Learning Parametric Constraints in High Dimensions from Demonstrations

Glen Chou, Necmiye Ozay, and Dmitry Berenson
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor
{gchou, necmiye, dmitryb}@umich.edu

Abstract: We present a scalable algorithm for learning parametric constraints in high dimensions from safe expert demonstrations. To reduce the ill-posedness of the constraint recovery problem, our method uses hit-and-run sampling to generate lower cost, and thus unsafe, trajectories. Both safe and unsafe trajectories are used to obtain a representation of the unsafe set that is compatible with the data by solving an integer program in that representation's parameter space. Our method can either leverage a known parameterization or incrementally grow a parameterization while remaining consistent with the data, and we provide theoretical guarantees on the conservativeness of the recovered unsafe set. We evaluate our method on high-dimensional constraints for high-dimensional systems by learning constraints for 7-DOF arm, quadrotor, and planar pushing examples, and show that our method outperforms baseline approaches.

Keywords: learning from demonstration, safe learning, constraint inference

1 Introduction

Learning from demonstration is a powerful paradigm for enabling robots to perform complex tasks. Inverse optimal control and inverse reinforcement learning (IOC/IRL) ([1, 2, 3, 4]) methods have been used to learn a cost function to replicate the behavior of an expert demonstrator. However, planning problems generally also require knowledge of constraints, which define the states or trajectories that are safe. For example, to get a robot arm to efficiently transport a cup of coffee without spilling it, one can optimize a cost function describing the length of the path, subject to constraints on the pose of the end effector. Constraints can represent safety requirements more strictly than cost functions, especially in safety-critical situations: enforcing a hard constraint can enable the robot to guarantee safe behavior, as opposed to using a "softened" cost penalty term. Furthermore, learning a global constraint shared across many tasks can help the robot generalize. Consider the arm, which must avoid spilling the coffee regardless of where the cup started off or needs to go.

While constraints are important, it can be impractical to exhaustively program all the possible constraints a robot should obey across all tasks. Thus, we consider the problem of extracting the latent constraints within expert demonstrations that are shared across tasks. We adopt the insight of [5] that each safe, optimal demonstration induces a set of lower-cost trajectories that must be unsafe due to violation of an unknown constraint. As in [5], we sample these unsafe trajectories, ensuring that they are also consistent with the system dynamics, control constraints, and start/goal constraints. The unsafe trajectories are used together with the safe demonstrations in an "inverse" integer program that recovers an unsafe set consistent with the safe and unsafe trajectories. We make the following additional contributions in this paper. First, by using a (potentially known) parameterization of the constraints, our method enables the inference of safe and unsafe sets in high-dimensional constraint spaces. Second, we relax the known parametrization assumption and propose a means to incrementally grow a parameterization with the data. Third, we introduce a method for extracting volumes of states which are guaranteed safe or guaranteed unsafe according to the data and parameterization. Fourth, we provide theoretical analysis showing that our method is guaranteed to output conservative estimates of the unsafe and safe sets under mild assumptions. Finally, we evaluate our method on high-dimensional constraints for high-dimensional systems by learning constraints for 7-DOF arm, quadrotor, and planar pushing examples, showing that our method outperforms baseline approaches.

2 Related Work

Inverse optimal control [6, 7] (IOC) and inverse reinforcement learning (IRL) [4] aim to recover an objective function that replicates provided expert demonstrations when optimized. Our method is

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

complementary to these approaches; if the demonstrator solves a constrained optimization problem, we are finding its constraints, given the cost; IOC/IRL finds the cost, given the constraints [8]. Risk-sensitive IRL [9] is complementary to our work, which learns hard constraints. Similarly, [10] learns a state-space constraint shared across tasks as a penalty term in the reward function of an MDP. However, when representing a constraint as a penalty, it is unclear if a demonstrated action was made to avoid a penalty or to improve the trajectory cost in terms of the true cost function (or both). Thus, learning a penalty generalizing across cost functions becomes difficult. To avoid this, we assume a known cost function to explicitly reason about the constraint. Also relevant is safe reinforcement learning, which aims to perform exploration while minimizing visitation of unsafe states. Several methods [11, 12, 13] use Gaussian process models to incrementally explore safe regions in the state space. We take a complementary approach to safe learning by using demonstrations in place of exploration to guide the learning of safe behaviors. Methods exist for learning geometric state space constraints [14, 15], task space equality constraints [16, 17], and convex constraints [18], which our algorithm generalizes by being able to learn arbitrary nonconvex parametric inequality constraints defined in some constraint space (not limited to the state space). Other methods aim to learn local trajectory-based constraints [19, 20, 21, 22, 23, 24] by reasoning over the constraints within a single trajectory or task. In contrast, our method aims to learn a global constraint shared across tasks.

The method closest to our work is [5], which learns a global shared constraint on a gridded constraint space; hence, the resulting constraint recovery method scales exponentially with the constraint space dimension and cannot exploit any side information on the structure of the constraint. This often leads to very conservative estimates of the unsafe set, and only grid cells visited by demonstrations can be learned guaranteed safe. We overcome these shortcomings with a novel algorithm that exploits constraint parameterizations for scalability and integration of prior knowledge, and also enables learning volumes of guaranteed safe/unsafe states in the original non-discretized constraint space, yielding less conservative estimates of the safe/unsafe sets under weaker assumptions than [5].

3 Problem Setup

Consider a system with discrete-time dynamics $x_{t+1} = f(x_t, u_t, t)$ or continuous-time dynamics $\dot{x} = f(x, u, t)$, where $x \in \mathcal{X}$ and $u \in \mathcal{U}$. The system performs tasks Π represented as constrained optimization problems over state/control trajectories ξ_x/ξ_u in state/control trajectory space $\mathcal{T}^x$/$\mathcal{T}^u$:

Problem 1 (Forward problem / "task" Π)**.**

$$\begin{aligned} \min_{\xi_x, \xi_u} \quad & c_\Pi(\xi_x, \xi_u) \\ \text{s.t.} \quad & \phi(\xi_x, \xi_u) \in \mathcal{S}(\theta) \subseteq \mathcal{C} \\ & \bar{\phi}(\xi_x, \xi_u) \in \bar{\mathcal{S}} \subseteq \bar{\mathcal{C}} \\ & \phi_\Pi(\xi_x, \xi_u) \in \mathcal{S}_\Pi \subseteq \mathcal{C}_\Pi \end{aligned} \tag{1}$$

where $c_\Pi(\cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \mathbb{R}$ is a cost function for task Π, and $\phi(\cdot, \cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \mathcal{C}$ is a known mapping from state-control trajectories to a constraint space $\mathcal{C}$, elements of which are referred to as "constraint states". Mappings $\bar{\phi}(\cdot, \cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \bar{\mathcal{C}}$ and $\phi_\Pi(\cdot, \cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \mathcal{C}_\Pi$ are known and map to constraint spaces $\bar{\mathcal{C}}$ and $\mathcal{C}_\Pi$, containing a known shared safe set $\bar{\mathcal{S}}$ and a known task-dependent safe set $\mathcal{S}_\Pi$, respectively. In this paper, we take $\mathcal{T}_{\mathcal{S}_\Pi}$ to be the set of trajectories satisfying start/goal state constraints and $\mathcal{T}_{\bar{\mathcal{S}}}$ to be the set of dynamically-feasible trajectories obeying control constraints, though the dynamics may not be known in closed form. $\mathcal{S}(\theta) = \{k \in \mathcal{C} \mid g(k, \theta) > 0\}$ is an unknown safe set defined by an unknown parameter $\theta \in \Theta$ and a possibly unknown parameterization $g(\cdot, \cdot)$. A demonstration, $\xi_{xu} \doteq (\xi_x, \xi_u) \in \mathcal{T}^{xu}$, is a state-control trajectory which approximately solves Problem 1, i.e. it satisfies all constraints and its cost is at most a factor of δ above the cost of a globally optimal solution ξ^*_{xu}, i.e. $c(\xi_x, \xi_u) \leq (1+\delta)c(\xi^*_x, \xi^*_u)$. For convenience, we summarize our frequently used notation in Appendix F. In this paper, our goal is to recover the safe set $\mathcal{S}(\theta)$ and its complement, the unsafe set $\mathcal{A}(\theta) \doteq \mathcal{S}(\theta)^c$, given N_s demonstrations $\{\xi^*_{s_j}\}_{j=1}^{N_s}$, $N_{\neg s}$ inferred unsafe trajectories $\{\xi_{\neg s_k}\}_{k=1}^{N_{\neg s}}$, the cost function $c_\Pi(\cdot)$, task-dependent constraints $\mathcal{S}_\Pi$, and a simulator generating dynamically-feasible trajectories satisfying control constraints.

4 Method

In this section, we describe our method (a full algorithm block is presented in Appendix A). In Section 4.1, we describe how to sample unsafe trajectories. In Sections 4.2 and 4.3, we present mixed integer programs which recover a consistent constraint for a fixed parameterization and extract volumes of guaranteed safe/unsafe states. In Section 4.4, we present how our method can be extended to the case of unknown parameterizations.

4.1 Sampling lower-cost trajectories

In this section, we describe the general sampling framework presented in [5] while also relaxing the assumption of known closed-form dynamics made in [5]. We define the set of unsafe state-control trajectories induced by an optimal, safe demonstration ξ^*_{xu}, $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$, as the set of state-control trajectories of lower cost that obey the known constraints, $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}} \doteq \{\xi_{xu} \in \mathcal{T}_{\bar{\mathcal{S}}} \cap \mathcal{T}_{\mathcal{S}_\Pi} \mid c(\xi_x, \xi_u) < c(\xi^*_x, \xi^*_u)\}$. We sample from $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ to obtain lower-cost trajectories obeying the known constraints using hit-and-run sampling [25], a method guaranteeing convergence to a uniform distribution of samples over $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ in the limit; an illustration is shown in Fig. 1. Hit-and-run starts from an initial point within the set, chooses a direction uniformly at random, moves a random amount in that direction such that the new point remains within the set, and repeats [5]. We sample from $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ indirectly by sampling control sequences and rolling them out through the dynamics to generate dynamically-feasible trajectories. We emphasize that $f(x, u, t)$ does not need to be known in closed form. Given a control sequence sampled by hit-and-run, a simulator can instead be used to output the resulting dynamically-feasible trajectory, which can then be checked for membership in $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ exactly as if the dynamics were known in closed form. Also, δ-suboptimality of the demonstration ξ^{dem}_{xu} can be handled in this framework by sampling instead from $\{\xi_{xu} \in \mathcal{T}_{\bar{\mathcal{S}}} \cap \mathcal{T}_{\mathcal{S}_\Pi} \mid c(\xi_x, \xi_u) < c(\xi^{\text{dem}}_x, \xi^{\text{dem}}_u)/(1+\delta)\}$. Optimal substructure in the cost function can be exploited to sample unsafe sub-trajectories over shorter time windows on the demonstrations; shorter unsafe trajectories provide less ambiguous information regarding $\mathcal{A}$ and can better reduce the ill-posedness of the constraint recovery problem [5].

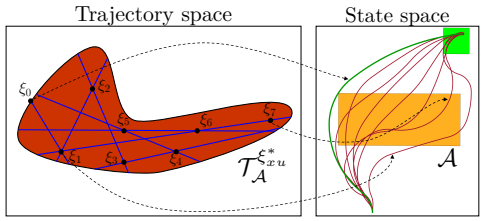

Figure 1: Hit-and-run. **Left**: Blue lines: sampled random directions; black dots: samples. **Right**: Each point in $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ corresponds to an unsafe trajectory in the constraint space $\mathcal{C}$ (here, $\mathcal{C} = \mathcal{X}$).

