Safe Human-Interactive Control via Shielding

Jeevana Priya Inala[1], Yecheng Jason Ma[2], Osbert Bastani[2], Xin Zhang[3], Armando Solar-Lezama[1]

***Abstract*—Ensuring safety for human-interactive robotics is important due to the potential for human injury. The key challenge is defining safety in a way that accounts for the complex range of human behaviors. We propose an approach for ensuring safety based on *backup actions* we believe the human always considers taking to avoid an accident—e.g., braking to avoid rear-ending the robot. Given a set of backup actions, our approach guarantees safety as long as the human takes the appropriate backup actions when necessary to ensure safety. We evaluate our approach on real humans interacting with a simulated robot.**[1]

I. Introduction

Robots are increasingly operating in environments where they must interact with humans, such as collaborative grasping [1, 2] and autonomous driving [3, 4, 5, 6]. Ensuring safety for such robots is paramount due to the potential to inflict harm on humans [7]. These challenges are particularly salient in settings such as autonomous driving, where robots and humans may have disjoint or conflicting goals—e.g., a self-driving car making an unprotected left turn [5].

The key challenge is how to define safety for human-interactive robots. Modeling the human as an adversary is one approach, but is typically prohibitively conservative. Another approach is to learn a model to predict human actions [8, 9, 10], and ensure safety with respect to this model. However, different humans may exhibit very different behaviors—e.g., people in different regions may drive differently. If a human behavior is not exhibited in the data used to train the model, then the model may not account for it. Another alternative, called *responsibility sensitive safety (RSS)* [11], is to manually specify the range of acceptable robot actions in various scenarios. In this approach, the designer of the robot controller is responsible for ensuring that acceptable actions only include safe actions. However, manually defining acceptable robot actions for all possible scenarios is challenging, especially for robots operating in open-world environments.

We propose a novel approach for ensuring safety in human-interactive robotics systems based on the following key ideas:

- **Bounding human behavior via backup actions:** The controller designer specifies *backup actions* that they believe the human always considers taking to avoid an accident (e.g., braking while steering in some direction). We assume the human may take any action in general, take these actions when necessary to ensure safety.
- **Ensuring safety:** We use *abstract interpretation* [12] to overapproximate the reachable set of the system for the above model of human behavior, and then ensure safety with respect to this overapproximation.

[1]Appendix and video are available at: https://osf.io/atbp4/?view_only=865b3796b5454594996bc5173ce75504.

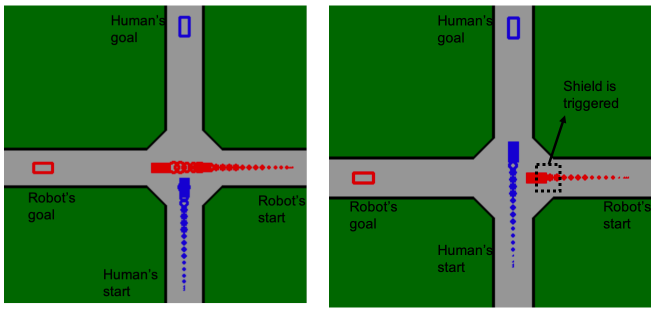

Fig. 1. Trajectories showing a robot (red) and a human (blue) interacting at an intersection (for 25 timesteps). Left: The robot passes before the human, leveraging the fact that a responsible human would slightly brake to allow the robot to cross safely. Right: Human arrives at the intersection first; the robot triggers the shield to brake and allow the human to cross first.

First, our notion of backup actions captures the idea that we reasonably believe the human will take a limited range of evasive maneuvers to avoid an accident—e.g., if the robot gradually slows to a stop, then we may expect the human to slow down to avoid rear-ending it. If the robot is on a highway, coming to a stop is more dangerous; in this case, we might conservatively restrict to the case where the robot pulls over to the shoulder before coming to a stop. Specifying backup actions provides a way to define safety; we refer to such a safety constraint as *safety modulo fault*. In particular, to instantiate our framework, the controller designer provides:

- **Robot backup action:** An action that the human anticipates the robot may take to ensure safety (e.g., to brake without changing directions), chosen based on intuitions based on traffic rules and common sense.
- **Human backup action set:** A set of actions that includes *at least one* action the human considers taking to ensure safety (e.g., braking while steering in some direction), chosen based on intuitions about human behavior.

