Learning Constraints from Locally-Optimal Demonstrations under Cost Function Uncertainty

Glen Chou, Necmiye Ozay, and Dmitry Berenson

Abstract—We present an algorithm for learning parametric constraints from locally-optimal demonstrations, where the cost function being optimized is uncertain to the learner. Our method uses the Karush-Kuhn-Tucker (KKT) optimality conditions of the demonstrations within a mixed integer linear program (MILP) to learn constraints which are consistent with the local optimality of the demonstrations, by either using a known constraint parameterization or by incrementally growing a parameterization that is consistent with the demonstrations. We provide theoretical guarantees on the conservativeness of the recovered safe/unsafe sets and analyze the limits of constraint learnability when using locally-optimal demonstrations. We evaluate our method on high-dimensional constraints and systems by learning constraints for 7-DOF arm and quadrotor examples, show that it outperforms competing constraint-learning approaches, and can be effectively used to plan new constraint-satisfying trajectories in the environment.

Index Terms—Learning from Demonstration, Robot Safety, Optimization and Optimal Control, Learning and Adaptive Systems

I. INTRODUCTION

LEARNING from demonstration has largely focused on learning cost and reward functions, through the frameworks of inverse optimal control and inverse reinforcement learning (IOC/IRL) [1]–[4], which replicate the behavior of an expert demonstrator when optimized. However, real-world planning problems, such as navigating a quadrotor in an urban environment, also require that the system obey hard constraints, that is, system trajectories must remain in a set of safe (constraint-satisfying) states. As constraints enforce safety more strictly than cost function penalties, which may "soften" a constraint and allow violations, they are better suited for planning in safety-critical situations. Furthermore, while different tasks may require different cost functions, oftentimes safety constraints are shared across tasks, and identifying them can help the robot generalize.

Initial work in [5], [6] has taken steps towards identifying constraints from approximately globally-optimal expert demonstrations, assuming that the demonstrator's cost function is known exactly. However, as humans are not always experts

at performing a task, requiring them to provide demonstrations which are nearly globally-optimal can be unreasonable. Furthermore, it is rare for the cost function being optimized to be known exactly by the learner. To address these shortcomings, we consider the problem of learning parametric constraints shared across tasks from approximately locally-optimal demonstrations under parametric cost function uncertainty. Our method is based on the insight that locally-optimal, constraint-satisfying demonstrations satisfy the Karush-Kuhn-Tucker (KKT) optimality conditions, which are first-order necessary conditions for local optimality of a solution to a constrained discrete-time optimal control problem. We solve a mixed integer linear program (MILP) to recover constraint and cost function parameters which make the demonstrations locally-optimal. We make the following specific contributions in this paper:

- We develop a novel algorithm for learning parametric, potentially non-convex constraints from approximately locally-optimal demonstrations, where the parameterization can either be provided or grown incrementally to be consistent with the data. The method can extract volumes of safe/unsafe states (states which satisfy/do not satisfy the constraints) for future guaranteed safe planning and enable planners to query states for safety.
- Our method can learn constraints despite uncertainty in the cost function and can also recover a cost function jointly with the constraint.
- Under mild assumptions, we prove that our method recovers guaranteed conservativeness estimates (that is, inner approximations) of the true safe/unsafe sets, and analyze the learnability of a constraint from locally-optimal compared to globally-optimal demonstrations.
- We evaluate our method on difficult constraint learning problems in high-dimensional constraint spaces (23 dimensions) on systems with complex nonlinear dynamics and demonstrate that our method outperforms previous methods for parametric constraint inference [5], [6].

II. RELATED WORK

Previous work in IOC [7]–[9] has used the KKT conditions to recover a cost function from demonstrations, assuming that the constraints are known. In contrast, our method learns both. The risk-sensitive IRL approach in [10] also uses the KKT conditions, and is complementary to our work, which learns hard constraints. Perhaps the closest to our work is [11], which aims to recover a cost function and constraint simultaneously using the KKT conditions. However, to avoid non-convexity in the cost/constraint recovery problem, [11]

restricts their method to recovering convex constraints and do not directly search for constraint parameters, instead enumerating an *a-priori* fixed, finite set of candidate constraint parameters using a method holding only for the convex case. In contrast, by formulating our problem as a MILP, our method avoids enumeration, directly searching the full continuous space of possible constraint parameters. It also enables us to learn non-convex, nonlinear constraints while retaining formal guarantees on the conservativeness of the recovered constraint.

There also exists prior work in learning geometric state space constraints [12], [13], which our method generalizes by learning non-convex constraints not necessarily defined in the state space. Learning local trajectory-based constraints has also been explored in [14]–[19] by reasoning over the constraints within a single trajectory or task. These methods complement our approach, which learns a global constraint shared across tasks. In the constraint-learning literature, our work is closest to [5] and [6], which learn a global shared constraint by sampling unsafe trajectories using globally-optimal demonstrations, a known cost function, and a system simulator. In addition to the drawbacks of global optimality and assuming a known cost as mentioned previously, sampling unsafe trajectories can be difficult for systems with complicated dynamics. By using the KKT conditions, which *implicitly* define the unsafe set instead of explicitly through unsafe trajectories, our method sidesteps both the need for an exact cost function to classify the safety of sampled trajectories as well as any sampling difficulties.

III. PRELIMINARIES AND PROBLEM SETUP

We consider discrete-time nonlinear systems $x_{t+1} = f(x_t, u_t, t)$, $x \in \mathcal{X}$ and $u \in \mathcal{U}$, performing tasks Π, which are represented as constrained optimization problems over state/control trajectories $\xi_{xu} \doteq (\xi_x, \xi_u)$:

Problem 1 (Forward problem / "task" Π):

$$
\begin{aligned}
\underset{\xi_{xu}}{\text{minimize}} \quad & c(\xi_{xu}, \gamma) \\
\text{subject to} \quad & \phi(\xi_{xu}) \in \mathcal{S}(\theta) \subseteq \mathcal{C} \\
& \bar{\phi}(\xi_{xu}) \in \bar{\mathcal{S}} \subseteq \bar{\mathcal{C}} \\
& \phi_\Pi(\xi_{xu}) \in \mathcal{S}_\Pi \subseteq \mathcal{C}_\Pi
\end{aligned}
\tag{1}
$$

where $c(\cdot, \gamma)$ is a potentially non-convex cost function, parameterized by $\gamma \in \Gamma$. In Sec. IV-A to IV-C, we assume that γ is known (through possibly inaccurate prior knowledge) to ease notation; we later relax this assumption and discuss how to learn γ from the demonstrations. Further, $\phi(\cdot)$ is a known mapping from state-control trajectories to a constraint space $\mathcal{C}$, elements of which are referred to as *constraint states* $p \in \mathcal{C}$. Mappings $\bar{\phi}(\cdot)$ and $\phi_\Pi(\cdot)$ 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 encode the system dynamics in $\bar{\mathcal{S}}$ and start/goal constraints in $\mathcal{S}_\Pi$. Grouping the constraints of Problem 1 as equality/inequality (eq/ineq) constraints and known/unknown $(k/\neg k)$ constraints, we can write:

$$
\begin{aligned}
h_{i,k}(\xi_{xu}) = 0, i = 1, ..., N_k^{\text{eq}} &\quad\Leftrightarrow \mathbf{h}_k(\xi_{xu}) = \mathbf{0} \\
g_{i,k}(\xi_{xu}) \leq 0, i = 1, ..., N_k^{\text{ineq}} &\quad\Leftrightarrow \mathbf{g}_k(\xi_{xu}) \leq \mathbf{0} \\
g_{i,\neg k}(\xi_{xu}, \theta) \leq 0, i = 1, ..., N_{\neg k}^{\text{ineq}} &\quad\Leftrightarrow \mathbf{g}_{\neg k}(\xi_{xu}, \theta) \leq \mathbf{0}
\end{aligned}
\tag{2}
$$