4.2 Recovering the constraint

Recall that the unsafe set can be described by some parameterization $\mathcal{A}(\theta) \doteq \{k \in \mathcal{C} \mid g(k, \theta) \leq 0\}$, where we assume for now that $g(\cdot, \cdot)$ is known, and θ are parameters to be learned. Intuitively, $g(k, \theta)$ tells us if constraint state k (which is any element of constraint space $\mathcal{C}$) is safe according to parameter θ. Then, a feasibility problem can be written to find a θ consistent with the data:

Problem 2 (Parametric constraint recovery problem)**.**

$$\begin{aligned} \text{find} \quad & \theta \\ \text{s.t.} \quad & g(k_i, \theta) > 0, \quad \forall k_i \in \phi(\xi^*_{s_j}), \quad \forall j = 1, \ldots, N_s && \text{(2a)} \\ & \exists k_i \in \phi(\xi_{\neg s_k}), \quad g(k_i, \theta) \leq 0, \quad \forall k = 1, \ldots, N_{\neg s} && \text{(2b)} \end{aligned}$$

Constraint (2a) enforces that each safe constraint state lies outside $\mathcal{A}(\theta)$ and constraint (2b) enforces that at least one constraint state on each unsafe trajectory lies inside $\mathcal{A}(\theta)$. Denote $\mathcal{F}$ as the feasible set of Problem 2. Further denote $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ as the set of constraint states which are learned guaranteed unsafe and safe, respectively; that is, a constraint state $k \in \mathcal{G}_{\neg s}$ or $k \in \mathcal{G}_s$ if k is classified unsafe or safe for all $\theta \in \mathcal{F}$:

$$\mathcal{G}_{\neg s} \doteq \bigcap_{\theta \in \mathcal{F}} \{k \mid g(k, \theta) \leq 0\} \qquad (3) \qquad\qquad \mathcal{G}_s \doteq \bigcap_{\theta \in \mathcal{F}} \{k \mid g(k, \theta) > 0\} \qquad (4)$$

In Problem 2, it is possible to learn that a constraint state is guaranteed safe/unsafe even if it does not lie directly on a demonstration/unsafe trajectory. This is due to the parameterization: for the given set of safe and unsafe trajectories, there may be no feasible $\theta \in \mathcal{F}$ where k is classified unsafe/safe. It is precisely this extrapolation which will enable us to learn constraints in high-dimensional spaces. We now identify classes of parameterizations for which Problem 2 can be efficiently solved:

Linear case: $g(k, \theta)$ is defined by a Boolean conjunction of linear inequalities, i.e. $\mathcal{A}(\theta)$ can be defined as the union and intersection of half-spaces. For this case, mixed-integer programming can be employed. If $g(k, \theta) \leq 0$ is a single polytope, i.e. $g(k, \theta) \leq 0 \Leftrightarrow H(\theta)k \leq h(\theta)$, where $H(\theta)$ and $h(\theta)$ are affine in θ, we can solve

Problem 3 (Polytopic constraint recovery problem)**.**

$$\begin{aligned} \text{find} \quad & \theta, \{b^i_s\}_{i=1}^{N_s}, \{b^i_{\neg s}\}_{i=1}^{N_{\neg s}} \\ \text{s.t.} \quad & H(\theta)k_i > h(\theta) - M(1 - b^i_s), \quad b^i_{s_j} \in \{0,1\}^{N_h}, \\ & \sum_{i=1}^{N_h} b^i_{s_j} \geq 1, \forall k_i \in \phi(\xi^*_{s_j}), i = 1, ..., T_j, j = 1, ..., N_s && \text{(5a)} \\ & H(\theta)k_i \leq h(\theta) + M(1 - b^i_{\neg s_k})\mathbf{1}_{N_h}, \quad b^i_{\neg s_k} \in \{0,1\}, \\ & \sum_{i=1}^{T_k} b^i_{\neg s_k} \geq 1, \quad \forall k_i \in \phi(\xi_{\neg s_k}), \quad \forall k = 1, ..., N_{\neg s} && \text{(5b)} \end{aligned}$$

Problem 3, a mixed integer feasibility problem, to find a feasible θ. In Problem 3, M is a large positive number and $\mathbf{1}_{N_h}$ is a vector of ones of length N_h, where N_h is the number of rows in $H(\theta)$. Constraints (5a) and (5b) use the big-M formulation [26] to enforce that each safe constraint state lies outside $\mathcal{A}(\theta)$ and that at least one constraint state on each unsafe trajectory lies inside $\mathcal{A}(\theta)$. Similar problems can be solved when the safe/unsafe set can be described by unions of polytopes. As an alternative to integer programming, satisfiability modulo theories (SMT) solvers [27] can also be used to solve Problem 2 if $g(k, \theta)$ is defined by a Boolean conjunction of linear inequalities.

Convex case: $g(k, \theta)$ is defined by a Boolean conjunction of convex inequalities, i.e. $\mathcal{A}(\theta)$ can be described as the union and intersection of convex sets. For this case, satisfiability modulo convex optimization (SMC) [28] can be employed to find a feasible θ.

We close this subsection with some remarks on implementation and extensions to Problems 2 and 3.

- For suboptimal demonstrations / imperfect lower-cost trajectory sampling, Problem 3 can become infeasible. To address this, slack variables can be introduced: replace constraint $\sum_{i=1}^{T_k} b^i_{\neg s_k} \geq 1$ with $\sum_{i=1}^{T_k} b^i_{\neg s_k} \geq v_k, v_k \in \{0, 1\}$ and change the feasibility problem to minimization of $\sum_{k=1}^{N_{\neg s}} (1 - v_k)$; this finds a θ that is consistent with as many unsafe trajectories as possible.
- In addition to recovering sets of guaranteed learned unsafe and safe constraint states, a probability distribution over possibly unsafe constraint states can be estimated by sampling unsafe sets $\mathcal{A}(\theta)$ from the feasible set of Problem 2 using hit-and-run sampling, starting from a feasible θ.

4.3 Extracting guaranteed safe and unsafe states

One can check if a constraint state $k \in \mathcal{G}_s$ or $k \in \mathcal{G}_{\neg s}$ by adding a constraint $g(k, \theta) \leq 0$ or $g(k, \theta) > 0$ to Problem 2 and checking feasibility of the resulting program; if the program is infeasible, $k \in \mathcal{G}_s$ or $k \in \mathcal{G}_{\neg s}$. In other words, solving this modified integer program can be seen as querying an oracle about the safety of a constraint state k. The oracle can then return that k is guaranteed safe (program infeasible after forcing k to be unsafe), guaranteed unsafe (program infeasible after forcing k to be safe), or unsure (program remains feasible despite forcing k to be safe or unsafe).

Unlike the gridded formulation in [5], Problem 2 works in the continuous constraint space. Thus, it is not possible to exhaustively check if each $k \in \mathcal{G}_{\neg s}$ or $k \in \mathcal{G}_s$. To address this, the neighborhood of some constraint state k_{query} can be checked for membership in $\mathcal{G}_{\neg s}$ by solving the following problem:

Problem 4 (Volume extraction)**.**

$$\begin{aligned} \min_{\theta, \varepsilon} \quad & \varepsilon \\ \text{s.t.} \quad & g(k_i, \theta) > 0, \quad \forall k_i \in \phi(\xi^*_{s_j}), \quad \forall j = 1, \ldots, N_s \\ & \exists k_i \in \phi(\xi_{\neg s_k}), \quad g(k_i, \theta) \leq 0, \quad \forall k = 1, \ldots, N_{\neg s} \\ & \exists k_{near} \in \{k_{near} \mid \|k_{near} - k_{query}\|_\infty \leq \varepsilon\}, \quad g(k_{near}, \theta) > 0 \end{aligned}$$

In words, Problem 4 finds the smallest ε-hypercube centered at k_{query} containing a $k \notin \mathcal{G}_{\neg s}$; thus, any hypercube of size $\hat{\varepsilon} < \varepsilon$ is contained within $\mathcal{G}_{\neg s}$: $\{k \mid \|k - k_{\text{query}}\|_\infty \leq \hat{\epsilon}\} \subseteq \mathcal{G}_{\neg s}$. We can write a similar problem to check the neighborhood of k_{query} for membership in $\mathcal{G}_s$. For some common parameterizations (axis-aligned hyper-rectangles, convex sets), there are even more efficient methods for recovering subsets of $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$, which are described in Appendix B. Volumes of safe/unsafe space can thus be produced by repeatedly solving Problem 4 for different k_{query}, and these volumes can be passed to a planner to generate new trajectories that are guaranteed safe.

4.4 Unknown parameterizations

For many realistic applications, we do not have access to a known parameterization which can represent the unsafe set. Despite this, complex unsafe/safe sets can often be approximated as the union of many simple unsafe/safe sets. Along this line of thought, we present a method for incrementally growing a parameterization based on the complexity of the demonstrations and unsafe trajectories.

Suppose that the true parameterization $g(k, \theta)$ of the unsafe set $\mathcal{A}(\theta) = \{k \mid g(k, \theta) \leq 0\}$ is unknown but can be exactly or approximately expressed as the union of N^* simple sets $\mathcal{A}(\theta) \approxeq \bigcup_{i=1}^{N^*} \{k \mid g_s(k, \theta_i) \leq 0\} \doteq \bigcup_{i=1}^{N^*} \mathcal{A}(\theta_i)$, where each simple set $\mathcal{A}(\theta_i)$ has a known parameterization $g_s(\cdot, \cdot)$ and N^*, the minimum number of simple sets needed to reconstruct $\mathcal{A}$, is unknown.

A lower bound on N^*, $\underline{N}$, can be estimated by incrementally adding simple sets until Problem 2 becomes feasible. However, for $\underline{N} < N^*$, the extracted $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ are not guaranteed to be conservative estimates of $\mathcal{S}$ and $\mathcal{A}$ (Theorem 3), and $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ are only guaranteed to be conservative if

$\hat{N} \geq N^*$, where $\hat{N}$ is the chosen number of simple sets (see Theorem 2). Unfortunately, inferring a guaranteed overestimation of N^* only from data is not possible, as there can always be subsets of the constraint which are not activated by the given demonstrations. Two facts mitigate this:

- If an upper bound on the number of simple sets needed to describe $\mathcal{A}(\theta)$, $\bar{N}_{\text{loose}} \geq N^*$, is known (where this bound can be trivially loose), $\mathcal{G}_s \subseteq \mathcal{S}$ and $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ by using $\bar{N}_{\text{loose}}$ simple sets in solving Problem 2. Hence, by using $\bar{N}_{\text{loose}}$, $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ can be made guaranteed conservative (see Theorem 2), at the cost of the resulting $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ being potentially small.
- As the demonstrations begin to cover the space, $\underline{N} \to N^*$. Hence, by using $\underline{N}$ simple sets, $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ are asymptotically conservative.

In our experiments, we choose our simple sets as axis-aligned hyper-rectangles in $\mathcal{C}$, which is motivated by: 1) any open set in $\mathcal{C}$ can be approximated as a countable/finite union of open axis-aligned hyper-rectangles [29]; 2) unions of hyper-rectangles are easily representable in Problem 3.

5 Theoretical Analysis

In this section, we present theoretical analysis on our parametric constraint learning algorithm. In particular, we analyze the conditions under which our algorithm is guaranteed to learn a conservative estimate of the safe and unsafe sets. For space, the proofs and additional results on conservativeness (Section C.2) and the learnability of a constraint (Section C.1) are presented in the appendix. We develop the theory for $\mathcal{C} = \mathcal{X}$ for legibility, but the results can be easily extended to general $\mathcal{C}$.

Theorem 1 (Conservativeness: Known parameterization)**.** *Suppose the parameterization $g(x, \theta)$ is known exactly. Then, for a discrete-time system, extracting $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ (as defined in (3) and (4), respectively) from the feasible set of Problem 2 returns $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$. Further, if the known parameterization is $H(\theta)x_i \leq h(\theta)$ and M in Problem 3 is chosen to be greater than*

$$\max\Big(\max_{x_i \in \xi_s} \max_{\theta} \max_{j} (H(\theta)x_i - h(\theta))_j, \max_{x_i \in \xi_{\neg s}} \max_{\theta} \max_{j} (H(\theta)x_i - h(\theta))_j\Big),$$

then extracting $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ from the feasible set of Problem 3 recovers $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$.

We also present conservativeness results for continuous-time dynamics in Corollary C.2.

