Safe Human-Interactive Control Modulo Fault

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 a 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 without modeling the human as an unconstrained adversary. We propose a novel approach to ensuring safety in these settings. Our approach focuses on defining actions that both the robot and human are expected to take to avoid an accident—e.g., brake to avoid rear-ending the other agent. These actions should be defined in so that if one of the agents does not take these actions and an accident occurs, then we can reasonably consider them to be at fault for that accident. We refer to this notion of safety as *safety modulo fault*. Then, we propose an algorithm that overrides an arbitrary given controller as needed to ensure that the robot is safe modulo fault. We evaluate our approach in a simulated environment, interacting with both real and simulated humans.**

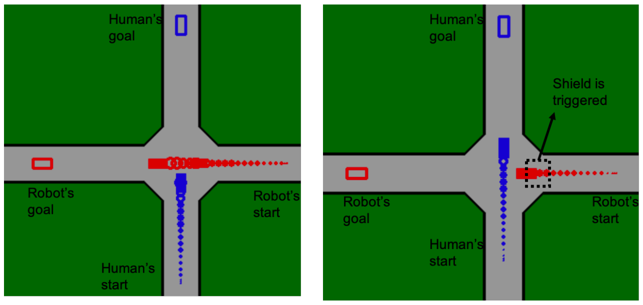

Fig. 1. Trajectories showing a robot (red) and a human (blue) interacting at an intersection (for 25 timesteps). Left: The robot passes before the human, leveraging the fact that a responsible human would slightly brake to allow the robot to cross safely. Right: Human arrives at the intersection first;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

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]. As a consequence, there has been much interest in designing planning and control algorithms for human-robot interaction.

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 that needs to make an unprotected left turn at a busy intersection [5]. The key challenge is how to define safety for human-interactive robots. Modeling the human as an adversary is one approach to defining safety, but would be prohibitively conservative.

Another approach is to train a machine learning model to predict human actions [8], [9], and ensure safety with respect to this model. If the model captures all actions exhibited by humans, then this approach ensures safety. However, there are a number of reasons why the model may not satisfy this property. For instance, different humans may exhibit very different behaviors [6]—e.g., people in a lab may act differently than people on a street, or people in different regions may act differently. Collecting data from all possible settings can be very challenging. If a human behavior is not exhibited in the data used to train the model, then the model may not account for it. More fundamentally, even the best machine learning models rarely make zero errors, and errors can correspond to actions missed by the model.

An alternative approach, which has been termed *responsibility-sensitive safety (RSS)* [10], is to manually specify the range of acceptable robot actions in various scenarios. As the designer of the robot controller, it is our job to ensure that acceptable actions only include safe actions—e.g., according to legal or social norms. In particular, if the robot acts within this range, then any accident should be considered the fault of the human—e.g., if they cause an accident by running a red light. However, manually defining acceptable robot actions for all possible scenarios is challenging, especially for robots operating in open-world environments. For instance, in [10], they formally define acceptable actions for a limited number of scenarios such as changing lanes, but autonomous driving is notoriously challenging precisely due to the vast number of corner cases.

We propose a novel approach for ensuring safety in human-interactive robotics systems, based on two concepts:

- **Modeling fault:** Rather than specify actions the robot is *allowed* to take (as in RSS), we specify actions the human is *expected* to take to avoid an accident.
- **Conservative overapproximation:** We conservatively overapproximate the dynamics for a given fault model.

First, our notion of fault captures the idea that we can reasonably expect the human to 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

[1]Jeevana Priya Inala and Armando Solar-Lezama are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA {jinala, asolar}@csail.mit.edu
[2]Yecheng Jason Ma and Osbert Bastani are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA {jasonyma,obastani}@seas.upenn.edu
[3]Xin Zhang is with the Department of Computer Science and Technology at Peking University, Beijing, 100871, China xin@pku.edu.cn

to also slow down to avoid rear-ending it.[1][2] 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. Similarly, we may restrict the robot from coming to a stop in an intersection. Specifying a notion of fault provides a way to define safety; we refer to such a safety constraint as *safety modulo fault*.