where $\mathbf{h}_k(\xi_{xu}) \in \mathbb{R}^{N_k^{\text{eq}}}$, $\mathbf{g}_k(\xi_{xu}) \in \mathbb{R}^{N_k^{\text{ineq}}}$, and $\mathbf{g}_{\neg k}(\xi_{xu}, \theta) \in \mathbb{R}^{N_{\neg k}^{\text{ineq}}}$. Note that unknown equality constraints $\mathbf{h}_{\neg k}(\xi_{xu}, \theta) = 0$ can be written equivalently as $\mathbf{h}_{\neg k}(\xi_{xu}, \theta) \leq 0, -\mathbf{h}_{\neg k}(\xi_{xu}, \theta) \leq 0$. As shorthand, let $g(p, \theta) \doteq \max_{i \in \{1, ..., N_{\neg k}^{\text{ineq}}\}} (g_{i, \neg k}(p, \theta))$. We now define

$$
\mathcal{S}(\theta) \doteq \{p \in \mathcal{C} \mid g(p, \theta) \leq 0\}
\tag{3}
$$

$$
\mathcal{A}(\theta) \doteq \mathcal{S}(\theta)^c = \{p \in \mathcal{C} \mid g(p, \theta) > 0\}
\tag{4}
$$

as an unknown safe/unsafe set defined by unknown parameter $\theta \in \Theta$, for possibly unknown parameterizations $g_{i, \neg k}(\cdot, \cdot)$. Last, we restrict Γ and Θ to be unions of polytopes.

Intuitively, a trajectory ξ_{xu} is locally-optimal if all trajectories within a neighborhood of ξ_{xu} have cost greater than or equal to $c(\xi_{xu})$. More precisely, for a trajectory to be locally-optimal, it necessarily satisfies the KKT conditions [20]. We define a demonstration ξ^{loc} as a state-control trajectory which we assume approximately solves Problem 1 to local optimality, i.e. it satisfies all constraints and is in the neighborhood of a local optimum.

Our goal is to recover the safe set $\mathcal{S}(\theta)$ and unsafe set $\mathcal{A}(\theta)$, given N_s demonstrations $\{\xi_j^{\text{loc}}\}_{j=1}^{N_s}$, known shared safe set $\bar{\mathcal{S}}$, and task-dependent constraints $\mathcal{S}_\Pi$. As a byproduct, our method can also recover unknown cost parameters γ.

IV. METHOD

We detail our constraint-learning algorithm. First, we formulate the general KKT-based constraint recovery problem (Sec. IV-A) and then develop specific optimization problems for the cases where the constraint is defined as a union of offset-parameterized (Sec. IV-B) or affinely-parameterized constraints (Sec. IV-D). We show how to extract guaranteed safe/unsafe states (Sec. IV-C), handle unknown constraint parameterizations (Sec. IV-E), and handle cost function uncertainty (Sec. IV-F). In closing, we show how our method can be used within a planner to guarantee safety (Sec. IV-G).

A. Constraint recovery via the KKT conditions

Recall that the KKT conditions are necessary conditions for local optimality of a solution of a constrained optimization problem [20]. For constraints (2) and Lagrange multipliers λ and ν, the KKT conditions for the jth locally-optimal demonstration ξ_j^{loc}, denoted KKT(ξ_j^{loc}), are:

Primal feasibility:	$\mathbf{h}_k(\xi_j^{\text{loc}}) = \mathbf{0}$,	(5a)
	$\mathbf{g}_k(\xi_j^{\text{loc}}) \leq \mathbf{0}$,	(5b)
	$\mathbf{g}_{\neg k}(\xi_j^{\text{loc}}, \theta) \leq \mathbf{0}$,	(5c)
Lagrange mult.	$\lambda_{i,k}^j \geq 0, \; i = 1, ..., N_k^{\text{ineq}} \;\Leftrightarrow\; \boldsymbol{\lambda}_k^j \geq \mathbf{0}$	(5d)
nonnegativity:	$\lambda_{i,\neg k}^j \geq 0, \; i = 1, ..., N_{\neg k}^{\text{ineq}} \;\Leftrightarrow\; \boldsymbol{\lambda}_{\neg k}^j \geq \mathbf{0}$	(5e)
Complementary	$\boldsymbol{\lambda}_k^j \odot \mathbf{g}_k(\xi_j^{\text{loc}}) = \mathbf{0}$	(5f)
slackness:	$\boldsymbol{\lambda}_{\neg k}^j \odot \mathbf{g}_{\neg k}(\xi_j^{\text{loc}}, \theta) = \mathbf{0}$	(5g)
Stationarity:	$\nabla_{\xi_{xu}} c_\Pi(\xi_j^{\text{loc}}) + \boldsymbol{\lambda}_k^{j\top} \nabla_{\xi_{xu}} \mathbf{g}_k(\xi_j^{\text{loc}})$	
	$\quad + \boldsymbol{\lambda}_{\neg k}^{j\top} \nabla_{\xi_{xu}} \mathbf{g}_{\neg k}(\xi_j^{\text{loc}}, \theta)$	(5h)
	$\quad + \boldsymbol{\nu}_k^{j\top} \nabla_{\xi_{xu}} \mathbf{h}_k(\xi_j^{\text{loc}}) = \mathbf{0}$	

where $\nabla_{\xi_{xu}}(\cdot)$ takes the gradient with respect to a flattened trajectory ξ_{xu} and $\odot$ denotes elementwise multiplication. For compactness, we vectorize the multipliers $\boldsymbol{\lambda}_k^j \in \mathbb{R}^{N_k^{\text{ineq}}}$, $\boldsymbol{\lambda}_{\neg k}^j \in \mathbb{R}^{N_{\neg k}^{\text{ineq}}}$, and $\boldsymbol{\nu}_k^j \in \mathbb{R}^{N_k^{\text{ineq}}}$. We drop the γ dependency,

as the cost is assumed known for now, as well as (5a)-(5b), as they involve no decision variables. Then, finding a constraint consistent with the local optimality conditions of the N_s demonstrations amounts to finding a constraint parameter θ which satisfies the KKT conditions for each demonstration. That is, we can solve the following feasibility problem:

Problem 2 (KKT inverse, locally-optimal):

$$
\begin{aligned}
&\text{find} && \theta, \boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j, \quad j = 1, ..., N_s \\
&\text{subject to} && \{\text{KKT}(\xi_j^{\text{loc}})\}_{j=1}^{N_s}
\end{aligned} \tag{6}
$$

Further, to address suboptimality (i.e. approximate local-optimality) in the demonstrations, we can relax the stationarity (5h) and complementary slackness constraints (5f)-(5g) and place corresponding penalties into the objective function:

Problem 3 (KKT inverse, suboptimal):

$$
\begin{aligned}
&\underset{\theta, \boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j}{\text{minimize}} && \sum_{j=1}^{N_s} \left(\|\text{stat}(\xi_j^{\text{loc}})\|_1 + \|\text{comp}(\xi_j^{\text{loc}})\|_1 \right) \\
&\text{subject to} && \text{(5c)} - \text{(5e)}, \ \forall \xi_j^{\text{loc}}, \ j = 1, \ldots, N_s
\end{aligned} \tag{7}
$$

where $\text{stat}(\xi_j^{\text{loc}})$ denotes the LHS of Eq. (5h) and $\text{comp}(\xi_j^{\text{loc}})$ denotes the concatenated LHSs of Eqs. (5f) and (5g).