Next, we propose an algorithm for ensuring safety modulo fault. We build on *model predictive shielding* [13, 14], which takes an arbitrary controller designed to reach the goal, but then overrides it if needed to ensure safety. In particular, our algorithm, called *MPS modulo fault*, uses on-the-fly verification based on abstract interpretation to determine whether the goal-reaching controller is safe; if so, it uses the given controller, but otherwise, it switches to a safe backup controller. Figure 1 shows how our algorithm ensures safety while interacting with a human driver without being overly cautious.

We empirically evaluate our approach on a real human interacting with a simulated robot via keyboard. We demonstrate that our algorithm enables the robot to avoid accidents, even when combined with a naïve controller that ignores the human.

II. Preliminaries

a) Human-robot system: We consider a robot R and a human H. As in prior work [5], we assume they act in alternation, which is reasonable if the time steps are small. For a state x_t where R is acting, we have

$$x_t' = f_R(x_t, u_{R,t}) \qquad \text{and} \qquad x_{t+1} = f_H(x_t', u_{H,t}),$$

where $\mathcal{X} \subseteq \mathbb{R}^{n_X}$ is the state space, $\mathcal{U}_A \subseteq \mathbb{R}^{n_{U,A}}$ are the actions for $A \in \{R, H\}$, and $f_A : \mathcal{X} \times \mathcal{U}_A \to \mathcal{X}$ are the dynamics for A. Given initial state $x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$ where R is acting, and actions $\vec{u}_A = (u_{A,0}, u_{A,1}, ...) \in \mathcal{U}_A^\infty$ for each $A \in \{R, H\}$, we define the trajectory $\zeta_R(x_0, \vec{u}_R, \vec{u}_H) = (x_0, x_0', x_1, ...) \in \mathcal{X}^\infty$, where $x_t' = f_R(x_t, u_{R,t})$ and $x_{t+1} = f_H(x_t', u_{H,t})$. Similarly, given initial state $x_0' \in \mathcal{X}$ where H is acting, we define $\zeta_H(x_0', \vec{u}_H, \vec{u}_R) = (x_0', x_0, x_1', ...)$, where $x_t = f_H(x_t', u_{H,t})$ and $x_{t+1}' = f_R(x_t, u_{R,t})$. We can replace each $\vec{u}_A$ by a policy $\pi_A : \mathcal{X} \to \mathcal{U}_A$. Our goal is to ensure the system stays in a given safe region $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$.

Definition II.1. A trajectory $\zeta = (x_0, x_0', x_1, ...)$ (or $\zeta = (x_0', x_0, x_1', ...)$) is *safe* if $x_t, x_t' \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$.

III. Safety Modulo Fault

Here, we formalize our assumptions and safety notion.

a) Human objective: We assume the human acts according to a maximin objective, where the "min" is the worst-case over actions the human anticipates the robot may take, and the "max" is over the human's own actions. That is, the human plans optimally while conservatively accounting for actions they anticipate the robot might take. In particular, at state x', the human takes an action $\pi_H(x') = u_{H,0}^*$ such that

$$\vec{u}_H^* \in \operatorname*{arg\,max}_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(x', \vec{u}_H), \tag{1}$$

where the $\arg\max$ denotes the set of all optimal values, and

$$J_H(\vec{u}_H) = \min_{\vec{u}_R \in \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x', \vec{u}_H, \vec{u}_R))$$

$$J_H((x_0', x_0, x_1, ...)) = \sum_{t=0}^{\infty} \gamma^t r_H(x_t', u_{H,t}, x_t),$$

where $\hat{\mathcal{U}}_R \subseteq \mathbb{R}^{n_{U,R}}$ is the set of actions the human anticipates the robot may take, $r_H : \mathcal{X} \times \mathcal{U}^H \times \mathcal{X} \to \mathbb{R} \cup \{-\infty\}$ is the human reward function, and $\gamma \in (0, 1)$ is a discount factor.

The key challenge for the robot to plan safely is that it does not know the human reward function r_H, the human action set $\mathcal{U}_H$ or the human-anticipated robot action set $\hat{\mathcal{U}}_R$. Assuming we know these values exactly is implausible. Instead, we assume access to minimal knowledge about each of these objects, which we formalize in the next section.

b) Assumption on human objective: First, we assume that the human reward for reaching an unsafe state is $-\infty$.