Our concept of fault is related to RSS, but differs in two key ways: (i) we specify constraints on the human actions rather than the robot actions, and (ii) rather than specify the set of *all* actions that the robot is allowed to take, we only need to specify a *single* sequence of actions that the human is expected to take if necessary to avoid an accident.

Next, given a definition of fault, we propose an algorithm for ensuring safety modulo fault. Rather than design a specific controller that tries to achieve the desired goal subject to safety modulo fault, we decompose the problem into two parts: (i) designing a controller that achieves the goal, and (ii) modifying this controller to ensure safety modulo fault. More precisely, we build on the concept of *shielding*, which composes a high-performing controller with a backup controller in a way that guarantees safety [12], [13]. In particular, we extend the *model predictive shielding (MPS)* algorithm [14], [15] to the setting of safety modulo fault. Our algorithm, called *MPS modulo fault*, converts an arbitrary given controller into one that is guaranteed to be safe, while trying to use the given controller as frequently as possible. At a high level, it does so by using on-the-fly verification to determine whether it is safe modulo fault to use the given controller. If so, it uses the given controller; otherwise, it uses an appropriate backup action.

Finally, we empirically evaluate our approach in a simulation, including both settings where the humans are simulated and settings where the human is controlled by an real person via keyboard inputs. We demonstrate that our MPS modulo fault algorithm enables the robot to avoid accidents both with real and simulated humans, even when combined with a naïve controller that altogether ignores the humans.

Figure 1 illustrates how our MPS modulo fault algorithm ensures safety while interacting with a human driver without being overly cautious. It assumes that the human will at least slightly brake to avoid an accident (left). If it still cannot guarantee safety, then it allows the human to go first (right).

In summary, our contributions are: a formal definition of safety modulo fault for human-interactive robotics systems (Section III), an algorithm for ensuring safety modulo fault of an arbitrary given controller (Sections IV), and an empirical evaluation of our approach (Section V).

[1]We note that the human is *not* expected to take evasive maneuvers such as swerving or speeding up to avoid an accident; also, we assume a reasonable amount of time for the human to brake and come to a stop.

[2]This kind of behavior is exhibited by human drivers, who occasionally come to a stop in the middle of the road to pick up or drop off passengers [11]. Thus, in this situation, we believe it is reasonable to expect human drivers to slow down to avoid a collision. Furthermore, in our approach, the robot would only take this kind of action to avoid an accident.

II. Preliminaries

We consider a system with a robot R and a human H. In particular, we have $x_{t+1} = f(x_t, u_{R,t}, u_{H,t})$, where $f : \mathcal{X} \times \mathcal{U}_R \times \mathcal{U}_H \to \mathcal{X}$ is the dynamics, $\mathcal{X} \subseteq \mathbb{R}^{n_X}$ is the joint state space, $\mathcal{U}_R \subseteq \mathbb{R}^{n_{U,R}}$ are the robot actions, and $\mathcal{U}_H \subseteq \mathbb{R}^{n_{U,H}}$ are the human actions. We assume the robot acts first, and then the human (i.e., a Stackelberg game) [5]. Given an initial state $x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$ and two action sequences

$$\vec{u}_R = (u_{R,0}, u_{R,1}, ...) \subseteq \mathcal{U}_R$$
$$\vec{u}_H = (u_{H,0}, u_{H,1}, ...) \subseteq \mathcal{U}_H$$

for R and H, respectively, the *trajectory* generated by these actions from x_0 is the sequence $(x_0, x_1, ...) \subseteq \mathcal{X}$ where $x_{t+1} = f(x_t, u_{R,t}, u_{H,t})$.

As a running example, we consider an autonomous car. For this problem, R is an autonomous driving robot and H is a pedestrian or a human driver. A state $x \in \mathcal{X} = \mathbb{R}^8$ is a vector $x = (x_R, y_R, v_R, \theta_R, x_H, y_H, v_H, \theta_H)$ representing the positions $(x_R, y_R), (x_H, y_H)$, velocities v_R, v_H, and angles θ_H, θ_R of agents H and R, respectively. The actions are $\mathcal{U}_R = \mathcal{U}_H = \mathcal{U} = \mathbb{R}^2$, where $u = (\phi, a)$ represents the steering angle ϕ and the acceleration a; we assume that $|\phi| \le \phi_{\text{max}}$ and $|a| \le a_{\text{max}}$ are bounded by constants $\phi_{\text{max}}, a_{\text{max}} \in \mathbb{R}_{>0}$. The dynamics are