Now, let's consider the case where the true parameterization is not known and we use the incremental method described in Section 4.4, where $g_s(x, \theta)$ is the simple parameterization. We consider the over-parameterized case (Theorem 2) and the under-parameterized case (Theorem 3). We analyze the case where the true, under-, and over-parameterization are defined respectively as:

$$g(x,\theta) \leq 0 \Leftrightarrow \bigvee_{i=1}^{N^*} \big(g_s(x,\theta_i) \leq 0\big) \quad (6) \qquad g(x,\theta) \leq 0 \Leftrightarrow \bigvee_{i=1}^{\underline{N}} \big(g_s(x,\theta_i) \leq 0\big), \quad \underline{N} < N^* \quad (7)$$

$$g(x,\theta) \leq 0 \Leftrightarrow \bigvee_{i=1}^{\bar{N}} \big(g_s(x,\theta_i) \leq 0\big), \quad \bar{N} > N^*. \quad (8)$$

Theorem 2 (Conservativeness: Over-parameterization)**.** *Suppose the true parameterization and over-parameterization are defined as in (6) and (8). Then, $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$.*

Theorem 3 (Conservativeness: Under-parameterization)**.** *Suppose the true parameterization and under-parameterization are defined as in (6) and (7). Furthermore, assume that we incrementally grow the parameterization as described in Section 4.4. Then, the following are true:*

1. *$\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ are not guaranteed to be contained in $\mathcal{A}$ (unsafe set) and $\mathcal{S}$ (safe set), respectively.*
2. *Each recovered simple unsafe set $\mathcal{A}(\theta_i)$, $i = 1, \ldots, \underline{N}$, for any $\theta_1, \ldots, \theta_{\underline{N}} \in \mathcal{F}$, touches the true unsafe set (there are no spurious simple unsafe sets): for $i = 1, \ldots, \underline{N}$, for $\theta_1, \ldots, \theta_{\underline{N}} \in \mathcal{F}$, $\mathcal{A}(\theta_i) \cap \mathcal{A} \neq \emptyset$ ($\underline{N}$ is as defined in Section 4.4).*

6 Results

We evaluate our method, showing that our method can be applied to constraints with unknown parameterizations (Section 6.1), high-dimensional constraints defined for high-dimensional systems (Section 6.2), and settings where the dynamics are not known in closed form (Section 6.3). We also compare our performance with a neural network (NN) baseline[1]. We further compare with the grid-based method [5] on the 2D examples. For space, experimental details are provided in Appendix E.

[1] In all experiments, 1) the NN is trained with the safe/unsafe trajectories and predicts at test time if a queried constraint state is safe/unsafe; 2) error bars are generated by initializing the NN with 10 different random seeds and evaluating accuracy after training. The architectures/training details are presented in Appendix E.

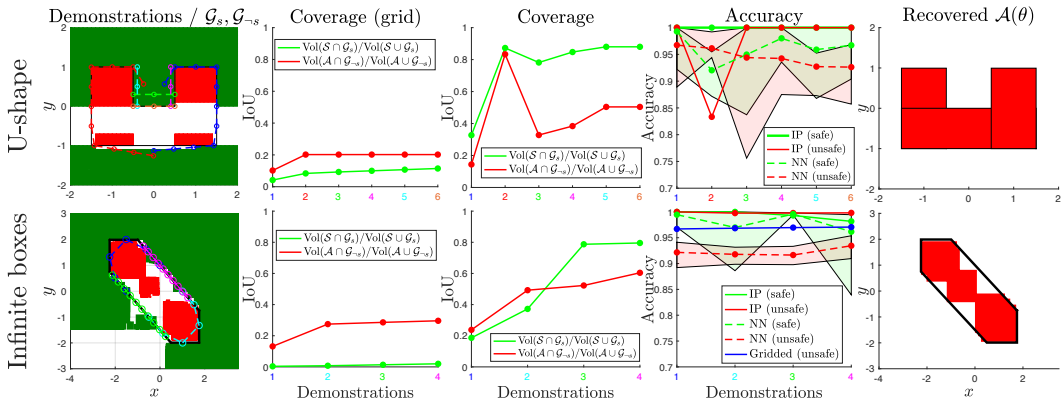

Figure 2: Unknown parameterization. **Col. 1**: Red: $\mathcal{G}_{\neg s}$; Green: $\mathcal{G}_s$. Demonstrations are overlaid. **Col. 2**: Coverage of $\mathcal{A}$ and $\mathcal{S}$ with [5]. In this (and all later examples), the demonstrations are color-coded with x-axis. **Col. 3**: Coverage of $\mathcal{A}$ and $\mathcal{S}$ with our method. **Col. 4**: Classification accuracy (dotted: average NN accuracy, shaded: range of NN accuracies over 10 random seeds). **Col. 5**: Recovered constraint with multi-polytope variant of Problem 3.

6.1 Unknown parameterization

U-shape: We first present a kinematic 2D example where a U-shape $\mathcal{A}$ is to be learned, but the number of simple unsafe sets needed to represent $\mathcal{A}$ (three) is unknown. In Row 1, Column 1 of Fig. 2, we outline $\mathcal{A}$ in black and overlay $\mathcal{G}_{\neg s}$, $\mathcal{G}_s$, and the six provided demonstrations, synthetically generated via trajectory optimization. We note that due to the chosen control constraints and U-shape, there are parts of $\mathcal{A}$ (a subset of the white region in Fig. 2, Row 1, Column 1) which cannot be implied unsafe by sampled unsafe trajectories and the parameterization (see Theorem C.1). As a result, $\mathcal{G}_{\neg s}$ may not fully cover $\mathcal{A}$, even with more demonstrations (Fig. 2, Row 1, Column 3). Note that the decrease in coverage[2] at the third demonstration is due to a increase from a two-box parameterization to a three-box parameterization. Likewise, the accuracy[3] decreases at the second demonstration due to over-approximation of $\mathcal{A}$ with two boxes (Fig. 2, Row 1, Column 4), but this over-approximation vanishes when switching to the three-box parameterization (which is exact; hence $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ are guaranteed conservative, c.f. Theorem 1). The grid-based method in [5] always has perfect accuracy, since it does not extrapolate beyond the observed trajectories. However, as a result of that, it also yields low coverage (Fig. 2, Row 1, Column 2). The NN baseline achieves lower accuracy for the unsafe set as it misclassifies some corners of the U. Recovering a feasible θ using a multi-box variant of Problem 3 recovers $\mathcal{A}$ exactly (Fig. 2, Row 1, Column 5). Finally, we note that in this (and future) examples, demonstrations were specifically chosen to be informative about the constraint. We present a version of this example in Appendix D with random demonstrations and show that the constraint is still learned (albeit needing more demonstrations).

Infinite boxes: To show that our method can still learn a constraint that cannot be easily expressed using a chosen parameterization, we limit our parameterization to an unknown number of axis-aligned boxes and attempt to learn a diagonal "I" unsafe set (see Fig. 2, Row 2). This is a particularly difficult example, since an infinite number of axis-aligned boxes will be needed to recover $\mathcal{A}$ exactly. However, for finite data, only a finite number of boxes will be needed; in particular, for 1, 2, 3, and 4 demonstrations (which are synthetically generated assuming kinematic system constraints), 3, 5, 6, and 6 boxes are required to generate a parameterization consistent with the data (see Fig. 2, Row 2, Column 1). Also overlaid in Fig. 2, Row 2, Column 1 are $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$, which are approximated by solving Problem 4 for randomly sampled k_{center}. Compared to the gridded formulation in [5] (see Fig. 2, Row 2, Column 3), $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ cover $\mathcal{S}$ and $\mathcal{A}$ far better due to the parameterization enabling the IP to extrapolate more from the demonstrations. Furthermore, we note that while the gridded case has perfect accuracy for the safe set, it does not for the unsafe set, due to grid alignment [5]. Overall, the multi-box variant of Problem 3 recovers $\mathcal{A}$ well (Fig. 2, Row 2, Column 5), and the remaining gap can be improved with more data. Last, we note that the NN baseline

[2]Coverage is measured as the intersection over union (IoU) of the relevant sets (see legends for exact formula).

[3]In all experiments, computed accuracies are: IP (safe) = $\mathrm{Vol}(\mathcal{G}_s \cap \mathcal{S})/\mathrm{Vol}(\mathcal{G}_s)$, IP (unsafe) = $\mathrm{Vol}(\mathcal{G}_{\neg s} \cap \mathcal{A})/\mathrm{Vol}(\mathcal{G}_{\neg s})$, NN (safe) = $\left(\sum_{i=1}^q \mathbf{I}_{(x_i \in \mathcal{S}) \wedge (\text{NN classified } x_i \text{ as safe})}\right) / \sum_{i=1}^q \mathbf{I}_{x_i \in \mathcal{S}}$, NN (unsafe) = $\left(\sum_{i=1}^q \mathbf{I}_{(x_i \in \mathcal{A}) \wedge (\text{NN classified } x_i \text{ as unsafe})}\right) / \sum_{i=1}^q \mathbf{I}_{x_i \in \mathcal{A}}$, where $x_1, \ldots, x_q$ are query states sampled from $\mathcal{G}_{\neg s} \cup \mathcal{G}_s$ and $\mathbf{I}_{(\cdot)}$ is the indicator function. Note that NN accuracy is computed only on $(\mathcal{G}_s \cup \mathcal{G}_{\neg s}) \subseteq \mathcal{C}$.

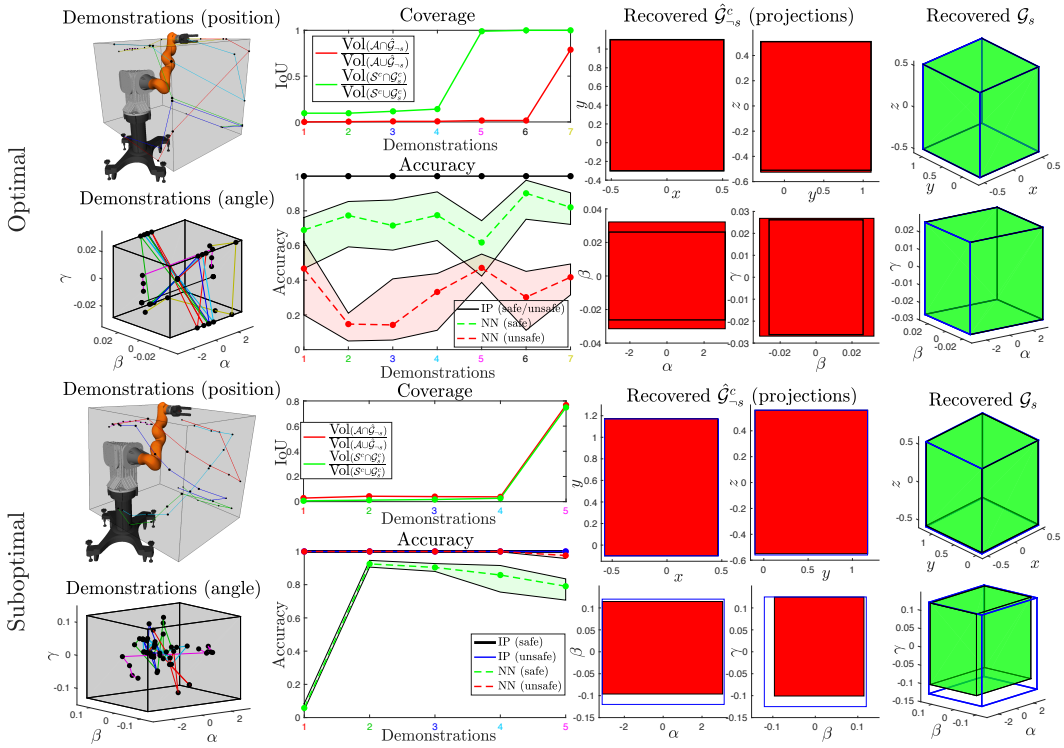

Figure 3: **Rows 1:2**: 7-DOF arm, optimal demonstrations **Col. 1**: Experimental setup. Gray boxes are projections of $\mathcal{A}$. Projections of demonstrations in position/angle space are overlaid. **Col. 2**: *Top*: Comparing safe/unsafe set coverage as a function of demonstrations. *Bottom*: Prediction accuracy. **Cols. 3-4**: projections of $\hat{\mathcal{G}}_{\neg s}$ using all demonstrations. For the optimal case, the red boxes over-approximate the blue boxes, as the complement of $\hat{\mathcal{G}}_{\neg s}$ (not $\hat{\mathcal{G}}_{\neg s}$ itself) is plotted. **Col. 5**: projections of $\mathcal{G}_s$ using all demonstrations. **Rows 3:4**: Same for 7-DOF arm, suboptimal demonstrations.

reaches comparable accuracies here (Fig. 2, Row 2, Column 4), since our method suffers from a few disadvantages for this particular example. First, attempting to represent the "I" with a finite number of boxes introduces a modeling bias that the NN does not have. Second, since the system is kinematic and the constraint is low-dimensional, many unsafe trajectories can be sampled, providing good coverage of the unsafe set. We show later that for higher dimensional constraints/systems with highly constrained dynamics, it becomes difficult to gather enough data for the NN to perform well.

6.2 High-dimensional examples

6D pose constraint for a 7-DOF robot arm: In this example, we learn a 6D hyper-rectangular pose constraint for the end effector of a 7-DOF Kuka iiwa arm. One such setting is when the robot is to bring a cup to a human while ensuring its contents do not spill (angle constraint) and proxemics constraints (i.e. the end effector never gets too close to the human) are satisfied (position constraint). We examine this problem for the cases of optimal and suboptimal demonstrations.

