The root of this apparent paradox is in the computation of the reachability cache used by both controllers as the basis of their safety assurance. Though the 7-state relative dynamics model (8) subsumes a single-track vehicle model that has proven successful in predicting the evolution of highly dynamic vehicle maneuvers Funke et al. (2017); Brown et al. (2017), the way it is employed in computing the value function V omits relevant components of the dynamics. In particular, when computing the optimal avoidance control (2) as part of solving the HJI partial differential equation, we assume total freedom over the choice of robot steering angle δ and longitudinal force command F_x , up to maximum control limits. This does not account for, e.g., limits on the steering slew rate (traversing $[\delta_{\min}, \delta_{\max}]$ takes approximately 1 second), and thus the value function is computed under the assumption that the robot can brake/swerve far faster than it actually can.

We note that simply tuning the safety buffer ϵ is insufficient to account for these unmodeled dynamics. In Figure 8a we see that V may drop from approximately 0.5 (the value when the two cars traveling at 8m/s start side-by-side in lanes) to -0.3 in the span of a few tenths of a second. Selecting $\epsilon > 0.5$ might give enough time for the steering to catch up, but such a selection would prevent the robot car from accomplishing the traffic weaving task even under nominal conditions, i.e., when the human car

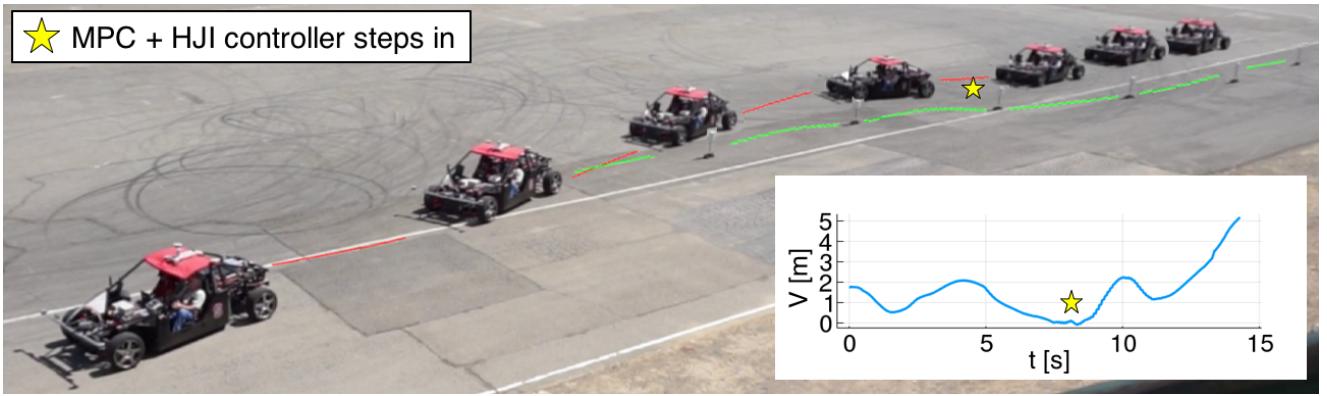


Figure 12. Time-lapse of pairwise vehicle interaction: X1 with 1/10-scale human-driven RC car. The RC car (green trajectory) nudges into X1 (red trajectory), which swerves gently to avoid. Inset: The value function over time of the X1-human driven RC car experiment

is equally concerned about collision avoidance. Even for $\epsilon = 0.05$, we see that the human car can cause the robot to swerve by just staying its lane but with a small offset towards the robot’s lane (see Figure 11). This is because the safety controller would push the robot car outside of its lane from the outset to maintain the buffer. This behavior follows from wide level sets associated with the transient control authority asymmetry (recall that in the HJI relative dynamics the human car may adjust its trajectory curvature discontinuously), assumed as a conservative safety measure as well as a way to keep the relative state dimension manageable.