Assumption III.1. For any $x', x \in \mathcal{X}$ and $u_H \in \mathcal{U}_H$, we have $r_H(x', u_H, x) = -\infty$ if $x' \notin \mathcal{X}_{\text{safe}}$ or $x \notin \mathcal{X}_{\text{safe}}$.

That is, the human driver always acts to avoid an accident. Other than Assumption III.1, r_H can be arbitrary.

With this assumption, there are two reasons accidents may happen: (i) there was a safe action sequence $\vec{u}_H \in \mathcal{U}_H^\infty$ that the human driver failed to take, or (ii) if the robot takes an action $u_R \notin \hat{\mathcal{U}}_R$ that the human driver failed to anticipate. Thus, we can always conservatively take $\hat{\mathcal{U}}_R$ to be smaller than it actually is. Conversely, we can always take $\mathcal{U}_H^0$ to be larger than it actually is. Thus, we make minimal assumptions about what actions are contained in $\hat{\mathcal{U}}_R$ and $\mathcal{U}_H$.

First, we make the following assumption on the set of actions $\vec{\mathcal{U}}_R$ that the human anticipates the robot may take:

Assumption III.2. We are given a *robot backup action* $u_R^0 \in \mathcal{U}_R$ that is anticipated by the human—i.e., $u_R^0 \in \hat{\mathcal{U}}_R$.

That is, the human always accounts for the possibility that the robot might take action u_R^0. For example, we might assume that u_R^0 is gradually braking and coming to a stop.

Next, we make the following assumption about the human:

Assumption III.3. We are given a *human backup action set* $\mathcal{U}_H^0 \subseteq \mathcal{U}_H$ such that if $\max_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(\vec{u}_H) = -\infty$, then the human takes an action $\pi_H(x') \in \mathcal{U}_H^0$.

That is, if the human is unable to guarantee safety (i.e., their objective value is $-\infty$), then they take *some* action in $\mathcal{U}_H^0$. For example, $\mathcal{U}_H^0$ may contain all actions where the human driver decelerates by at least some rate; this choice allows them to slow down more quickly or steer in any direction.

c) Problem formulation: Our goal is to ensure that the robot acts in a way that ensures safety for an infinite horizon for any human that satisfies our assumptions.

Definition III.4. A robot policy $\pi_R : \mathcal{X} \to \mathcal{U}_R$ is *safe modulo fault* for initial states $\mathcal{X}_0 \subseteq \mathcal{X}$ if for any human policy π_H satisfying Assumptions III.1, III.2, & III.3, and any $x_0 \in \mathcal{X}_0$, the trajectory $\zeta_R(x_0, \pi_R, \pi_H) \in \mathcal{X}^\infty$ is safe.

That is, π_R that ensures safety as long as the human acts in a way that satisfies our assumptions. Our goal is to design a policy π_R that is safe modulo fault.

Finally, we cannot guarantee safety starting from an arbitrary state x_0. For instance, if the robot is about to crash into a wall, no action can ensure safety. We assume that the initial states $\mathcal{X}_0$ are ones where we can guarantee safety.

Definition III.5. A *safe equilibrium state* $x \in \mathcal{X}$ satisfies (i) $x \in \mathcal{X}_{\text{safe}}$, and (ii) $x = f(x, u_R^0, u_H)$ for all $u_H \in \mathcal{U}_H^0$.

We denote the set of safe equilibrium states by $\mathcal{X}_{\text{eq}}$. At a state $x \in \mathcal{X}_{\text{eq}}$, the robot and human can together ensure safety for an infinite horizon by taking actions u_R^0 and u_H for any $u_H \in \mathcal{U}_H^0$. In our driving example, $\mathcal{X}_{\text{eq}}$ contains states where both agents are at rest (i.e., their velocity is zero).

Assumption III.6. We have $\mathcal{X}_0 \subseteq \mathcal{X}_{\text{eq}}$.

In other words, the system starts at a safe equilibrium state where we can ensure safety for an infinite horizon.