Demonstration setup: The end effector orientation (parametrized in Euler angles) and position are constrained to satisfy $(\alpha, \beta, \gamma) \in [\underline{\alpha}, \bar{\alpha}] \times [\underline{\beta}, \bar{\beta}] \times [\underline{\gamma}, \bar{\gamma}]$ and $(x, y, z) \in [\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}] \times [\underline{z}, \bar{z}]$ (see Fig. 3, Column 1). For the optimal case, we synthetically generate seven demonstrations minimizing joint-space trajectory length. For the suboptimal case, five suboptimal continuous-time demonstrations approximately optimizing joint-space trajectory length are recorded in a virtual reality environment, where a human demonstrator moves the arm from desired start to goal end effector configurations using an HTC Vive (see Fig. E.1). The demonstrations are time-discretized for lower-cost trajectory sampling [5]. In both cases, the constraint is recovered with Problem 3, where $H(\theta) = [I, -I]^\top$ and $h(\theta) = \theta = [\bar{x}, \bar{y}, \bar{z}, \bar{\alpha}, \bar{\beta}, \bar{\gamma}, \underline{x}, \underline{y}, \underline{z}, \underline{\alpha}, \underline{\beta}, \underline{\gamma}]^\top$. For the suboptimal case, slack variables are added to ensure feasibility of Problem 3, and for a suboptimal demonstration of cost $\hat{c}$, we only use trajectories of cost less than $0.9\hat{c}$ as unsafe trajectories.

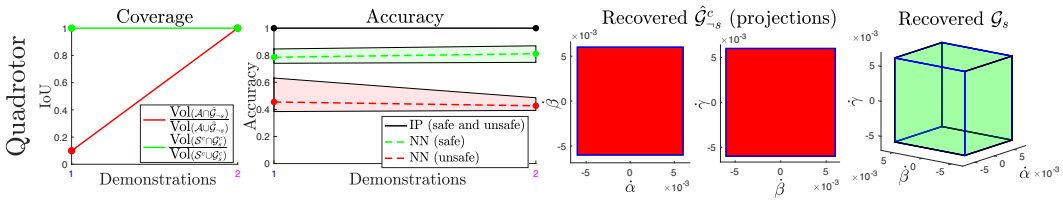

Figure 5: Constraint recovery for a 12D quadrotor. **Col. 1**: Coverage of $\mathcal{A}$ and $\mathcal{S}$. **Col. 2**: Classification error between $\mathcal{G}_s/\mathcal{S}$ and $\mathcal{G}_{\neg s}/\mathcal{A}$. **Cols. 3-4**: $\hat{\mathcal{G}}_{\neg s}$ using all demonstrations. **Col. 5**: $\mathcal{G}_s$ using all demonstrations.

Results: The coverage plots (Fig. 3, Rows 1 and 3, Col. 2) show that as the number of demonstrations increases, $\mathcal{G}_s/\mathcal{G}_{\neg s}$ approach the true safe/unsafe sets $\mathcal{S}/\mathcal{A}$ [4]. For the suboptimal case, the low IoU values for lower numbers of demonstrations is due to overapproximation of the unsafe set in the α component (arising from continuous-time discretization and imperfect knowledge of the suboptimality bound); the fifth demonstration, where α takes values near $-\pi, \pi$ greatly reduces this overapproximation. The accuracy plots (Fig. 3, Rows 2 and 4, Col. 2) present results consistent with the theory: for the optimal case, all constraint states in $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ are truly safe and unsafe (Theorem 1), and the small over-approximation for the suboptimal case is consistent with the continuous-time conservativeness (Theorem C.2). Note that the NN accuracy is lower and can oscillate with demonstrations, since it finds just a single constraint which is approximately consistent with the data, while our method classifies safety by consulting all possible constraints which are exactly consistent with the data, thus performing more consistently. The NN performs better on the suboptimal case than it does on the optimal case, as more unsafe trajectories are sampled due to the suboptimality, improving coverage of the unsafe set. The projections of $\hat{\mathcal{G}}^c_{\neg s}$ (Fig. 3, Cols. 3-4, in red), where $\hat{\mathcal{G}}_{\neg s} \subseteq \mathcal{G}_{\neg s}$ is obtained using the method in Appendix B, are compared to the safe set (blue outline), showing that the two match nearly exactly (though the gap for the suboptimal case is larger), and the gap can be likely reduced with more demonstrations. The projections of $\mathcal{G}_s$ (Fig. 3, Col. 5) match exactly with $\mathcal{A}$ for the optimal case (true safe set is outlined in blue) and match closely for the suboptimal case. Note that $\mathcal{G}_s \subseteq \mathcal{S}$, as is the case for all axis-aligned box parameterizations.

3D constraint for 12D quadrotor model: We learn a 3D box angular velocity constraint for a quadrotor with discrete-time 12D dynamics (see Appendix E for details). In this scenario, the quadrotor must avoid an a priori known unsafe set in position space while also ensuring that angular velocities are below a threshold: $(\dot{\alpha}, \dot{\beta}, \dot{\gamma}) \in [\underline{\dot{\alpha}}, \bar{\dot{\alpha}}] \times [\underline{\dot{\beta}}, \bar{\dot{\beta}}] \times [\underline{\dot{\gamma}}, \bar{\dot{\gamma}}]$. The $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$ safe set is to be inferred from two demonstrations (see Fig. 4). The constraint is recovered with Problem 3, where $H(\theta) = [I, -I]^\top$ and $h(\theta) = \theta = [\bar{\dot{\alpha}}, \bar{\dot{\beta}}, \bar{\dot{\gamma}}, \underline{\dot{\alpha}}, \underline{\dot{\beta}}, \underline{\dot{\gamma}}]^\top$. Fig. 5 shows that with more demonstrations, $\mathcal{G}_s$ approaches the true safe set $\mathcal{S}$ and $\mathcal{G}_{\neg s}$ approaches the true unsafe set $\mathcal{A}$, respectively. Consistent with Theorem 1, our method has perfect accuracy in $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$. Here, the NN struggles more compared to the arm examples since due to the more constrained dynamics, fewer unsafe trajectories can be sampled, and a parameterization needs to be leveraged in order to say more about the unsafe set. The remaining columns of Fig. 5 show that we recover $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ exactly (the true safe set is outlined in blue).

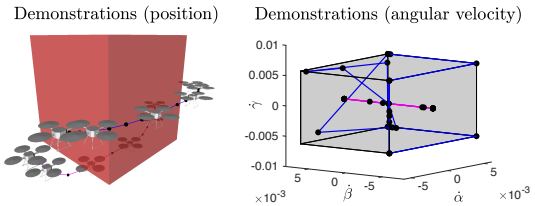

Figure 4: **Left**: Known unsafe set in (x, y, z) (red); (x, y, z) components of demonstrations are overlaid. **Right**: Unknown unsafe set in $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$ (gray); $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$ components of demonstrations are overlaid.

6.3 Planar pushing example

In this section, using the `FetchPush-v1` environment in OpenAI Gym [30], we aim to learn a 2D box unsafe set on the center-of-mass (CoM) of a block pushed by the Fetch arm (see Fig. 6) using two demonstrations. Here, the dynamics of the block CoM are not known in closed form, but rollouts can still be sampled using the simulator. Since the block CoM is highly underactuated, it is not possible to sample short sub-trajectories. Thus, without leveraging a parameterization, the constraint recovery problem is very ill-posed. Furthermore, while our method can explicitly consider

[4] For the unsafe sets, the IoUs are computed between $\mathcal{G}^c_{\neg s}$ and $\mathcal{A}^c$, as in high dimensions, the IoU changes more smoothly for the complements than the IoU between $\mathcal{G}_{\neg s}$ and $\mathcal{A}$, so we plot the the former for visual clarity.

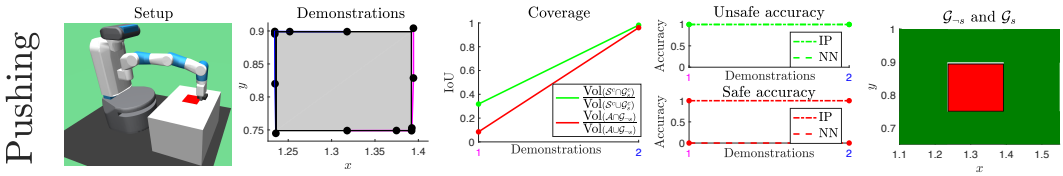

Figure 6: Constraint recovery without closed-form dynamics. **Cols. 1-2**: Setup (unsafe set in red) and demonstrations (unsafe set in gray). **Cols. 3-4**: Coverage of $\mathcal{A}$ and $\mathcal{S}$; classification accuracy. **Col. 5**: $\mathcal{G}_{\neg s}$ / $\mathcal{G}_s$ using all demonstrations.

the unsafeness in longer unsafe trajectories (at least one state is unsafe), the NN struggles with this example as it fails to accurately model that fact. Overall, Fig. 6 presents that $\mathcal{G}_{\neg s}$/$\mathcal{G}_s$ match up well with $\mathcal{A}$/$\mathcal{S}$, and our classification accuracy for safeness/unsafeness is perfect across demonstrations.

7 Discussion and Conclusion

In this paper, we present a method capable of learning parametric constraints in high-dimensional spaces with and without known parameterizations. We also present a method for extracting volumes of guaranteed safe and guaranteed unsafe states, information which can be directly used in a planner to enforce safety constraints. We analyze our algorithm, showing that these recovered guaranteed safe/unsafe states are truly safe/unsafe under mild assumptions. We evaluate the method by learning a variety of constraints defined in high-dimensional spaces for systems with high-dimensional dynamics. One shortcoming of our work is scalability with the amount of data, due to the number of integer variables growing linearly with the number of safe/unsafe trajectories. As a result, learning constraints without extensive sampling of unsafe trajectories is a direction of future work.

Acknowledgments

This work was supported in part by a National Defense Science and Engineering Graduate (NDSEG) Fellowship, Office of Naval Research (ONR) grants N00014-18-1-2501 and N00014-17-1-2050, and National Science Foundation (NSF) grants ECCS-1553873 and IIS-1750489.

References

[1] N. D. Ratliff, J. A. Bagnell, and M. Zinkevich. Maximum margin planning. In *International Conference on Machine Learning*, pages 729–736, 2006. doi:10.1145/1143844.1143936.

[2] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004. doi:10.1145/1015330.1015430.

[3] B. Argall, S. Chernova, M. M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[4] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning '00*, pages 663–670, San Francisco, CA, USA, 2000.

[5] G. Chou, D. Berenson, and N. Ozay. Learning constraints from demonstrations. *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018. URL `http://arxiv.org/abs/1812.07084`.

[6] R. E. Kalman. When is a linear control system optimal? *Journal of Basic Engineering*, 86(1): 51–60, Mar 1964. ISSN 0098-2202. doi:10.1115/1.3653115.

[7] A. Keshavarz, Y. Wang, and S. P. Boyd. Imputing a convex objective function. In *International Symposium on Intelligent Control*, pages 613–619. IEEE, 2011.

[8] P. Englert, N. A. Vien, and M. Toussaint. Inverse kkt: Learning cost functions of manipulation tasks from demonstrations. *International Journal of Robotics Research*, 36(13-14):1474–1488, 2017. doi:10.1177/0278364917745980.

[9] S. Singh, J. Lacotte, A. Majumdar, and M. Pavone. Risk-sensitive inverse reinforcement learning via semi- and non-parametric methods. *International Journal of Robotics Research*, 37 (13-14), 2018. doi:10.1177/0278364918772017.

[10] K. Amin, N. Jiang, and S. P. Singh. Repeated inverse reinforcement learning. In *Neural Information Processing Systems*, pages 1813–1822, 2017.

[11] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. Safe exploration for active learning with gaussian processes. In *European Conference on Machine Learning*, 2015.

[12] M. Turchetta, F. Berkenkamp, and A. Krause. Safe exploration in finite markov decision processes with gaussian processes. In *Neural Information Processing Systems*, pages 4305–4313, 2016.

[13] A. Akametalu, J. Fisac, J. Gillula, S. Kaynama, M. Zeilinger, and C. Tomlin. Reachability-based safe learning with gaussian processes. In *Conference on Decision and Control*, Dec 2014. doi:10.1109/CDC.2014.7039601.

[14] L. Armesto, J. Bosga, V. Ivan, and S. Vijayakumar. Efficient learning of constraints and generic null space policies. In *International Conference on Robotics and Automation* , pages 1520–1526. IEEE, 2017.

[15] C. Pérez-D'Arpino and J. A. Shah. C-LEARN: learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *International Conference on Robotics and Automation*, 2017.