The simplest remedy for both of these issues is to increase the fidelity of the relative dynamics model by incorporating additional integrator states $\dot{\delta}$ for the robot and $\dot{\omega}$ for the human. Naively increasing the state dimension to 8 or 9, however, might not be computationally feasible (even offline) without devising more efficient HJI solution techniques or choosing an extremely coarse discretization over the additional states. By literature standards we already use a relatively coarse discretization grid for solving the HJI PDE; associated numerical inaccuracies may be another source of the observed safety mismatch. We believe that simulation, accounting for slew rates, could be a good tool to prototype such efforts, noting that as it stands we have relatively good agreement between simulation and the experimental platform in our testing.

Takeaway 2: Interpretability of the value function V should be a key consideration in future work.

In this work the terminal value function $V(0, x_{\mathcal{R}})$ is specified as the separation/penetration distance between the bounding boxes of the two vehicles, a purely geometric quantity dependent only on $p_{x_{\mathcal{R}}}$, $p_{y_{\mathcal{R}}}$, and $\psi_{\mathcal{R}}$. Recalling that V ($= V_{\infty}$) represents the worst-case eventual outcome of a differential game assuming optimal actions from both robot and human, we may interpret the above results through the lens of worst-case outcomes, i.e., a value of -0.3 may be thought of as 30cm of collision penetration assuming optimal collision seeking/avoidance from human/robot. When extending this work to cases with environmental obstacles (e.g., concrete highway boundaries that preclude large deviations from the lane), or multi-agent settings where the robot must account for the uncertainty in multiple other

parties’ actions, for many common scenarios it may be the case that guaranteeing absolute safety is impossible. Instead of avoiding a BRS of states that might lead to collision, we should instead treat the value function inside the BRS as a cost. In particular we should specify more contextually relevant values of $V(0, x_{\mathcal{R}})$ for states in collision, e.g., negative kinetic energy or another notion of collision severity as a function of the velocity states $U_{x_{\mathcal{R}}}$, $U_{y_{\mathcal{R}}}$, v_h , and $r_{\mathcal{R}}$ in addition to the relative pose. This would lead to a controller that prefers, in the worst case, collisions at lower speed, or perhaps “glancing blows” where the velocities of the two cars are similar in magnitude and direction.

Takeaway 3: Static obstacles should be accounted for using MPC tracking state constraints and not HJI constraints.

It is natural to use HJI when reasoning about potential dangers from a dynamic, sentient agent but for static obstacles, there is no uncertainty in how the obstacle may act. Using HJI to avoid a static wall may give strict safety guarantees, but offers little foresight over the entire MPC horizon—safe feasible control actions are boiled down to a single control constraint over one (in practice, 3) MPC time step(s). Consequently, it is not surprising that adding lateral error constraints along the entire MPC horizon instead of an additional HJI-wall constraint results in higher tracking performance as it is able to optimize over the entire planning trajectory. In general, this approach is more favorable because (1) for autonomous cars, passenger comfort is a key planning consideration, (2) this approach is applicable to any general road geometry without the need for any offline computation, and (3) the number of constraints remains constant regardless of the number of static obstacles. In contrast, the HJI-wall constraint results in sudden braking and steering maneuvers, and the number of constraints increases with the number of static obstacles (assuming access to the BRS for each static obstacle). [Bajcsy et al. \(2019\)](#) computes the BRS for arbitrary static obstacles at run time, but is not sufficiently fast for our 100Hz MPC frequency requirement especially with bicycle model robot dynamics.

Due to the model mismatch discussed in Takeaway 1, whenever safety is violated between the robot and the wall, it is likely that the robot will enter the robot-wall BRS. Since the wall is static, it will result in an inevitable collision,

which is not necessarily the case between the human and robot car if the human is not following an optimal adversarial policy. Further, in cases where the robot car is squeezed between the road boundary wall and the human, having two HJI constraints may lead to chattering in the controls. This also leads to the discussion of how one should weigh the relative importance of human car and wall constraint via the slack variable penalty. Since the wall is static, the penalty for the wall constraint should be higher since entering the set implies inevitable collision. However, if the penalty on avoiding the human car is not high enough, then the robot is less incentivized to avoid the human.

