

Safe MPC Alignment With Human Directional Feedback

Zhixian Xie[✉], Graduate Student Member, IEEE, Wenlong Zhang[✉], Member, IEEE, Yi Ren[✉], Zhaoran Wang, George J. Pappas[✉], Fellow, IEEE, and Wanxin Jin[✉]

I. INTRODUCTION

Abstract—In safety-critical robot planning or control, manually specifying safety constraints or learning them from demonstrations can be challenging. In this article, we propose a certifiable alignment method for a robot to learn a safety constraint in its model predictive control policy with human online directional feedback. To the best of authors' knowledge, it is the first method to learn safety constraints from human feedback. The proposed method is based on an empirical observation: human directional feedback, when available, tends to guide the robot toward safer regions. The method only requires the direction of human feedback to update the learning hypothesis space. It is certifiable, providing an upper bound on the total number of human feedback in the case of successful learning, or declaring the hypothesis misspecification, i.e., the true implicit safety constraint cannot be found within the specified hypothesis space. We evaluated the proposed method using numerical examples and user studies in two simulation games. In addition, we implemented and tested the proposed method on a real-world Franka robot arm performing mobile water-pouring tasks. The results demonstrate the efficacy and efficiency of our method, showing that it enables a robot to successfully learn safety constraints with a small handful (tens) of human directional corrections.

Index Terms—Constraint inference, human–robot interaction, learning from human feedback, model predictive control (MPC).

Received 30 October 2025; accepted 7 November 2025. Date of publication 12 January 2026; date of current version 26 January 2026. This work was supported by ABRC under Grant RFGA2024-022-013. This article was recommended for publication by Associate Editor A. Cherubini and Editor J. Kober upon evaluation of the reviewers' comments. (Corresponding author: Wanxin Jin.)

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by ASU's Institutional Review Board under Application No. STUDY00020060, and performed in the line with Title 45 of the Code of Federal Regulations, Part 46 (45 CFR 46).

Zhixian Xie and Wanxin Jin are with the Intelligent Robotics and Interactive Systems (IRIS) Lab, School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ 85281 USA (e-mail: zxxie@asu.edu; wanxinjin@gmail.com).

Wenlong Zhang is with Polytechnic School, Arizona State University, Tempe, AZ 85281 USA (e-mail: wenlong.zhang@asu.edu).

Yi Ren is with the School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ 85281 USA (e-mail: yiren@asu.edu).

Zhaoran Wang is with the Departments of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208 USA (e-mail: zhaoranwang@gmail.com).

George J. Pappas is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: pappasg@seas.upenn.edu).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TRO.2026.3651678>, provided by the authors.

Digital Object Identifier 10.1109/TRO.2026.3651678

IN ROBOT policy design, safety is always the top priority, especially in safety-critical applications, such as robots operating near humans, where safety violations can lead to irreversible losses. While extensive work in safe control [1] and reinforcement learning [2] focuses on designing safe policy given robot motion constraints, less attention has been paid to how to specify the safety constraints in the first place. In many applications, specifying proper safety constraints can be challenging. First, *explicit safety boundary for robot behavior can be hard to model*. For example, when delivering food in a crowded restaurant, a service robot must balance the tray to avoid spills; this requires constraints on robot speed, food weight distributions, and interaction with its surroundings, and all those constraint aspects can be complex to model. Second, *user's perception of safety may vary*. In the same example, some diner may feel secure with the robot staying within a foot when delivering, while others may feel uncomfortable with such proximity and prefer the robot at a slight distance.

The challenges in safety specifications lead to two questions. Q1: can a robot learn and align its safety constraints with human intent in an *interactive* manner? Q2: can such a learning process be *user-data efficient* such that a robot can quickly adapt to different users? Recent work [3], [4], [5] has explored learning safety constraints from human demonstrations. However, acquiring safety-informative demonstrations could be challenging. Learning from human preference [6], [7], [8], [9], [10], [11], [12] offers a new paradigm, where a user gives preference of one trajectory over the other, and such information for robot learning. Despite the success in reward alignment [6], the query trajectories can be difficult to judge, especially in early learning stages. To address such, the authors in [13], [14], [15], [16], [17], [18], and [19] proposed learning from human physical corrections. While promising, the recent work [20] suggests those methods are potentially sensitive to correction magnitude: human corrections can lead to overshooting and inefficiency of learning. Importantly, all those feedback learning methods primarily focus on learning rewards and do not address the problem of learning safety constraints.

This article aims to answer the previous two questions by proposing a new paradigm for a robot to efficiently learn safety constraints in its model predictive control (MPC) policy with human feedback. We name the method *safe MPC alignment*. As shown in Fig. 1, a human user, observing the robot's potentially unsafe behavior, provides feedback to the robot. Upon receiving human feedback, the robot immediately updates the safety constraints in its MPC policy and continues with the updated policy. Different from learning from feedback, our method only requires human *directional corrections* [20]. For example, when

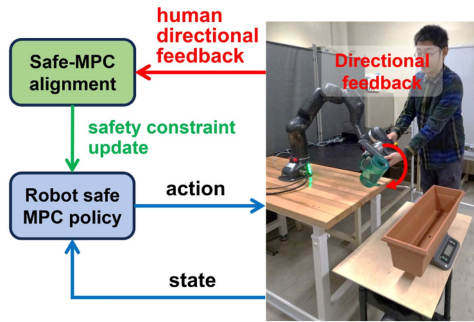


Fig. 1. Illustration of the proposed safe MPC alignment. Our method online updates the constraint in robot's safe MPC policy based on human directional feedback, such that the robot eventually aligns the safe MPC policy with human intent for the safety-critical task.

a human user corrects an autonomous car, he/she only needs to give the command of “turn left” or “turn right,” without dictating how much the car needs to turn. The *safe MPC alignment* only utilizes the direction of human correction, regardless of its magnitude. As shown in our previous work [20], directional feedback provides a consistently large and flexible correction space, overcoming the over-shooting correction. Importantly, the proposed alignment is able to guarantee the exponential convergence with finite rounds of human corrections, or output failure with certified hypothesis misspecification. Specifically, we highlight our main contributions as follows.

- 1) We propose the method of *safe MPC alignment* with human directional feedback. It enables a robot to learn a safe MPC policy online from human directional corrections. To the best of authors' knowledge, it is the first work for interactive learning of safety constraints from human feedback.
- 2) We theoretically show the human-data efficiency of the *safe MPC alignment*: the proposed method is guaranteed to have exponential convergence with finite human corrections. Furthermore, the proposed method enables the verification of hypothesis space misspecification.
- 3) Extensive user studies have been conducted to validate the proposed method, including two MuJoCo-based simulation games and a real-world experiment on the Franka robot arm. The results show the effectiveness and efficiency of our method.

The rest of this article is organized as follows. Section II reviews the related work, and Section III formulates the safe MPC alignment problem. The main method is presented in Section IV. Section V gives the implementation details and the main results. Sections VI–VIII provide the experiments, user studies and discussions. Finally, Section IX concludes this article.

II. RELATED WORK

A. Learning From Demonstrations

Under the general theme of learning from human data, the proposed method is loosely connected to the topic of learning from demonstrations, where costs/rewards are learned from expert data using inverse reinforcement learning (IRL) methods [21]. IRL methods typically run offline: human demonstrations are first collected, and then reward/cost learning is performed [21],

[22], [23], [24], [25], [26]. As pointed by [27], acquiring optimal demonstration data can be difficult in practice, due to either the high degrees of freedom of the robot, or the lack of prior knowledge about the desired robot behavior [28].

In this work, we focus on interactive learning of safety constraints. Different from learning from demonstrations that needs the exhibition of desired robot motion, the proposed method only requires a user to incrementally improve the current robot motion by providing directional feedback.

B. Learning From Human Feedback

This line typically focuses on learning rewards/costs from human feedback given in forms of preferences and corrections. Human preference is typically given as user's ranking of two or multiple robot trajectories [6], [7], [8], [9], [10], [11], [12], and the Bradley–Terry model [29] is used to model human preference. Recent variants seek to directly learn a policy from human preference [30].

Human feedback can also come as corrections, where a user may intervene and correct the robot at runtime, from which a intended reward/cost can be inferred. Human corrections can include language command [31], [32], [33], [34], [35] and physical corrections [13], [15], [18]. Since human corrections can be sparse in time, many methods [13], [14], [15], [16], [17], [18], [19] use trajectory deformation methods to interpolate sparse corrections as a continuous trajectory. Some strategies are further proposed to deal with the uncertainty and improve the learning robustness, like updating one parameter dimension at a time [15] or formulating the reward/cost learning in the framework of Kalman filter [14].

Our work belongs to the category of learning from human corrections. But instead of learning rewards/costs, we focus on learning safety constraints from human corrections, which has not been addressed before. Methodologically, we propose a certifiable method, which guarantees the exponential learning convergence with finite human corrections, and certifies the hypothesis misspecification if the learning hypothesis space does not contain the true human intent. Those theoretical results are missing in existing methods.

C. Learning Constraints From Demonstrations

An increasing number of work focuses on learning safety constraints from demonstrations. One vanilla idea is to formulate constraint learning as a binary classification. However, since a robot usually runs within a safe region, negative demonstration samples that violate safety are challenging to collect. The authors in [3] and [36] considered the trajectories of lower cost as the unsafe samples. Gaurav et al. [37] proposed adjusting the constraints toward decreasing the safety margin of feasible policies. Safety constraints can also be learned via mixed integer programming [3], [36], pessimistic adjustment [37], and no-regret algorithm [38].

Another line of work attempts to learn safety constraints only from safe demonstration. For example, some work exploits the optimality of demonstration with unknown constraints [39], [40], [41]. Different probabilistic formulations are also proposed for learning constraints, such as maximum likelihood [5], [42], [43], maximum causal entropy [44], [45], and Bayesian inference [46], [47]. Safety constraints can be represented as a set of

safe states [39], the belief of potential constraints [40], Gaussian process [41], or directly as a convex hull in policy space [4].

The vast majority of the above methods formulate the constraint learning problem in the setting of reinforcement learning, and thus, the learning process usually involves repetitive policy optimization. Those methods cannot be deployed in an online setting, where a robot is expected to on-the-fly update its safe policy and respond to human feedback. To fill this gap, we will develop the first method, to the best of authors' knowledge, to enable online learning of safe constraints from human feedback.

III. PRELIMINARY AND PROBLEM FORMULATION

A. Safe Model Predictive Control (Safe MPC)

We consider the following Safe MPC for a robotic system, where the safety constraint is learnable and parameterized by $\theta \in \mathbb{R}^r$

$$\begin{aligned} \min_{\mathbf{u}_{0:T-1}} \quad & J(\mathbf{u}_{0:T-1}; \mathbf{x}_0) = \sum_{t=0}^{T-1} c(\mathbf{x}_t, \mathbf{u}_t) + h(\mathbf{x}_T) \\ \text{subject to} \quad & \mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \text{ given } \mathbf{x}_0 \\ & g_\theta(\mathbf{x}_{0:T}, \mathbf{u}_{0:T-1}) \leq 0. \end{aligned} \quad (1)$$

Here, $\mathbf{x}_t \in \mathbb{R}^n$ and $\mathbf{u}_t \in \mathbb{R}^m$ are robot state and action, respectively; $t = 0, 1, 2, \dots, T$ is the prediction time steps based on the dynamic model over a short horizon T within the Safe MPC optimization, starting from a given $\mathbf{x}_0$. Define $\mathbf{x}_{0:T} = (\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T)$ and $\mathbf{u}_{0:T-1} = (\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{T-1})$; $J(\cdot)$ is the given task cost function including the running cost $c(\cdot)$ and final cost $h(\cdot)$; $\mathbf{f}(\cdot, \cdot)$ is the system dynamics; $g_\theta(\cdot)$ is the safety constraint parameterized by $\theta \in \mathbb{R}^r$. Given starting state $\mathbf{x}_0$ and parameter θ , the optimal plan (solution) to (1), denoted as $\xi_\theta = \{\mathbf{u}_{0:T-1}^\theta, \mathbf{x}_{0:T}^\theta\}$, is an implicit function of θ .

In a receding horizon implementation on a real robot, $\mathbf{x}_0$ is set as the real robot's current state (estimation) $\mathbf{x}_k^{\text{real}}$, where $k = 0, 1, 2, \dots$, is the global rollout time step of the Safe MPC policy. Only the first action $\mathbf{u}_0^\theta$ in ξ_θ solved from (1) is applied to the robot to move the robot to the next state $\mathbf{x}_{k+1}^{\text{real}}$. This process repeats at each rollout time step $k = 0, 1, \dots$. Such a receding horizon implementation leads to a closed-loop Safe MPC policy, denoted as π_θ , mapping from robot current state $\mathbf{x}_k^{\text{real}}$ to its action $\mathbf{u}_0^\theta$.

B. Safe Asymptotic Approximation of Safe MPC

We present a penalty-based approximation to the Safe MPC in (1). Define trajectory variable $\xi = \{\mathbf{u}_{0:T-1}, \mathbf{x}_{0:T}\}$ and the following penalty objective:

$$B(\xi, \theta) = J(\xi) - \gamma \ln(-g_\theta(\xi)). \quad (2)$$

Here, $B(\xi, \theta)$ includes the original cost in (1) and safety constraint $g_\theta(\xi)$ in a logarithmic barrier transformation, $\gamma > 0$ is named the barrier parameter, controlling the weight of safety penalty $-\ln(-g_\theta(\xi))$ relative to the control cost. With $B(\xi, \theta)$, one can establish the following penalty-based MPC

$$\min_{\mathbf{u}_{0:T-1}} B(\xi, \theta) \quad \text{subject to } \mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t). \quad (3)$$

We denote the optimal plan (solution) to the above (3) as $\xi_\theta^\gamma = \{\mathbf{u}_{0:T-1}^{\theta,\gamma}, \mathbf{x}_{0:T}^{\theta,\gamma}\}$. The following result is from our prior work [48], stating that the penalty-based MPC (3) is an asymptotic and safe approximation to the Safe MPC (1).

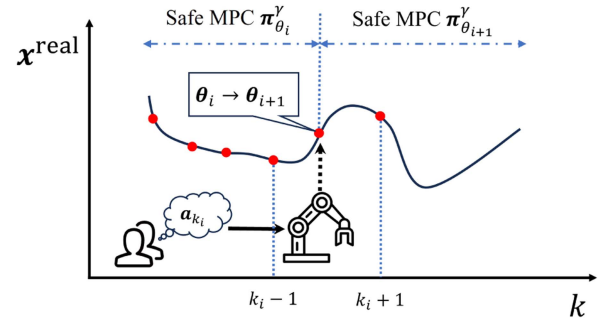


Fig. 2. Update of Safe MPC policy with human directional corrections.

Lemma 1 (Theorem 2 in [48]): If the original Safe MPC (1) satisfies certain mild conditions (i.e., local second-order condition and strict complementarity), then for any small $\gamma > 0$, the penalty-based MPC (3) has a (local) solution ξ_θ^γ , and ξ_θ^γ is safe with respect to the original constraint, i.e., $g_\theta(\xi_\theta^\gamma) < 0$, and ξ_θ^γ satisfies the first-order condition of (3). Furthermore, ξ_θ^γ is asymptotically approximates ξ_θ : $\xi_\theta^\gamma \rightarrow \xi_\theta$ as $\gamma \rightarrow 0$.

The above lemma asserts that solving the Safe MPC (1) can be replaced by solving penalty-based MPC (3). The solution approximation is controlled by γ for arbitrary accuracy, as $\xi_\theta^\gamma \rightarrow \xi_\theta$ as $\gamma \rightarrow 0$. A good approximation is achieved by a smaller γ . On the contrary, if γ is larger, penalty-based MPC (3) will lead to more safety-conservative solution ξ_θ^γ . In the development below, we fix the parameter γ in (3). We denote the penalty-based MPC policy in (3) as π_θ^γ .

C. Problem Formulation

1) Representation of Human Intended Safety: Consider a human is supervising the robot's behavior. The user has his/her intent of robot motion safety. In Safe MPC (1), we hypothesize that user-intended safety can be represented by a subset of the trajectory space

$$\mathcal{S}(\theta_H) = \{(\mathbf{x}_{0:T}, \mathbf{u}_{0:T-1}) \mid g_{\theta_H}(\mathbf{x}_{0:T}, \mathbf{u}_{0:T-1}) \leq 0\}. \quad (4)$$

Here, θ_H is intent parameter from an implicit set $\Theta_H \in \Theta_H$. Θ_H is implicit to human and unknown to the robot. For the robot policy π_θ^γ , if $\theta \in \Theta_H$, the user considers the robot motion to be safe; otherwise if $\theta \notin \Theta_H$, the robot motion by π_θ^γ might not be safe at certain time, and thus, the user provides correction to improve its safety. We use an implicit set $\Theta_H \in \mathbb{R}^r$, instead of a single point θ_H , to capture the potential variation of user intents. We suppose that a volume measure of Θ_H , $\text{Vol}(\Theta_H)$, is nonzero, and a user intent $\theta_H \in \Theta_H$ can vary at different steps and contexts, but always within Θ_H .