[16] H. Lin, P. Ray, and M. Howard. Learning task constraints in operational space formulation. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 309–315, 2017.

[17] H. Lin, M. Howard, and S. Vijayakumar. Learning null space projections. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 2613–2619, 2015.

[18] M. Menner, P. Worsnop, and M. N. Zeilinger. Predictive modeling by infinite-horizon constrained inverse optimal control with application to a human manipulation task. *CoRR*, abs/1812.11600, 2018. URL http://arxiv.org/abs/1812.11600.

[19] C. Li and D. Berenson. Learning object orientation constraints and guiding constraints for narrow passages from one demonstration. In *International Symposium on Experimental Robotics*. Springer, 2016.

[20] N. Mehr, R. Horowitz, and A. D. Dragan. Inferring and assisting with constraints in shared autonomy. In *(Conference on Decision and Control)*, pages 6689–6696, Dec 2016. doi:10.1109/CDC.2016.7799299.

[21] A. L. Pais, K. Umezawa, Y. Nakamura, and A. Billard. Learning robot skills through motion segmentation and constraints extraction. International Conference on Human Robot Interaction, 2013.

[22] G. Ye and R. Alterovitz. Demonstration-guided motion planning. In *International Symposium on Robotics Research*, 2011. doi:10.1007/978-3-319-29363-9_17.

[23] S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *International Conference on Intelligent Robots and Systems*, 2008. doi:10.1109/IROS.2008.4650593.

[24] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *International Conference on Human Robot Interaction 2007*, pages 255–262, 2007. doi:10.1145/1228716.1228751.

[25] S. Kiatsupaibul, R. L. Smith, and Z. B. Zabinsky. An analysis of a variation of hit-and-run for uniform sampling from general regions. *Transactions on Modeling and Computer Simulation*, 2011.

[26] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

[27] L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, pages 337–340, 2008. doi:10.1007/978-3-540-78800-3_24. URL https://doi.org/10.1007/978-3-540-78800-3_24.

[28] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada. SMC: satisfiability modulo convex programming. *Proceedings of the IEEE*, 106 (9):1655–1679, 2018.

[29] T. Tao. *Analysis II*. Springer Texts and Readings in Mathematics. Springer Singapore, 2016.

[30] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018. URL http://arxiv.org/abs/1802.09464.

[31] G. Allaire, F. Jouve, and G. Michailidis. Thickness control in structural optimization via a level set method. *Struct. and Multidisciplinary Optimization*, 2016. URL https://hal.archives-ouvertes.fr/hal-00985000.

[32] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006. doi:10.1007/s10107-004-0559-y.

[33] F. Sabatino. Quadrotor control: modeling, nonlinear control design, and simulation, 2015.

A Detailed algorithm block

Algorithm 1: Overall method

Output: θ (a feasible unsafe/safe set describing the safe/unsafe trajectories),
$\mathcal{G}_s$ (the set of guaranteed safe constraint states),
$\mathcal{G}_{\neg s}$ (the set of guaranteed unsafe constraint states)

Input : $\xi_s = \{\xi_1^*, \ldots, \xi_{N_s}^*\}$, $c_\Pi(\cdot)$, known constraints, $\{k_{\text{query}}^q\}_{q=1}^Q$

$\xi_{\neg s} \leftarrow \{\}$;

`/* Sample unsafe trajectories` $\xi_{\neg s}$ `*/`

for i = $1{:}N_s$ **do**

  $\xi_{\neg s} \leftarrow \xi_{\neg s} \cup \mathsf{HitAndRun}(\xi_i^*)$;

end

`/* Constraint recovery */`

$\theta \leftarrow$ Problem Y$(\xi_s, \xi_{\neg s})$;

`/* Y = 2 if general parameterization */`

`/* Y = 3 if polytope parameterization */`

$\mathcal{G}_s, \mathcal{G}_{\neg s} \leftarrow \{\}, \{\}$;

`/* Guaranteed safe/unsafe recovery */`

if *general parameterization* **then**

  for $q = 1, \ldots, Q$ **do**

    `/* Extract safe/unsafe volume around query point` k_{query}^q `*/`

    $\mathcal{G}_s(k_{\text{query}}^q), \mathcal{G}_{\neg s}(k_{\text{query}}^q) \leftarrow$ Problem 4(k_{query}^q);

    $\mathcal{G}_s \leftarrow \mathcal{G}_s \cup \mathcal{G}_s(k_{\text{query}}^q)$;

    $\mathcal{G}_{\neg s} \leftarrow \mathcal{G}_{\neg s} \cup \mathcal{G}_{\neg s}(k_{\text{query}}^q)$;

  end

else if *axis-aligned hyper-rectangle parameterization* **then**

  $\mathcal{G}_s, \hat{\mathcal{G}}_{\neg s} \leftarrow$ Procedure in Appendix B.1;

else if *convex parameterization* **then**

  $\mathcal{G}_s, \hat{\mathcal{G}}_{\neg s} \leftarrow$ Procedure in Appendix B.2;

B Extraction of $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$

In this section, we discuss specific ways of extracting sets of guaranteed safe/unsafe states for axis-aligned hyper-rectangles (this method is used for all numerical examples in Section 6.2 and Section 6.3) and for convex parameterizations.

B.1 Axis-aligned hyper-rectangle parameterization

In this parameterization, $\mathcal{C} \subseteq \mathbb{R}^n$, $\theta = [\underline{k}_1, \bar{k}_1, \ldots, \underline{k}_n, \bar{k}_n]$, and $g(k, \theta) \leq 0 \Leftrightarrow H(\theta)k \leq h(\theta)$, where $H(\theta)k = [I_{n\times n}, -I_{n\times n}]^\top$ and $h(\theta) = [\bar{k}_1, \ldots, \bar{k}_n, \underline{k}_1, \ldots, \underline{k}_n]^\top$. Here, $\underline{k}_i$ and $\bar{k}_i$ are the lower and upper bounds of the hyper-rectangle for coordinate i.

As the set of axis-aligned hyper-rectangles is closed under intersection, $\mathcal{G}_{\neg s}$ is also an axis-aligned hyper-rectangle, the axis-aligned bounding box of any two constraint states $k_1, k_2 \in \mathcal{G}_{\neg s}$ is also contained in $\mathcal{G}_{\neg s}$. This also implies that $\mathcal{G}_{\neg s}$ can be fully described by finding the top and bottom corners $[\underline{k}_1, \ldots, \underline{k}_n]^\top$ and $[\bar{k}_1, \ldots, \bar{k}_n]^\top$. Suppose we start with a known $k \in \mathcal{G}_{\neg s}$. Then, finding $[\underline{k}_1, \ldots, \underline{k}_n]^\top$ amounts to performing a binary search for each of the n dimensions, and the same holds for finding $[\bar{k}_1, \ldots, \bar{k}_n]^\top$.

Recovering $\mathcal{G}_s$ is not as straightforward, as the complement of axis-aligned boxes is not closed under intersection. While we can still solve Problem 4 to recover $\mathcal{G}_s$, an inner approximation of $\mathcal{G}_s$ can be more efficiently obtained: starting at a constraint state $k \in \mathcal{G}_{\neg s}$, $2n$ line searches can be performed to find the two points of transition to $\mathcal{G}_{\neg s}$ in each constraint coordinate. Denote as $\hat{\mathcal{G}}_s$ the complement of the axis-aligned bounding box of these $2n$ points; $\hat{\mathcal{G}}_s$ is an inner approximation of $\mathcal{G}_s$, as $\mathcal{G}_s = (\bigcap_{\theta\in\mathcal{F}}\{x \mid g(x,\theta) \leq 0\})^c \supseteq \text{AABB}(\bigcap_{\theta\in\mathcal{F}}\{x \mid g(x,\theta) \leq 0\})^c$, where AABB$(\cdot)$

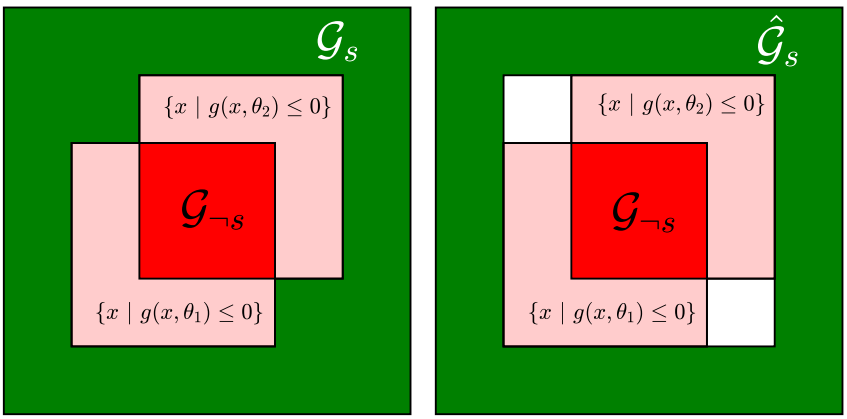

Figure B.1: Comparison of the true $\mathcal{G}_s$ (left, in green) and the extracted inner approximation $\hat{\mathcal{G}}_s$ (right, in green).

denotes the axis-aligned bounding box of a set of points. For example, consider the scenario in Fig. B.1 where there are only two feasible parameters, θ_1 and θ_2. Here, $\mathcal{G}_s$ is $(\mathcal{A}(\theta_1) \cup \mathcal{A}(\theta_2))^c$ and $\hat{\mathcal{G}}_s$ under-approximates the safe set ($\mathcal{G}_s$ is in general not representable as the complement of an axis-aligned box).

B.2 Convex parameterization

In this parameterization, for fixed θ, $\{k \mid g(k,\theta) \leq 0\}$ is convex.

While apart from solving Problem 4 it is hard to recover $\mathcal{G}_{\neg s}$ exactly, an inner approximation of $\mathcal{G}_{\neg s}$ can be extracted more efficiently by taking the convex hull of any $k_1, k_2, \ldots \in \mathcal{G}_{\neg s}$, as the convex hull is the minimal convex set containing $k_1, k_2, \ldots$.

The same approaches apply for recovering $\mathcal{G}_s$ when it is instead the safe set which is an axis-aligned hyper-rectangle or a convex set.

C Theoretical Analysis (Expanded)

In this section, we present theoretical analysis on our parametric constraint learning algorithm. In particular, we analyze the limits of what constraint states can be learned guaranteed unsafe/safe (Section C.1) as well as the conditions under which our algorithm is guaranteed to learn a conservative estimate of the safe and unsafe sets (Section C.2). For ease of reading, we repeat the theorem statements from the main body (the corresponding theorem numbers from the main body are listed in the theorem statement). We develop the theory for $\mathcal{C} = \mathcal{X}$ for legibility, but the results can be easily extended to general $\mathcal{C}$.

C.1 Learnability

In this section, we develop results for learnability of the unsafe set in the parametric case. We begin with the following notation:

Definition C.1 (Signed distance)**.** *Signed distance from point $p \in \mathbb{R}^m$ to set $\mathcal{S} \subseteq \mathbb{R}^m$, $\mathsf{sd}(p, \mathcal{S}) = -\inf_{y \in \partial\mathcal{S}} \|p - y\|$ if $p \in \mathcal{S}$; $\inf_{y \in \partial\mathcal{S}} \|p - y\|$ if $p \in \mathcal{S}^c$.*

Definition C.2 (Δx-shell)**.** *For a discrete time system satisfying $\|x_{t+1} - x_t\| \leq \Delta x$ for all t, denote the Δx shell of the unsafe set as: $\mathcal{A}_{\Delta x} \doteq \{x \in \mathcal{A} \mid -\Delta x \leq \mathsf{sd}(x, \mathcal{A}) \leq 0\}$.*

Definition C.3 (Implied unsafe set)**.** *For some set $\mathcal{B} \subseteq \Theta$, denote $I(\mathcal{B}) \doteq \bigcap_{\theta \in \mathcal{B}} \{x \mid g(x,\theta) \leq 0\}$ as the set of states that are implied unsafe by restricting the parameter set to $\mathcal{B}$. In words, $I(\mathcal{B})$ is the set of states for which all $\theta \in \mathcal{B}$ mark as unsafe.*

Definition C.4 (Feasible set $\mathcal{F}$)**.** *Denote as $\mathcal{F}$ the feasible set of Problem 2 with N_s demonstrations and $N_{\neg s}$ unsafe trajectories sampled using the hit-and-run method presented in Section 4.1:*

$$\mathcal{F} = \{\theta \mid \forall i \in \{1, \dots, N_s\}, \forall x \in \xi_i^*, g(x, \theta) > 0, \\ \forall j \in \{1, \dots, N_{\neg s}\}, \exists x \in \xi_j, g(x, \theta) \leq 0\}.$$