Denote the projection of the feasible set of Problem 2 onto Θ as $\mathcal{F}$. We define the set of learned guaranteed safe/unsafe constraint states as $\mathcal{G}_s/\mathcal{G}_{\neg s}$, respectively. For Problem 2, a constraint state p is learned guaranteed safe/unsafe if p is marked safe/unsafe for all $\theta \in \mathcal{F}$. Formally, we have:

$$
\mathcal{G}_s \doteq \bigcap_{\theta \in \mathcal{F}} \{p \mid g(p, \theta) \leq 0\} \ (8) \qquad \mathcal{G}_{\neg s} \doteq \bigcap_{\theta \in \mathcal{F}} \{p \mid g(p, \theta) > 0\} \ (9)
$$

We now formulate variants of Problem 2 which are efficiently solvable for specific constraint parameterizations. For legibility, we describe the method assuming $\mathcal{C} = \mathcal{X}$ and $\phi(\cdot)$ is the identity. Due to the bilinearity between decision variables in Problems 2 and 3 for some parameterizations, we describe exact (Sec. IV-B) and relaxed (Sec. IV-D) formulations for recovering the unknown parameters.

B. Unions of offset-parameterized constraints

Consider when Problem 1 involves avoiding an unsafe set $\mathcal{A}(\theta)$ described by the union and intersection of offset-parameterized half-spaces (i.e. θ does not multiply p):

$$
\mathcal{A}(\theta) = \bigcup_{m=1}^{N_c} \bigcap_{n=1}^{N_c^m} \{p \mid a_{m,n}^\top p < b_{m,n}(\theta)\} \tag{10}
$$

This parameterization can represent any arbitrarily-shaped unsafe set if N_c is sufficiently large (i.e. as a union of polytopes) [21], though in practice our method may not be efficient for large N_c. We will often use the specific case of unions of axis-aligned N_c^m-dimensional hyper-rectangles,

$$
\mathcal{A}(\theta) = \bigcup_{m=1}^{N_c} \bigcap_{n=1}^{N_c^m} \{p \mid p < \overline{\theta}_n^m, -p < -\underline{\theta}_n^m\}, \tag{11}
$$

where $\overline{\theta}_n^m/\underline{\theta}_n^m$ are the upper/lower extents of dimension n of box m. We now modify the KKT conditions to handle the "or" constraints in (10). Primal feasibility (5c) changes to

$$
\forall p \in \xi_j^{\text{loc}}, \forall m = 1, ..., N_c, \ \bigvee_{n=1}^{N_c^m} \left(a_{m,n}^\top p \geq b_{m,n}(\theta)\right), \tag{12}
$$

which can be implemented using the big-M formulation [22]:

$$
\forall p, \forall m, \mathbf{A}_m^\top p \geq \mathbf{b}_m(\theta) - M(\mathbf{1} - \mathbf{z}_m^{j,p}), \ \sum_{n=1}^{N_c^m} \mathbf{z}_m^{j,p}(n) \geq 1, \tag{13}
$$

where M is a large positive number, $\mathbf{A}_m \in \mathbb{R}^{N_c^m \times |p|}$ and $\mathbf{b}_m \in \mathbb{R}^{N_c^m}$ are the vertical concatenation of $a_{m,n}$ and $b_{m,n}$ for all n, $\mathbf{z}_m^{j,p} \in \{0,1\}^{N_c^m}$ are binary variables encoding that at least one half-space constraint must hold, and $\mathbf{z}_m^{j,p}(n)$ is the nth entry of $\mathbf{z}_m^{j,p}$. For demonstration ξ_j^{loc} to be locally-optimal, we know that for each $p \in \xi_j^{\text{loc}}$, the complementary slackness condition, $\lambda_{(m,n),\neg k}^{j,p}(a_{m,n}^\top p - b_{m,n}(\theta)) = 0$, must hold for at least one n and for all m in Eq. (12). Furthermore, in the stationarity condition (5h), $\lambda_{(m,n),\neg k}^{j,p}\nabla_p g_{(m,n),\neg k}(\xi_j^{\text{loc}}, \theta)$ terms should only be included for (m,n) pairs where the complementary slackness condition is enforced. To implement this, we can enforce that $\lambda_{(m,n),\neg k}^{j,p}(a_{m,n}^\top p - b_{m,n}(\theta)) = 0$ holds for all $p \in \xi_j^{\text{loc}}$, for all $m \in \{1, ..., N_c\}$, and for some $n \in \{1, ..., N_c^m\}$ by writing:

$$
\forall p, m, \begin{bmatrix} \boldsymbol{\lambda}_{m,\neg k}^{j,p} \\ \mathbf{A}_m^\top p - \mathbf{b}_m(\theta) \end{bmatrix} \leq M \begin{bmatrix} \mathbf{z}_{m,1}^{j,p} \\ \mathbf{z}_{m,2}^{j,p} \end{bmatrix}, \quad \mathbf{z}_{m,1}^{j,p} + \mathbf{z}_{m,2}^{j,p} \leq 2 - \mathbf{q}_m^{j,p},
$$
$$
\sum_{n=1}^{N_c^m} \mathbf{q}_m^{j,p}(n) \geq 1, \quad \mathbf{z}_{m,1}^{j,p}, \mathbf{z}_{m,2}^{j,p}, \mathbf{q}_m^{j,p} \in \{0,1\}^{N_c^m} \tag{14}
$$

together with (5d) and (5e), where we use a big-M formulation with binary variables z (encoding the complementary slackness condition) and q (encoding if the complementary slackness condition is being enforced). We have denoted $\boldsymbol{\eta}_m^{j,p} \doteq [\eta_{(m,1)}^{j,p}, \ldots, \eta_{(m,N_c^m)}^{j,p}]^\top$ for $\eta \in \{\lambda, z, q\}$. Next, we replace line 2 of constraint (5h) with:

$$
\sum_{m=1}^{N_c} \left[(\boldsymbol{\lambda}_{m,\neg k}^{j,p} \odot \mathbf{q}_m^{j,p})^\top \nabla_p(\mathbf{b}_n(\theta) - \mathbf{A}_m^\top p) \right] \doteq \sum_{m=1}^{N_c} \mathbf{q}_m^{j,p\top} \mathbf{L}_m^{j,p} \tag{15}
$$

for all $p \in \xi_j^{\text{loc}}$, where the (i,n)-th entry of $\mathbf{L}_m^{j,p} \in \mathbb{R}^{N_c^m \times |p|}$, $\mathbf{L}_m^{j,p}(i,n)$, refers to $\lambda_{(m,n),\neg k}^{j,p}\nabla_{p(i)}(b_{m,n}(\theta) - a_{m,n}^\top p)$. Note that λ (continuous variables) and q (binary variables) are bilinear (as $\nabla_p(b_{m,n}(\theta) - a_{m,n}^\top p)$ has no decision variables as θ does not multiply p). By assuming bounds $\underline{M} \leq \mathbf{L}_m^{j,p}(i,n) \leq \overline{M}$, this can be reformulated **exactly** in a linear fashion (i.e. *linearized*) [23] by replacing each bilinear product $\mathbf{q}_m^{j,p}(n)\mathbf{L}_m^{j,p}(i,n)$ in (15) with slack variables $\mathbf{R}_m^{j,p}(i,n)$ and adding constraints (where $\tilde{\mathbf{q}}_m^{j,p}(n) \doteq 1 - \mathbf{q}_m^{j,p}(n)$ for short):

$$
\begin{aligned}
\min(0, \underline{M}) &\leq \mathbf{R}_m^{j,p}(i,n) \leq \overline{M} \\
\underline{M}\mathbf{q}_m^{j,p}(n) &\leq \mathbf{R}_m^{j,p}(i,n) \leq \overline{M}\mathbf{q}_m^{j,p}(n) \\
\mathbf{L}_m^{j,p}(i,n) - \tilde{\mathbf{q}}_m^{j,p}(n)\overline{M} &\leq \mathbf{R}_m^{j,p}(i,n) \leq \mathbf{L}_m^{j,p}(i,n) - \tilde{\mathbf{q}}_m^{j,p}(n)\underline{M} \\
\mathbf{R}_m^{j,p}(i,n) &\leq \mathbf{L}_m^{j,p}(i,n) + \tilde{\mathbf{q}}_m^{j,p}(n)\overline{M}
\end{aligned} \tag{16}
$$