For nominal cases, such as the ones explored in this work, a combination of the MPC+HJI controller with lateral error constraints for avoiding static obstacles is the preferred approach. Future work will investigate how to properly prioritize which obstacle, human or wall, to avoid especially in cases where collision is inevitable (e.g., the robot is boxed in) and the goal is to minimize collision severity instead of penetration distance.

9 Conclusions

We have investigated a control scheme for providing real-time safety assurance to underpin the guidance of a probabilistic planner for human-robot vehicle-vehicle interactions. By essentially projecting the planner’s desired trajectory into the set of safety-preserving controls whenever safety is threatened, we preserve more of the planner’s intent than would be achieved by adopting the optimal control with respect to separation distance. Our experiments show that with our proposed minimally interventional safety controller, we accomplish the high level objective (traffic weaving) despite the human car swerving directly onto the path of the robot car, and accomplish this relatively smoothly compared to using a switching controller that results in the robot car swerving more violently off the road. Further, we investigate the addition of a road boundary wall to our formulation to prevent the robot car from swerving completely out of the lane which could be dangerous in realistic road settings.

We note that this work represents only a promising first step towards the integration of reachability-based safety guarantees into a probabilistic planning framework. Future work includes investigating this framework for cases where the human and robot have very different dynamics, such as a pedestrian or cyclist interacting with a car, or a human interacting with a robotic manipulator, and for interactions involving multiple (more than two) agents. We may also consider adapting our approach to ensure high planning performance while guaranteeing satisfaction of constraints other than safety, e.g., task requirements specified by temporal logic constraints as considered in Chen et al. (2018). In the context of human-robot vehicle interactions, we have already discussed the concrete modifications to this controller we believe are necessary to improve the practical impact of our theoretical guarantees; further study should also consider better fitting of the planning objective at the controller level. That is, instead of performing a naive projection, i.e., the one that minimizes trajectory tracking error, it is likely that a more nuanced selection informed by the planner’s prediction model would represent a better

“backup choice” in the case that safety is threatened. We recognize that ultimately, guaranteeing absolute safety on a crowded roadway may not be realistic, but we believe that in such situations value functions derived from reachability may provide a useful metric for near-instantly evaluating the future implications of a present action choice.

Acknowledgements

This work was supported by the Office of Naval Research (Grant N00014-17-1-2433), by Qualcomm, and by the Toyota Research Institute (“TRI”). This article solely reflects the opinions and conclusions of its authors and not ONR, Qualcomm, TRI, or any other Toyota entity. The authors would like to thank the X1 team past and present, in particular Matt Brown, Larry Cathey, and Amine Elhafsi, Matteo Zallio, and the Thunderhill Raceway Park for accommodating testing.

References

- M. Althoff and J. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.
- M. Althoff and J. M. Dolan. Set-based computation of vehicle behaviors for the online verification of autonomous vehicles. In *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, 2011.
- M. Althoff and B. H. Krogh. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Transactions on Automatic Control*, 59(2):371–383, 2014.
- S. Arora, S. Choudhury, D. Althoff, and S. Scherer. Emergency maneuver library – ensuring safe navigation in partially known environments. In *Proc. IEEE Conf. on Robotics and Automation*, 2015.
- A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. *arXiv preprint arXiv:1905.00532*, 2019.
- O. Bokanowski, N. Forcadel, and H. Zidani. Reachability and minimal times for state constrained nonlinear problems without any controllability assumption. *SIAM Journal on Control and Optimization*, 48(7):4292–4316, 2010.
- M. Brown, J. Funke, S. Erlien, and J. C. Gerdes. Safe driving envelopes for path tracking in autonomous vehicles. *Control Engineering Practice*, 61:307–316, 2017.
- M. Chen and C. J. Tomlin. Hamilton–Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):333–358, 2018. doi: 10.1146/annurev-control-060117-104941.
- M. Chen, Q. Hu, J. Fisac, K. Akametalu, C. Mackin, and C. Tomlin. Reachability-based safety and goal satisfaction of unmanned aerial platoons on air highways. *AIAA Journal of Guidance, Control, and Dynamics*, 40(6):1360 – 1373, 2017.
- M. Chen, Q. Tam, S. C. Livingston, and M. Pavone. Signal temporal logic meets Hamilton–Jacobi reachability: Connections and applications. In *Workshop on Algorithmic Foundations of Robotics*, 2018. Submitted.