Algorithm 1 Model predictive shielding modulo fault.

```
procedure $\pi_R(x)$
    $x' \leftarrow f_R(x, \hat{\pi}_R(x))$
    return if IsRec$(x')$ then $\hat{\pi}_R(x)$ else $u_R^0$ end if
end procedure
procedure IsRec$(x')$
    $X_0' \leftarrow \{x'\}$
    for $t \in \{0, ..., k-1\}$ do
        if $X_t' \not\subseteq \mathcal{X}_{\text{safe}}$ then return false end if
        $U_{R,t} \leftarrow$ if $t = 0$ then $\{u_R\}$ else $\{u_R^0\}$ end if
        $U_{H,t} \leftarrow \mathcal{U}_H^0$
        $X_{t+1}' \leftarrow F(X_t, U_{R,t}, U_{H,t})$
    end for
    return if $X_k \subseteq \mathcal{X}_{\text{eq}}$ then true else false end if
end procedure
```

IV. Model Predictive Shielding Modulo Fault

We describe our algorithm for constructing a robot controller $\pi_R : \mathcal{X} \to \mathcal{U}_R$ that is safe modulo fault. Our approach is based on *model predictive shielding (MPS)* [13, 14], which converts an arbitrary controller $\hat{\pi}_R : \mathcal{X} \to \mathcal{U}_R$ into a controller π_R that uses $\hat{\pi}_R$ but overrides it when it cannot ensure it is safe. The challenge is checking whether it is safe to use $\hat{\pi}_R$. The idea is to maintain the invariant that the current state is *recoverable*—i.e., that there is some sequence of actions each agent can take that safely brings the system to a stop.

Definition IV.1. Given $k \in \mathbb{N}$, a state $x' \in \mathcal{X}$ is *recoverable* (denoted $x' \in \mathcal{X}_{\text{rec}}$) if for $\vec{u}_R = (u_R^0, u_R^0, ...) \in \mathcal{U}_R^\infty$ and any $\vec{u}_H \in (\mathcal{U}_H^0)^\infty$, $\zeta_H(x', \vec{u}_H, \vec{u}_R) = (x_0', x_0, x_1', ...)$ satisfies (i) $x_t', x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \{0, ..., k\}$, and (ii) $x_k \in \mathcal{X}_{\text{eq}}$.

Now, our MPS modulo fault algorithm for computing π_R is shown in Algorithm 1. Here, IsRec checks whether $x' = f_R(x, \hat{\pi}_R(x))$ is recoverable. If so, π_R returns $\hat{\pi}_R(x)$; otherwise, it returns u_R^0. To check recoverability, IsRec overapproximates the reachable set of states after t steps as a set $X_t \subseteq \mathcal{X}$. It assumes given a *dynamics overapproximation* $F : 2^{\mathcal{X}} \times 2^{\mathcal{U}_R} \times 2^{\mathcal{U}_H} \to 2^{\mathcal{X}}$ mapping sets of states $X \subseteq \mathcal{X}$, sets of robot actions $U_R \subseteq \mathcal{U}_R$, and sets human action $U_H \subseteq \mathcal{U}_H$ to sets of states $F(X, U_R, U_H) \subseteq 2^{\mathcal{X}}$, that satisfies

$$f(x, u_R, u_H) \in F(X, U_R, U_H) \tag{2}$$

for all $x \in X$, $u_R \in U_R$, and $u_H \in U_H$. Then, IsRec checks whether (i) safety holds for every state $x_t \in X_t$ (i.e., $X_t \subseteq \mathcal{X}_{\text{safe}}$), and (ii) every state $x_k \in X_k$ is a safe equilibrium state (i.e., $X_k \subseteq \mathcal{X}_{\text{eq}}$). If both hold, then x is recoverable. We have the following (see Appendix A for a proof):

Theorem IV.2. *Assuming (2) holds, then our policy π_R is safe modulo fault (i.e., it satisfies Definition III.4).*

V. Evaluation

We have implemented our approach in a simulation for three robotics tasks. We consider an aggressive robot controller with and without the shield as well as a cross entropy method controller (CEM) designed to avoid humans. Also, we consider real humans interacting with the simulation via keyboard.

We focus on understanding whether our approach can ensure safety in aggressive driving scenarios; our approach can easily be tailored to drive more conservatively, which would further improve safety (but may reduce performance). We focus on settings where the human and the robot must compete to reach their goals. We tune the parameters of our MPS modulo fault algorithm (i.e., the robot backup action u_R^0 and the human backup action set $\mathcal{U}_H^0$) to be as aggressive as possible while still ensuring safety on the simulated humans. Furthermore, for our experiments with real-world humans, we strongly encourage them to try and reach their goal before the robot, albeit keeping safety as the top priority. Then, our results are designed to answer the following questions:

- Can MPS modulo fault can be used to ensure safety?
- Can MPS modulo fault outperform a handcrafted MPC based on CEM in terms of performance?