Finally, let $\mathbf{R}_m^j$ / $\mathbf{L}_m^j$ be the horizontal concatenation of $\mathbf{R}_m^{j,p}$ / $\mathbf{L}_m^{j,p}$, for all $p \in \xi_j^{\text{loc}}$. We can now pose the full problem:

Problem 4 (KKT inverse, unions):

find $\theta, \boldsymbol{\lambda}_k^{j,p}, \boldsymbol{\lambda}_{\neg k}^{j,p}, \boldsymbol{\nu}_k^j, \mathbf{R}_m^j, \mathbf{L}_m^j, \mathbf{q}_m^{j,p}, \mathbf{z}_{m,1}^{j,p}, \mathbf{z}_{m,2}^{j,p},$
$\quad \forall p \in \xi_j^{\text{loc}}, m = 1, ..., N_c, j = 1, ..., N_s$

s.t. -------- Primal feasibility --------
Equation (13), $j = 1, ..., N_s$
--- Lagrange mult. nonnegativity ---
$\boldsymbol{\lambda}_k^{j,p} \geq \mathbf{0}, \ \forall p \in \xi_j^{\text{loc}}, \ j = 1, \ldots, N_s$
$\boldsymbol{\lambda}_{m,\neg k}^{j,p} \geq \mathbf{0}, \ \forall p \in \xi_j^{\text{loc}}, \ m = 1, ..., N_c, \ j = 1, ..., N_s$ $\qquad$ (17)
------- Complementary slackness -----
Equation (14), $j = 1, \ldots, N_s$
----------- Stationarity -----------
$\nabla_{\xi_{xu}} c_\Pi(\xi_j^{\text{loc}}) + \boldsymbol{\lambda}_{\neg k}^{j\top}\nabla_{\xi_{xu}} \mathbf{g}_k(\xi_j^{\text{loc}}) + \sum_{m=1}^{N_c} \mathbf{1}_{N_c^m}^\top \mathbf{R}_m^j$
$\quad + \boldsymbol{\nu}_{\neg k}^{j\top}\nabla_{\xi_{xu}} \mathbf{h}_k(\xi_j^{\text{loc}}) = 0, \ j = 1, ..., N_s$
Equation (16), $\forall p \in \xi_j^{\text{loc}}, m = 1, ..., N_c,$
$\quad n = 1, ..., N_c^m, j = 1, ..., N_s$

C. Extraction of safe and unsafe states

Before moving onto affine parameterizations, we first detail how to check guaranteed safeness/unsafeness (as defined in (8)-(9)). One can check if a constraint state $p \in \mathcal{G}_s$ or $p \in \mathcal{G}_{\neg s}$ by adding a constraint $g(p, \theta) > 0$ or $g(p, \theta) \leq 0$ to Problem 2 and checking feasibility of the resulting program:

Problem 5 (Query if p is guaranteed safe OR unsafe):

$$
\begin{aligned}
&\text{find} && \theta, \boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j, \quad j = 1, \ldots, N_s \\
&\text{s.t.} && \{\mathrm{KKT}(\xi_j^{\mathrm{loc}})\}_{j=1}^{N_s}, \quad g(p, \theta) > 0 \text{ OR } g(p, \theta) \leq 0
\end{aligned}
\tag{18}
$$

If Problem 5 is infeasible, then $p \in \mathcal{G}_s$ or $p \in \mathcal{G}_{\neg s}$. Solving this problem is akin to querying an oracle about the safety of p. The oracle can return that p is guaranteed safe (program infeasible after forcing p to be unsafe), guaranteed unsafe (program infeasible after forcing p to be safe), or unsure (program is feasible despite forcing p to be safe or unsafe).

Since the constraint space is continuous, it is not possible to check via enumeration if each $p \in \mathcal{G}_{\neg s}$ or $p \in \mathcal{G}_s$. To address this, we can check the neighborhood of a constraint state p_{query} for membership in $\mathcal{G}_{\neg s}$ by solving the following:

Problem 6 (Volume extraction):

$$
\begin{aligned}
&\underset{\varepsilon, p_{\mathrm{near}}, \theta, \boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j}{\text{minimize}} && \varepsilon \\
&\text{subject to} && \{\mathrm{KKT}(\xi_j^{\mathrm{loc}})\}_{j=1}^{N_s} \\
& && \|p_{\mathrm{near}} - p_{\mathrm{query}}\|_\infty \leq \varepsilon \\
& && g(p_{\mathrm{near}}, \theta) > 0
\end{aligned}
\tag{19}
$$

Intuitively, Problem 6 finds the largest box centered at p_{query} contained within $\mathcal{G}_s$. An analogous problem can also be posed to recover the largest hypercube centered at p_{query} contained within $\mathcal{G}_{\neg s}$. For some common parameterizations (axis-aligned hyper-rectangles, convex sets), subsets of $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ can be even more efficiently recovered by performing line searches or taking convex hulls of guaranteed safe/unsafe states, details of which are in Appendix B of [6]. Volumes of safe/unsafe space can thus be produced by repeatedly solving Problem 6 for different p_{query}.

D. KKT relaxation for unions of affine constraints

Now, consider when Problem 1 involves avoiding an unsafe set $\mathcal{A}(\theta)$ described by a union of affine constraints:

$$
\mathcal{A}(\theta) = \bigcup_{i=1}^{N_c} \{p \mid g_{i,\neg k}(p, \theta) > 0\}
\tag{20}
$$

where $g_{i,\neg k}(p, \theta)$ is an affine function of θ for fixed p. Unlike in Sec. IV-B, formulating the recovery problem like Problem 4 yields trilinearity in the stationarity condition between continuous variables θ and λ and binary variables q, since for the affine case, $\nabla_{\xi_{xu}} g_{i,\neg k}(\cdot, \theta)$ remains a function of θ. As the product of two continuous decision variables cannot be linearized exactly, one must solve a MINLP to recover θ in this case, which can be inefficient. However, a relaxation which enables querying of guaranteed safeness/unsafeness via Problem 5 can be formulated as a MILP. For legibility, we present the $N_c = 1$ case, where there is only one affine constraint (and hence the binary variables q seen in Problem 4 are all set to 1 and can thus be dropped). Each bilinear term $\lambda_{1,\neg k}^{j,p} \nabla_p g_{1,\neg k}(p, \theta)$ is replaced with $l_1^{j,p} z_{1,1}^{j,p}$, where $l_1^{j,p}$

is a variable which represents the bilinear term and $z_{1,1}^{j,p}$ is an indicator variable encoding that if $z_{1,1}^{j,p}$ is 0, then $\lambda_{1,\neg k}^{j,p}$ must be 0. Hence, by linearizing the bilinear term as such, there is no relaxation gap when the Lagrange multipliers are zero; the only loss is when the Lagrange multipliers are non-zero (i.e. when the demonstration touches the constraint boundaries). In this case, coupling between λ and θ is lost by introducing the $l_1^{j,p}$ variables. We further linearize $l_1^{j,p} z_{1,1}^{j,p}$ (product of continuous, binary variables) with the same procedure in Sec. IV-B by again introducing slack variables $r_1^{j,p}$ and constraining them accordingly with (16), where the $q_{m,n}^{j,p}$ are replaced with $z_{1,1}^{j,p}$. Putting things together, we can write the following relaxed constraint recovery problem for $N_c = 1$:

Problem 7 (KKT relaxation, affine):

find $\theta, \boldsymbol{\lambda}_k^{j,p}, \boldsymbol{\lambda}_{\neg k}^{j,p}, \boldsymbol{\nu}_k^j, \mathbf{r}_1^j, \boldsymbol{\ell}_1^j, z_{1,1}^{j,p}, z_{1,2}^{j,p}, \forall p \in \xi_j^{\mathrm{loc}}, \ j = 1, \ldots, N_s$

s.t.
```
-------- Primal feasibility --------
```
$g_{1,\neg k}(p, \theta) \leq 0, \ \forall p \in \xi_j^{\mathrm{loc}}, \ j = 1, \ldots, N_s$
```
--- Lagrange mult. nonnegativity ---
```
$\boldsymbol{\lambda}_k^{j,p} \geq \mathbf{0}, \ \forall p \in \xi_j^{\mathrm{loc}}, \ j = 1, \ldots, N_s$
$\lambda_{1,\neg k}^{j,p} \geq 0, \ \forall p \in \xi_j^{\mathrm{loc}}, \ j = 1, \ldots, N_s$
```
------- Complementary slackness -----
```
$$
\begin{bmatrix} \lambda_{1,\neg k}^{j,p} \\ -g_{1,\neg k}(p, \theta) \end{bmatrix} \leq M \begin{bmatrix} z_{1,1}^{j,p} \\ z_{1,2}^{j,p} \end{bmatrix}, \ z_{1,1}^{j,p} + z_{1,2}^{j,p} \leq 1,
\tag{21}
$$

$\forall p \in \xi_j^{\mathrm{loc}}, j = 1, \ldots, N_s$
```
---------- Stationarity ----------
```
$\nabla_{\xi_{xu}} c_{\Pi}(\xi_j^{\mathrm{loc}}) + \boldsymbol{\lambda}_k^j \nabla_{\xi_{xu}} \mathbf{g}_k(\xi_j^{\mathrm{loc}}) + \mathbf{r}_1^j$
$\quad + \boldsymbol{\nu}_k^j \nabla_{\xi_{xu}} \mathbf{h}_k(\xi_j^{\mathrm{loc}}) = 0, \ j = 1, \ldots, N_s$
$\min(0, \underline{M})\mathbf{1} \leq \mathbf{r}_1^j \leq \overline{M}\mathbf{1}, \quad \underline{M}\mathbf{z}_{1,1}^j \leq \mathbf{r}_1^j \leq \overline{M}\mathbf{z}_{1,1}^j,$
$\boldsymbol{\ell}_1^j - (\mathbf{1} - \mathbf{z}_{1,1}^j)\overline{M} \leq \mathbf{r}_1^j \leq \boldsymbol{\ell}_1^j - (\mathbf{1} - \mathbf{z}_{1,1}^j)\underline{M},$
$\mathbf{r}_1^j \leq \boldsymbol{\ell}_1^j + (\mathbf{1} - \mathbf{z}_{1,1}^j)\overline{M}, \quad j = 1, \ldots, N_s$

where $\mathbf{r}_1^j, \mathbf{z}_{1,1}^j, \boldsymbol{\ell}_1^j$ denote horizontal concatenation of $r_1^{j,p}$, $z_{1,1}^{j,p}, l_1^{j,p}$ for all $p \in \xi_j^{\mathrm{loc}}$. The case if the constraint is a union of affine constraints yields quadrilinearity and can be handled similarly, requiring one extra step to linearize the products of binary variables $q_m^{j,p}$ and $z_{1,1}^{j,p}$, which can be done exactly.

While Problem 7 cannot recover the constraint parameter θ directly, one can still check if a constraint state is guaranteed safe/unsafe using Problem 5 (see Theorem 2 for reasoning).

E. Unknown constraint parameterization

In many applications, we may not know a constraint parameterization a priori. However, complex unsafe/safe sets can often be approximated as the union of many simple unsafe/safe sets. Thus, we adapt the method in [6] for incrementally growing a parameterization based on the complexity of the provided demonstrations. More precisely, suppose the true parameterization $g(p, \theta)$ of the unsafe set $\mathcal{A}(\theta) = \{p \mid g(p, \theta) > 0\}$ is unknown but can be exactly/approximately written as the union of N^* simple sets $\mathcal{A}(\theta) \approxeq \bigcup_{i=1}^{N^*} \{p \mid g_s(p, \theta_i) > 0\} \doteq \bigcup_{i=1}^{N^*} \mathcal{A}(\theta_i)$. Each simple set $\mathcal{A}(\theta_i)$ has a known parameterization $g_s(\cdot, \cdot)$ but N^*, the minimum number of simple sets needed to reconstruct $\mathcal{A}$, is unknown. We can estimate a lower bound on N^*, $\underline{N}$, by incrementally adding simple sets until Problem 2 is feasible (i.e. there exists a sufficiently complex constraint which can satisfy the demonstrations' KKT conditions). Issues with conservativeness of the recovered constraint when $\underline{N} < N^*$ are discussed in [6] and also hold here, which we omit for brevity.

F. Handling cost function uncertainty

We now extend the KKT conditions presented in (5) and Problems 2 and 3 to learn constraints with parametric uncertainty in the cost function (i.e. if γ in Problem 1 is unknown). To address this, the first term in the stationarity condition (5h) must be changed to $\nabla_{\xi_{xu}} c_{\Pi}(\xi_j^{\text{loc}}, \gamma)$. Then, if $c_{\Pi}(\cdot, \gamma)$ is affine in γ, γ can be found using a MILP.

Querying/volume extraction holds just as before; the only difference is that γ is now a decision variable in Problem 5/6. Note we are extracting constraint states that are guaranteed safe/unsafe for all possible cost parameters; that is, we are extracting safe/unsafe sets that are robust to cost uncertainty.

We summarize what we can solve for when using various parameterizations. For the exact cases, we can solve for θ and γ, but when relaxing, we can only solve for $\mathcal{S}$ and $\mathcal{A}$ via queries. Note the constraint/cost can be nonlinear in p without inducing relaxation, though it precludes usage of Problem 6 (as p is a decision variable in the latter, but not the former):

Constraint param.	Cost param.	Recover θ, γ?	$\mathcal{G}_s/\mathcal{G}_{\neg s}$
θ: form of (10); p: affine	γ: affine; p: nonlin.	Yes: Prob. 4	Prob. 5/6
θ: form of (10); p: nonlin.	γ: affine; p: nonlin.	Yes: Prob. 4	Prob. 5
θ: form of (20); p: nonlin.	γ: affine; p: nonlin.	No	Prob. 5

This only describes what we can solve for; the actual accuracy of the recovered θ and γ and the size of the recovered $\mathcal{G}_s$ / $\mathcal{G}_{\neg s}$ depends on how informative the demonstrations are, i.e. the demonstrations should interact with the constraint.

G. Applications to safe planning

As learned constraints can be reused for novel tasks with the same safety requirements, we end this section by describing how our method can be used within a planner to guarantee the safety of trajectories planned for such tasks. Recall that Problems 5 and 6 can be used to query if a constraint state p or a region around p is guaranteed safe/unsafe. The planner can use this information by either:

- Extracting an *explicit* representation of the constraint by repeatedly solving Problem 6 for different p_{query} to cover $\mathcal{S}$ and $\mathcal{A}$. Denote these extracted sets as $\hat{\mathcal{S}} \subseteq \mathcal{S}$ and $\hat{\mathcal{A}} \subseteq \mathcal{A}$ (the conservativeness of our method is proved in Sec. V-A). Then, $\hat{\mathcal{S}}$ can be passed to a planner and quickly used for constraint/collision checking via set-containment checks. A planned trajectory is *guaranteed safe* if each state on it lies in $\hat{\mathcal{S}}$, since $\hat{\mathcal{S}}$ is contained in true safe set $\mathcal{S}$. If $\hat{\mathcal{S}}$ is small and the planner cannot find a feasible trajectory, we can at least guarantee that a trajectory is *not definitely unsafe* it it does not intersect with $\hat{\mathcal{A}}$, as $\hat{\mathcal{A}}$ is contained in the true unsafe set $\mathcal{A}$.
- Extracting an *implicit* representation of the constraint by solving Problem 5 as needed by the planner. This may be less computationally efficient than the explicit case, but we demonstrate in Sec. VI-C that we still achieve reasonable planning times for a 7-DOF arm.