$$\begin{aligned}
f(x, u_R, u_H) &= x + g_R(x, u_R) + g_H(x, u_H) \\
g_R(x, u_R) &= (v_R \cos\theta_R, v_R \sin\theta_R, a_R, v_R\phi_R, 0, 0, 0, 0) \\
g_H(x, u_H) &= (0, 0, 0, 0, v_H \cos\theta_H, v_H \sin\theta_H, a_H, v_H\phi_H).
\end{aligned}$$

For simplicity, we assume the agents cannot go backwards—i.e., the dynamics implicitly impose $v_R, v_H \ge 0$.

Given a safe region $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$, our goal is to ensure that the system stays in $\mathcal{X}_{\text{safe}}$. In our example, safety means the robot and the human have not collided, i.e.,

$$\mathcal{X}_{\text{safe}} = \{x \in \mathcal{X} \mid \|(x_R, y_R) - (x_H, y_H)\| \ge d_{\text{safe}}\},$$

for some constant $d_{\text{safe}} \in \mathbb{R}_+$.

Definition II.1. A trajectory $x_0, x_1, ...$ is *safe* if $x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$, and *unsafe* otherwise.

III. Safety Modulo Fault

Ensuring safety in the presence of an adversarial human would be impossible or at least significantly degrade performance—e.g., to avoid an accident with an adversarial human driver, the robot would have to maintain a very large distance. Thus, to ensure safety, we must make assumptions about the behavior of the human. Ideally, we want to make the minimal possible assumptions about the behavior of the human while still accounting for all possible behaviors of a human acting in a responsible way. Then, as long as the robot acts in a way that is safe according to these assumptions, the human would be at fault for any resulting accident.

The key challenge is devising a reasonable set of assumptions on the human. Intuitively, our assumptions are based on the idea that if the human can act in a safe way to avoid an accident, then they do so (Assumption III.2). In addition, we need to formalize what it means for the human to "be able to act in a safe way". We give the human great leeway in what safe actions they consider—for instance, we could assume the human always considers slowing down in *some* manner to ensure safety (Assumption III.8). Finally, just as we make assumptions about how the human acts, we expect the human to make assumptions about how the robot may act. Again, we give the human great leeway in doing so—for instance, we could assume that the human always accounts for the possibility that the robot may take a safe action such as gradually braking to avoid an accident (Assumption III.6). We formalize our assumptions and safety notion below.

a) Assumptions on the human objective: Our definition of safety is based on a model of a human acting according to a maximin objective in a receding horizon fashion. In this objective, the "min" is the worst-case over a set of action sequences that the human predicts the robot might take, and the "max" portion is over the human's own actions. In other words, the human plans optimally according to their own objective, while conservatively accounting for all actions they anticipate the robot might take.

Assumption III.1. Given $x_0 \in \mathcal{X}$ and $u_{R,0} \in \mathcal{U}_R$, let

$$\vec{u}_H^* = \underset{\vec{u}_H \subseteq \mathcal{U}_H}{\arg\max} \min_{\vec{u}_R \subseteq \hat{\mathcal{U}}_R} \sum_{t=0}^{\infty} \gamma^t \cdot r_H(x_t, u_{R,t}, u_{H,t}), \quad (1)$$

where $\gamma \in (0,1)$ is a discount factor, r_H is the human reward function, $(x_0, x_1, ...)$ is the trajectory generated by $\vec{u}_R$ and $\vec{u}_H$ from x_0, and $\hat{\mathcal{U}}_R$ is the set of actions the human predicts that the robot may take. Then, the human takes action $u_{H,0}^*$.

Thus, (1) says the human conservatively assumes the robot may take any action $\vec{u}_R \subseteq \hat{\mathcal{U}}_R$. In addition, we assume the human reward for reaching an unsafe state is $-\infty$.

Assumption III.2. For any $u_R \in \mathcal{U}_R$ and $u_H \in \mathcal{U}_H$, we have $r_H(x, u_R, u_H) = -\infty$ if and only if $x \notin \mathcal{X}_{\text{safe}}$.