Definition C.5 (Learnability and learnable set $\mathcal{G}_{\neg s}^*$)**.** *A state $x \in \mathcal{A}$ is learnable if there exists **any** set of N_s demonstrations and $N_{\neg s}$ unsafe trajectories sampled using the hit-and-run method presented in Section 4.1, where N_s and $N_{\neg s}$ may be infinite, such that $x \in \mathcal{I}(\mathcal{F})$. Accordingly, we define the learnable set of unsafe states $\mathcal{G}_{\neg s}^*$ as the union of all learnable states. Note that by this definition, a state $x_s \in \mathcal{S}$ is always learnable, since there always exists some safe demonstration passing through x_s.*

Lemma C.1. *Suppose $\mathcal{B} \subseteq \hat{\mathcal{B}}$, for some other set $\hat{\mathcal{B}}$. Then, $I(\hat{\mathcal{B}}) \subseteq I(\mathcal{B})$.*

Proof. By definition,

$$\begin{aligned} I(\hat{\mathcal{B}}) &= \bigcap_{\theta \in \hat{\mathcal{B}}} \{x \mid g(x, \theta) \leq 0\} \\ &= \bigcap_{\theta \in \left(\mathcal{B} \cup (\hat{\mathcal{B}} \setminus \mathcal{B})\right)} \{x \mid g(x, \theta) \leq 0\} \\ &\subseteq \bigcap_{\theta \in \mathcal{B}} \{x \mid g(x, \theta) \leq 0\} \\ &= I(\mathcal{B}). \end{aligned}$$

□

Lemma C.2. *Each unsafe trajectory ξ_j with start and goal states in the safe set contains at least one state in the Δx-shell $\mathcal{A}_{\Delta x}$: $\forall j \in \{1, \dots, N_{\neg s}\}, \exists x \in \xi_j, x \in \mathcal{A}_{\Delta x}$.*

Proof. For each unsafe trajectory ξ_j with start and goal states in the safe set, there exists $x \in \xi_j, x \in \mathcal{A}$. Further, if there exists $x \in \xi_j \in (\mathcal{A} \setminus \mathcal{A}_{\Delta x})$, then there also exists $x \in \xi_j \in \mathcal{A}_{\Delta x}$. For contradiction, suppose there exists a time $\hat{t} \in \{1, \dots, T_j\}$ for which $\xi_j(\hat{t}) \in (\mathcal{A} \setminus \mathcal{A}_{\Delta x})$ and $\nexists t \in \{1, \dots, T_j\}$ for which $\xi_j(t) \in \mathcal{A}_{\Delta x}$. But this implies $\exists t < \hat{t}, \|\xi(t) - \xi(t+1)\| > \Delta x$ or $\exists t > \hat{t}, \|\xi(t) - \xi(t-1)\| > \Delta x$, i.e. to skip deeper than Δx into the unsafe set without first entering the Δx shell, the state must have changed by more than Δx in a single time-step. Contradiction. An analogous argument holds for the continuous-time case. □

The following result states that in discrete time, the learnable set of unsafe states $\mathcal{G}_{\neg s}^*$ is contained by the set of states which must be implied unsafe by setting $\mathcal{A}_{\Delta x}$ as unsafe. Furthermore, in continuous time, the same holds, except the $\mathcal{A}_{\Delta x}$ is replaced by the boundary of the unsafe set, $\partial\mathcal{A}$.

Theorem C.1 (Discrete time learnability for parametric constraints)**.** *For trajectories generated by discrete time systems, $\mathcal{G}_{\neg s} \subseteq \mathcal{G}_{\neg s}^* \subseteq I(\mathcal{F}_{\Delta x})$, where*

$$\mathcal{F}_{\Delta x} = \{\theta \mid \forall i \in \{1, \dots, N_s\}, \quad \forall x \in \xi_i^*, g(x, \theta) > 0, \quad \forall x \in \mathcal{A}_{\Delta x}, g(x, \theta) \leq 0\}.$$

Proof. Recall that $\mathcal{G}_{\neg s} \doteq \bigcap_{\theta \in \mathcal{F}} \{x \mid g(x, \theta) \leq 0\}$, where as previously defined, $\mathcal{F}$ is the feasible set of Problem 2. We can then show that $\mathcal{F}_{\Delta x} \subseteq \mathcal{F}$, since enforcing that $g(x, \theta) \leq 0$ for all $x \in \mathcal{A}_{\Delta x}$ implies that there exists $x \in \xi_j$, for all $j \in \{1, \dots, N_{\neg s}\}$ such that $g(x, \theta) \leq 0$, via Lemma C.2. Then, via Lemma C.1, $\mathcal{G}_{\neg s} = I(\mathcal{F}) \subseteq I(\mathcal{F}_{\Delta x})$. As this holds for any arbitrary set of trajectories, $\mathcal{G}_{\neg s}^* \subseteq I(\mathcal{F}_{\Delta x})$ as well, and $\mathcal{G}_{\neg s} \subseteq \mathcal{G}_{\neg s}^*$. □

Corollary C.1 (Continuous-time learnability for parametric constraints)**.** *For trajectories generated by continuous time systems, $\mathcal{G}_{\neg s} \subseteq \mathcal{G}_{\neg s}^* \subseteq I(\mathcal{F}_{\partial\mathcal{A}})$, where*

$$\mathcal{F}_{\partial\mathcal{A}} = \{\theta \mid \forall x \in \xi_i^*, \quad \forall i \in \{1, \dots, N_s\}, g(x, \theta) > 0, \quad \forall x \in \partial\mathcal{A}, g(x, \theta) \leq 0\}.$$

Proof. Since going from discrete time to continuous time implies $\Delta x \to 0$, $\mathcal{A}_{\Delta x} \to \partial\mathcal{A}$. Then, the logic from the proof of Theorem C.1 can be similarly applied to show the result. □

C.2 Conservativeness: Parametric

We write conditions for conservative recovery of the unsafe set and safe set when solving Problems 2 and 3 for discrete time and continuous time systems.

Theorem C.2 (Conservativeness: Known parameterization (Theorem 1 in the main body))**.** *Suppose the parameterization $g(x, \theta)$ is known exactly. Then, for a discrete-time system, extracting $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ (as defined in (3) and (4), respectively) from the feasible set of Problem 2 returns $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$. Further, if the known parameterization is $H(\theta)x_i \leq h(\theta)$ and M in Problem 3 is chosen to be greater than*

$$\max\Big(\max_{x_i \in \xi_s} \max_{\theta} \max_{j} (H(\theta)x_i - h(\theta))_j, \max_{x_i \in \xi_{\neg s}} \max_{\theta} \max_{j} (H(\theta)x_i - h(\theta))_j\Big),$$

then extracting $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ from the feasible set of Problem 3 recovers $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$.

Proof. We first prove that $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$. Consider first the case of Problem 2, or equivalently the case of Problem 3 where $M = \infty$ (in this case, Problem 3 exactly enforces that at least one state in each unsafe trajectory is unsafe and all states on demonstrations are safe).

Suppose for contradiction that there exists some $x \in \mathcal{G}_{\neg s}, x \notin \mathcal{A}$. By definition of $\mathcal{G}_{\neg s}$, $g(x, \theta) \leq 0$, for all $\theta \in \mathcal{F}$, where $\mathcal{F}$ is the feasible set of parameters θ in Problem 2. However, as $x \notin \mathcal{A}$, but for all $\theta \in \mathcal{F}, g(x, \theta) \leq 0$ we know that $\theta_{\mathcal{A}} \notin \mathcal{F}$, where $\theta_{\mathcal{A}}$ is the parameter associated with the true unsafe set $\mathcal{A}$. However, $\mathcal{F}$ will always contain $\theta_{\mathcal{A}}$, since:

- $\theta_{\mathcal{A}}$ satisfies $g(x, \theta_{\mathcal{A}}) > 0$ for all x in safe demonstrations, since all demonstrations are safe with respect to the true $\theta_{\mathcal{A}}$.
- For each trajectory $\xi_{\neg s}$ sampled using the hit-and-run procedure in Section 4.1, there exists $x \in \xi_{\neg s}$ such that $g(x, \theta_{\mathcal{A}}) \leq 0$.

We come to a contradiction, and hence for Problem 2 and for Problem 3 where $M = \infty$, $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$.

Now, we consider the conditions on M such that choosing $M \geq$ const or $M = \infty$ causes no changes in the solution of Problem 3. M must be chosen such that 1) $H(\theta)x_i - h(\theta) > -M\mathbf{1} \Leftrightarrow H(\theta)x_i - h(\theta) > -\infty\mathbf{1}$, for all safe states $x_i \in \xi_s$, and 2) $H(\theta)x_i - h(\theta) \leq M\mathbf{1} \Leftrightarrow H(\theta)x_i - h(\theta) \leq \infty\mathbf{1}$ for all states x_i on unsafe trajectories $\xi_{\neg s}$. Condition 1 is met if $-M < \min_{x_i \in \xi_s} \min_{\theta} \min_{j} (H(\theta)x_i - h(\theta))_j$, where v_j denotes the j-th element of vector v; denote as M_1 an M which satisfies this inequality. Condition 2 is met if $M \geq \max_{x_i \in \xi_{\neg s}} \max_{\theta} \max_{j} (H(\theta)x_i - h(\theta))_j$; denote as M_2 an M which satisfies this inequality. Then, M should be chosen to satisfy $M > \max(M_1, M_2)$.

The proof that $\mathcal{G}_s \subseteq \mathcal{S}$ is analogous. If there exists $x \in \mathcal{G}_s, x \notin \mathcal{S}$, $g(x, \theta) > 0$, for all $\theta \in \mathcal{F}$, then $\theta_{\mathcal{A}} \notin \mathcal{F}$. We follow the same reasoning from before to show that $\theta_{\mathcal{A}} \in \mathcal{F}$ for $M = \infty$. Now, provided the condition on M holds, we reach a contradiction. □

Remark. A simple corollary from Theorem C.2 is that by solving Problem 4 repeatedly for different query centers x_{query} for a discrete-time system and unioning over the resulting volumes will also provide conservative estimates of $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$. Further, if the assumption on M holds, then the volume extraction analogue of Problem 3 will also return conservative estimates of $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$.

As discussed in [5], with continuous-time system dynamics, assigning unsafeness in lower-cost trajectories difficult since there are an infinite number of states on the continuous trajectory. To ameliorate this, as in [5], we time-discretize the sampled lower-cost trajectories and feed the resulting discrete-time trajectories into Problems 2 and 3. This can potentially cause a mild overapproximation of the unsafe set, which we quantify after introducing some notation.

Definition C.6 (Normal vectors)**.** *Denote the outward-pointing normal vector at a point $p \in \partial\mathcal{A}$ as $\hat{n}(p)$. Furthermore, at non-differentiable points on $\partial\mathcal{A}$, $\hat{n}(p)$ is replaced by the set of normal vectors for the sub-gradient of the Lipschitz function describing $\partial\mathcal{A}$ at that point ([31]).*

Definition C.7 (γ-offset padding)**.** *Define the γ-offset padding $\partial\mathcal{A}_\gamma$ as: $\partial\mathcal{A}_\gamma = \{x \in \mathcal{X} \mid x = y + d\hat{n}(y), d \in [0, \gamma], y \in \partial\mathcal{A}\}$.*

Definition C.8 (γ-padded set)**.** *We define the γ-padded set of the unsafe set $\mathcal{A}$, $\mathcal{A}(\gamma)$, as the union of the γ-offset padding and $\mathcal{A}$: $\mathcal{A}(\gamma) \doteq \partial\mathcal{A}_\gamma \cup \mathcal{A}$.*

Definition C.9 (Maximum distance on trajectories)**.** *Denote* $D_\xi([a,b]) \doteq \sup_{t_1\in[a,b],t_2\in[t_1,b]} \|\xi(t_1) - \xi(t_2)\|_2$*, for some trajectory* ξ*. Denote* $D^* \doteq \max_{i\in\{1,\dots,N_{\neg s}\}} D^*_{\xi_i}([a_i,b_i])$*. In words,* $D_\xi([a,b])$ *is the maximum distance between any two points on trajectory* ξ *from time* a *to time* b*, and* D^* *takes the maximum distance over all* $N_{\neg s}$ *trajectories.*

Lemma C.3 (Maximum distance)**.** *Consider a continuous time trajectory* $\xi : [0,T] \to \mathcal{X}$*. Suppose it is known that in some time interval* $[a,b], a \le b, a, b \in [0,T]$*,* ξ *is unsafe; denote this subsegment as* $\xi([a,b])$*. Consider any* $t \in [a,b]$*. Then, the signed distance from* $\xi(t)$ *to the unsafe set,* $\mathsf{sd}(\xi(t),\mathcal{A})$*, is bounded by* $D_\xi([a,b]) \doteq \sup_{t_1\in[a,b],t_2\in[t_1,b]} \|\xi(t_1) - \xi(t_2)\|_2$*.*

Proof. Since there exists $\tilde{t} \in [a,b]$ such that $\xi(\tilde{t}) \in \mathcal{A}$, $\sup_{t\in[a,b]} \mathsf{sd}(\xi(t),\mathcal{A}) = \sup_{t\in[a,b]} \mathsf{sd}(\xi(t),\xi(\tilde{t})) \le \sup_{t_1\in[a,b],t_2\in[t_1,b]} \|\xi(t_1) - \xi(t_2)\|_2$. □

Corollary C.2. *For a continuous-time system where demonstrations and sampled unsafe trajectories are time-discretized, if* M *is chosen as in Theorem C.2,* $\mathcal{G}_s \subseteq \mathcal{S}$*, where* $\mathcal{S}$ *is the safe set, and* $\mathcal{G}_{\neg s} \subseteq \mathcal{A}(D^*)$*, where* D^* *is as defined in Definition C.9.*

Proof. The reasoning for $\mathcal{G}_s \subseteq \mathcal{S}$ follows from the proof of $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ in the proof of Theorem C.2.