- A. Dhinakaran, M. Chen, G. Chou, J. C. Shih, and C. J. Tomlin. A hybrid framework for multi-vehicle collision avoidance. In *Proc. IEEE Conf. on Decision and Control*, 2017.
- J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Hybrid Systems: Computation and Control*, 2015.
- J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems, 2017. Available at <https://arxiv.org/abs/1705.01292>.
- G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proc. Int. Conf. Computer Aided Verification*, 2011.
- J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes. Collision avoidance and stabilization for autonomous vehicles in emergency scenarios. *IEEE Transactions on Control Systems Technology*, 25(4):1204 – 1216, 2017.
- A. Gattami, A. Al Alam, K. H. Johansson, and C. J. Tomlin. Establishing safety for heavy duty vehicle platooning: A game theoretical approach. *IFAC World Congress*, 44(1):3818–3823, 2011.
- A. Gray, Y. Gao, J. K. Hedrick, and F. Borrelli. Robust predictive control for semi-autonomous vehicles with an uncertain driver model. In *IEEE Intelligent Vehicles Symposium*, 2013.
- M. R. Greenstreet and I. Mitchell. Integrating projections. In *Hybrid Systems: Computation and Control*, 1998.
- T. Koolen and contributors. Parametron.jl: Efficiently solving parameterized families of optimization problems in Julia. Available at <https://github.com/tkoolen/Parametron.jl>.
- A. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis: Internal approximation. *System and Control Letters*, 41(3):201–211, 2000.
- S. B. Liu, H. Roehm, C. Heinemann, I. Ltkbohle, J. Oehlerking, and M. Althoff. Provably safe motion of mobile robots in human environments. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2017.
- A. Majumdar, R. Vasudevan, M. M. Tobenkin, and R. Tedrake. Convex optimization of nonlinear feedback controllers via occupation measures. *Int. Journal of Robotics Research*, 33(9):1209–1230, 2014.
- K. Margellos and J. Lygeros. Hamilton-Jacobi formulation for reach-avoid differential games. *IEEE Transactions on Automatic Control*, 56(8):1849–1861, 2011. doi: 10.1109/TAC.2011.2105730.
- J. Mattingley and S. Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 12(1):1–27, 2012.
- I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.
- H. Pacejka. *Tire and Vehicle Dynamics*. Elsevier, 2002.
- V. Patterson, S. Thornton, and J. C. Gerdes. Tire modeling to enable model predictive control of automated vehicles from standstill to the limits of handling. In *Int. Symp. on Advanced Vehicle Control*, 2018.
- J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in julia, 2016. Available at <https://arxiv.org/abs/1607.07892>.
- D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, 2016.
- E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *Proc. IEEE Conf. on Robotics and Automation*, 2018.
- B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. 2017. Available at <https://arxiv.org/abs/1711.08013>.
- K. Tanabe and M. Chen. BEACLS: Berkeley Efficient API in C++ for Level Set methods. Available at <https://github.com/HJReachability/beacls>.
- M. T. Wolf and J. W. Burdick. Artificial potential functions for highway driving with collision avoidance. In *Proc. IEEE Conf. on Robotics and Automation*, 2008.
- S. Yoon, A. Fern, R. Givan, and S. Kambhampati. Probabilistic planning via determinization in hindsight. In *Proc. AAAI Conf. on Artificial Intelligence*, 2008.