2) Human Directional Correction: As shown in Fig. 2, we consider human correcting a robot in an online fashion while the robot is executing π_θ^γ . Specifically, let i be the index of learning update, and suppose robot current policy is $\pi_{\theta_i}^\gamma$. During rollout of $\pi_{\theta_i}^\gamma$, the robot receives a human correction $a_{k_i} \in \mathbb{R}^m$ at the rollout time step k_i .

Here, we do not assume precise timing k_i or duration of the human correction due to the sparsity or intermittency of human feedback. In other words, the human can apply one or multiple corrections at any time in runtime as he/she find necessary. But we assume human correction a_{k_i} is in the robot action space. For example, in autonomous driving where the robot action

space is steering angle and acceleration, human can directly apply feedback with the steering wheel and pedals. In some other applications, the action space correction can be achieved via a specific mapping or interface. For example, to correct a torque-controlled robot arm, human's correction on the robot end-effector can be mapped into joint torque via the Jacobian transpose. The reason why we only consider the action space correction is because first, corrections in input spaces may be easier to implement, as shown in our later experiments, and second, corrections in state spaces can be infeasible for the underactuated robot system [20].

After receiving each $\mathbf{a}_{k_i}$, the robot updates the safety parameter $\theta_i \rightarrow \theta_{i+1}$ in (1) for its Safe MPC. Then, the robot continues with $\pi_{\theta_{i+1}}^\gamma$ until the next possible user correction. This process repeats until the convergence of the policy update or the user is satisfied with the robot's motion.

In the above learning process, while the correction signal $\mathbf{a}$ given by human is real-valued, only its direction is used in the learning process. This means that we will use the sign (for scalar signal) or the unit vector (for vector-valued correction signal) to represent the direction of correction. The magnitude of the correction is NOT taken into account in the learning process. For example, when correcting a mobile robot, the user's correction can be "left" or "right," without specifying how much the robot should move. As we will show later, directional feedback provides a consistently large and flexible correction space, effectively overcoming the overshooting correction, compared to magnitude feedback.

3) *Robot Operation Safety and Emergency Stop*: In the aforementioned learning procedure, the robot will respond to human correction by executing the updated safe MPC policy π_{θ}^γ . However, it is possible that the robot may still exit the user-intended safety. In such cases, the human may trigger an emergency stop to stop the robot and potentially restart it from a desired safety region. In such case, the executed trajectory before the emergency stop is assumed safe. In other words, until an emergency stop is triggered, human correction typically occur when the robot is still in the intended safety set; otherwise, the correction would be useless. For instance, correcting a vehicle after it is crashed is futile.

4) *Parametric Safety Constraints*: In (4), the safety set is defined as a zero sublevel set of a safety value function g_θ . If g_θ has sufficient representation power, the safety set can represent a complex safety region. In the following, we focus on a weighted-features parameterization of g_θ , which has been commonly used by the prior work on learning cost/reward [4], [20], [49], [50], [51]. Specifically, we parameterize g_θ

$$g_\theta(\xi) = \phi_0(\xi) + \theta_1\phi_1(\xi) + \theta_2\phi_2(\xi) + \dots + \theta_r\phi_r(\xi) \quad (5)$$

with $\theta = [\theta_1, \theta_2, \dots, \theta_r]^\top \in \mathbb{R}^r$ and $\phi = [\phi_1, \phi_2, \dots, \phi_r]^\top$ the learnable weight and task-relevant feature vectors, respectively. We fix the weight of ϕ_0 as unit, because otherwise, θ has scaling ambiguity, i.e., any scaled $c \cdot \theta$ with $c > 0$ lead to the same constraint. In this work, we assume that the task-relevant features in ϕ are given a priori. These features, defined in system states (e.g., position, velocity) and actions, can be either manually defined or obtained from pretrained neural representations, leveraging prior knowledge of the environment and tasks, as demonstrated in our experiments. The feature selection itself is beyond the scope of this article and is left for future work. A

more detailed discussion of this assumption and its limitations is provided in Section VIII-D3.

5) *Problem of Safe MPC Alignment*: Following the procedure of human correction and robot policy update in Section II-C, we aim to develop a policy update rule: $\theta_i \rightarrow \theta_{i+1}$ for each received human directional correction, such that $\theta_i \rightarrow \theta_H \in \Theta_H$, as $i = 1, 2, 3, \dots$, i.e., safety constraints in the robot's safe MPC policy will be aligned with human intent.

IV. OVERVIEW OF SAFE MPC ALIGNMENT

A. Hypothesis on Human Directional Corrections

Informally, we have the following intuitions on human feedback. First, human corrections, on average, improve the robot's safety. Second, human correction on robot safety is typically proactive and happens before the robot leaves the human-intended safety set. In other words, human correction is made when the robot is still in $\mathcal{S}(\theta_H)$ in (4). For example, it is useless to correct a vehicle after its crash. Third, human feedback is typically safety-margin aware: as a robot approaches the safety boundary, human users tend to prioritize safety over the task objective. For instance, when a vehicle moves closer to obstacles, humans may instinctively shift their focus to avoiding collisions rather than pursuing the original target. In the following, we will formalize those intuitions.

Without loss of generality, suppose the motion plan ξ_θ^γ is a local solution to the penalty MPC (3), and when the robot execute the first action $\mathbf{u}_0^{\theta, \gamma}$ (receding horizon implementation), a human applies a directional correction $\mathbf{a}$ in robot action space. In this section, the subscription k_i is omitted for general exposition and notation simplicity. Define

$$\mathbf{a} = (\mathbf{a}^\top, 0, 0, \dots) \in \mathbb{R}^{mT}. \quad (6)$$

We assume there is a nonzero volume set $\bar{\Theta}_H$ (containing a ball) in user implicit set Θ_H , i.e., $\bar{\Theta}_H \subseteq \Theta_H$, such that for any $\theta_H \in \bar{\Theta}_H$, the human correction $\mathbf{a}$, on average, indicates a direction of lowering the penalty value $B(\xi_\theta^\gamma, \theta_H)$ at the robot motion plan ξ_θ^γ . Concretely, it means $\mathbf{a}$ on averages points in the direction of the negative gradient of $B(\xi, \theta_H)$ at ξ_θ^γ , i.e.,

$$\begin{aligned} \forall \theta_H \in \bar{\Theta}_H, \mathbb{E}_{\mathbf{a}} \langle \mathbf{a}, -\nabla B(\xi_\theta^\gamma, \theta_H) \rangle \\ = \langle \bar{\mathbf{a}}, -\nabla B(\xi_\theta^\gamma, \theta_H) \rangle \geq 0. \end{aligned} \quad (7)$$

Here, $\langle \cdot, \cdot \rangle$ denotes the inner product operation and $\mathbb{E}_{\mathbf{a}}(\mathbf{a}) = \bar{\mathbf{a}}$ denotes the expectation value of $\mathbf{a}$. Since $\nabla B(\xi, \theta)$ is not dependent on $\mathbf{a}$, the expectation can be moved inside the inner product. $\nabla B(\xi, \theta)$ denotes the full gradient of B with respect to $\mathbf{u}_{0:T-1}$ evaluated at trajectory $\xi = \{\mathbf{u}_{0:T-1}, \mathbf{x}_{0:T}\}$. Similar notations apply to J and g as follows. Since state sequence $\mathbf{x}_{0:T}$ is the rollout of $\mathbf{u}_{0:T-1}$ on dynamics, the differentiation goes through the dynamics equation.

Equation (7) shows that although the human correction $\mathbf{a}$ is provided as a real-valued vector, only its direction—not its magnitude—affects the inequality. This brings one advantage of directional correction compared to the magnitude-based correction: regardless of robot plan ξ_θ^γ , the allowable human correction satisfying (7) always account for half of the action space, making directional corrections more accessible for human users, as shown in Fig. 3. Recall that $B(\xi, \theta) = J(\xi) - \gamma \ln(-g_\theta(\xi))$, by extension of the gradient, (7) becomes

$$\forall \theta_H \in \bar{\Theta}_H, \langle \bar{\mathbf{a}}, -\nabla J(\xi_\theta^\gamma) \rangle$$

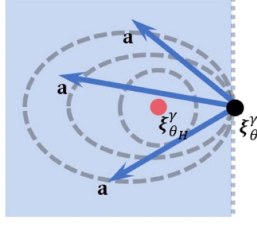


Fig. 3. Illustration of directional correction in the space of robot plan ξ s. The grey circles represent the level contours of $B(\xi, \theta_H)$. The black dot stands for the current motion plan ξ_θ^γ and the red point stands for optimal plan $\xi_{\theta_H}^\gamma$ with ground truth safety constraint. The blue region stands for the half-space of all possible human correction under our assumption (7). Some samples of human corrections are shown by blue arrows.

$$+ \gamma \frac{1}{-g_{\theta_H}(\xi_\theta^\gamma)} \langle \bar{\mathbf{a}}, -\nabla g_{\theta_H}(\xi_\theta^\gamma) \rangle \geq 0. \quad (8)$$

For the above equation, we have the following interpretations.

- 1) The first and second terms of (8) correspond to human correction toward lowering the cost and improving the safety of ξ_θ^γ , respectively. γ weighs between two improvements.
- 2) If we interpret the value of $-g_{\theta_H}(\xi_\theta^\gamma)$ as the safety margin of the plan ξ_θ^γ , (8) suggests that the human correction is safety-margin aware: the closer the robot motion get to the unsafe region, the more will human improve the robot's safety than minimizing the cost. This property is consistent with our previous intuition.
- 3) Equation (7) implies $B(\xi_\theta^\gamma, \theta_H)$ is well-defined in the first place

$$\forall \theta_H \in \bar{\Theta}_H, \quad g_{\theta_H}(\xi_\theta^\gamma) < 0. \quad (9)$$

This corresponds to the third intuition: a user typically gives correction only when the robot is in user intended safety region.

In sum, the two constraints (7) and (9) formalize the intuitions we have observed at the beginning of this subsection. Those constraints will be used in the robot learning in the following section.

B. Alignment Algorithm: Learning by Hypothesis Space Cutting

To find $\theta_H \in \bar{\Theta}_H \subseteq \Theta$ for the safe MPC (1), our idea is that for each human directional correction $i = 0, 1, 2, \dots$, the robot performs one learning iteration, by updating a hypothesis space $\Theta_i \in \mathbb{R}^r$, a set containing all possible $\bar{\Theta}_H$. Initially, the robot may choose a large box hypothesis space Θ_0

$$\Theta_0 = \{\theta \mid c_l \leq \theta \leq c_u\} \subset \mathbb{R}^r \quad (10)$$

in which c_l, c_u are the vectors of lower and upper bounds of the box space, respectively, and the inequalities are applied element-wise. By saying a Θ_0 large enough, we assume

$$\bar{\Theta}_H \subset \Theta_0. \quad (11)$$

In the following, we will show our method can self-access the misspecification, where (11) is not met.

We now return to the notation $\mathbf{a}_{k_i}$ to explicitly denote the time index of human correction. Recall the procedure of human correction in Section III-C and Fig. 2. At i th learning iteration, $i = 1, 2, 3, \dots$, the robot executes $\pi_{\theta_i}^\gamma$ with $\theta_i \in \Theta_{i-1}$.

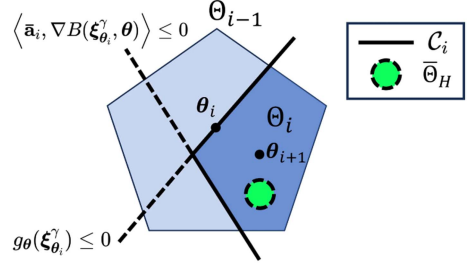


Fig. 4. Update of hypothesis space (2-D example). Two cutting lines are used to cut Θ_{i-1} to Θ_i . The user intent set $\bar{\Theta}_H$ is contained in Θ_i . New θ_{i+1} is chosen from Θ_i for the updated safe MPC policy.

The user gives a correction $\mathbf{a}_{k_i}$ at rollout step k_i . By letting $\bar{\mathbf{a}}_i = (\mathbf{a}_{k_i}^T, 0, 0, \dots)$ as shown in (6), we can establish the following two constraints from the previous hypotheses (7) and (9):

$$\bar{\Theta}_H \subset \mathcal{C}_i = \{\theta \mid \langle \bar{\mathbf{a}}_i, \nabla B(\xi_{\theta_i}^\gamma, \theta) \rangle \leq 0, \quad g_{\theta}(\xi_{\theta_i}^\gamma) < 0\} \quad (12)$$

which we refer to as the *cutting set*. Note that the first inequality $\langle \bar{\mathbf{a}}_i, \nabla B(\xi_{\theta_i}^\gamma, \theta) \rangle \leq 0$ is from (7). Using $\mathcal{C}_i$, one can update the hypothesis space

$$\Theta_i = \Theta_{i-1} \cap \mathcal{C}_i. \quad (13)$$

The above update can be viewed as a step of using (12) to cut the hypothesis space Θ_{i-1} to remove the portion where $\bar{\Theta}_H$ is impossible. The remaining hypothesis space Θ_i will be new hypothesis used for the next iteration. With the above update, we have the following result.

Lemma 2: Given $\bar{\Theta}_H \subset \Theta_0$, if one follows the update (13), then $\bar{\Theta}_H \subset \Theta_i$ for all $i = 1, 2, 3, \dots$

Proof: See the proof in Appendix A-A. $\square$

As shown in Fig. 4, the procedure of robot learning is to cut and move the hypothesis space based on human directional correction. Therefore, we name our alignment method *hypothesis space cutting*. As a summary, we present the three main steps of our algorithm as follows.

Safe MPC Alignment by Hypothesis Space Cutting

- Initialize Θ_0 that contains $\bar{\Theta}_H$. Repeat the following three steps for $i = 1, 2, \dots, K$ until $\theta_i \in \bar{\Theta}_H$.
- Step 1: Robot chooses parameter $\theta_i \in \Theta_{i-1}$ for its current safe MPC policy $\pi_{\theta_i}^\gamma$.
 - Step 2: Human user applies a directional correction $\mathbf{a}_{k_i}$ while robot executes $\pi_{\theta_i}^\gamma$.
 - Step 3: Perform hypothesis space cut using (13) to get new hypothesis space Θ_i .

At i th learning iteration, the robot chooses $\theta_i \in \Theta_{i-1}$ for its current safe MPC $\pi_{\theta_i}^\gamma$. While executing $\pi_{\theta_i}^\gamma$, the robot receives human directional correction $\mathbf{a}_{k_i}$ at the rollout time step k_i . This correction leads to a cutting set $\mathcal{C}_i$ (12), which will update Θ_{i-1} to Θ_i in (13). The above iteration repeats $i = 1, 2, 3, \dots$. From (13), one can easily see that $\Theta_0 \supseteq \Theta_1 \supseteq \dots \supseteq \bar{\Theta}_H$. Given a volume measure, it follows $\text{Vol} \Theta_0 \geq \text{Vol} \Theta_1 \geq \dots \geq \text{Vol} \bar{\Theta}_H$. Thus, the volume of the hypothesis space is at least not increasing after each user correction. It is hoped that the volume of the hypothesis space decreases quickly, such that new θ_{i+1} increases its chance of choosing from $\bar{\Theta}_H$ —meaning a successful safety alignment between the robot and human.

For the *Safe MPC alignment*, we will later (see Section V) show within finite number K of correction, the method will either, as follows:

- 1) surely have $\theta_i \in \bar{\Theta}_H \subset \Theta_H, i < K$ and
- 2) declare the failure due to misspecification ($\bar{\Theta}_H \cap \Theta_0 = \emptyset$).

Also in Section V, we will provide implementation details for the above hypothesis space cutting procedure.

C. Properties and Geometric Interpretations

We next show properties and geometric interpretations for the proposed safe MPC alignment. Those properties will be used to establish the convergence of the algorithm.

1) *Piecewise Linear Cut*: By recalling the weight-feature parameterization of the safety constraint (5), one can examine that the inequities in (12) are both linear in θ . This means that the cutting set $\mathcal{C}_i$ in (12) is a piecewise linear cut, as stated formally in the following lemma.