V. THEORETICAL ANALYSIS

In this section, we prove that our method provides a conservative estimate of the guaranteed learned safe/unsafe sets $\mathcal{G}_s, \mathcal{G}_{\neg s}$ (Sec. V-A) and prove learnability results using locally-optimal demonstrations (Sec. V-B).

A. Conservativeness

Definition 1 (Implied unsafe/safe set): For some set $\mathcal{B} \subseteq \Theta$, let $I_{\neg s}(\mathcal{B}) \doteq \bigcap_{\theta \in \mathcal{B}} \{x \mid g(x, \theta) > 0\}$ be the set of states implied unsafe by restricting the parameter set to $\mathcal{B}$, i.e. $I_{\neg s}(\mathcal{B})$ is the set of states that all $\theta \in \mathcal{B}$ mark as unsafe. Similarly, let $I_s(\mathcal{B}) \doteq \bigcap_{\theta \in \mathcal{B}} \{x \mid g(x, \theta) \leq 0\}$ be the set of states implied safe by restricting the parameter set to $\mathcal{B}$.

We further introduce the following lemma:

Lemma 1 (Lemma C.1 in [6]): Suppose $\mathcal{B} \subseteq \hat{\mathcal{B}}$, for some other set $\hat{\mathcal{B}}$. Then, $I_{\neg s}(\hat{\mathcal{B}}) \subseteq I_{\neg s}(\mathcal{B})$ and $I_s(\hat{\mathcal{B}}) \subseteq I_s(\mathcal{B})$.

Theorem 1 (Conservativeness of Problem 2): Suppose the constraint parameterization $g(x, \theta)$ is known exactly. Then, extracting $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ (as defined in (8) and (9), respectively) from the feasible set of Problem 2 projected onto Θ (denoted as $\mathcal{F}$) returns $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$.

Proof 1: We first prove that $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$. Suppose that there exists $x \in \mathcal{G}_{\neg s}$ such that $x \notin \mathcal{A}$. Then by definition, for all $\theta \in \mathcal{F}$, $g(x, \theta) > 0$. However, we know that all locally-optimal demonstrations satisfy the KKT conditions with respect to the true parameter θ^*; hence, $\theta^* \in \mathcal{F}$. Then, $x \in \mathcal{A}(\theta^*)$. Contradiction. Similar logic holds for proving that $\mathcal{G}_s \subseteq \mathcal{S}$. Suppose that there exists $x \in \mathcal{G}_s$ such that $x \notin \mathcal{S}$. Then by definition, for all $\theta \in \mathcal{F}$, $g(x, \theta) \leq 0$. However, we know that all locally-optimal demonstrations satisfy the KKT conditions with respect to the true parameter θ^*; hence, $\theta^* \in \mathcal{F}$. Then, $x \in \mathcal{S}(\theta^*)$. Contradiction.

Remark 1: Unfortunately, it is difficult to guarantee conservativeness when using suboptimal demonstrations (solving Problem 3), as the relationship between cost suboptimality and KKT violation is generally unknown. However, we note in practice that the $\mathcal{G}_s, \mathcal{G}_{\neg s}$ recovered using suboptimal demonstrations still tend to be conservative (see Sec. VI-B).

Theorem 2 (Conservativeness of Problem 7): Suppose the constraint parameterization $g(x, \theta)$ is known exactly. Then, extracting $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ (as defined in (8) and (9), respectively) from the feasible set of Problem 7 (denoted as $\mathcal{F}$) returns $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$.

Proof 2: Denote the Θ-projected feasible set of the original unrelaxed problem (i.e. variables r_i are not introduced and the bilinear terms between θ and λ remain) as $\mathcal{F}_{\text{MINLP}}$ and the Θ-projected feasible set of Problem 7 as $\mathcal{F}$. Using the logic in Theorem 1, extracting $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ from $\mathcal{F}_{\text{MINLP}}$ yields $\mathcal{G}_s \subseteq \mathcal{S}$ and $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$ (since $\mathcal{F}_{\text{MINLP}}$ is the true feasible set, like assumed in Theorem 1). Furthermore, $\mathcal{F}_{\text{MINLP}} \subseteq \mathcal{F}$, since relaxing the bilinear terms to linear terms in Problem 7 expands the feasible set compared to the unrelaxed problem. By definition, $\mathcal{G}_s = I_s(\mathcal{F}_{\text{MINLP}})$ and $\mathcal{G}_{\neg s} = I_{\neg s}(\mathcal{F}_{\text{MINLP}})$, and via Lemma 1, $I_s(\mathcal{F}) \subseteq I_s(\mathcal{F}_{\text{MINLP}})$ and $I_{\neg s}(\mathcal{F}) \subseteq I_{\neg s}(\mathcal{F}_{\text{MINLP}})$. Hence, $I_s(\mathcal{F}) \subseteq \mathcal{S}$ and $I_{\neg s}(\mathcal{F}) \subseteq \mathcal{A}$.

Remark 2: For brevity, we omit conditions on $M, \underline{M}, \overline{M}$ for conservativeness; it is well-known that this is achieved by choosing the big-M constants to be sufficiently large [22].

B. Global vs local learnability

Definition 2 (Local learnability): A state $x \in \mathcal{A}$ is locally learnable if there exists **any** set of N_s locally-optimal demonstrations, where N_s may be infinite, such that $x \in \mathcal{I}_{\neg s}(\mathcal{F})$, where $\mathcal{F}$ is the Θ-projected feasible set of Problem 2. We also

define the *locally learnable* set of unsafe states $\mathcal{G}_{\neg s}^{\text{loc},*}$ as the union of all locally learnable states.

Definition 3 (Global learnability): A state $x \in \mathcal{A}$ is globally learnable if there exists **any** set of N_s globally-optimal demonstrations and $N_{\neg s}$ sampled strictly lower-cost (and hence unsafe) trajectories, where N_s and $N_{\neg s}$ may be infinite, such that $x \in \mathcal{I}(\mathcal{F}_{\text{glo}})$, where $\mathcal{F}_{\text{glo}}$ is the feasible set of Problem 2 in [6] (which recovers a constraint consistent with the demonstrations and sampled unsafe trajectories). Accordingly, we define the *globally learnable* set of unsafe states $\mathcal{G}_{\neg s}^{\text{glo},*}$ as the union of all globally learnable states.

Note that a safe state $x_s \in \mathcal{S}$ can always be learned guaranteed safe, as there always exists a safe globally-optimal or locally-optimal demonstration passing through x_s. Armed with these definitions, we show the following:

Theorem 3 (Global vs local): Suppose the initial constraint parameter set Θ is identical for both the local and global problems. Then, $\mathcal{G}_{\neg s}^{\text{loc},*} \subseteq \mathcal{G}_{\neg s}^{\text{glo},*}$.

Proof 3: Any globally-optimal demonstration must also satisfy the KKT conditions, as it is also locally-optimal. Further conditions (in the form of lower-cost trajectories being infeasible) must be imposed on a constraint parameter for it to be globally-optimal. Hence, $\mathcal{F}_{\text{glo}} \subseteq \mathcal{F}$. By Lemma 1, $\mathcal{I}(\mathcal{F}) \subseteq \mathcal{I}(\mathcal{F}_{\text{glo}})$, and thus $\mathcal{G}_{\neg s}^{\text{loc},*} \subseteq \mathcal{G}_{\neg s}^{\text{glo},*}$.

Note that Theorem 3 holds in the limit of having sampled **all** unsafe trajectories. In practice, the sampling is nowhere near complete, especially for nonlinear dynamics. We see in these cases (Sec. VI-C) that our KKT-based method learns more compared to sampling-based techniques. Finally, we note that cost function uncertainty can only decrease learnability, as it enlarges the feasible set of Problem 2.