That is, the human driver always acts to avoid an accident. Other than Assumption III.2, r_H can be arbitrary.

Remark III.3. The assumption that the human acts optimally is strong. We use it for simplicity; in fact, we only need to assume that the human chooses an action sequence $\vec{u}_H^*$ with a reward $> -\infty$. Then, Assumption III.2 & III.1 say that the human chooses actions $\vec{u}_H^*$ that will avoid an accident assuming the robot takes actions in $\hat{\mathcal{U}}_R$.

Remark III.4. For simplicity, we have assumed that $\mathcal{U}_H$ and $\hat{\mathcal{U}}_R$ are time-invariant. Our approach can easily be extended to the case where they are time varying.

Remark III.5. The fact that real-world human drivers have accidents contradicts Assumption III.2. There are two reasons such an accident may happen: (i) there was a safe action sequence $\vec{u}_H \subseteq \mathcal{U}_H$ that the human driver failed to take, or (ii) if the other driver (i.e., the "robot") takes an action $u_R \notin \hat{\mathcal{U}}_R$ that the first driver failed to anticipate. We expect (i) might happen if one driver does not see the other. Alternatively, (ii) might happen if one of the two drivers is acting aggressively—i.e., either $\hat{\mathcal{U}}_R$ is overly optimistic and one driver aggressively ignores other's actions (e.g., braking), or one driver takes aggressive actions outside of $\hat{\mathcal{U}}_R$ (e.g., cutting in front of other driver).

b) Assumption on action sets: Assuming the human is acting according to (1), the key challenge for the robot to plan safely is that it does not know the human action set $\mathcal{U}_H$ or the human-predicted 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 sets.

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

Assumption III.6. We are given a *robot backup action* $u_R^0 \in \mathcal{U}_R$ that is predicted 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.

Remark III.7. For simplicity, we have assumed that u_R^0 is time-invariant. Our approach easily extends to the case where u_R^0 is time varying.

Next, we could make a similar assumption that we are given a human backup action $u_H^0 \in \mathcal{U}_H$—e.g., applying the brakes. However, we might not know how quickly or slowly the human driver is able to brake, or in what direction they may steer. Thus, we make the following *weaker* assumption:

Assumption III.8. We are given a *human backup action set* $\mathcal{U}_H^0$ that satisfies $\mathcal{U}_H^0 \cap \mathcal{U}_H \neq \varnothing$.

That is, there is *some* action $u_H^0 \in \mathcal{U}_H^0$ that the human considers taking (i.e., $u_H^0 \in \mathcal{U}_H$), but we may not know u_H^0.

As its name suggests, $\mathcal{U}_H^0$ should only include actions that are considered to be backup actions that might be taken by the human driver to avoid an accident. For example, $\mathcal{U}_H^0$ may contain all actions where the human driver decelerates by at least some rate; this choice allows the human to slow down more quickly or to steer in any direction while slowing down. Intuitively, our algorithm uses $\mathcal{U}_H^0$ to check that the human always has some way to come to a stop to avoid an accident.

Remark III.9. Our approach ensures safety modulo fault for any given $\mathcal{U}_H^0$, but having a set of backup actions such as braking is necessary for the robot to not be overly conservative.

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.10. 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 $\pi_H : \mathcal{X} \to \mathcal{U}_H$ satisfying Assumptions III.1, III.2, III.6, & III.8, and any $x_0 \in \mathcal{X}_0$, the trajectory $x_0, x_1, ...$ generated using $u_{R,t} = \pi_R(x_t)$ and $u_{H,t} = \pi_H(x_t)$ is safe.

Finally, we cannot guarantee safety starting from an arbitrary state x_0. For instance, if the robot is about to crash

Algorithm 1 Model predictive shielding modulo fault.

```
procedure $\pi_R(x)$
    if IsRec$(x, \hat{\pi}_R(x))$ then
        return $\hat{\pi}_R(x)$
    else
        return $u_R^0$
    end if
end procedure
procedure IsRec$(x, u_R)$
    $X_0 \leftarrow \{x\}$
    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
    if $X_k \subseteq \mathcal{X}_{\text{eq}}$ then
        return true
    else
        return false
    end if
end procedure
```

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.11. 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.12. 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.