Lemma 3: $\forall i, \mathcal{C}_i$ in (12) can be expressed as an intersection of two half-spaces in $\mathbb{R}^r$

$$\mathcal{C}_i = \{\theta \mid \theta^\top h_i \leq b_i\} \cap \{\theta \mid \theta^\top \phi(\xi_{\theta_i}^\gamma) < -\phi_0(\xi_{\theta_i}^\gamma)\} \quad (14)$$

with

$$h_i = -\langle \bar{a}_i, \nabla J(\xi_{\theta_i}^\gamma) \rangle \phi(\xi_{\theta_i}^\gamma) + \gamma \frac{\partial \phi}{\partial u}(\xi_{\theta_i}^\gamma) \bar{a}_i \quad (15)$$

$$b_i = \langle \bar{a}_i, \nabla J(\xi_{\theta_i}^\gamma) \rangle \phi_0(\xi_{\theta_i}^\gamma) - \gamma \langle \bar{a}_i, \nabla \phi_0(\xi_{\theta_i}^\gamma) \rangle. \quad (16)$$

Proof: See Appendix A-B. $\square$

According to the above lemma, $\Theta_{i-1} \cap \mathcal{C}_i$ can be viewed as cutting Θ_{i-1} with two hyperplanes (see Fig. 4)

$$\begin{aligned} \text{bd } \mathcal{C}_i &= \{\theta \mid \theta^\top h_i = b_i, \theta^\top \phi(\xi_{\theta_i}^\gamma) \leq -\phi_0(\xi_{\theta_i}^\gamma)\} \cup \\ &\quad \{\theta \mid \theta^\top h_i \leq b_i, \theta^\top \phi(\xi_{\theta_i}^\gamma) = -\phi_0(\xi_{\theta_i}^\gamma)\}. \end{aligned} \quad (17)$$

As the initial hypothesis space $\Theta_0 \subset \mathbb{R}^r$ in (10) is a convex polytope, iteratively cutting with hyperplanes yields a series of convex polytopes in $\mathbb{R}^r$. Consequently, all Θ_i s, $i = 0, 1, 2, \dots$, are convex polytopes.

2) *Relationship Between θ_i TEXT and $\text{bd } \mathcal{C}_i$ TEXT*: The following assertion states a relationship between θ_i and $\text{bd } \mathcal{C}_i$.

Lemma 4: $\forall i, \theta_i$ is on $\text{bd } \mathcal{C}_i$ in (17). Specifically, θ_i satisfies

$$\theta_i^\top h_i = b_i \quad \text{and} \quad \theta_i^\top \phi(\xi_{\theta_i}^\gamma) < -\phi_0(\xi_{\theta_i}^\gamma) \quad (18)$$

for all $i = 1, 2, \dots$

Proof: See Appendix A-C. $\square$

Lemma 4 states $\theta_i \in \text{bd } \mathcal{C}_i$. Its geometric interpretation is in Fig. 5(c): the cut $\text{bd } \mathcal{C}_i$ always passes θ_i . This suggests that the choice of $\theta_i \in \Theta_{i-1}$ is critical in the learning process: it directly affects the position of the cut $\text{bd } \mathcal{C}_i$ and consequently the removal volume. It is important to choose a good $\theta_i \in \Theta_{i-1}$ such that a large hypothesis space is removed after each cut.

We briefly discuss the implication of different choices of $\theta_i \in \Theta_{i-1}$. When choosing θ_i near the boundary of Θ_{i-1} , as shown in Fig. 5(a) and (b), the removal of the hypothesis space depends on the specific direction of cut $\text{bd } \mathcal{C}_i$. Since the direction of $\mathcal{C}_i$ will be eventually determined by the direction of user correction according to (17), the removal of the hypothesis space can be relatively small, as shown in Fig. 5(b), or large, as shown in Fig. 5(a), depending on the specific direction of the correction.

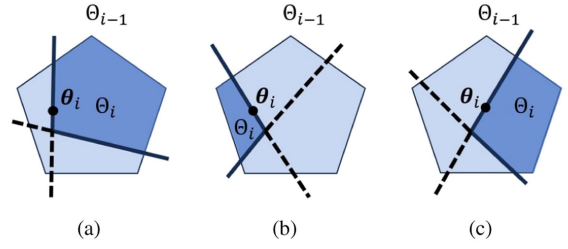


Fig. 5. Illustration of the different choices of θ_i . (a) and (b) When θ_i is chosen near the set boundary, the removed volume depends on the specific direction of $\text{bd } \mathcal{C}_i$. (c) When θ_i is near the center, the cut volume is large on average regardless of the specific user directional correction.

As the specific direction of user correction is hard to assume, a good choice of $\theta_i \in \Theta_{i-1}$ will be near the center of Θ_{i-1} . As such, it can lead to an averaged large cut of the hypothesis space regardless of the direction of $\mathcal{C}_i$, as shown in Fig. 5(c). We will rigorously discuss the centering choice of θ_i and volume reduction rate of the hypothesis space in the following section.

V. MAIN RESULTS AND IMPLEMENTATION DETAILS

Next, we will present the details for how to select $\theta_i \in \Theta_{i-1}$ at each learning iteration i in Safe MPC Alignment. Based on the previous geometric intuition, our choice of θ_i is to achieve a large averaged reduction in the hypothesis space for all possible human directional corrections. The reduction rate enables us to establish a certification of the learning process.

A. Choice of θ_i TEXT

As discussed previously, θ_i needs to be picked from the “center” of Θ_{i-1} in each iteration. One option is to choose the mass center of Θ_{i-1} . However, computing the centroid of a polytope can be #P-Hard [52]. While methods like the Hit-And-Run algorithm [53] estimate the mass center by sampling, they can suffer from high sample complexity. Another possible choice is to use the center of the maximum inscribed ball of a polytope [54], but it lacks of the guarantee in volume reduction rate. For computational efficiency, we will choose θ_i as the center of the maximum volume inscribed ellipsoid (MVE) in the convex set Θ_{i-1} .

Definition 1 (Maximum Volume inscribed Ellipsoid (MVE) [55]): The MVE of a given convex set $\Theta \subset \mathbb{R}^r$ is defined as the image of the unit ball under an affine transformation

$$\mathcal{E} = \{\bar{H}\theta + \bar{d} \mid \|\theta\|_2 \leq 1\} \quad (19)$$

where the affine transformation parameter $\bar{H} \in \mathbb{S}_{++}^r$ (a $r \times r$ symmetric positive definite matrix), and $\bar{d} \in \mathbb{R}^r$ is called the center of $\mathcal{E}$. $\bar{H}, \bar{d}$ are determined via the following optimization problem to maximize the volume of the ellipsoid $\det H$:

$$\begin{aligned} \bar{H}, \bar{d} &= \arg \max_{H \in \mathbb{S}_{++}^r, \bar{d}} \log \det H \\ \text{s.t.} \quad &\sup_{\|\theta\|_2 \leq 1} \mathbb{1}_{\Theta}(H\theta + \bar{d}) \leq 0 \end{aligned} \quad (20)$$

where the indicator $\mathbb{1}_{\Theta}(\theta) = 0$ for $\theta \in \Theta$ and otherwise $\mathbb{1}_{\Theta}(\theta) = +\infty$. Specifically, when Θ is a polytope $\{\theta^\top n_i \leq c_i, i = 1, \dots, m\}$, the problem can be converted to a convex

optimization with m constraints, see [55, ch. 8.4.2]

$$\begin{aligned} \bar{H}, \bar{\mathbf{d}} &= \arg \min_{H \in \mathbb{S}_{++}^r, \mathbf{d}} \log \det H^{-1} \\ \text{s.t. } &\|H\mathbf{n}_i\|_2 + \mathbf{d}^\top \mathbf{n}_i \leq c_i, \quad i = 1, \dots, m. \end{aligned} \quad (21)$$

Recall that in Safe MPC alignment the learning update (13) is to iteratively cut the hypothesis space by hyperplanes (17), and that the initial hypothesis space (11) is a box. By induction, it leads to that any updated hypothesis space Θ_i is a convex polytope, $i = 0, 1, 2, \dots$. Thus, for any Θ_{i-1} , by the above definition, let $\mathcal{E}_{i-1}$ denote the MVE of Θ_{i-1} and $\bar{\mathbf{d}}_{i-1}$ denote the center of $\mathcal{E}_{i-1}$. Then, the choice of θ_i is

$$\theta_i = \bar{\mathbf{d}}_{i-1}. \quad (22)$$

The convex optimization (20) can be solved efficiently via existing tools like CVX [56].

B. Result 1: Finite Iteration Convergence

When choosing θ_i as the center of MVE of Θ_{i-1} , the following lemma states that the volume ratio $\text{Vol}(\Theta_i)/\text{Vol}(\Theta_{i-1})$ after each hypothesis space cut can be upper bounded.

Lemma 5: In the Safe MPC Alignment, when choosing $\theta_i \in \mathbb{R}^r$ as the center of MVE of Θ_{i-1} , one has

$$\frac{\text{Vol}(\Theta_i)}{\text{Vol}(\Theta_{i-1})} \leq 1 - \frac{1}{r} \quad (23)$$

for all $i = 1, 2, 3, \dots$ and any enclosure volume measure Vol .

Proof: See Appendix A-D. $\square$

The lemma above asserts an upper bound of the volume of the hypothesis space $\text{Vol}(\Theta_i)$ relative to the previous volume. With this, we can now establish the finite iteration convergence of the Safe MPC Alignment, stated as follows.

Theorem 1 (Successful learning in finite iterations): In the Safe MPC Alignment algorithm, θ_i is selected as the center of MVE of Θ_{i-1} , $i = 1, 2, 3, \dots$. If $\bar{\Theta}_H \subset \Theta_0$, the algorithm will terminate successfully within

$$K = \left\lceil \frac{\ln \frac{\tau_r \rho_H^r}{\text{Vol}(\bar{\Theta}_0)}}{\ln \left(1 - \frac{1}{r}\right)} \right\rceil \quad (24)$$

iterations, where ρ_H is the radius of a ball contained within $\bar{\Theta}_H$ and τ_r is the volume of a unit ball in $\mathbb{R}^r$ space. That is, there exists $i < K$, such that $\theta_i \in \bar{\Theta}_H \subset \Theta_H$.

Proof: See Appendix A-E. $\square$

The above result asserts the maximum iteration for the proposed algorithm to find $\theta_i \in \Theta_H$. In other words, within K rounds of human correction, the robot can successfully learn a safety constraint. The above result requires a preset of ρ_H^r , which is the radius of a ball contained in $\bar{\Theta}_H$. Since the size of $\bar{\Theta}_H$ is unknown in reality, $\rho_H > 0$ can be interpreted as an underestimated size of the unknown user intent set $\bar{\Theta}_H$. It also serves as a termination threshold for learning convergence: a smaller ρ_H can lead to a more concentrated θ_H at convergence.

The finite iteration convergence in Theorem 1 comes with a prerequisite: $\bar{\Theta}_H \in \Theta_0$ has to be satisfied in the first place for the Safe MPC alignment. Although such a condition can be achieved by empirically choosing Θ_0 large enough to contain all possible $\bar{\Theta}_H$ (10), it is also desired to have a guarantee to verify the hypothesis misspecification, where such condition is

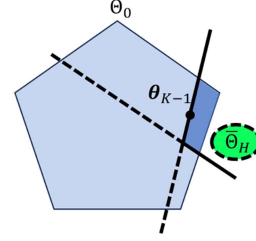


Fig. 6. When there is misspecification, the Safe MPC Alignment will converge to some point on the boundary of Θ_0 . So we can use the distance of θ_{K-1} to the boundary as an indicator for misspecification.

not met. Next, we will discuss the declaration of hypothesis misspecification.

C. Result 2: Hypothesis Misspecification Certification

Consider hypothesis space misspecification, where $\bar{\Theta}_H \cap \Theta_0 = \emptyset$. In this case, no $\theta \in \Theta_0$ can account for the user intended safety (4). The hypothesis misspecification can lead to the undesired behavior of the robot [51], [57]. The following result states the Safe MPC alignment algorithm can automatically certify the misspecification of Θ_0 .

Theorem 2 (Certifying misspecification): In the Safe MPC Alignment algorithm, θ_i is selected as the center of MVE of Θ_{i-1} , $i = 1, 2, 3, \dots$. If $\bar{\Theta}_H \cap \Theta_0 = \emptyset$, the algorithm converges to a set Θ^* as $i \rightarrow \infty$, such that $\text{Vol}(\Theta^*) = 0$ and $\Theta^* \cap \text{int } \Theta_0 = \emptyset$.

Proof: See Appendix A-F. $\square$

The above result states when the initial hypothesis space is misspecified, $\bar{\Theta}_H \cap \Theta_0 = \emptyset$, the Safe MPC Alignment algorithm will finally converge to a set of zero volume on the boundary of the initial hypothesis space Θ_0 . The above theorem inspires us to empirically identify the misspecification by monitoring the position of the MVE center of Θ_i during the learning update. As shown in Fig. 6, if after the maximum iteration $K - 1$ in (24), the MVE center θ_{K-1} is very close to the boundary of the initial hypothesis set Θ_0 , there is likely hypothesis space misspecification, and the algorithm terminates with misspecification failure.

Specifically, recall that Θ_0 in (10) is a box with bounds c_l, c_u , we can predefine a small threshold ϵ to examine if θ_{K-1} is on the boundary of Θ_0 . If θ_{K-1} satisfies

$$\min_{j=1, \dots, r} \min (|c_l[j] - \theta_{K-1}[j]|, |c_u[j] - \theta_{K-1}[j]|) \leq \epsilon \quad (25)$$

it means θ_{K-1} is very close to the boundary of Θ_0 . We consider there is likely misspecification in this case.

D. Detailed Algorithm of Safe MPC Alignment

With the main results on convergence and misspecification, we present the detailed safe MPC alignment algorithm in Algorithm 1. With initializing Θ_0 and predefining a termination threshold ρ_H , three steps in the hypothesis space cutting are executed until the user satisfies robot motion or the iteration count reaches the maximum count K in (24). Note that during the execution of the MPC policy $\pi_{\theta_i}^\gamma$, the human can perform emergency stopping and reset the robot's state, which corresponds to Section III-C3. After $K - 1$ iterations, the algorithm checks the misspecification by (25).

Algorithm 1: Safe MPC Alignment Algorithm.

Initial: Initial hypothesis space Θ_0 , termination threshold ρ_H (i.e., estimated radius of $\bar{\Theta}_H$), $i = 1$

Compute maximum iteration count K from ρ_H in (24);

while $i < K$ **do**

 Choose $\theta_i \in \Theta_{i-1}$ using (22);

 Robot executes $\pi_{\theta_i}^\gamma$;

if Human provides directional correction $\mathbf{a}_{k_i}$ **then**

 Generate $\bar{\mathbf{a}}_i$ using (6);

 Calculate the cutting set $\mathcal{C}_i$ using (14);

 Update hypothesis space $\Theta_i = \Theta_{i-1} \cap \mathcal{C}_i$;

$i \leftarrow i + 1$;

else if Human triggers emergency stop **then**

 Human restarts robot from safe region;

else if Human user is satisfied **then**

return θ_i ;

end

if θ_{K-1} satisfies (25) **then**

return misspecification failure;

else

return θ_{K-1}

end

VI. NUMERICAL EXAMPLES

In this section, we will evaluate the Safe MPC Alignment using two numerical examples: A pendulum task, where we will illustrate all theoretical aspects of the proposed method, and a thrust-controlled quadrotor navigation task, where we will demonstrate the capability of the method to learn safety constraints in complex 3-D environments with various features, as well as to learn safe velocity constraint. Simulated human correction knowing ground-truth safety constraints will be used. We will compare the proposed method with baselines, and ablate the hyperparameters of the algorithm. All numerical experiments are implemented in Python, running on a desktop with Intel i9-13900 K CPU. All our experiment codes are open-sourced at <https://github.com/asu-iris/Safe-MPC-Alignment>.

A. Inverted Pendulum

1) *Setup:* The dynamics of an inverted pendulum is

$$\ddot{\alpha} = \frac{3}{ml^2} \left(-\frac{1}{2}mgl \sin \alpha + u - d\dot{\alpha} \right) \quad (26)$$

with α being the angle between the pendulum and gravity, g the gravity constant, u input torque, m , l and d the length, inertia, and damping ratio. The system state is $\mathbf{x} = [\alpha, \dot{\alpha}]^T$, (26) discretized by the Euler method with $\Delta t = 0.02$ s. A zero-mean Gaussian noise with covariance $\Sigma = \text{diag}(1, 4) \times 10^{-5}$ is added to $\mathbf{x}$ at each MPC rollout to simulate the state noise. The task is to control the pendulum to a target $\mathbf{x}^* = [\pi, 0]^T$. We pose an unknown global state constraint on pendulum motion

$$0.6\alpha + \dot{\alpha} = \theta^H \mathbf{x} \leq 3 \quad (27)$$

where $\theta^H = [0.6, 1]^T$ is to be learned. All other parameter values are given in the Appendix.