A. Experimental Setup

a) Robotics tasks: We consider three non-cooperative tasks (see Figure 2): (i) "merge": there are two lanes that merge—i.e., the robot is coming in from one lane and the humans from another; both their goals are to navigate the merge and reach their goal, (ii) "cross": both agents are moving towards an intersection from different directions—i.e., the robot is moving horizontally and the human is moving vertically; both their goals are to get to the other side of the intersection, (iii) "turn": an unprotected left turn—i.e., the human is driving without turning and the robot needs to make a left turn that crosses the human path.

b) Safety property: We assume the robot and human are each a rectangle; then, the safety property is that the the robot and human rectangles should not intersect.

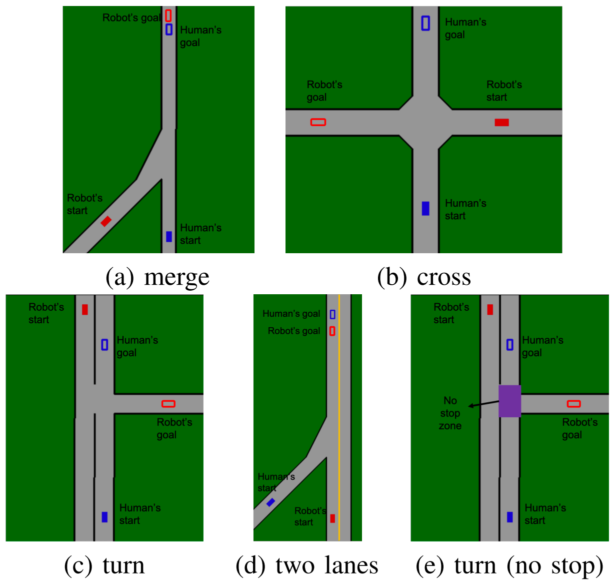

Fig. 2. Visualizations of the different tasks along with the initial positions and the goals for the robot and the human. The red box is the robot and the blue box is the human.

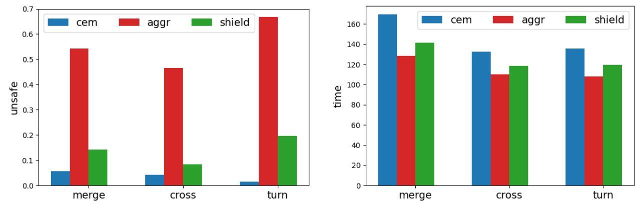

Fig. 3. Results with real humans, for the aggressive controller (red), the CEM MPC (blue), and our shielded aggressive policy (green). Left: Fraction of unsafe runs. Right: Time the robot takes to reach its goal in seconds.

c) Robot dynamics: The robot dynamics are the ones in our running example—i.e., its state is (x, y, v, θ), where (x, y) is position, v is velocity, and θ is orientation, and its actions are (a, ϕ), where a is acceleration and ϕ is steering angle.

d) Humans: We consider real human users interacting with the simulation via keyboard. They control the human using the up/down arrows to control acceleration and the left/right arrows to control steering angle. We asked the human users to prioritize safety first, but to drive aggressively to try and reach their goal before the robot. We also considered simulated humans, including multiple humans; see Appendix B.

e) Controllers: We consider three controllers for the robot: (i) an aggressive controller, (ii) a handcrafted MPC based on the cross-entropy method (CEM) designed to ensure safety without a shield, and (iii) our MPS modulo fault algorithm used in conjunction with the aggressive controller. We give details in Appendix B.

B. Experimental Results

We describe our experimental results, based on 18 users.

a) MPS modulo fault ensures safety for real humans: Next, we had real human users interact with our simulated robot via keyboard input. we show both the fraction of unsafe runs (left), and the time taken by the robot to reach the goal (right), including the aggressive controller (red), the MPC based on CEM (blue), and our shielded aggressive controller (green). As can be seen, for the aggressive controller, the robot gets to its goal the fastest, but is frequently unsafe. The MPC based on CEM is significantly safer; in this case, it is somewhat safer than our shielded aggressive controller. On the other hand, our shield controller reaches its goal significantly faster than the MPC, while being almost as safe as the MPC CEM. As described above, we set the shield parameters aggressively based on the simulated humans to ensure it could reach its goal; in practice, we could further improve safety by setting these parameters more conservatively and by tuning them to the real human driver data.