Now we prove $\mathcal{G}_{\neg s} \subseteq \mathcal{A}(D^*)$. Suppose in this case, there exists a state $x = \xi_j(t_i) \notin \mathcal{A}$ which is truly safe but lies on a sampled unsafe trajectory $\xi_j([a_j,b_j])$, and suppose that $\{t_1,\dots,t_N\}$ is chosen such that for all $k \in \{1,\dots,N\} \setminus \{i\}$, $\xi_j(t_k)$ belongs to a known safe cell. Then, we may incorrectly learn that $\xi_j(t_i)$ is unsafe, as we force at least one point in the sampled trajectory to be unsafe. Via Lemma C.3, we know that $\xi_j(t_i)$ is at most $D_{\xi_j}([a_j,b_j])$ signed distance away from $\mathcal{A}$. Hence, for this trajectory, any learned guaranteed unsafe state must be contained in the $D_{\xi_j}([a_j,b_j])$-padded unsafe set. For this to hold for all unsafe trajectories sampled with the hit-and-run procedure presented in Section 4.1, we must pad the unsafe set by D^*. Hence, under this assumption, the algorithm returns a conservative estimate of the D^*-padded unsafe set. □

Let's consider the case where the true parameterization is not known and we use the method described in Section 4.4, where $g_s(x,\theta)$ is the simple parameterization. We consider the under-parameterized case (Theorem 3) and the over-parameterized case (Theorem 2). In particular, we analyze the case where the true parameterization, the under-parameterization, and the over-parameterization are defined respectively as:

$$g(x,\theta) \le 0 \Leftrightarrow \bigvee_{i=1}^{N^*} \big(g_s(x,\theta_i) \le 0\big) \quad (9) \qquad g(x,\theta) \le 0 \Leftrightarrow \bigvee_{i=1}^{\underline{N}} \big(g_s(x,\theta_i) \le 0\big), \quad \underline{N} < N^* \quad (10)$$

$$g(x,\theta) \le 0 \Leftrightarrow \bigvee_{i=1}^{\bar{N}} \big(g_s(x,\theta_i) \le 0\big), \quad \bar{N} > N^*. \quad (11)$$

Theorem C.3 (Conservativeness: Over-parameterization (Theorem 2 in the main body))**.** *Suppose the true parameterization and over-parameterization are defined as in (9) and (11). Then,* $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ *and* $\mathcal{G}_s \subseteq \mathcal{S}$*.*

Proof. Note that (9) is equivalent to $\Big(\bigvee_{i=1}^{\bar{N}} \big(g_s(x,\theta_i) \le 0\big)\Big)$, where $\theta_{N^*+1},\dots,\theta_{\bar{N}}$ are constrained to satisfy $\{x \mid g_s(x,\theta_i) \le 0\} = \emptyset, i = N^*+1,\dots,\bar{N}$. Thus, the true θ is equivalent to adding additional constraints on a loosened parameterization (the over-parameterization). Let $\hat{\mathcal{F}}$ be the feasible set of Problem 2 with θ loosened as above, i.e. $\mathcal{F} = \hat{\mathcal{F}} \cap \{\theta \mid \{x \mid g_s(x,\theta_i) \le 0\} = \emptyset, i = N^*+1,\dots,\bar{N}\}$. Via Lemma C.1, $\mathcal{F} \subseteq \hat{\mathcal{F}}$; thus, $I_{\neg s}(\hat{\mathcal{F}}) \subseteq I_{\neg s}(\mathcal{F}) \subseteq \mathcal{A}$, where the last set containment follows from Theorem 1. Vice versa, $I_s(\hat{\mathcal{F}}) \subseteq I_s(\mathcal{F}) \subseteq \mathcal{S}$, where again the last set containment follows from Theorem 1. □

Theorem C.4 (Conservativeness: Under-parameterization (Theorem 3 in the main body))**.** *Suppose the true parameterization and under-parameterization are defined as in (9) and (10). Furthermore, assume that we incrementally grow the parameterization as described in Section 4.4. Then, the following are true:*

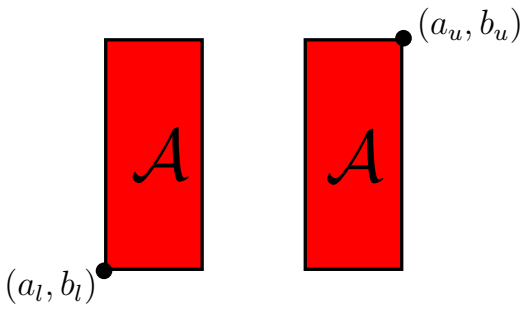

Figure C.1: Counterexample used in the proof of the first statement in Theorem C.4.

1. *$\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ are not guaranteed to be contained in $\mathcal{A}$ (unsafe set) and $\mathcal{S}$ (safe set), respectively.*
2. *Each recovered simple unsafe set $\mathcal{A}(\theta_i)$, $i = 1, \ldots, \underline{N}$, for any $\theta_1, \ldots, \theta_{\underline{N}} \in \mathcal{F}$, touches the true unsafe set (there are no spurious simple unsafe sets): for $i = 1, \ldots, \underline{N}$, for $\theta_1, \ldots, \theta_{\underline{N}} \in \mathcal{F}$, $\mathcal{A}(\theta_i) \cap \mathcal{A} \neq \emptyset$ ($\underline{N}$ is as defined in Section 4.4).*

Proof. 1. We first formally prove the statement with a counterexample and then follow up with logic related to the proof of Theorem C.3.

Consider the example in Fig. C.1, where the parameterization is chosen as a single axis-aligned box $[I_{2\times 2}, -I_{2\times 2}]^\top x \leq \theta$ but $\mathcal{A}$ is only representable with at least two boxes. Suppose demonstrations are provided which imply that (a_l, b_l) and (a_u, b_u) are unsafe; then $\text{AABB}(\{(a_l, b_l), (a_u, b_u)\}) \not\subseteq \mathcal{A}$ is implied unsafe.

Note that (10) is equivalent to $\left(\bigvee_{i=1}^{N^*} \left(g_s(x, \theta_i) \leq 0\right)\right)$, where $\theta_{\underline{N}+1}, \ldots, \theta_{N^*}$ are constrained to satisfy $\{x \mid g_s(x, \theta_i) \leq 0\} = \emptyset, i = \underline{N} + 1, \ldots, N^*$. Thus, restricting the parameterization is equivalent to adding additional constraints on the true θ. Let $\hat{\mathcal{F}}$ be the feasible set of Problem 2 with θ restricted as above, i.e. $\hat{\mathcal{F}} = \mathcal{F} \cap \{\theta \mid \{x \mid g_s(x, \theta_i) \leq 0\} = \emptyset, i = \underline{N} + 1, \ldots, N^*\}$. Via Lemma C.1, $\hat{\mathcal{F}} \subseteq \mathcal{F}$; thus, $I_{\neg s}(\mathcal{F}) \subseteq I_{\neg s}(\hat{\mathcal{F}})$. Since $I_{\neg s}(\mathcal{F})$ can equal $\mathcal{A}$, potentially $\mathcal{G}_{\neg s} = I_{\neg s}(\hat{\mathcal{F}}) \cap \mathcal{S} \neq \emptyset$. Vice versa, $I_s(\mathcal{F}) \subseteq I_s(\hat{\mathcal{F}})$, and since $I_s(\mathcal{F})$ can equal $\mathcal{S}$, potentially $\mathcal{G}_s = I_s(\hat{\mathcal{F}}) \cap \mathcal{S} \neq \emptyset$.

2. Assume, by contradiction, that Problem 2 outputs a simple unsafe set $\mathcal{A}(\theta_i), i \in \{1, \ldots, \underline{N}\}$, which does not touch the true unsafe set: $\exists i \in \{1, \ldots, \underline{N}\}, \mathcal{A}(\theta_i) \cap \mathcal{A}(\theta^*) = \emptyset$. Then, $\theta_j, j \in \{1, \ldots, \underline{N}\} \setminus \{i\}$ would be a feasible point for Problem 2 with a parametrization that contains only $\underline{N} - 1$ simple sets. However, we know Problem 2 with $\underline{N} - 1$ simple sets is infeasible. Contradiction.

□

D Extra numerical examples

D.1 U-shape (random demonstrations)

In this example, we show what the performance of our method looks like with random demonstrations on the U-shape example. On the left of Fig. D.1, we show that our coverage grows more slowly than for the case where demonstrations are chosen for their informativeness; furthermore, coverage for the safe set is higher and coverage for the unsafe set is lower in the random demonstration case. This is because by using random demonstrations, we cover a good deal of $\mathcal{S}$, so $\mathcal{G}_s$ becomes larger; on the other hand, many of these safe demonstrations may not come in contact with the constraint, so there are relatively few unsafe trajectories that can be sampled, so $\mathcal{G}_{\neg s}$ is not as large. In the center of Fig. D.1, we show that the accuracy of our method doesn't change much, though the relative performance of the NN gets worse for classifying safe states; this is because the accuracy for the NN is now being evaluated on a larger region since $\mathcal{G}_s$ is larger due to more demonstrations. As in previous examples, the NN error bars are generated by training the NN ten times with initializations

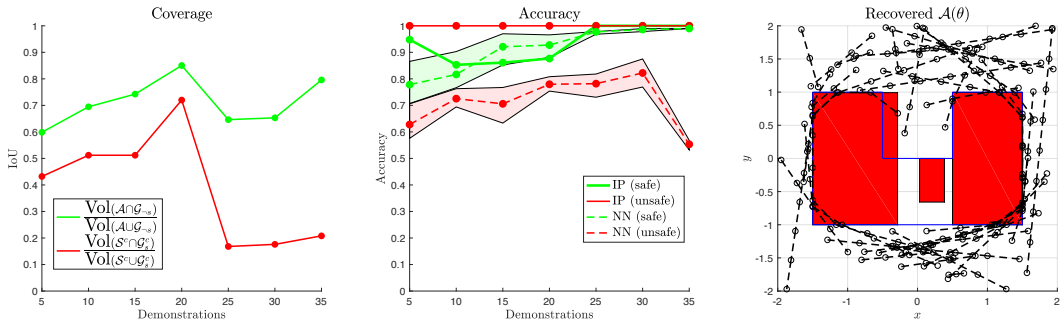

Figure D.1: U-shape performance with random demonstrations. **Left**: Coverage of $\mathcal{A}$ and $\mathcal{S}$. **Center**: Classification accuracy. **Right**: A recovered feasible $\mathcal{A}(\theta)$, overlaid with demonstrations, and the true unsafe set $\mathcal{A}$ is outlined in blue.

using different random seeds. On the right of Fig. D.1, we display a feasible $\mathcal{A}(\theta)$ recovered by solving a multi-box variant of Problem 3. With more demonstrations, the gap between $\mathcal{A}(\theta)$ and the true unsafe set $\mathcal{A}$ will continue to shrink.

The main takeaways from this experiment are: 1) when demonstrations are not informative (in the sense that they do not interact with the constraint), it can take many demonstrations to learn the unsafe set (this holds for any constraint recovery method), and 2) our accuracy remains just as high as for the case with specifically chosen demonstrations and is not much affected by the coverage.