In Safe MPC (1), we set $T = 40$ and the cost functions as

$$\begin{aligned} c(\mathbf{x}_t, u_t) &= Ru_t^2 \\ h(\mathbf{x}_T) &= (\mathbf{x}_T - \mathbf{x}^*)^T Q(\mathbf{x}_T - \mathbf{x}^*) \end{aligned} \quad (28)$$

respectively, with $R = 0.1$ and $Q = \text{diag}(25, 10)$. To ensure the safe MPC to satisfy a global constraint, it is sufficient to learn a safety constraint defined on the closest controllable state. Thus, we define

$$g_\theta(\xi) = -3 + \theta^T \mathbf{x}_1. \quad (29)$$

In the penalty-based MPC (3), we set $\gamma = 0.1$.

Suppose the user intent set $\bar{\Theta}_H = \{\theta \mid \|\theta - \theta_H\| \leq 0.02\}$. The simulated human correction $\mathbf{a}$ are generated as follows: when the system state is near the safety boundary, i.e., $g_{\theta_H}(\xi_\theta^\gamma) > -\epsilon_g$ ($\epsilon_g = 0.25$ here), a directional correction $\mathbf{a}$ is generated in probability $p = 0.3$ using $\text{sign}(-\nabla B(\xi_\theta^\gamma, \theta_H))$ [from which only the first step is taken considering (6)]. The correction is probabilistic to simulate the randomness of human providing corrections. It is easy to verify that the simulated $\mathbf{a}$ is valid in term of satisfying the hypotheses in Section IV-A.

2) *Convergence Analysis:* We first evaluate the convergence of our algorithm without hypothesis misspecification. In Algorithm 1, we set the initial hypothesis Θ_0 with box boundaries $\mathbf{c}_l = [-6, -6]^T$ and $\mathbf{c}_u = [2, 2]^T$; thus, Θ_0 includes $\bar{\Theta}_H$. The initial $\theta_1 = [-2, -2]^T$ (i.e., the center of Θ_0).

With each simulated human correction, we track θ_i in each learning update. Each update of Θ_i takes less than 30 ms. θ_i falls in the ground-truth set $\bar{\Theta}_H$ after 14 corrections. The MPC rollout trajectory (phase plot) with each θ_i is shown in Fig. 7. The ground-truth constraint (27) and the learned constraints g_{θ_i} are also plotted. Fig. 7 shows that the true safety constraint can be successfully learned from simulated correction.

To analyze the convergence, we run the algorithm for 10 times. In each run, we keep track of two metrics over correction (iteration) count and show them in Fig. 8. The first metric (left panel) is the logarithm volume measure of the solved MVE $\log \det \bar{H}$ in (20); this provides a lower bound of the volume of Θ_i to show the exponential convergence. The second (right panel) is the parameter error $\|\theta_i - \theta_H\|_2$. From Fig. 8, we can conclude the proposed method is

- 1) *Fast-converging:* The logarithm volume $\log \det \bar{H}$ has a near-linear decreasing speed. This is consistent with the theory in Lemma 5.
- 2) *Correction data-efficient:* $\|\theta_i - \theta_H\|_2$ converges to the value lower than 0.02 within 17 corrections. This is consistent with the theoretical upper bound of correction count in Theorem 1

$$\ln \frac{\tau_r \rho_H^r}{\text{Vol}(\Theta_0)} / \ln \left(1 - \frac{1}{r} \right) = 17.28. \quad (30)$$

3) *Misspecification Analysis:* To analyze the performance of our algorithm to certify misspecification cases, we set the initial hypothesis space Θ_0 as $\mathbf{c}_l = [-1, -1]^T$ and $\mathbf{c}_u = [0.8, 0.8]^T$, which does not contain $\bar{\Theta}_H$. In this setting, the algorithm automatically stops after seventh correction. The path of θ_i , $i = 1, \dots, 8$ is shown in Fig. 9. θ_i is continuously moving toward the boundary of Θ_0 . This fact verifies Theorem 2 and justifies the proposed method is capable of misspecification detection.

B. Quadrotor Navigation in Complex Environment

1) *Setup:* We consider a thrust-controlled quadrotor navigating in an 3-D zigzag tube, as shown in Fig. 10. The simulation environment is based on MuJoCo [58], and the quadrotor dynamics follows [20]. Initially, the quadrotor has no knowledge of the safety constraint. When it nears the tube wall, a simulated

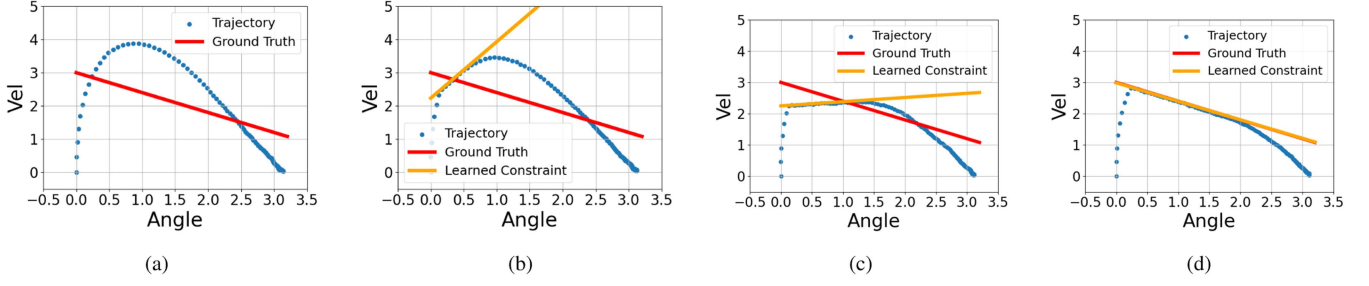


Fig. 7. Safe MPC rollout trajectory at different correction interaction. The learned constraint with θ_1 is omitted since it covers the whole first quadrant. As learning proceeds, the constraint boundary (orange line) converges to the ground-truth (red line). When the learning process terminates, the learned and ground-truth constraint boundaries almost overlap, suggesting the success of constraint learning from correction. (a) Initial trajectory (with θ_1). (b) After 3 corrections (with θ_4). (c) After 6 corrections (with θ_7). (d) After 14 corrections (with θ_{15}).

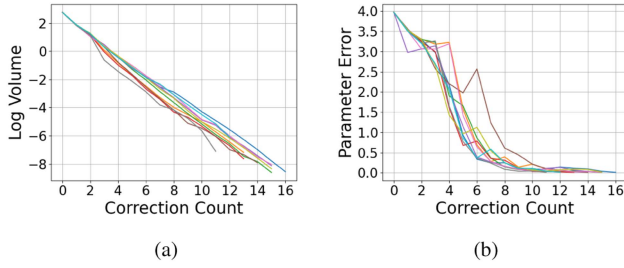


Fig. 8. (a) Plot of $\log \det \bar{H}$ with respect to correction numbers. In all algorithm runs (different colors), the plot shows the near-linear decrease of the logarithm volume, suggesting the exponential decrease of the volume of MVE of the hypothesis spaces, as asserted in Theorem 1. (b) Plot of the parameter estimation error $\|\theta_i - \theta_H\|_2$ versus correction numbers. In all runs, the parameter estimation error can converge to a value lower than 0.02 within 17 corrections, consistent with the theoretical bound in Theorem 1. (a) Logarithm MVE volume. (b) Parameter estimation error.

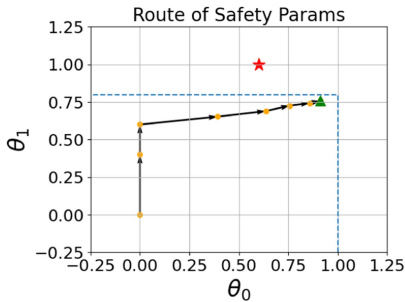


Fig. 9. Experiment of misspecification case. The blue dashed lines are the boundary of Θ_0 . Path of θ convergence, θ_i , $i = 1, \dots, 7$, is shown in orange dots with arrows showing the update direction. The ground-truth intent set is marked as the red star. Since misspecification happens for this case, θ will converge toward the boundary of Θ_0 (the green triangle). This is consistent with statement in Theorem 2.

human correction is generated to push it away from the tube wall. Details of the simulation and correction are provided in Appendix B-B. To learn positional safety constraints, we use two different types of feature vectors $\phi(\xi)$ in (5): polynomial feature vector and pretrained neural feature vector, respectively. In addition, we add velocity features to demonstrate that our method can simultaneously handle both positional and velocity constraints.

a) *Polynomial feature vector*: A vector of polynomial basis functions (up to third order to properly capture the zig-zag

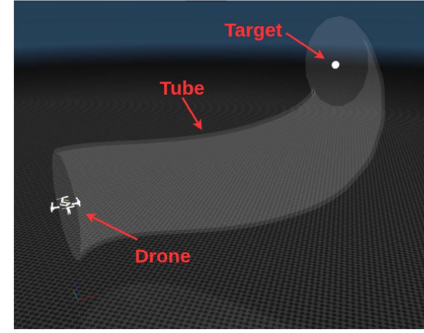


Fig. 10. Quadrotor navigation in a 3-D zigzag tube.

pattern of the tube) is defined in the quadrotors' 3-D position $[x, y, z]$ at the fifth timestep in motion plan ξ

$$\phi(\xi) = [x_5^3, y_5^3, z_5^3, x_5^2, y_5^2, z_5^2, x_5, y_5, z_5, x_5 y_5, y_5 z_5, x_5 z_5]^T. \quad (31)$$

We select the fifth time step for predictive collision avoidance. The hypothesis space is initialized as a box space with $c_l = [-80, \dots, -80]^T$ and $c_u = [200, \dots, 200]^T$. We select $\phi_0 = 1$ and $\gamma = 60$.

b) *Pretrained neural feature vector*: We obtain the neural spacial features from a pretrained signed distance function (SDF) [59]. Specifically, we pretrain an SDF neural network using the point cloud data samples near the tube surface. This simulates the application where some geometric or sensory data of the environment is available (such as via LIDAR and SLAM). The neural SDF is pretrained in a regression-like manner, with distance, normal, and Eikonal losses (see details in Appendix B-B). With the pretrained SDF function, we take the 16-D output of the second-to-last layer of the SDF network as our feature vector $\phi(\xi)$. The input to $\phi(\xi)$ is the quadrotor's 3-D position $[x, y, z]$ at the fifth timestep in MPC motion plan ξ . Such a choice of feature vector $\phi(\xi)$ means that we will use the proposed Safe MPC Alignment to fine tune the last layer of the learned SDF, $g(\xi) = \theta^T \phi(\xi)$, for its safe navigation control.

To facilitate the real-time safe MPC with the neural safety constraint $g(\xi) = \theta^T \phi(\xi)$, the optimization (1) uses the linearized $g(\xi)$ for calculation, which is linearized at the current robot state x_k^{real} . The initial bounds of the hypothesis space are

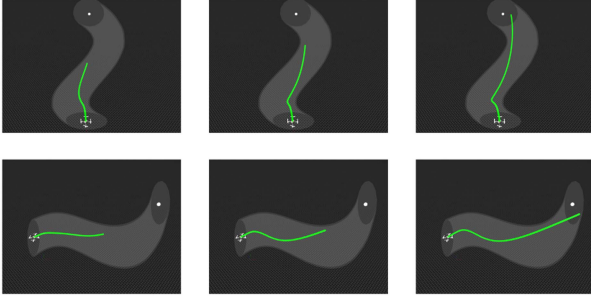


Fig. 11. Safe MPC rollout trajectories (in green) for learning safety constraints with polynomial features after 26, 38, 45 corrections (different columns). The first row and second row show the top and the side views of the navigation environment.

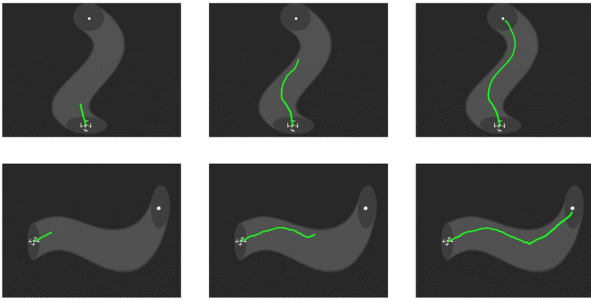


Fig. 12. Safe MPC rollout trajectories (in green) for learning safety constraints with neural features after 6, 9, 11 corrections (different columns). The first row show the top view, and second row shows the side view of the navigation environment.

set to $\mathbf{c}_l = [-0.2, \dots, -0.2]^\top$, $\mathbf{c}_u = [0.1, \dots, 0.1]^\top$. We select $\phi_0 = 0.1$ and $\gamma = 50$.

To ensure the feasibility of the learned constraint, in both cases of feature vectors, two additional cuts are added in each MVE update to enforce that the start and goal remain within the intended safe region. The task is successful if the drone reaches the goal without triggering this stop.

c) *Polynomial features with velocity features:* We consider the case where the safety constraints are additionally defined in the velocity of the quadrotor. After learning the positional constraint with the polynomial feature vector, a new set of velocity-based features over all time steps, $\phi(\xi) = [\|\mathbf{v}_1\|^2, \dots, \|\mathbf{v}_{10}\|^2]^\top$, is added. In this case, we will freeze the weights for polynomial features, and only learn the weights for velocity features to regulate the quadrotor's linear speed in navigation. Our objective is to impose an additional constraint that caps the quadrotor's velocity magnitude below 0.45 [m/s], while ensuring that the previously learned positional constraint remains satisfied for collision avoidance. When the quadrotor's velocity exceeds a correction triggering threshold (0.4), a simulated human correction is applied to push the quadrotor in the direction of negative velocity. The hypothesis space for the velocity features is initialized with bounds $\mathbf{c}_l = [0, \dots, 0]^\top$ and $\mathbf{c}_u = [0.16, \dots, 0.16]^\top$. We set the bias feature $\phi_0 = 1$ and select $\gamma = 100$.

2) *Result:* Figs. 11 and 12 show the rollout trajectories of the learned Safe MPC policy using polynomial and neural features, respectively. With polynomial features, the task is completed in

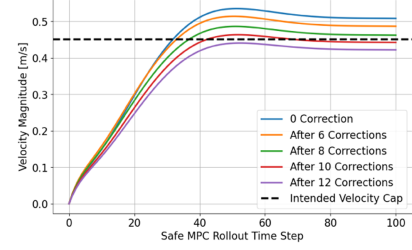


Fig. 13. Velocity profile during the first 100 timesteps (acceleration phase) of the drone trajectory. After 12 corrections, the quadrotor successfully learns to stay below the intended velocity cap of 0.45 (dashed line).

45 corrections, while the neural features require only 11. This is because the neural features, constructed from the second-to-last layer of pretrained neural SDF, have already encoded the partial knowledge of spatial constraints.

For the learning of velocity constraint, Fig. 13 shows the velocity profile of the quadrotor during the Safe MPC rollout in the 100 time steps (acceleration phase) after different number of simulated corrections (different colors). After 12 corrections, the quadrotor successfully learns to stay below the intended velocity cap of 0.45 (dashed line). Throughout this process, the positional safety constraint remains satisfied, with the quadrotor continuing to navigate within the tube to the target. This result demonstrates the flexibility of the proposed method in scenarios involving both positional and velocity constraints.

C. Baseline Comparison

In this session, we compare our method with two baseline methods using magnitude corrections in the pendulum environment (the task and model settings are identical to Section VI-A). Since the literature on directly learning safety constraints from human feedback is sparse, the chosen baselines are an adaptation from the most relevant methods.