b) Alternative robot backup actions: A key feature of our approach is that we can flexibly design the robot backup action to ensure safety. To demonstrate this flexibility, we design an alternative backup action that pulls the robot over to the shoulder of a highway. Note that this backup policy is time varying—i.e., the robot steering depends on the current state. We test this backup policy with simulated humans on the task in Figure 2 (d), where there are two lanes on the highway and an on-ramp that merges onto the highway. The human is on the on-ramp and the robot is on the highway. To avoid collisions, the robot can pull over to the right-most lane. Figure 4 shows the fraction of the unsafe runs (leftmost), and the time the robot takes to reach its goal (second from left) for all three controllers—aggressive (red), the MPC based on CEM (blue), and our shielded aggressive controller with the pull over backup policy (green). Our shielded controller is always safe and is significantly faster than the MPC.

We also design a robot backup action that avoids stopping in the middle of an intersection and blocking it, which is often illegal. To this end, we modify the turn task to include a no-stop zone (shown in Figure 2 (e)) where the robot is prohibited from stopping. In this zone, the robot backup action does not come to a stop immediately; instead, it drives through the zone until it crosses the intersection, and only brakes once it has fully cleared the intersection. The results for this experiment using simulated humans are shown in Figure 4 (right). We compare the original shielded controller (“shield”) that may stop in the intersection with the new one that adheres to the no-stop zone in Figure 2 (e) (“shield++”). In this case, both the controllers were always safe; instead, we show the fraction of runs where the robot stops in the intersection (second from the right), and the time the robot takes to reach its goal (rightmost). The new shielded controller takes slightly longer to reach the goal, but never stops in the intersection.

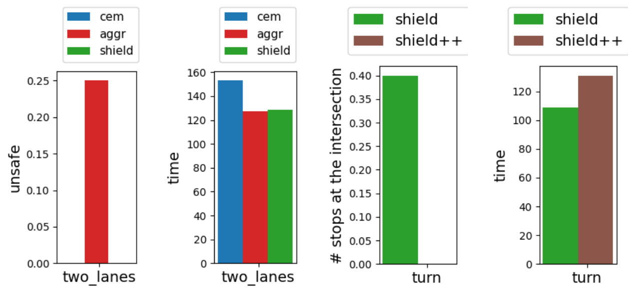

Fig. 4. Results for alternative robot backup actions with simulated humans. For the “pull over” backup action, we show the fraction of unsafe runs (leftmost) and the time the robot takes to reach its goal in seconds (second from the left), for the aggressive controller (red), the CEM MPC (blue), and our shielded aggressive controller (green). For the “no-stop zone” backup action, we show the number of stops in the intersection (second from the right) and the time the robot takes to reach its goal in seconds (rightmost), for the original (green, “shield”) and the new (brown, “shield++”) shielded controllers; both controllers are always safe.

VI. Conclusion

We have proposed an approach for ensuring safety in human-interactive robotics systems. We define a notion of safety that models human behavior by specifying their backup behaviors, and propose our MPS modulo fault algorithm for ensuring our safety with respect to this model. Finally, we validate our approach on both real and simulated humans.

References

[1] K. Strabala, M. K. Lee, A. Dragan, J. Forlizzi, S. S. Srinivasa, M. Cakmak, and V. Micelli, “Toward seam-

less human-robot handovers," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 112–132, 2013.

[2] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 301–308.

[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[4] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 163–168.

[5] 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*, vol. 2. Ann Arbor, MI, USA, 2016.

[6] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 66–73.

[7] K. Eder, C. Harper, and U. Leonards, "Towards the safety of human-in-the-loop robotics: Challenges and opportunities for safety assurance of robotic co-workers'," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2014, pp. 660–665.

[8] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[9] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," in *RSS*, 2018.

[10] D. Sadigh, S. S. Sastry, S. A. Seshia, and U. Berkeley, "Verifying robustness of human-aware autonomous cars," *IFAC-PapersOnLine*, vol. 51, no. 34, pp. 131–138, 2019.

[11] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[12] P. Cousot, "Abstract interpretation," in *In POPL*. Citeseer, 1977.

[13] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 7130–7135.

[14] S. Li and O. Bastani, "Robust model predictive shielding for safe reinforcement learning with stochastic dynamics," in *ICRA*, 2019.

[15] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.