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 *shielding* [13]—it takes as input an arbitrary controller $\hat{\pi}_R : \mathcal{X} \to \mathcal{U}_R$ and modifies it to construct π_R. Intuitively, π_R overrides $\hat{\pi}_R$ when it cannot ensure it is safe.

The challenge is checking whether it is safe to use $\hat{\pi}_R$. Model predictive shielding (MPS) is an approach to shielding that checks safety online based on the following [14], [15]:

Definition IV.1. Given hyperparameter $k \in \mathbb{N}$, $(x, u_R) \in \mathcal{X} \times \mathcal{U}_R$ is *recoverable* if for the robot action sequence $\vec{u}_R = (u_R, u_R^0, ..., u_R^0) \in \mathcal{U}_R^k$, for all human action sequences

$$\vec{u}_H = (u_{H,0}, u_{H,1}, ..., u_{H,k-1}) \in (\mathcal{U}_H^0)^k,$$

the trajectory $(x_0, x_1, ..., x_k)$ generated from $x_0 = x$ using actions $\vec{u}_R, \vec{u}_H$ satisfies (i) $x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \{0, ..., k\}$, and (ii) $x_k \in \mathcal{X}_{\text{eq}}$.

In other words, (x, u_R) is recoverable if, after using action u_R, the robot can subsequently ensure safety by using its backup action u_R^0 for any sequence of backup actions taken by the human. We let $\mathcal{Z}_{\text{rec}}$ denote the set of recoverable pairs.

Now, our MPS modulo fault algorithm for computing π_R is shown in Algorithm 1. Here, IsRec checks whether $(x, \hat{\pi}_R(x))$ is recoverable. If so, π_R returns $\hat{\pi}_R(x)$; otherwise, it returns the robot backup action u_R^0.

Next, we describe how IsRec checks recoverability. The challenge is that we need to guarantee safety with respect to *all* human backup action sequences $\vec{u}_H \in (\mathcal{U}_H^0)^k$. To do so, IsRec *conservatively overapproximates* recoverability—i.e., if it says that (x, u_R) is recoverable, then it must be recoverable, but it may say (x, u_R) is not recoverable even if it is. In particular, IsRec overapproximates the reachable set of states after t steps as a subset $X_t \subseteq \mathcal{X}$.

More precisely, the *dynamics overapproximation* $F : 2^{\mathcal{X}} \times 2^{\mathcal{U}_R} \times 2^{\mathcal{U}_H} \to 2^{\mathcal{X}}$ is a function from 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 $F(X, U_R, U_H) \subseteq 2^{\mathcal{X}}$; it should satisfy

$$f(x, u_R, u_H) \in F(X, U_R, U_H) \qquad (2)$$

for all $x \in X$, $u_R \in U_R$, and $u_H \in U_H$. In other words, $F(X, U_R, U_H)$ should contain *at least* the states that can be reached from $x \in X$ by taking actions $u_R \in U_R$ and $u_H \in U_H$. Intuitively, computing F in a way that this property holds with equality may be computationally intractable, but there exist tractable overapproximations—e.g., based on polytopes [16], [17] or ellipsoids [18], [19]. In general, this approach is known as *abstract interpretation* [20]. We describe the overapproximation we use for our autonomous driving example in Appendix B.[3]

Finally, 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 these properties hold, then x is guaranteed to be recoverable. We have the following guarantee (see Appendix A for a proof):

Theorem IV.2. *Assuming (2) holds, π_R is safe modulo fault.*

V. Evaluation

We have implemented our approach in a simulation for three robotics tasks. For the robot, we consider an aggressive controller with and without the shield as well as a cross entropy method controller (CEM) that is designed to avoid humans. For the human, we use both simulated humans based on a social forces model of pedestrians [21], as well as real humans interacting with the simulation via keyboard inputs.