Baseline 1: gradient matching. This baseline is inspired by [60], where raw human correction (with magnitude) is directly used as the gradient direction. In a similar spirit, when a human correction $\mathbf{a}$ is made on robot motion ξ_θ^γ , we directly use the human correction as the true gradient label to learn the constraint θ . This leads to a learning loss

$$\mathcal{L} = \|\mathbf{a} + \nabla B(\xi_\theta^\gamma)\|^2. \quad (32)$$

Baseline 2: maximum likelihood learning. This baseline follows [61], where it views $e^{g_\theta(\xi)}$ as a probability of human applying the “stop-feedback” to robot motion ξ , when the robot reaches in a ϵ_γ -proximity of the safety boundary, i.e., when $g_{\theta_H}(\xi_\theta^\gamma) > -\epsilon_\gamma$. Thus, a dataset of trajectory-correction pairs, i.e., $\mathcal{D} = \{\mathbf{a}_i, \xi_i\}_{i=0}^N$, can be used as the training data to learn θ using maximum likelihood, while also penalizing positivity of $g_\theta(\xi_i)$ as a regularizer term. This leads to the following loss:

$$\mathcal{L} = \sum_{i=1}^N (-g_\theta(\xi_i) + \alpha \cdot \text{softplus}(g_\theta(\xi_i))). \quad (33)$$

We choose $N = 15$ (similar to the correction number needed for our method to converge) and $\alpha = 5$ for its best performance.

The comparison between our method and the gradient matching baseline focuses on data efficiency. The convergence criteria of $\|\theta - \theta_H\| \leq 0.02$ is set for both methods. A total of 50 independent runs are performed and the total number of correction

TABLE I
COMPARISON OF SAFE MPC ALIGNMENT AND BASELINE I

Method	Correction count for convergence
Safe MPC Alignment	11.76 ± 2.14
Gradient Matching	240.38 ± 120.70

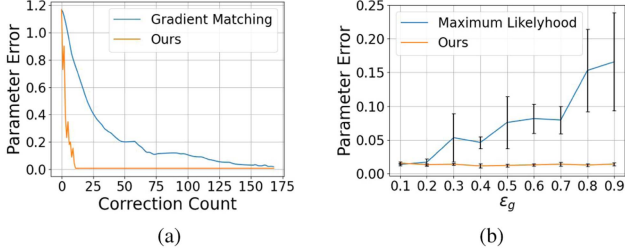


Fig. 14. (a) Comparison with gradient matching baseline: the parameter error $\|\theta - \theta_H\|$ versus learning iteration. This is best run with the fewest corrections for the baseline. Our method converges after 12 corrections, while the baseline needs more than 160 corrections. (b) Comparison with the maximum likelihood baseline under the different ϵ_g : the y-axis is $\|\theta - \theta_H\|$ at convergence with the error bar denoting the standard deviation. The maximum likelihood baseline fails to learn the correct weights when ϵ_g is large. (a) Comparison: gradient matching. (b) Comparison: maximum likelihood.

at convergence are counted. We use Adam optimizer [62] with a learning rate of 0.02. We summarize the results in Table I. The parameter error $\|\theta - \theta_H\|$ versus learning iteration for a best run (for the baseline) is plotted in Fig. 14(a). The comparison results demonstrate data efficiency of the proposed method compared to the gradient matching baseline, even though the latter uses both magnitude and direction information of corrections to learn.

The comparison between our method and the maximum likelihood baseline uses the varying positions of simulated correction, i.e., varying ϵ_g . Recall that the simulated corrections are generated only when $g_{\theta_H}(\xi_{\theta}^{\gamma}) > -\epsilon_g$. The value of ϵ_g determines the position of correction. In our comparison, we vary ϵ_g from 0.1 to 0.9 and track the converged parameter error for both methods. As shown in Fig. 14(b), safe MPC alignment consistently converges to the true θ_H regardless of ϵ_g , while the maximum likelihood baseline fails when $|\epsilon_g|$ is large. The results suggest the robustness of the proposed method against different correcting conditions compared to the maximum likelihood baseline method.

D. Ablation Study

We conduct two ablation studies to show the performance of our algorithm given different design aspects. The task is still based on the pendulum task in Section VI-A.

1) *Choosing θ_i : MVE Centering Versus Random Sampling*: We compare two strategies in selecting θ_i from the hypothesis space Θ_{i-1} : MVE centering (used in our method) and random sampling. For the latter, θ_i is sampled from the uniform distribution supported by Θ_{i-1} . We compare the total number of corrections at learning convergence under the two strategies.

With 50 independent runs, we observe the MVE centering requires 11.76 ± 2.14 corrections, while random sampling 11.88 ± 3.50 corrections. While the mean value of the correction count looks similar, the random sampling shows a slightly larger variance. This is understandable given sampling randomness. We also observed that the sampling process introduced a large

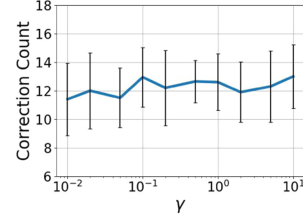


Fig. 15. Total number of corrections needed for convergence under different γ , the x-axis is of log scale.

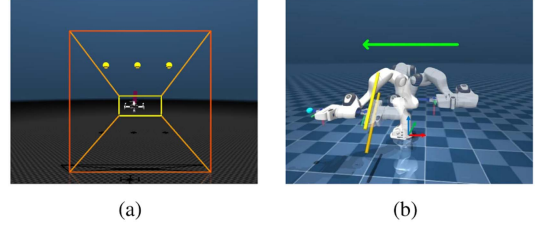


Fig. 16. Two simulated games in user study. (a) Drone navigation game. (b) Franka arm reaching game.

computational overhead, which is around 25% time slower than MVE centering. In contrast, the MVE centering used in our algorithm shows low variance and computational efficiency. Because selecting θ_i is cast as a convex programming for MVE centering, such an advantage would be more significant for a high-dimensional parameter space.

2) *Choosing γ* : We study how the setting of γ affects the learning performance. We vary γ from 0.01 to 10. Fig. 15 shows the total number of corrections needed at convergence versus different γ s. The results show that the learning performance remains the same given different settings of γ . Since γ is a parameter to weigh the safety conservativeness in the penalty-based MPC (3), it means our method can successfully learn a safety constraint when the robot executes with a safe MPC policy of different conservativeness levels.

VII. USER STUDY IN SIMULATION GAMES

We evaluate the Safe MPC Alignment in two human user studies for two simulation games. One is the drone navigation game and the other the Franka robot arm reaching game. In both games, the robot is controlled by an MPC policy and human participants provide real-time corrections through a keyboard. The objective of each game is to teach a robot to successfully learn safe MPC to move through environments with obstacles. Section VII-A describes the setting of the two games. Section VI-B details the user study procedure. The results and analysis are given in Section VII-C.

A. Two Simulation Games

In both user studies, human participants are asked to give correction to the following two physics-based simulation games developed based on MuJoCo [58].

1) *Drone navigation game*. As shown in Fig. 16(a), the goal of this game is to let a drone to successfully navigate through a narrow gate (yellow rectangle), reaching different targets

(shown in yellow points behind the gate). The safety constraint $\theta^T \phi$ is parameterized using a linear combination of spacial radial basis functions (RBFs), defined on yz plane at $x = 10$. The drone's control inputs are the thrusts of four rotors and human correction is applied via keyboard, with specific keys mapping to specific correction commands to the drone's control space. See the detailed task setup and keyboard interface in Appendix B-C.

- 2) *Franka arm reaching game*: As shown in Fig. 16(b), the goal of the Franka arm reaching game is to teach a Franka arm such that its gripper successfully reaches different target positions (cyan ball) through a narrow slot (two yellow bars). Note that safety constraint is only applied to end effector and the collision between the link and the slot is ignored. The safety constraint function $\theta^T \phi$ is parameterized as a weighted combination of RBFs defined on the z -axis and rotations around the x -axis. Human corrections are applied via keyboard, with specific keys mapping to specific correction commands applied to the robot control space $\mathbb{R}^6$ (we use operational space control). See the detailed task setup and keyboard interface in Appendix B-D.

In both games, the robots are controlled with MPC policy (3) but fail to maintain safety with the initial safety constraints. To simulate an emergency stop, the "Enter" key is used as a reset command, allowing users to reset the environment to its initial state whenever the participant deems the robot violating the safety constraint.

B. Participants, Procedures, and Metrics

1) *Participants*: We recruited 15 participants from Arizona State University, including undergraduates, masters, and Ph.D. students with different backgrounds. Among all participants, 13 are male and 2 female. All participants are new to the user study. This user study had been reviewed and approved by the Institutional Review Board of Arizona State University, and all participants had signed the consent forms.

2) *Procedure*: Each participant was asked to play the two simulation games. In each game, each participant plays 10 trials. In the drone navigation game, a trial is successful when the drone navigates through the gate without collision and reaches different targets 6 consecutive times with the learned safe MPC policy. A trial is deemed failed when there are too many corrections (exceeding 25), or the user believes the robot cannot be corrected anymore and gives up. In the Franka reaching game, a trial is deemed successful when the robot arm reaches different targets with the end-effector having no collision with the slot. The failure of one trial is determined when there are too many corrections (exceeding 25) or the user believes the robot cannot be corrected anymore and gives up. The result of every trial is recorded. Before formally starting the study, every user is instructed about the settings of the game and is given 2 chances to interact with the environment hands-on to get familiar with the keys.

3) *Metrics for Evaluation*: Two metrics are used to evaluate the performance of the proposed method: 1) *Success rate*: The overall success rate suggests the *effectiveness* of the proposed method. 2) *Numbers of corrections needed for a successful trial*: The averaged number of corrections performed by users in successful trials is a measure for the *efficiency* of the proposed

TABLE II
RESULT OF USER STUDY FOR SIMULATION GAMES

Environment	Success Rate	Correction count
Drone navigation	89.3% $\pm$ 7.7%	15.7 $\pm$ 10.6
Franka arm reaching	89.3% $\pm$ 11.2%	16.5 $\pm$ 6.1

method. Fewer corrections in successful trials suggest better human data efficiency.

C. Result and Analysis

1) *Results*: For the user study, we report the success rate and correction count in a successful game trial in Table II. Both metrics are averaged across users. The success rate is 89.3% $\pm$ 7.7% for the drone navigation game and 89.3% $\pm$ 11.2% for the Franka arm reaching game. The number of corrections for a successful trial is 15.7 $\pm$ 10.6 for the drone navigation game and 16.5 $\pm$ 6.1 Franka arm reaching game. In both games, each learning update of θ takes approximately 0.15 s, during which the simulation briefly pauses. This brief pause is imperceptible to the user, especially since human corrections occur infrequently.

The above result shows that the proposed method is effective (i.e., with a high success rate) in learning a safety constraint for varying tasks (recall targets are changing across trials). It also shows the efficiency of the proposed method—successful learning only requires tens of human corrections. Since this method is online and the user can perform multiple corrections in one robot rollout trajectory, the constraint can be learned in a few runs. In Table II, one can note that the drone navigation task has a higher variance in the correction count than the Franka arm reaching task. This could be because the drone game has a larger motion range and longer rollout horizon. As a result, human correction strategies, such as correction position and timing, can vary significantly, leading to a larger variance of correction count.

2) *Analysis*: To illustrate learning progress, in Figs. 17 and 18, we show the intermediate robot motion after different numbers of human corrections. Fig. 17 shows the safe MPC rollout trajectory of the drone after different numbers of human corrections. Fig. 18 shows the pose of the robot gripper when passing the slot after a different number of human corrections.

To understand the learned safety constraint $g_{\theta}(\xi)$, in Fig. 19, we draw the heat map of the learned $g_{\theta^*}(\xi)$ for the drone navigation game, with θ^* the averaged of weights learned. Recall in drone navigation game, g_{θ} is defined on yz plane at $x = 10$. In the heat map, we draw 0-level contour of g_{θ^*} in dashed lines, which is the learned safety boundary. Also, we plot yz -plane projection of the gate, which is the closed rectangle in the center.

Fig. 19 shows that the learned safety boundary (corresponding to $g_{\theta^*}(\xi) \leq 0$) is only similar to the top edge of the gate. This is explained below. As shown in Fig. 16(a), the targets are set higher than the gate, and thus during the learning process, the motion of the drone is always going up, as shown in Fig. 17. As a result, the majority of human corrections happen when the drone approaches the top edge of the gate, leaving the learned safety boundary accurate near the top edge of the gate and bottom edge untouched. It is expected that to learn a safety constraint that fully represents the gate, lower target positions should be added so that the drone will receive human corrections near the bottom edge of the gate.

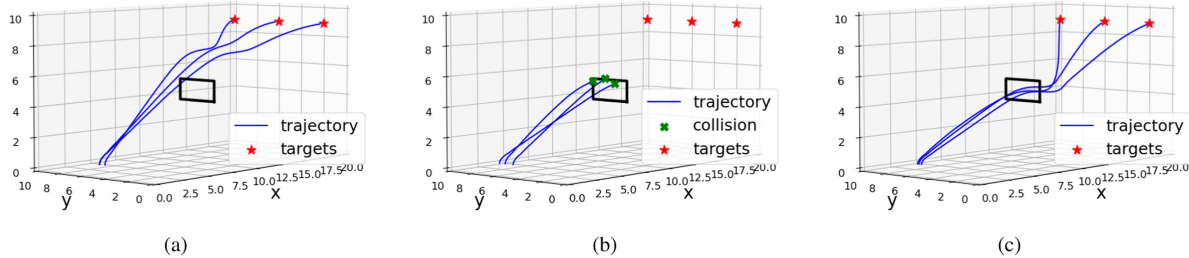


Fig. 17. Illustrations of the drone MPC rollout trajectory after 2, 6, 12 corrections. The red stars denote the targets and blue curves denote the trajectories. The gate is drawn as black frame and the collision is highlighted in green crosses. User corrections typically lower the drone to make it pass through the gate. (a) After 2 corrections. (b) After 6 corrections. (c) After 12 corrections.

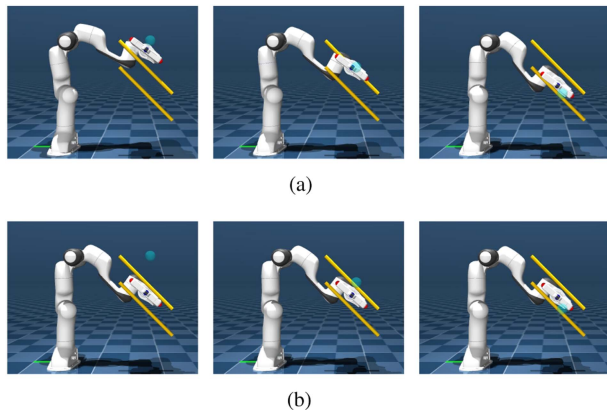


Fig. 18. Screenshots of the robot gripper when passing through the slot after $k = 4, 17$ corrections, given three different targets (cyan ball in different columns). (a) After 4 corrections, the gripper collides with the slot. (b) After 17 corrections, the gripper succeeds in passing the slot for all targets.

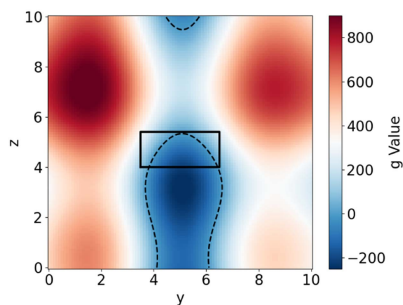


Fig. 19. Heatmap for the learned constraint in the drone game. The black solid line denotes the actual position of the gate in yz -plane. The black dashed curve denotes the boundary of the learned safety zone. Since the majority of human directional corrections happen when the drone approaches the top edge of the gate, the learned safety boundary is accurate near the top edge of the gate and the bottom boundary is untouched.

VIII. USER STUDY IN REAL-WORLD EXPERIMENT

We conduct a user study to evaluate the performance of our safe MPC alignment in a hardware Franka Research 3 experiment. We focus on the task of *mobile liquid pouring*: the robot performs liquid pouring while moving from one location to targets. In certain scenarios, this kind of task can be safety-critical: for example, if the liquid is corrosive/chemical/toxic, inappropriate mobile pouring will cause safety concerns. The

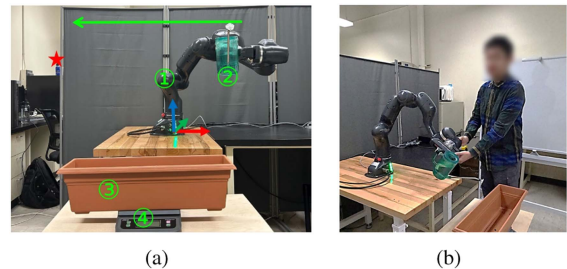


Fig. 20. Illustration images of the real-world experiment. (a) Mobile water pouring task using Franka robot arm. ① The robot arm. ② The watering pot. ③ The bar sink. ④ The scale to measure the amount of water poured into the sink. The target is marked as the red star. The long arrow indicates the motion direction, and the RGB frame is the world frame. (b) Illustration of human correction to the robot in the real-world experiment, by applying a contact force to the robot arm. The delta position of the end effector resulting from human contact interaction is used as human directional correction.

task is challenging because it is hard to manually define a safety constraint to accomplish liquid pouring without spilling. The Franka robot arm will learn a safe MPC controller from human physical corrections.