E Experimental details

For all neural network baseline results in every experiment, the network is trained with weights initialized using ten different random seeds, and the resulting performance range (displayed as a shaded region) and average performance over the ten random seeds are plotted in the figures.

E.1 Unknown parameterizations

We emphasize that for all examples with unknown parameterization, by following the incremental procedure detailed in Section 4.4, we are finding the minimum number of boxes required to represent the data; in other words, we are always operating with the minimal feasible parameterization.

U-shape and infinite boxes:

- For both experiments, the system dynamics are $x_{t+1} \doteq [\chi_{t+1}, y_{t+1}]^\top = [\chi_t, y_t]^\top + [u_t^\chi, u_t^y]^\top$. The U-shape experiment uses control constraints $\|[u_t^\chi, u_t^y]\|_2 \leq 0.5$, while the infinite-box experiment uses control constraints $\|[u_t^\chi, u_t^y]\|_2 \leq 1$.
- For both experiments, the cost function is $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|x_{t+1} - x_t\|_2^2$.
- Since the cost function has optimal substructure, 100000 unsafe trajectories for each sub-trajectory are sampled. The dataset is downsampled to 50 unsafe trajectories for each sub-trajectory, which are to be fed into the multi-box variant of Problem 3.
- For both experiments, the initial parameter set is restricted to $[-5, -5, -3, -3]^\top \leq \theta_i \leq [8, 8, 3, 3]^\top$, for each θ_i (the parameter for box i). For the infinite-box experiment, each box is restricted to be at least 1.25×1.25 in width/height.
- Sampling time is around 15 seconds per demonstration (for the U-shape experiment) and 10 seconds per demonstration (for the infinite-box experiment). Computation time for solving Problem 3 is around 40 seconds (for the U-shape experiment) and 15-20 seconds (for the infinite-box experiment).
- The same data is used for training the neural network (7800 trajectories total for the U-shape case, 2000 trajectories for the infinite-box case). The neural network architecture used for this example is a fully connected (FC) layer, $2 \times 10 \rightarrow$ LSTM, $10 \times 10 \rightarrow$ FC 10×1 (the recurrent layer is used since we have variable length trajectories as training input). The network is trained using Adam.

U-shape with random demonstrations:

- The system dynamics are $x_{t+1} \doteq [\chi_{t+1}, y_{t+1}]^\top = [\chi_t, y_t]^\top + [u_t^\chi, u_t^y]^\top$ with control constraints $\|[u_t^\chi, u_t^y]\|_2 \leq 0.5$.
- The cost function is $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|x_{t+1} - x_t\|_2^2$.
- Demonstrations are generated for 35 pairs of start/goal states sampled uniformly at random over $(\chi, y) \in [-2, 2] \times [-2, 2]$, rejecting any start/goal states that lie in $\mathcal{A}$.
- Since the cost function has optimal substructure, 10000 unsafe trajectories for each sub-trajectory are sampled. The dataset is downsampled to 25 unsafe trajectories for each sub-trajectory, which are to be fed into the multi-box variant of Problem 3.
- The initial parameter set is restricted to $[-5, -5, -3, -3]^\top \leq \theta_i \leq [8, 8, 3, 3]^\top$, for each θ_i (the parameter for box i).
- Sampling time is around 2 minutes total. Computation time for solving the multi-box variant of Problem 3 is around 90 seconds.
- The same data is used for training the neural network (10100 trajectories total). The neural network architecture used for this example is a fully connected (FC) layer, $2 \times 10 \rightarrow$ LSTM, $10 \times 10 \rightarrow$ FC 10×1. The network is trained using Adam.

E.2 High-dimensional examples

7-DOF arm, optimal/suboptimal demonstrations

- The system dynamics are $\doteq \theta_{t+1}^i = \theta_t^i + u_t^i$, $i = 1, \ldots, 7$, with control constraints $-2 \leq u_t^i \leq 2$, $i = 1, \ldots, 7$, where the state is $x = [\theta^1, \ldots, \theta^7]$.
- The cost function is $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|x_{t+1} - x_t\|_2^2$. Note that the generate demonstrations (displayed in Fig. 3) push up against the position constraint, since the trajectory minimizing joint-space path length without the position constraint is an arc that exceeds the bounds of the position constraint; the position constraint ends up increasing the cost by truncating that arc.
- The true safe set is $(x, y, z, \alpha, \beta, \gamma) \in [-0.51, 0.51] \times [-0.3, 1.1] \times [-0.51, 0.51] \times [-\pi, \pi] \times [-\pi/120, \pi/120] \times [-\pi/120, \pi/120]$ for the optimal case and the true safe set is $(x, y, z, \alpha, \beta, \gamma) \in [-0.57, 0.47] \times [-0.10, 1.17] \times [-0.56, 0.56] \times [-\pi, \pi] \times [-0.12, 0.12] \times [-0.125, 0.125]$ for the suboptimal case.
- Since the cost function has optimal substructure, 250000 unsafe trajectories for each sub-trajectory are sampled. For the suboptimal case, the continuous-time demonstrations are time-discretized down to $T = 10$ time-steps. The dataset is downsampled to 500 unsafe trajectories for each sub-trajectory, which are to be fed into Problem 3.
- For the optimal case, the demonstrations are obtained by solving trajectory optimization problems solved with the IPOPT solver [32]. For the suboptimal case, the demonstrations are recorded in a virtual reality (VR) environment displayed in Fig. E.1.
- The initial parameter set is restricted to $[-1.5, -1.5, -1.5, -\pi, -\pi, -\pi]^\top \leq [x, y, z, \alpha, \beta, \gamma]^\top \leq [1.5, 1.5, 1.5, \pi, \pi, \pi]^\top$.
- Sampling time is 12.5 minutes total for the optimal case and 9 minutes total for the suboptimal case. Computation time for solving Problem 2 is around 2 seconds for both the optimal/suboptimal case.
- The same data is used for training the neural network (70000 trajectories total for the optimal case, 49900 trajectories total for the suboptimal case). The neural network architecture used for this example is a fully connected (FC) layer, $3 \times 20 \rightarrow$ LSTM, $20 \times 20 \rightarrow$ FC 20×1. The network is trained using Adam.

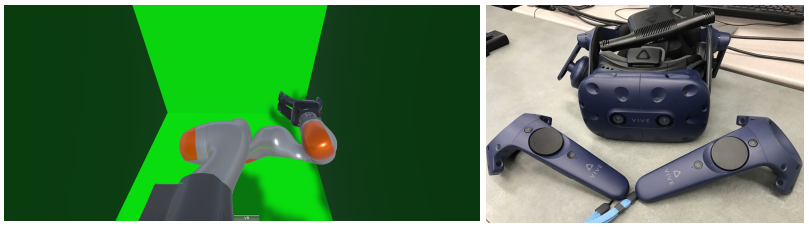

Figure E.1: VR setup. **Left**: VR environment as viewed from the Vive headset. The green box represents the position constraints on the end effector. The end effector is commanded to move by dragging it with the HTC Vive controllers (**right**).

12D quadrotor example

- The system dynamics [33] are

$$\begin{bmatrix} \dot{\chi} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \\ \ddot{\chi} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\alpha} \\ \ddot{\beta} \\ \ddot{\gamma} \end{bmatrix} = \begin{bmatrix} \dot{\chi} \\ \dot{y} \\ \dot{z} \\ \dot{\beta}\frac{\sin(\gamma)}{\cos(\beta)} + \dot{\gamma}\frac{\cos(\gamma)}{\cos(\beta)} \\ \beta\cos(\gamma) - \dot{\gamma}\sin(\gamma) \\ \dot{\alpha} + \dot{\beta}\sin(\gamma)\tan(\beta) + \dot{\gamma}\cos(\gamma)\tan(\beta) \\ -\frac{1}{m}[\sin(\gamma)\sin(\alpha) + \cos(\gamma)\cos(\alpha)\sin(\beta)]u_1 \\ -\frac{1}{m}[\cos(\alpha)\sin(\gamma) - \cos(\gamma)\sin(\alpha)\sin(\beta)]u_1 \\ g - \frac{1}{m}[\cos(\gamma)\cos(\beta)]u_1 \\ \frac{I_y - I_z}{I_x}\dot{\beta}\dot{\gamma} + \frac{1}{I_x}u_2 \\ \frac{I_z - I_x}{I_y}\dot{\alpha}\dot{\gamma} + \frac{1}{I_y}u_3 \\ \frac{I_x - I_y}{I_z}\dot{\alpha}\dot{\beta} + \frac{1}{I_z}u_4 \end{bmatrix}, \tag{12}$$

  with control constraints $[0, -0.02, -0.02, -0.02]^\top \leq u_t \leq [mg, 0.02, 0.02, 0.02]^\top$. For our purposes, we convert the dynamics to discrete time by performing forward Euler integration with discretization time $\delta t = 0.4$ seconds. The state is $x = [\chi, y, z, \alpha, \beta, \gamma, \dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}, \dot{\gamma}]^\top$, and the constants are $g = -9.81\text{m/s}^2$, $m = 1\text{kg}$, $I_x = 0.5\text{kg}\cdot\text{m}^2$, $I_y = 0.1\text{kg}\cdot\text{m}^2$, and $I_z = 0.3\text{kg}\cdot\text{m}^2$.
- The known unsafe set in (χ, y, z) is $(\chi, y, z) \notin [-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$.
- The true safe set in $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$ is $(\dot{\alpha}, \dot{\beta}, \dot{\gamma}) \in [-0.006, 0.006]^3$.
- The cost function is $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|[\chi_{i+1}, y_{i+1}, z_{i+1}, \dot{\alpha}_{i+1}, \dot{\beta}_{i+1}, \dot{\gamma}_{i+1}]^\top - [\chi_i, y_i, z_i, \dot{\alpha}_i, \dot{\beta}_i, \dot{\gamma}_i]^\top\|_2$ (penalizing acceleration and path length).
- The demonstrations are obtained by solving trajectory optimization problems solved with the IPOPT solver [32].
- Since the cost function has optimal substructure, 10000 unsafe trajectories for each sub-trajectory are sampled. The dataset is downsampled to 500 unsafe trajectories for each sub-trajectory, which are to be fed into Problem 3.
- The initial parameter set is restricted to $[-\pi/2, -\pi/2, -\pi/2]^\top \leq [\dot{\alpha}, \dot{\beta}, \dot{\gamma}]^\top \leq [\pi/2, \pi/2, \pi/2]^\top$.
- Sampling time is 8.5 minutes total for the optimal case and 9 minutes total for the suboptimal case. Computation time for solving Problem 2 is 12 seconds.
- The same data is used for training the neural network (30000 trajectories total). The neural network architecture used for this example is a fully connected (FC) layer, $6 \times 36 \rightarrow$ LSTM, $36 \times 42 \rightarrow$ FC 42×1. The network is trained using Adam.

E.3 Black-box system dynamics

Pushing example

- The cost function is $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|x_{t+1} - x_t\|_2^2$. The two demonstrations are manually generated and are not exactly optimal.
- 1000 unsafe trajectories for each demonstrations are sampled.

- The initial parameter set is restricted to $[-5, -5, -3, -3]^\top \leq \theta_i \leq [8, 8, 3, 3]^\top$.
- Sampling time is 2 hours for each demonstration (using the simulator is slower than using the closed form dynamics). Computation time for solving Problem 2 is around 1 second.
- Demonstrations are time-discretized to 40 simulator timesteps when input to Problem 3.
- The same data is used for training the neural network (2700 trajectories total). The neural network architecture used for this example is a fully connected (FC) layer, $8 \times 10 \rightarrow$ FC, $10 \times 10 \rightarrow$ FC 10×1. No recurrent layer is used this time since all trajectories are of the same length (no sub-trajectories were sampled this time due to speed). The network is trained using Adam.

F Summary of frequently used notation

Meaning	Notation
State, state space	x, $\mathcal{X}$
Control, control space	$u, \mathcal{U}$
State/control trajectory	ξ_x, ξ_u
Constraint state, constraint space	$k, \mathcal{C}$
Safe set, unsafe set	$\mathcal{S}$, $\mathcal{A}$
Parameterized safe set	$\mathcal{S}(\theta) = \{k \mid g(k, \theta) > 0\}$
Parameterized unsafe set	$\mathcal{A}(\theta) = \{k \mid g(k, \theta) \leq 0\}$
Safe demonstration j	$\xi^*_{s_j}$
Sampled unsafe trajectory k	$\xi_{\neg s_k}$
Guaranteed safe set	$\mathcal{G}_s$
Guaranteed unsafe set	$\mathcal{G}_{\neg s}$

Table 1: Notation.