Our goal is to understand how our approach can ensure safety in aggressive driving scenarios. Thus, we focus on settings where the human (either simulated or real) and the robot must compete to reach their goals. We tune the

[3]https://obastani.github.io/docs/safehumancontrol.pdf

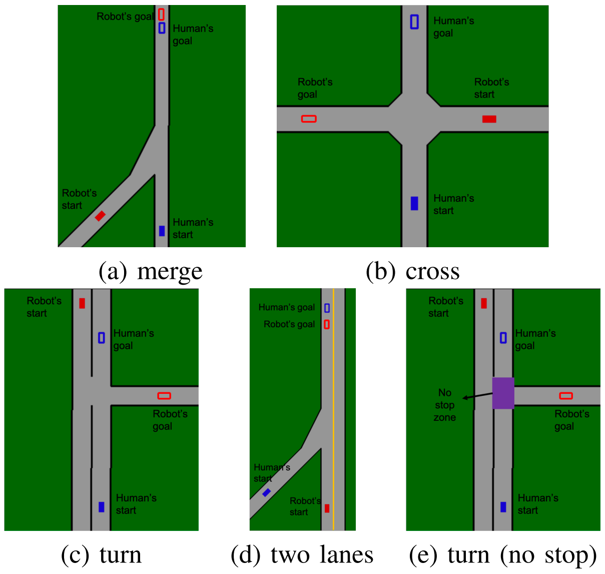

Fig. 2. Visualizations of the different tasks along with the initial positions and the goals for the robot and the human. The red box is the robot and the blue box is the human.

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 with real and simulated humans?
- 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 robotics tasks (depicted in Figure 2). In the first task ("merge"), there are two lanes that merge—i.e., the robot is coming in from one lane and the humans from another; the robot and human goals are to navigate the merge and reach their goal. The second task ("cross") has both the human and the robot moving towards an intersection from different directions—i.e., the robot is moving horizontally and the human is moving vertically; the robot and human goals are to get to their goal on the other side of the intersection. The third task ("turn") is an unprotected left turn—i.e., the humans are 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.

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. We assume $|a| \leq a_{\text{max}}$, $|\phi| \leq \phi_{\text{max}}$, and $0 \leq v \leq v_{\text{max}}$.

d) Simulated humans: For simulated humans, we use the social force model [21], which includes potential forces that cause each human to avoid the robot, other humans, and walls, while trying to reach their goal.

e) Real humans: We also considered 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.

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

The first controller is an "aggressive controller" that ignores the humans and moves directly towards the goal as fast as possible. For tasks where nonlinear trajectories are required to reach the goal, we manually specify a sequence of intermediate subgoals that the controller targets; once it reaches each subgoal, it continues to the next one.

The second controller is a model-predictive controller (MPC) that aims to avoid colliding with the human. We use a planning algorithm based on the cross-entropy method (CEM). Then, it chooses the action that attempts to optimize its objective over the planning horizon. We use a handcrafted objective that provides a positive reward for progressing towards its goal and a large negative penalty for colliding with the human. To predict collisions, it forecasts the behavior of the human over the planning horizon by extrapolating their position based on their current velocity (i.e., constant velocity assumption). Finally, for the goal-reaching portion of the objective, we use subgoals the same way we do for the aggressive controller.

The third controller is our MPS modulo fault algorithm used with the aggressive controller. The robot backup action is $u_R^0 = (0, -1)$ where $\phi = 0$ is the steering angle and $a = -1$ is the acceleration. The human backup action set is

$$\mathcal{U}_H^0 = \left\{ (\phi, a) \mid \phi \in \left[-\frac{\pi}{10}, \frac{\pi}{10}\right], \; a \in \left[-1, -\frac{1}{2}\right] \right\}.$$

That is, the human predicts that the robot may gradually brake without changing direction, and the human considers braking gradually (or hard) while steering up to some angle.

B. Experimental Results

We describe our experimental results. For simulated humans, all results shown are averaged over 100 simulations. For real humans, the results are based on 18 users.

a) MPS modulo fault ensures safety for simulated humans: In Figure 3, 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 rate of unsafe runs is very high since the robot ignores the human

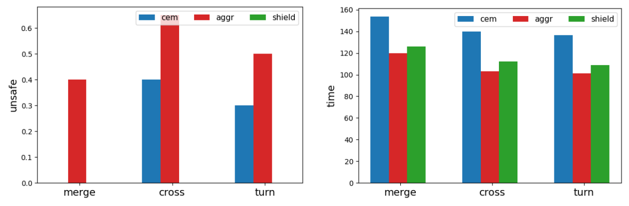

Fig. 3. Results with simulated humans, for the aggressive controller (red), the CEM MPC (blue), and our shielded aggressive controller (green). Left: Fraction of unsafe runs. Right: Time the robot takes to reach its goal in seconds.