A. Hardware Setup

The setup of the mobile water pouring task is shown in Fig. 20(a). A water pot is gripped by the robot's end-effector. The robot moves from the starting point $[0.22, -0.5, 0.48]^T$ (with quaternion $[0, -0.707, 0, 0.707]^T$) to target point $[-0.65, -0.5, 0.5]^T$ (with quaternion $[0, -0.707, 0, 0.707]^T$). During the motion, the robot needs to pour more than 50 grams of water into a bar sink. The bar sink is placed in the middle of the robot's path, but the precise size and location of the bar sink are unknown.

As shown in Fig. 21, the hardware Franka arm platform has three main components: the parameterized safe MPC controller (high-level controller), the low-level controller, and the safe MPC alignment module. The safe MPC controller follows the safe MPC formulation of Franka arm reaching game in Appendix B-D (recall that the learnable safety constraint is parameterized by RBFs). The parameters of this Safe MPC controller are listed in Table III. The safe MPC controller receives the end-effector's current pose $\mathbf{x}$ and generates the action $\mathbf{u}$ [see (59) in the Appendix], which is the desired velocity of the end-effector. The low-level controller implements the operational space control to track the $\mathbf{u}$ given by the safe MPC controller.

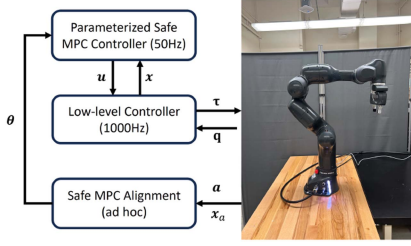


Fig. 21. System diagram for the hardware implementation. Different blocks indicate separate processes communicated via ROS.

TABLE III
PARAMETER LIST IN REAL-WORLD USER STUDY

Parameter	Value
Q_r	$\text{diag}(0.8, 0.8, 0.8)$
k_q	0
R_v	$\text{diag}(30, 30, 1)$
R_w	$\text{diag}(0.85, 0.85, 0.85)$
S_r	$\text{diag}(80, 80, 80)$
μ_q	80

The safe MPC alignment module updates the parameterize θ of the safe MPC controller. The update only happens when the alignment module detects a human correction α . The module implements our safe MPC alignment algorithm. Note that the alignment module and controller modules runs different process, thus the learning update does not affect the real-time MPC control of the robot arm. ROS is used for communication between different processes.

In the safe MPC, the cost function is set to penalize the current and target, as in Appendix B-D. Since it is difficult to explicitly define a parameterized safety constraint using the state of pouring water, we instead define the constraints only using the robot state. Specifically, we use the weighted RBFs to parameterize the safety constraint, similar to Appendix B-D.

As shown in Fig. 20(b), to give a physical correction to the Franka robot arm, a human user needs to physically touch/hold any link of the robot arm, and the applied force will immediately activate the zero-torque control mode. Suppose the joint positions of the robot arm before and after human physical correction are $\mathbf{q}_{\text{before}}$ and $\mathbf{q}_{\text{after}}$, respectively, we calculate the corresponding raw human correction $\mathbf{a}_r$ using

$$\mathbf{a}_r = \mathbf{J}(\mathbf{q}_{\text{before}})(\mathbf{q}_{\text{after}} - \mathbf{q}_{\text{before}}) \quad (34)$$

with $\mathbf{J}$ the Jacobian of the end-effector. The calculated $\mathbf{a}_r \in \mathbb{R}^6$ is considered as the displacement of the end-effector pose, intended by the human user.

B. Participants, Procedures, and Metrics

All participants recruited in Section VII-B also participate in the hardware experiment. They are told of the objective before starting trials. One episode is defined as one robot motion from the starting position to the targets, during which the human can apply corrections at any time instances. One learning trial may include multiple episodes (with a maximum of 15 corrections). During learning, the nozzle of the water pot is sealed, and we switch to the unsealed nozzle and measure the amount of water poured into the flower pot after the user is satisfied with the motion of the robot. The successful trial is defined when the

amount of poured water is over 50 g. Each user performs 5 trials, and the same metrics in Section VII-B are recorded.

C. Result and Analysis

In the real-world experiment, the success rate is $84.0\% \pm 16.7\%$ and the average number of corrections needed for successful trials is 6.88 ± 3.26 . The results show the efficacy and efficiency of our method in real-world applications. To illustrate the learning progress, Fig. 22 shows the robot behavior after different numbers of corrections in one trial.

Since this real-world Franka arm experiment is similar to the simulated Franka arm reaching game in Appendix B-D, we compare the user study results for both and have the following conclusion and analysis.

- 1) *Real-world experiment has slightly lower success rate:* While the difference between simulation and real-world studies is small in success rate, we have identified some factors for such a difference. On human aspects, most of the participants have no experience or knowledge in operating robot hardware, and thus, when directly interacting with hardware, their corrections can be conservative or ineffective due to safety considerations. On hardware side, noisy correction signals and the possible timing delay between the recorded correction and the robot state can contribute to failure.
- 2) *Real-world experiment needs less human corrections:* This could be attributed to two reasons. First, the constraint needed for accomplishing the water pouring task could be more flexible than that for reaching task in the simulation game. This makes the safety constraints easy to learn for the real-world task. Second, different from simulation games, where human users can only perceive the robot arm from the screen, a human user has a better perception of the robot's motion in the real world. A good perception of the robot's state could help humans give more effective feedback each time, reducing the total number of human interactions for successful robot learning.

D. Discussion

Based on the above hardware and simulation user studies, we discuss some design choices and potential limitations of the proposed method.

1) *Learning Task-Independent Safety Constraints:* In the real-world and simulation experiments, the learned safety constraint could be task-specific, instead of task-independent. For instance, in the drone navigation game, since the three task targets are placed above the gate, the learned constraint shown in Fig. 19 can only reflect the upper edge of the unknown gate. One limitation is that the learned constraints have poor generalization to new tasks. For example, in drone navigation, if a new target is placed below the gate, the drone with the learned safety constraint Fig. 19 will fail.

To improve the generalization of the learned safety constraints or learning task-independent safety constraints, one solution is to increase the diversity of task samples. By doing so, it is expected that the robot can receive human correction on the entire boundary of the safety region.

2) *Robustness Against Human Mistakes:* The proposed *safe MPC alignment* is based on the hypothesis cutting. In our formulation, a hypothesis cut (7) is based on an average of

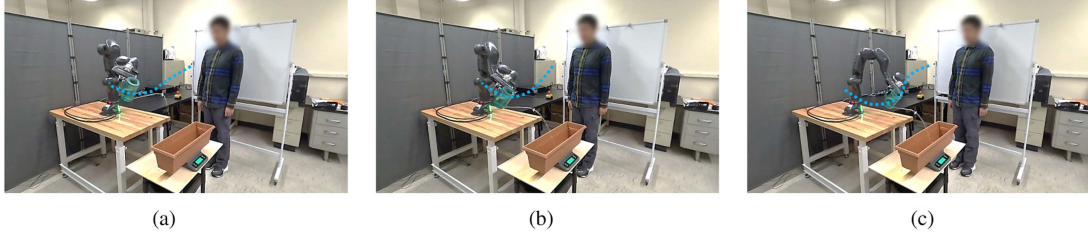


Fig. 22. Illustrations of one trial of the real-world experiment. The trajectories of the watering pot is denoted by blue dashed lines. (a) After 2 corrections, the arm begins tilting the pot but spills out the water. (b) After 3 corrections, the arm can pour water into the sink but the human is not satisfied with its height and the altitude of pouring. (c) After 5 corrections, the arm learns to tilt only along the direction of the x -axis and successfully aligns with human intent.

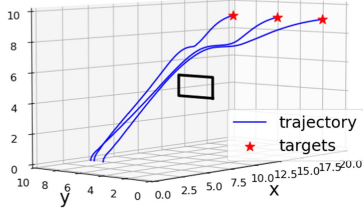


Fig. 23. Failure case in the drone game user study. A user makes a wrong “up” correction at the beginning and fails to correct the motion of the drone.

multiple human corrections, such an averaging treatment can mitigate human mistakes for one cut. But still, in practice, human irrationality can lead to the wrong cut of the hypothesis space, which is the main reason for failure in user studies. For example, in the drone navigation game, we found that failure game trials usually happen when a human gives too many wrong directional corrections. A failure case is in Fig. 23.

To further improve the robustness of the method against human mistakes, two extensions of the current method can be explored. The first is to add a relaxation factor $\epsilon \geq 0$ in our cutting hypothesis. For example, we can rewrite (7) as

$$\forall \theta_H \in \bar{\Theta}_H, \mathbb{E}_a \langle \mathbf{a}, -\nabla B(\xi_{\theta}^{\gamma}, \theta_H) \rangle + \epsilon \geq 0. \quad (35)$$

Another is to add a selection mechanism for human correction. For example, one can use some statistical method to rule out mistake corrections before performing a hypothesis cut. We will leave those possible extensions in our future work.

3) *Limitation of Weight-Feature Constraints*: In this work, we assume the set of task-relevant features in $\phi(\xi)$ is selected as a priori. These features define the space in which the safety constraints are represented. However, in practice, identifying appropriate features can be challenging. Poorly chosen features may fail to capture the intended safety constraints, leading to undesired learning outcomes. Furthermore, feature representations are often task-dependent—features that are effective in one task may not generalize to another. This task-specific dependence inherently limits the development of a unified framework for constraint learning across diverse tasks, making the lack of task-generalizable feature selection a key limitation of the proposed method. At the same time, learning unstructured constraints entirely from scratch would demand extensive human interaction and involvement, rendering such approaches impractical for most real-world applications. This challenge likely explains why recent works on learning from human feedback [13], [18],

[20] commonly assume task-specific feature structures or prior knowledge of the environment/tasks.

We recognize that identifying or learning a universal feature representation that can flexibly describe a wide variety of safety constraints remains an important and open research challenge. We consider this an exciting direction for future research, as it could significantly increase the applicability and effectiveness of safety constraint learning in real world.

IX. CONCLUSION

This article proposes a safe MPC alignment method to online learn a safety constraint in safe MPC controllers from human directional feedback. The method, named hypothesis space cutting, has a geometric interpretation: it exponentially cuts the hypothesis space based on each received human directional feedback. The method has a provable guarantee for the total number of human feedback needed for learning, and also can declare hypothesis misspecification. Numeric examples, user studies on simulation games, and a user study on the real-world robot arm, covering varying robot tasks, were used to validate the effectiveness of the proposed approach. The results show that the proposed method can learn safety constraints effectively and efficiently. To the best of authors' knowledge, this is the first certifiable method to enable online learning of safety constraints with human feedback.

APPENDIX A

PROOF OF LEMMAS AND THEOREMS

A. Proof of Lemma 2

Proof: We prove this lemma by induction. First, we have $\bar{\Theta}_H \subset \Theta_0$. Suppose $\bar{\Theta}_H \subset \Theta_{i-1}$, $i = 1, 2, 3, \dots$. Due to the condition $\bar{\Theta}_H \subset \mathcal{C}_i$ in (12), following the update (13), we have $\bar{\Theta}_H \subset \Theta_i = \Theta_{i-1} \cap \mathcal{C}_i$, which completes the proof. $\square$

B. Proof of Lemma 3

Proof: Equation (14) can be directly obtained from (12), (2), and (5). Based on the definition of the penalty objective function (2), $\langle \bar{\mathbf{a}}_i, \nabla B(\xi_{\theta_i}^{\gamma}, \theta) \rangle \leq 0$ is a linear fractional inequality

$$\langle \bar{\mathbf{a}}_i, \nabla J(\xi_{\theta_i}^{\gamma}) \rangle + \gamma \frac{1}{-g_{\theta}(\xi_{\theta_i}^{\gamma})} \langle \bar{\mathbf{a}}_i, \nabla g_{\theta}(\xi_{\theta_i}^{\gamma}) \rangle \leq 0 \quad (36)$$

with a positive denominator $-g_{\theta}(\xi_{\theta_i}^{\gamma}) > 0$ due to (12).

Multiplying the denominator on both side and substituting the weight-feature expression of g_{θ} in (5) yields $\theta^T \mathbf{h}_i \leq$

b_i in Lemma 3. The condition $-g_{\theta}(\xi_{\theta_i}^\gamma) > 0$ itself yields $\theta^\top \phi(\xi_{\theta_i}^\gamma) < -\phi_0(\xi_{\theta_i}^\gamma)$ in Lemma 3. $\square$

C. Proof of Lemma 4

Proof: For any $i = 1, 2, 3, \dots$, the plan $\xi_{\theta_i}^\gamma$ is a solution to the optimization (3), with $\theta = \theta_i$. Recall Lemma 1, $g_{\theta_i}(\xi_{\theta_i}^\gamma) < 0$. Extending the feature-weight expression of $g_{\theta_i}(\xi_{\theta_i}^\gamma)$, one can get $\theta_i^\top \phi(\xi_{\theta_i}^\gamma) < -\phi_0(\xi_{\theta_i}^\gamma)$. Since the optimal solution $u_{0:T-1}^{\theta_i, \gamma}$ is a minimizer to (3), it means

$$\nabla B(\xi_{\theta_i}^\gamma, \theta_i) = 0. \quad (37)$$

Note that with $g_{\theta_i}(\xi_{\theta_i}^\gamma) < 0$, the above $\nabla B(\xi_{\theta_i}^\gamma, \theta_i)$ is well defined. With (37)

$$\langle \mathbf{a}, \nabla B(\xi_{\theta_i}^\gamma, \theta_i) \rangle = 0 \quad (38)$$

always holds. Extending the above (38) leads to

$$\langle \mathbf{a}, \nabla J(\xi_{\theta_i}^\gamma) \rangle - \gamma \frac{1}{g_{\theta_i}(\xi_{\theta_i}^\gamma)} \langle \mathbf{a}, \nabla g_{\theta_i}(\xi_{\theta_i}^\gamma) \rangle = 0. \quad (39)$$

Due to $-g_{\theta_i}(\xi_{\theta_i}^\gamma) > 0$, one can multiple $-g_{\theta_i}(\xi_{\theta_i}^\gamma)$ on both sides of the above equality, which leads to $\theta_i^\top \mathbf{h}_i = b_i$ based on the definition (15). The second term in (18) directly follows $g_{\theta_i}(\xi_{\theta_i}^\gamma) < 0$. $\square$

D. Proof of Lemma 5

Proof: By Lemma 4, the MVE center $\theta_i = \bar{\mathbf{d}}_{i-1}$ is on the cutting hyperplane $\theta^\top \mathbf{h}_i = b_i$. Directly from [63], one has

$$\frac{\text{Vol}(\Theta_{i-1} \cap \{\theta | \theta^\top \mathbf{h}_i \leq b_i\})}{\text{Vol}(\Theta_{i-1})} \leq 1 - \frac{1}{r}. \quad (40)$$

Since

$$\text{Vol}(\Theta_{i-1} \cap \{\theta | \theta^\top \mathbf{h}_i \leq b_i\}) \geq \text{Vol}(\Theta_{i-1} \cap \mathcal{C}_i) = \text{Vol}(\Theta_i) \quad (41)$$

based on (12), (13), one can get

$$\frac{\text{Vol}(\Theta_i)}{\text{Vol}(\Theta_{i-1})} \leq 1 - \frac{1}{r} \quad (42)$$

which concludes the proof. $\square$

E. Proof of Theorem 1

Proof: We complete the proof in two parts. First, we prove the number of learning iterations is less than K , then we use this to prove there will be a $\theta_k \in \bar{\Theta}_H$, $k = 0, 1, 2, \dots, K-1$.

We first prove the total number of iterations must be less than K by contradiction. Suppose the learning iteration number $i \geq K$. With recursively applying lemma 5, one can obtain

$$\text{Vol}(\Theta_i) \leq \left(1 - \frac{1}{r}\right)^i \text{Vol}(\Theta_0). \quad (43)$$

When $i \geq K$

$$\text{Vol}(\Theta_i) < \frac{\tau_r \rho_H^r}{\text{Vol}(\Theta_0)} \text{Vol}(\Theta_0) = \tau_r \rho_H^r \leq \text{Vol}(\bar{\Theta}_H). \quad (44)$$

However, from Lemma 2, one has $\bar{\Theta}_H \subset \Theta_i$, meaning $\text{Vol}(\bar{\Theta}_H) \leq \text{Vol}(\Theta_i)$. This leads to a contradiction with (44). Therefore, the total number of iterations must be less than K .