VI. RESULTS

We show our method, first on 2D examples (Sec. VI-A) for intuition, and then on high-dimensional 7-DOF arm (Sec. VI-B) and quadrotor (Sec. VI-C) constraint-learning problems (see the accompanying video for experiment visualizations). All computation times are recorded on a laptop with a 3.1 GHz Intel Core i7 processor and 16 GB RAM.

A. 2D examples

<u>*Global vs. local:*</u> Assuming global demonstration optimality can enlarge $\mathcal{G}_s/\mathcal{G}_{\neg s}$ compared to assuming local optimality (Theorem 3). In this example, we show some common differences in the learned constraints when assuming global/local optimality. Consider a 2D kinematic system $\chi_{t+1} = \chi_t + u_t$, $\chi = [x, y]^\top$, $\|u_t\| \leq 1$ avoiding the pink obstacle in Fig. 1. We use an axis-aligned box constraint parameterization. In Fig. 1 (left), by assuming the demonstrations (cyan, green) are globally-optimal and sampling lower-cost trajectories (the middle state on each trajectory is plotted in red), the hatched area is implied guaranteed unsafe, as any axis-aligned box containing the sampled unsafe states (in red) must also contain the hatched area. In contrast, assuming local optimality gives us zero volume learned guaranteed safe/unsafe, as a measure-zero horizontal line obstacle (orange dashed) can make the demonstrations locally-optimal: as the line supports the middle state on each demonstration, the cost cannot be locally
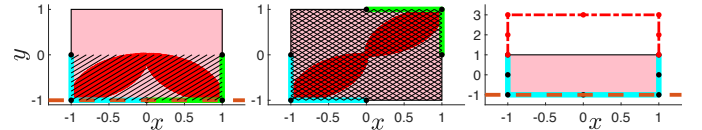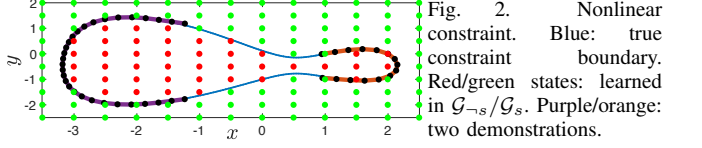

Fig. 1. Left: local learns less than global. Center: local learns the same as global. Right: global recovers non-conservative solution. Red: sampled unsafe trajectories. Pink: true constraint. Green/cyan: demonstrations.

Fig. 2. Nonlinear constraint. Blue: true constraint boundary. Red/green states: learned in $\mathcal{G}_{\neg s}/\mathcal{G}_s$. Purple/orange: two demonstrations.

improved. In Fig. 1, center, we show a case where there is no gap in learnability: without assuming a parameterization, the demonstrations can be explained by two horizontal line obstacles, but together with the box parameterization, we recover $\mathcal{G}_s = \mathcal{S}$ and $\mathcal{G}_{\neg s} = \mathcal{A}$. Fig. 1 (right) shows that assuming global optimality may result in non-conservative constraint recovery (e.g. if the dotted red line were a sampled unsafe trajectory), while a horizontal line obstacle (orange dashed line) can explain local optimality of the demonstration, yielding conservative constraint recovery.

<u>*Effects of cost uncertainty:*</u> We show that learnability under cost uncertainty is more related to the possible behaviors that a cost uncertainty set can represent, rather than the actual size of the cost parameter space. For the demonstrations/constraint in the center plot of Fig. 1, consider the following cost uncertainty sets: A) $c(\xi) = \sum_{t=1}^{T-1} \gamma_1(x_{t+1} - x_t)^2 + \gamma_2(y_{t+1} - y_t)^2$, where $\gamma_i \in [-5, 5], i = 1, 2$ and B) $c(\xi) = \sum_{k=1}^{10} \sum_{t=1}^{T-1} \left(\gamma_{1,k}(x_{t+1} - x_t)^{2k} + \gamma_{2,k}(y_{t+1} - y_t)^{2k} \right)$, where $\gamma_{i,k} \in [0.001, 5]$ for all i, k. While Set A has a much smaller parameter space compared to Set B (2 vs 20 parameters), allowing γ_1, γ_2 to take negative values enables the case where the demonstrator wants to maximize path length (i.e. set $\gamma_1 = \gamma_2 = -1$). For fixed start/goal states and control constraints, the observed demonstrations are actually locally-optimal with respect to a cost function which maximizes path length in an environment with no box state space constraint. Hence, for Set A, our method returns $\mathcal{G}_s = \mathcal{G}_{\neg s} = \emptyset$. In contrast, while Set B has a much larger parameter space, the range of allowable behaviors is small (all cost terms must penalize path length). Thus, despite the large cost parameter space, $\mathcal{G}_s = \mathcal{S}$ and $\mathcal{G}_{\neg s} = \mathcal{A}$.

<u>*Nonlinear constraint:*</u> We emphasize that while our method requires an affine parameterization in the *constraint parameters*, constraints that are nonlinear in the *constraint state* can still be learned. Consider a parameterization $g_{1,\neg k}(p, \theta) = 2(x^4 + y^4) - 5(x^3 + y^3) + 5(x - 1)^3 + 5(y + 1)^3 - \theta$, which yields a highly nonlinear state space constraint. With two demonstrations for $\theta = 2$ (see Fig. 2), $\mathcal{G}_s = \mathcal{S}$ and $\mathcal{G}_{\neg s} = \mathcal{A}$.

B. 7-DOF arm

<u>*Robot bartender:*</u> Consider a 7-DOF Kuka iiwa robot bartender which must deliver a drink from the bar cabinet (Fig. 3, brown box) or from another bartender to a customer at the counter (Fig. 3, gray box). To do this, the arm must satisfy an end-effector pose constraint to avoid spilling the drink, and the swept volume of the arm must not collide with the bar furniture while satisfying proxemics constraints (Fig. 3, green
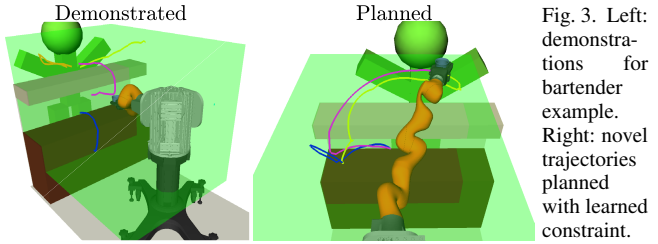
Fig. 3. Left: demonstrations for bartender example. Right: novel trajectories planned with learned constraint.
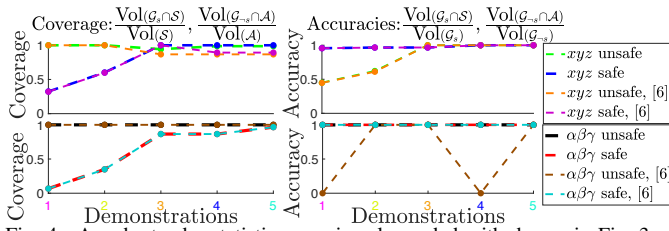

Fig. 4. Arm bartender statistics; x-axis color-coded with demos in Fig. 3.

box) with respect to the customer. We use a kinematic model of the arm, $j_{t+1}^i = j_t^i + u_t^i, i = 1, ..., 7$, where $\|u_t^i\|_2^2 \leq 0.8$ for all t. Five suboptimal human demonstrations, optimizing joint-space path length $c(\xi) = \sum_{i=1}^{T-1} \|j_{t+1} - j_t\|_2^2$, are captured in a virtual reality environment using an HTC Vive. The proxemics constraints encoded in the demonstrations (Fig. 3, left) disallow the arm from getting too close to the customer and from making large sweeping motions from the left, right, and particularly the top, as the customer can perceive such motions as aggressive. We aim to learn these 15 constraint parameters: $[\underline{x}^{\text{ctr}}, \underline{z}^{\text{ctr}}, \overline{z}^{\text{ctr}}]$ (unknown extents of the bar top), $[\underline{x}^{\text{cab}}, \overline{z}^{\text{cab}}]$ (unknown extents of the bar cabinet), $[\underline{\alpha}, \overline{\alpha}, \underline{\beta}, \overline{\beta}, \underline{\gamma}, \overline{\gamma}]$ (unknown pose constraint), and $[\overline{x}^{\text{prox}}, \underline{y}^{\text{prox}}, \overline{y}^{\text{prox}}, \overline{z}^{\text{prox}}]$ (box proxemics constraint).