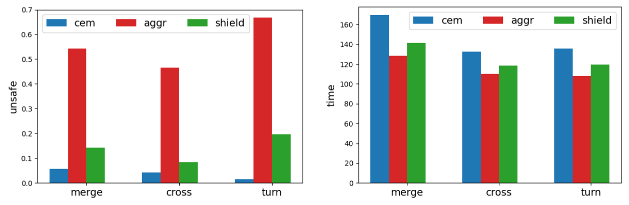

Fig. 4. Results with real humans, for the aggressive controller (red), the CEM MPC (blue), and our shielded aggressive policy (green). Left: Fraction of unsafe runs. Right: Time the robot takes to reach its goal in seconds.

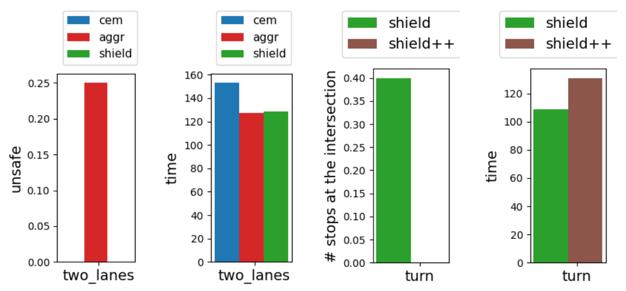

Fig. 5. Results for alternative robot backup actions with simulated humans. For the "pull over" backup action, we show the fraction of unsafe runs (leftmost) and the time the robot takes to reach its goal in seconds (second from the left), for the aggressive controller (red), the CEM MPC (blue), and 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.

to get to its goal. Next, for the MPC based on CEM, the rate of unsafety is lower but still not zero. However, the CEM policy takes significantly longer to reach its goal compared to the aggressive policy. Finally, our shielded aggressive controller is always safe, yet only takes a small amount of time longer to reach its goal compared to the aggressive policy; in particular, it is significantly faster than the MPC. These comparisons demonstrate that our approach greatly improves safety without significantly reducing time to goal.

b) 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. 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 ensure safety by setting these parameters more conservatively and by tuning them to the real human driver data.

c) 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. In contrast to the previous backup policy, this one 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 5 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.

In addition, we can also design a robot backup action that avoids stopping in the middle of an intersection and blocking it, which is illegal in most places. 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 5 (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.

VI. Conclusion

We have proposed an approach for ensuring safety in human-interactive robotics systems. We define a notion of safety that models fault rather than human behaviors, and propose our MPS modulo fault algorithm that ensures safety for an arbitrary robot controller. We evaluate our approach on both real and simulated humans. There are a number of important directions for future work—e.g., designing algorithms to infer the shield parameters or adapting our approach to collaborative robotics settings.

References

[1] K. Strabala, M. K. Lee, A. Dragan, J. Forlizzi, S. S. Srinivasa, M. Cakmak, and V. Micelli, "Toward seamless 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 *2013 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] 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.

[9] 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.

[10] 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.

[11] F. Bastani, O. Bastani, S. He, A. Balasingam, Z. Jiang, R. Mittal, M. Alizadeh, H. Balakrishnan, T. Kraska, and S. Madden, "Skyquery: Optimizing video queries over uavs," 2020. [Online]. Available: https://favyen.com/skyquery.pdf

[12] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1424–1431.

[13] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] 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.

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

[16] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear analysis: hybrid systems*, vol. 4, no. 2, pp. 233–249, 2010.

[17] S. Sadraddini and R. Tedrake, "Linear encodings for polytope containment problems," *arXiv preprint arXiv:1903.05214*, 2019.

[18] L. Asselborn, D. Gross, and O. Stursberg, "Control of uncertain nonlinear systems using ellipsoidal reachability calculus," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 50–55, 2013.

[19] T. F. Filippova, "Ellipsoidal estimates of reachable sets for control systems with nonlinear terms," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 15 355–15 360, 2017.

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

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