Next, we prove $\exists k < K$ such that $\theta_k \in \bar{\Theta}_H$. In contradiction, suppose $\forall k < K$, $\theta_k \notin \bar{\Theta}_H$. By Lemma 2, this means that for all $i < K$, $\bar{\Theta}_H \subset \mathcal{C}_i$. Then, the algorithm will continue for $i = K$. Consequently, the total number of iterations can reach K , which contradicts that the algorithm will terminate within K iterations. $\square$

F. Proof of Theorem 2

Proof: We complete the proof in two parts. First we prove the algorithm converges to Θ^* , which has $\text{Vol}(\Theta^*) = 0$. Based on this, we next prove $\Theta^* \cap \text{int } \Theta_0 = \emptyset$.

We first prove the algorithm converges to Θ^* which has $\text{Vol}(\Theta^*) = 0$. Given the condition $\Theta_0 \cap \bar{\Theta}_H = \emptyset$, the algorithm will never terminate because $\forall i = 1, 2, 3, \dots, \infty$, $\mathcal{C}_i \supset \bar{\Theta}_H$ due to Lemma 2. Since the volume of the initial hypothesis space is finite, by applying Lemma 5 we have

$$\text{Vol}(\Theta^*) = \lim_{i \rightarrow \infty} \left(1 - \frac{1}{r}\right)^i \text{Vol}(\Theta_0) = 0. \quad (45)$$

Next, we prove $\Theta^* \cap \text{int } \Theta_0 = \emptyset$ by contradiction. Suppose there exists $\hat{\theta} \in \Theta^* \cap \text{int } \Theta_0$. Define the ball contained in $\bar{\Theta}_H$ to be $\mathcal{B}(\theta_H^c, \rho_H) = \{\theta | \|\theta - \theta_H^c\| \leq \rho_H\} \subseteq \bar{\Theta}_H$ with θ_H^c as the center. By the equation (13), we can note

$$\Theta^* = \Theta_0 \cap \mathcal{C} \quad (46)$$

with $\mathcal{C} = \bigcap_{i=1}^{\infty} \mathcal{C}_i$ a convex set. So we have

$$\Theta^* \subseteq \mathcal{C}. \quad (47)$$

By (12)

$$\mathcal{B}_H \subseteq \bar{\Theta}_H \subseteq \mathcal{C}. \quad (48)$$

From the relationship between sets above, $\theta_H^c \in \mathcal{C}$, $\hat{\theta} \in \mathcal{C}$.

Since $\theta_H^c \notin \Theta_0$, there exists $\mu > 0$ such that

$$\tilde{\theta} = \hat{\theta} + \mu(\theta_H^c - \hat{\theta}) \in \text{int } \Theta_0. \quad (49)$$

Then, one can define a ball $\mathcal{B}(\tilde{\theta}, d)$ centered at $\tilde{\theta}$ with radius

$$d = \min_{\theta \in \text{bd } \Theta_0} \|\tilde{\theta} - \theta\|. \quad (50)$$

Thus, $\mathcal{B}(\tilde{\theta}, d) \subset \text{int } \Theta_0$.

Next, we will show $\mathcal{B}(\tilde{\theta}, \mu\rho_H) \subset \mathcal{C}$. In fact, for any θ satisfying $\|\theta - \tilde{\theta}\| \leq \mu\rho_H$, one has

$$\begin{aligned} \theta &= \mu \left(\frac{1}{\mu}(\theta - \tilde{\theta}) + \theta_H^c \right) + \tilde{\theta} - \mu\theta_H^c \\ &= \underbrace{\mu \left(\frac{1}{\mu}(\theta - \tilde{\theta}) + \theta_H^c \right)}_{\in \mathcal{B}_H \subseteq \mathcal{C}} + (1 - \mu) \underbrace{\tilde{\theta}}_{\in \Theta^* \subseteq \mathcal{C}}. \end{aligned} \quad (51)$$

The convexity of $\mathcal{C}$ leads to $\theta \in \mathcal{C}$, which means the ball $\mathcal{B}_\mu \triangleq \{\theta | \|\tilde{\theta} - \theta\| \leq \mu\rho_H\} \subset \mathcal{C}$.

TABLE IV
PHYSICS PARAMETERS IN PENDULUM NUMERICAL STUDY

Parameter	Value	Interpretation
m	1 kg	Weight of the pendulum
l	1 m	Length of the pendulum
g	10 m/s ²	Gravitational Acceleration
d	0.1	Dampening Coefficient

Let a new ball be $\mathcal{B}(\tilde{\theta}, \min(\mu\rho_H, d))$. Based on the previous two results of $\mathcal{B}(\tilde{\theta}, d) \subset \text{int}\Theta_0$ and $\mathcal{B}(\tilde{\theta}, \mu\rho_H) \subset \mathcal{C}$, one has $\mathcal{B}(\tilde{\theta}, \min(\mu\rho_H, d)) \subset (\Theta_0 \cap \mathcal{C}) = \Theta^*$. As a result

$$\text{Vol}(\Theta^*) \geq \text{Vol}(\mathcal{B}(\tilde{\theta}, \min(\mu\rho_H, d))) > 0 \quad (52)$$

which contradicts $\text{Vol}(\Theta^*) = 0$ from (45). Thus, $\Theta^* \cap \text{int}\Theta_0 = \emptyset$ must hold. $\square$

APPENDIX B

SETTINGS OF EXPERIMENTS AND USER STUDIES

A. Inverted Pendulum

In the dynamics of an inverted pendulum (26), the physical parameters is set in Table IV. When the system state violates $g_{\theta_H}(\xi_{\theta}^T) \leq 0$ or it is near the target, the pendulum state is uniformly randomly reset between $[0, 0]^T$ and $[\frac{2\pi}{3}, 3]^T$. Simulated human correction will continue to be applied until the learned θ_i fall in $\bar{\Theta}_H$, which terminates one trial.

B. Simulated Quadrotor Navigation

1) *Environment*: The quadrotor starts from $[0.0, 0.0, 5.0]^T$ [m]. The starting quaternion $\mathbf{q}_{\text{init}} = [1, 0, 0, 0]^T$. The target is to navigate to the target position $[15.0, 0.0, 10.0]^T$ [m] within a zigzag tube. The average radius of the tube section is set to be 2.5 m.

2) *Learning Neural Features*: We construct the neural feature ϕ using a neural SDF model trained on synthetic 3-D point samples around the tube surface. Each input coordinate (scaled for stability) is first encoded using sinusoidal positional embeddings, then passed through a five-layer MLP (width 128, ReLU activations). The positional embeddings are skip-connected to the input of the fifth hidden layer. The penultimate layer outputs a 16-D feature vector used as ϕ ; the final layer predicts a scalar signed distance supervised during training.

Training data are generated by randomly perturbing surface points on the tube. Signed distance labels are computed as the offset from the nearest point on the tube surface to the sample point, and normals are estimated from the same displacement vector pointing from the nearest surface point to the sample point. The model is trained with three loss terms: L_2 distance loss, cosine similarity loss on normals (via auto-diff), and Eikonal loss to regularize gradient norm, which is similar to the work [64]. Optimization is done using Adam for 250 epochs with a batch size of 1024.

3) *Simulated Correction and Stop Behavior*: Correction signals $\mathbf{a}_x = [-1, 0, 1, 0]^T$, $\mathbf{a}_y = [0, 1, 0, -1]^T$, $\mathbf{a}_z = [1, 1, 1, 1]^T$ in drone control space is used throughout the experiments, corresponding to different motion directions in cartesian space.

During the process of learning positional constraints with polynomial features or neural features, if the distance between the quadrotor and the nearest point on the surface of the tube is less than 0.6, a simulated correction $\mathbf{a}$ is applied in the action space of the drone, based on the vector $\mathbf{d} = (d_x, d_y, d_z)$ pointing from the closest tube surface point to drone position

$$\mathbf{a} = d_y \mathbf{a}_y + d_z \mathbf{a}_z. \quad (53)$$

Intuitively, the simulated correction “pulls” the drone to the center of the tube (designated safe region). After each correction, the navigation restarts, emulating an emergency stop. During the process of learning velocity constraint, simulated human corrections against the velocity direction of form

$$\mathbf{a} = -v_x \mathbf{a}_x - v_y \mathbf{a}_y - v_z \mathbf{a}_z \quad (54)$$

are applied whenever the velocity $\mathbf{v} = (v_x, v_y, v_z)$ exceeds 0.4, and an emergency stop is triggered if it surpasses 0.45.

C. Quadrotor Navigation Game With Real Human Feedback

1) *Environment*: The drone starts from $[0.1, y_{\text{init}}, 0.5]^T$ with y_{init} uniformly randomly in an interval $[4, 6]$. The starting quaternion $\mathbf{q}_{\text{init}} = [1, 0, 0, 0]^T$. We set three different targets at positions $[19, 1, 9]^T$, $[19, 5, 9]^T$ and $[19, 9, 9]^T$, with quaternion $[1, 0, 0, 0]^T$. For the narrow gate (yellow rectangle) in Fig. 16(a), both its size and position are unknown, and one only roughly knows its x position $x_{\text{gate}} \approx 9.8$.

Since the location of the gate is unknown, the drone needs to learn a safety constraint in its safe MPC policy with humans guidance in order to successfully fly through the gate. In the drone MPC policy (1), the dynamics and its parameters are set following [20]; and we generically set the constraint function $g_{\theta}(\xi)$ using a set of N Gaussian RBFs

$$g_{\theta}(\xi) = \phi_0 + \sum_{i=1}^N \theta_i \phi_i(\xi). \quad (55)$$

As discussed in Section III-C4, we fix $\phi_0 = -1$ to avoid learning ambiguity, and use $N = 20$ temporally accumulated RBFs defined as

$$\phi_i(\xi) = \sum_{t=0}^T \beta_t \text{RBF}_i(\mathbf{x}_t) \quad (56)$$

where β_t is a weighting factor to balance the immediate and future constraint satisfaction. If one emphasizes instant safety, more weights will be given to the initial time step. In our implementation, we choose an exponentially decaying β_t starting at $t = 5$, i.e., $\beta_t = 0$, $t < 5$ and $\beta_t = 0.9^{t-5}$, $t \geq 5$ to achieve better predictive safety performance. In (56), each RBF is defined as

$$\text{RBF}_i(\mathbf{x}_t) = \begin{cases} \exp(-\epsilon^2 \|r_{xy,t} - c_{xy,i}\|^2), & i = 1, \dots, 10 \\ \exp(-\epsilon^2 \|r_{xz,t} - c_{xz,i}\|^2), & i = 11, \dots, 20. \end{cases} \quad (57)$$

Here, the variables $r_{xy,t} = [r_{x,t}, r_{y,t}]^T$ and $r_{xz,t} = [r_{x,t}, r_{z,t}]^T$ with r_x, r_y, r_z from $\mathbf{x}_t$. The i th center $c_{xy,i} = [c_{x,i}, c_{y,i}]^T$ or $c_{xz,i} = [c_{x,i}, c_{z,i}]^T$ is defined with $c_{x,i} = x_{\text{gate}} = 9.8$, $c_{y,i}$ and $c_{z,i}$ are evenly chosen on interval $[0, 10]$. We set $\epsilon = 0.45$. The initial hypothesis space Θ_0 for possible θ in (55) is set as $\mathbf{c}_l = [-80, \dots, -80]^T$ and $\mathbf{c}_u = [100, \dots, 100]^T$. γ is 50. This

TABLE V
KEYBOARD COMMANDS IN UAV USER STUDY

Key	Correction	Interpretation
"up"	$[-1, 0, 1, 0]^T$	Go forward (adding positive y-axis torque)
"down"	$[1, 0, -1, 0]^T$	Go backward (adding negative y-axis torque)
"left"	$[0, 1, 0, -1]^T$	Go left (adding negative x-axis torque)
"right"	$[0, -1, 0, 1]^T$	Go right (adding positive x-axis torque)
"shift"	$[1, 1, 1, 1]^T$	Go up (increasing all thrusts)
"ctrl"	$[-1, 1, 1, 1]^T$	Go down (decreasing all thrusts)
"enter"	N/A	Reset robot position and switch target.

TABLE VI
KEYBOARD COMMANDS IN ROBOT ARM USER STUDY

Key	Correction	Interpretation
"left"	$[0, 0, 0, 0, 1]^T$	Clock-wise rotation of the gripper
"right"	$[0, 0, 0, 0, -1]^T$	Anti-clock-wise rotation of the gripper
"shift"	$[0, 0, 0, -1, 0]^T$	Go up in world frame
"ctrl"	$[0, 0, 0, 1, 0]^T$	Go down in world frame
"Enter"	N/A	Reset robot position and switch target.

initialization blocks the drone from advancing to the gate at the beginning of the game, leading to conservative behavior in the learning process. The drone will proceed toward the gate after few corrections.

2) *Human Correction Interface*: The motion of the drone is rendered in two camera views in MuJoCo visualization for better human perception: one is the first-person view tracking the drone, the other a third-person fixed view. The gate, the current MPC planning solution, and the target position are visible to the users.

Human users apply corrections to the drone via a keyboard during the drone flying. The user is informed of the mappings between the key press and the corresponding correction in the game, listed in Table V. The safety constraint (55) in the drone's MPC policy is immediately updated once a user correction is detected. When the user presses "ENTER," the drone is reset to the initial position and target will randomize.

D. Franka Constrained Reaching Game

1) *Environment*: The arm starts at a configuration in which the end-effector position $\mathbf{r}_{\text{init}} = [0.22, -0.5, 0.48]^T$ and altitude $q_{\text{init}} = [0, -0.707, 0, 0.707]^T$. The reaching target position $\mathbf{r}^*$ will be randomly chosen from three options $[-0.6, -0.5, 0.4]^T$, $[-0.6, -0.5, 0.5]^T$, $[-0.6, -0.5, 0.6]^T$. The reaching target attitude is $\mathbf{q}^* = [0, -0.707, 0, 0.707]^T$. The slot size and position are unknown except its rough x -position, which is $x_{\text{slot}} \approx -0.15$.

We adopt a two-level control scheme, where a safe MPC policy (high-level) is defined for the end effector, which generates the desired velocity for a given low-level operational space controller to track. In the safe MPC policy, we model the end-effector dynamics as

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v}^d \\ \dot{\mathbf{q}} &= \frac{1}{2} \Omega(R^T(\mathbf{q})\mathbf{w}^d)\mathbf{q} \end{aligned} \quad (58)$$

where $\mathbf{r}$ and $\mathbf{q}$ is the position and the quaternion of the arm end-effector in the world frame. $R^T(\mathbf{q})$ is the rotation matrix

from the end-effector frame to the body frame. $\Omega(\cdot)$ is the matrix operator of the angular velocity for quaternion multiplication. Define the robot state and control input

$$\mathbf{x}_t = [\mathbf{r}_t, \mathbf{q}_t] \in \mathbb{R}^7, \quad \mathbf{u}_t = [\mathbf{v}_t^d, \mathbf{w}_t^d] \in \mathbb{R}^6 \quad (59)$$

respectively.

The above end-effector dynamics is discretized with $\Delta t = 0.1\text{s}$. In (1), the MPC horizon is $T = 20$. Let $\mathbf{x}^* = [\mathbf{r}^*, \mathbf{q}^*]$ denote the target end-effector pose. The step cost $c(\mathbf{x}_t, \mathbf{u}_t)$ and terminal cost $h(\mathbf{x}_T)$ in (1) is set to

$$\begin{aligned} c(\mathbf{x}_t, \mathbf{u}_t) &= (\mathbf{r}_t - \mathbf{r}^*)^T Q_r (\mathbf{r}_t - \mathbf{r}^*) + k_q \text{diff}(\mathbf{q}_t, \mathbf{q}^*) \\ &\quad + \mathbf{v}_{I,t}^T R_v \mathbf{v}_{I,t} + \mathbf{w}_{I,t}^T R_w \mathbf{w}_{I,t} \\ h(\mathbf{x}_T) &= (\mathbf{r}_T - \mathbf{r}^*)^T S_r (\mathbf{r}_T - \mathbf{r}^*) + \mu_q \text{diff}(\mathbf{q}_T, \mathbf{q}^*). \end{aligned} \quad (60)$$

In this game, the cost parameters are set as $Q = \text{diag}(2, 2, 2)$, $k_q = 1$, $R_v = \text{diag}(30, 30, 1)$, $R_w = \text{diag}(0.85, 0.85, 0.85)$, $S_r = \text{diag}(80, 80, 80)$, and $\mu_q = 60$. The output of the MPC controller is sent to the low-level controller to track.

Similar to Appendix B-C, to constrain r_z and $\mathbf{q}$ for proper height and tilt in order to pass through the slot. We use a set of $N = 20$ Gaussian RBFs to parameter the safety constraint

$$g_\theta(\boldsymbol{\xi}) = \phi_0 + \sum_{i=1}^N \theta_i \phi_i(\boldsymbol{\xi}) \quad (61)$$

with

$$\phi_i(\boldsymbol{\xi}) = \sum_{t=0}^T \beta_t \text{RBF}_i(\mathbf{x}_t), \quad \text{and} \quad (62)$$

$\text{RBF}_i(\mathbf{x}_t)$

$$= \begin{cases} \sigma(r_{x,t}) k_z \exp(-\epsilon_z^2 (r_{z,t} - c_{z,i})^2), & i=1, \dots, 10 \\ \sigma(r_{x,t}) \exp(-\epsilon_q^2 \text{diff}(\mathbf{q}_t, \mathbf{q}_{c,i})), & i=11, \dots, 20. \end{cases} \quad (63)$$

Here, k_z is used to balance the scaling between position and orientation RBFs, $\sigma(r_{x,t})$ is defined as

$$\sigma(r_{x,t}) = \text{sigmoid} \left(50 \left((r_{x,t} - x_{\text{slot}})^2 - \left(\frac{w}{2} \right)^2 \right) \right) \quad (64)$$

which is used to constrain the RBF_i only taking effect around x range $[x_{\text{slot}} - \frac{w}{2}, x_{\text{slot}} + \frac{w}{2}]$ (recall slot's x location, slot, is an approximate value). In the above definition, $w = 0.3$, $k_z = 0.05$, $\epsilon_z = 12$, and $\epsilon_q = 1.8$. The center of positional RBF, $c_{z,i}$, is evenly chosen on interval $[0, 1]$, and the center of orientation RBF, $\mathbf{q}_{c,i}$, is generated by $(\cos \frac{\alpha}{2}, \sin \frac{\alpha}{2}, 0, 0) * \mathbf{q}_{\text{init}}$ with α evenly chosen from $[-\pi/4, 3\pi/4]$ to make sure $\mathbf{q}_{c,i}$ covers all possible orientations. $\beta_t = 0.9^t$ if $t < 5$ and 0 otherwise. The bounds of the initial hypothesis space Θ_0 is set to $\mathbf{c}_l = [-3, \dots, -3]^T$ and $\mathbf{c}_u = [5, \dots, 5]^T$. γ is set to 1.5. This initialization leads to a conservative MPC policy, preventing the arm from approaching the target at the beginning.

2) *Human Correction Interface*: The motion of the robot arm is rendered in the free-view mode in MuJoCo visualization [see Fig. 16(b)], and users can drag to select his/her comfortable view. The slot and the random reaching target are visible to the users.

Human users apply their corrections to the robot arm via a keyboard. Users are informed with the following key mapping between the key press and the corresponding correction in the game. The safety constraint (61) in Franka arm's safe MPC policy is updated immediately once a user correction is detected. When user press "ENTER," the arm is reset to the initial position and the reaching target will also change.

REFERENCES

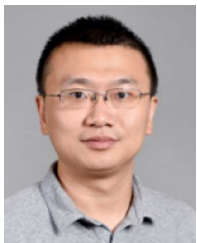
- [1] K. P. Wabersich et al., "Data-driven safety filters: Hamilton-Jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Syst. Mag.*, vol. 43, no. 5, pp. 137–177, Oct. 2023.
- [2] S. Gu et al., "A review of safe reinforcement learning: Methods, theory and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 11216–11235, 2024.
- [3] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations," in *Proc. Algorithmic Found. Robot. XIII: Proc. 13th Workshop Algorithmic Found. Robot.*, 2020, vol. 13, pp. 228–245.
- [4] D. Lindner, X. Chen, S. Tschitschek, K. Hofmann, and A. Krause, "Learning safety constraints from demonstrations with unknown rewards," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2024, pp. 2386–2394.
- [5] D. R. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2020.
- [6] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [7] D. Sadigh, A. D. Dragan, S. Shankar Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Proc. Robot.: Sci. Syst.*, Cambridge, MA, USA, 2017.
- [8] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.
- [9] K. Lee, L. M. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," in *Proc. 38th Int. Conf. Mach. Learn.*, Jul. 2021, vol. 139, pp. 6152–6163.
- [10] D. J. Hejna III and D. Sadigh, "Few-shot preference learning for human-in-the-loop RL," in *Proc. Conf. Robot Learn.*, 2023, pp. 2014–2025.
- [11] V. Myers, E. Bıyık, and D. Sadigh, "Active reward learning from online preferences," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 7511–7518.
- [12] V. Myers, E. Bıyık, N. Anari, and D. Sadigh, "Learning multimodal rewards from rankings," in *Proc. 5th Conf. Robot Learn.*, Nov. 2022, pp. 342–352.
- [13] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning robot objectives from physical human interaction," in *Proc. Conf. Robot Learn.*, 2017, pp. 217–226.
- [14] D. P. Losey and M. K. O'Malley, "Including uncertainty when learning from human corrections," in *Proc. Conf. Robot Learn.*, 2018, pp. 123–132.
- [15] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time," in *Proc. ACM/IEEE Int. Conf. Hum.-Robot Interact.*, 2018, pp. 141–149.
- [16] J. Y. Zhang and A. D. Dragan, "Learning from extrapolated corrections," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 7034–7040.
- [17] M. Li, A. Canberk, D. P. Losey, and D. Sadigh, "Learning human objectives from sequences of physical corrections," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 2877–2883.
- [18] D. P. Losey, A. Bajcsy, M. K. O'Malley, and A. D. Dragan, "Physical interaction as communication: Learning robot objectives online from human corrections," *Int. J. Robot. Res.*, vol. 41, no. 1, pp. 20–44, 2022.
- [19] S. A. Mehta and D. P. Losey, "Unified learning from demonstrations, corrections, and preferences during physical human-robot interaction," *ACM Trans. Hum.-Robot Interact.*, vol. 13, no. 3, 2024, Art. no. 39.
- [20] W. Jin, T. D. Murphey, Z. Lu, and S. Mou, "Learning from human directional corrections," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 625–644, Feb. 2023.
- [21] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 2000, pp. 663–670.
- [22] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, Art. no. 1.
- [23] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 729–736.
- [24] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, Chicago, IL, USA, 2008, vol. 8, pp. 1433–1438.
- [25] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—An inverse optimal control approach," *Auton. Robots*, vol. 28, pp. 369–383, 2010.
- [26] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations," *Int. J. Robot. Res.*, vol. 36, no. 13/14, pp. 1474–1488, 2017.
- [27] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration annual review of control," *Annu. Rev. Control Robot., Auton. Syst.*, vol. 3, pp. 297–330, 2020.
- [28] A. Jain, S. Sharma, T. Joachims, and A. Saxena, "Learning preferences for manipulation tasks from online coactive feedback," *Int. J. Robot. Res.*, vol. 34, no. 10, pp. 1296–1313, 2015.
- [29] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.
- [30] J. Hejna et al., "Contrastive preference learning: Learning from human feedback without RL," in *Proc. Int. Conf. Learn. Representations*, 2024.
- [31] S. Karamcheti, R. Palleti, Y. Cui, P. Liang, and D. Sadigh, "Shared autonomy for robotic manipulation with language corrections," in *Proc. ACL Workshop Learn. Natural Lang. Supervision*, 2022.
- [32] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh, "No, to the right: Online language corrections for robotic manipulation via shared autonomy," in *Proc. ACM/IEEE Int. Conf. Hum.-Robot Interaction*, 2023, pp. 93–101.
- [33] M. Srivastava, N. Goodman, and D. Sadigh, "Generating language corrections for teaching physical control tasks," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 32561–32574.
- [34] L. Zha et al., "Distilling and retrieving generalizable knowledge for robot manipulation via language corrections," in *Proc. Int. Conf. Robot. Autom.*, May 2024, pp. 15172–15179.
- [35] L. X. Shi et al., "Yell at your robot: Improving on-the-fly from language corrections," in *Proc. Robot.: Sci. Syst.*, Delft, The Netherlands, 2024.
- [36] G. Chou, N. Ozay, and D. Berenson, "Learning parametric constraints in high dimensions from demonstrations," in *Proc. Conf. Robot Learn.*, 2020, pp. 1211–1230.
- [37] A. Gaurav, K. Rezaee, G. Liu, and P. Poupert, "Learning soft constraints from constrained expert demonstrations," in *Proc. 11th Int. Conf. Learn. Representations*, 2023.
- [38] K. Kim, G. Swamy, Z. Liu, D. Zhao, S. Choudhury, and S. Z. Wu, "Learning shared safety constraints from multi-task demonstrations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 5808–5826, 2023.
- [39] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3682–3690, Apr. 2020.
- [40] G. Chou, D. Berenson, and N. Ozay, "Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations," in *Proc. Conf. Robot Learn.*, 2021, pp. 1612–1639.
- [41] G. Chou, H. Wang, and D. Berenson, "Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3827–3834, Apr. 2022.
- [42] K. C. Stocking, D. L. McPherson, R. P. Matthew, and C. J. Tomlin, "Maximum likelihood constraint inference on continuous state spaces," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 8598–8604.
- [43] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, "Inverse constrained reinforcement learning," in *Int. Conf. Mach. Learn.*, 2021, pp. 7390–7399.
- [44] D. L. McPherson, K. C. Stocking, and S. S. Sastry, "Maximum likelihood constraint inference from stochastic demonstrations," in *Proc. IEEE Conf. Control Technol. Appl.*, 2021, pp. 1208–1213.
- [45] M. Baert, P. Mazzaglia, S. Leroux, and P. Simoens, "Maximum causal entropy inverse constrained reinforcement learning," *Mach. Learn.*, vol. 114, no. 4, p. 103, 2025.
- [46] D. Park, M. Noseworthy, R. Paul, S. Roy, and N. Roy, "Inferring task goals and constraints using Bayesian nonparametric inverse reinforcement learning," in *Proc. Conf. robot Learn.*, 2020, pp. 1005–1014.
- [47] D. Papadimitriou, U. Anwar, and D. S. Brown, "Bayesian methods for constraint inference in reinforcement learning," *Trans. Mach. Learn. Res.*, 2022.
- [48] W. Jin, S. Mou, and G. J. Pappas, "Safe pontryagin differentiable programming," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 16034–16050, 2021.

- [49] R. Zhang et al., “A dual representation framework for robot learning with human guidance,” in *Proc. Conf. Robot Learn.*, 2023, pp. 738–750.
- [50] E. Bıyık, A. Talati, and D. Sadigh, “APReL: A library for active preference-based reward learning algorithms,” in *Proc. 17th ACM/IEEE Int. Conf. Hum.-Robot Interaction*, 2022, pp. 613–617.
- [51] A. Bobu, A. Bajcsy, J. F. Fisac, S. Deglurkar, and A. D. Dragan, “Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections,” *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 835–854, Jun. 2020.
- [52] L. A. Rademacher, “Approximating the centroid is hard,” in *Proc. 23rd Annu. Symp. Comput. Geometry*, 2007, pp. 302–305.
- [53] A. Laddha and S. S. Vempala, “Convergence of Gibbs sampling: Coordinate hit-and-run mixes fast,” *Discr. Comput. Geometry*, 2023, vol. 70, no. 2, pp. 406–425.
- [54] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, Nov. 2001.
- [55] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [56] Inc. CVX Research, “CVX: Matlab software for disciplined convex programming,” version 2.0, Aug. 2012. [Online]. Available: <http://cvxr.com/cvx>
- [57] G. R. Ghosal, M. Zurek, D. S. Brown, and A. D. Dragan, “The effect of modeling human rationality level on learning rewards from multiple feedback types,” in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, pp. 5983–5992.
- [58] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *Proc. IEEE/RSSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.
- [59] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.
- [60] D. P. Losey and M. K. O’Malley, “Learning the correct robot trajectory in real-time from physical human interactions,” *ACM Trans. Hum.-Robot Interaction*, vol. 9, no. 1, pp. 1–19, 2019.
- [61] S. Poletti, A. Testolin, and S. Tschitschek, “Learning constraints from human stop-feedback in reinforcement learning,” in *Proc. 2023 Int. Conf. Auton. Agents Multiagent Syst.*, 2023, pp. 2328–2330.
- [62] P. D. Kingma, “Adam: A method for stochastic optimization,” in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, May 7–9, 2015.
- [63] S. P. Tarasov, “The method of inscribed ellipsoids,” in *Soviet Mathematics-Doklady*, vol. 37. Providence, RI, USA: Amer. Mathematical Society, 1988, pp. 226–230.
- [64] J. Ortiz et al., “iSDF: Real-time neural signed distance fields for robot perception,” in *Proc. Robot. Sci. Syst. (RSS)*, New York City, NY, USA, 2022.



Zhixian Xie (Graduate Student Member, IEEE) received the B.Eng. degree in automation from Shanghai Jiaotong University, Shanghai, China, in 2022. He is currently working toward the Ph.D. degree in electrical engineering with Arizona State University, Tempe, AZ, USA.

His research focuses mainly on controls, robotic learning from human, and robot dexterous manipulation.



Wenlong Zhang (Member, IEEE) received the Doctoral degree in mechanical engineering from the University of California, Berkeley, Berkeley, CA, USA in 2015.

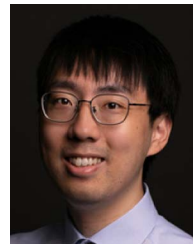
He is currently an Associate Professor with the School of Manufacturing Systems and Networks, Arizona State University, Tempe, AZ, USA, where he is currently the Research Director. He is also a core faculty member with the Robotics and Autonomous Systems and Systems Engineering Ph.D. Programs.

He joined ASU faculty in 2015 after the doctoral degree. At ASU, he leads the Robotics and Intelligent Systems Laboratory (RISE Lab), and teaches undergraduate- and graduate-level classes on dynamics, control systems, optimization, and robotics.



Yi (Max) Ren received the B.S. degree in automotive engineering from Tsinghua University, Beijing, China, in 2007 and the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2012.

He is currently an Associate Professor of Aerospace and Mechanical Engineering with the School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ, USA, where he directs the Design Informatics Laboratory. His group studies computational and data-driven methods that will augment or transform the activities of engineering and industrial design. His current research interests include optimization, product/configuration design, human–computer interaction, and machine learning.



Zhaoran Wang received the Ph.D. degree in operations research & financial engineering/ electrical engineering with Princeton University, Princeton, NJ, USA, in 2018.

He is currently an Associate Professor with Northwestern University, Evanston, IL, USA. His research interests include the interface of machine learning, statistics, and optimization.

Mr. Wang was the recipient of the Artificial Intelligence and Statistics Conference notable paper award, American Statistical Association best student paper in statistical learning and data mining, Institute for Operations Research and the Management Sciences best student paper finalist in data mining, Microsoft Ph.D. Fellowship, Simons Berkeley/J.P. Morgan AI Research Fellowship, Amazon Machine Learning Research Award, and NSF CAREER Award.

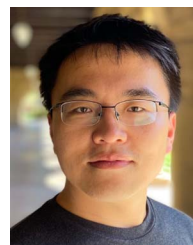


George J. Pappas (Fellow, IEEE) received his Ph.D. on Electrical Engineering and Computer Sciences from the University of California, Berkeley in 1999.

He is currently the UPS Foundation Professor with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA. He is also with the Departments of Computer and Information Sciences, and Mechanical Engineering and Applied Mechanics. He was the Deputy Dean for Research with the School of Engineering and Applied Science. His research interest focus on control

systems, robotics and autonomous systems, formal methods, machine learning for safe, and secure cyber-physical systems.

Mr. Pappas was the recipient of numerous awards including the NSF PECASE, the Antonio Ruberti Young Researcher Prize, the George S. Axelby Award, the O. Hugo Schuck Best Paper Award, and the George H. Heilmeier Faculty Excellence Award. Pappas has mentored more than fifty students and postdocs, now faculty in leading universities worldwide. He is a Fellow of IFAC and was elected to the National Academy of Engineering in 2024.



Wanxin Jin received the Doctorate degree in control and autonomy from Purdue University, West Lafayette, IN, USA, in 2021.

He is currently an Assistant Professor with the School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ, USA. He was a Postdoctoral Fellow with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA. His research includes robotics, control and machine learning, with a focus on the autonomy of robots interacting with humans and objects.