The constraint parameters are recovered using the suboptimal analogue of Problem 4 (i.e. using the objective function of Problem 3), taking 17.2 seconds to solve when using all demonstrations. For tractability, we approximate the swept volume constraint by sampling 18 points on the volume of the arm, mapping them through the arm's forward kinematics, and ensuring that the resulting points are consistent with the obstacle avoidance/proxemics constraint parameters. We solve Problems 5/6 to extract the learned guaranteed safe/unsafe sets $\mathcal{G}_s/\mathcal{G}_{\neg s}$, where each query takes 16.4/12.1 seconds on average over 10 queries. Fig. 4 shows the coverage of $\mathcal{G}_s/\mathcal{G}_{\neg s}$
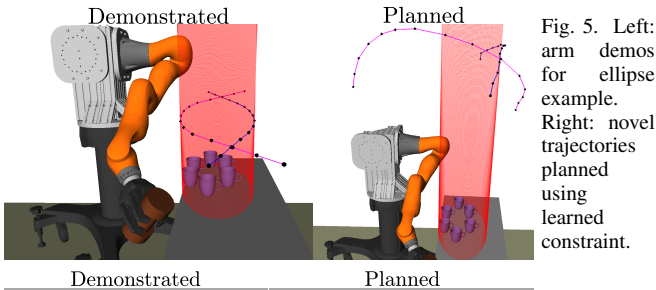

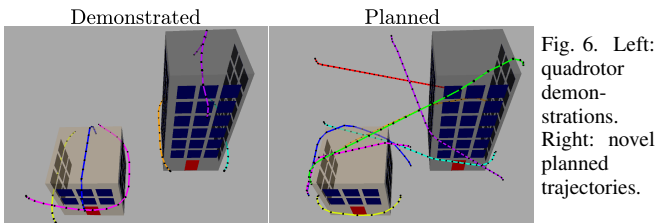Fig. 5. Left: arm demos for ellipse example. Right: novel trajectories planned using learned constraint.

Fig. 6. Left: quadrotor demonstrations. Right: novel planned trajectories.

compared to the true safe/unsafe sets $\mathcal{S}/\mathcal{A}$, as well as the accuracy of the claimed safe/unsafe sets (Fig. 4). We use the sampling-based approach described in [6] with Problem 3 of [6] as a baseline. We note that [6] will have difficulty with this example, since the swept volume constraint scales the number of decision variables in Problem 3 of [6] by a factor of 18; this limits the number of trajectory samples which can be tractably used in the constraint-recovery problem. Despite this, the pose constraint is learned fairly well by both the baseline and our method, though the baseline experiences accuracy dips due to suboptimality causing some safe lower-cost trajectories. However, the baseline performs poorly on the position constraints, as it does not learn that the bar top or the bar cabinet are unsafe and does not fully learn the safe set, due to insufficient trajectory samples. In contrast, our KKT-based approach recovers $\mathcal{G}_s = \mathcal{S}$ and $\mathcal{G}_{\neg s} = \mathcal{A}$. Finally, we extract volumes of guaranteed safe space using the procedure in Sec. IV-C and provide the extracted constraint to the CBiRRT planner [24], generating the novel safe trajectories in Fig. 3 (right). This experiment suggests that when Problem 1 has many constraints (in this case, due to the swept volume), sampling trajectories leads to worse scalability and constraint-recovery performance compared to our KKT-based approach. *Elliptical end-effector constraint:* This example is meant to demonstrate the efficacy of using Problem 5 in the planning loop. Suppose the arm is manipulating a heavy object near some glassware. For safety, the end effector's center of mass is constrained to lie outside an elliptical cylinder containing the glassware: $\chi_t^\top A(\theta)\chi_t - 2b(\theta)^\top \chi_t + c(\theta) > 0$, where $\chi_t = [x_t, y_t]^\top, A(\theta) = \text{diag}([0.5, 2]), b(\theta) = [0, 1.1]^\top$, and $c(\theta) = 0.505$. We modify Problem 5 to use the affine-relaxed KKT conditions, and solving this problem using two demonstrations (Fig. 5, left) is enough to recover $\mathcal{G}_s = \mathcal{S}, \mathcal{G}_{\neg s} = \mathcal{A}$ via queries. To plan novel constraint-satisfying trajectories, we use STOMP [25], where the usual collision/constraint checker is replaced with Problem 5. We show two planned trajectories (Fig. 5, right), where the planning times were 2 and 6 minutes. Averaged over 10 different queries, solving Problem 5 takes 0.073 seconds. We note that this can be sped up by warm-starting Problem 5 with the results of previous queries (since like many trajectory optimizers, STOMP samples points near previous iterates).

C. Quadrotor

We consider the scenario of a quadrotor carrying a delicate payload in an urban environment (see Fig. 6). Accordingly, the quadrotor is constrained to not collide with surrounding buildings (i.e. $(x, y, z) \notin ([\underline{x}_1, \overline{x}_1] \times [\underline{y}_1, \overline{y}_1] \times [0, \overline{z}_1]) \vee ([\underline{x}_2, \overline{x}_2] \times [\underline{y}_2, \overline{y}_2] \times [0, \overline{z}_2]))$, satisfy control constraints $\|u_t\| \leq \overline{U}$, pose constraints $\alpha \in [\underline{\alpha}, \overline{\alpha}], \beta \in [\underline{\beta}, \overline{\beta}], \gamma \in [\underline{\gamma}, \overline{\gamma}]$, and angular velocity constraints $\dot{\alpha} \in [\underline{\dot{\alpha}}, \overline{\dot{\alpha}}], \dot{\beta} \in [\underline{\dot{\beta}}, \overline{\dot{\beta}}], \dot{\gamma} \in [\underline{\dot{\gamma}}, \overline{\dot{\gamma}}]$. In this problem, we aim to recover all of these constraints (23 unknown constraint parameters total) using the six demonstrations in Fig. 6 (left) by solving Problem 4, which takes 19.4 seconds when using all demonstrations. We start from a single box parameterization for each constraint and detect from infeasibility that another box should be added to the position constraint parameterization (see Sec. IV-E).
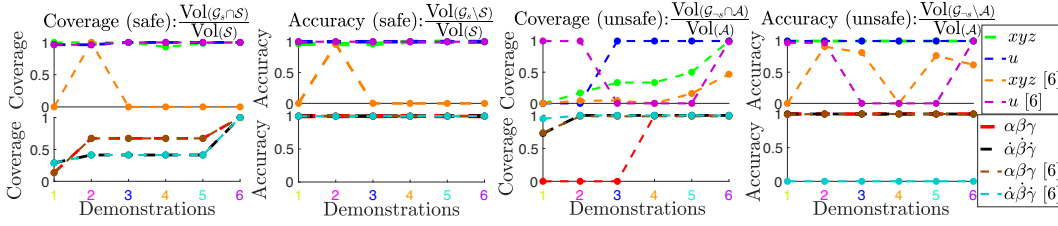
Fig. 7. Quadrotor statistics: coverage and accuracy for $\mathcal{G}_s$, $\mathcal{G}_{\neg s}$. Demonstration axis is color coded with the demonstrations shown on the left in Fig. 6.

The demonstrations are synthetically generated by solving trajectory optimization problems for the cost function $c(\xi) = \sum_{r \in R} \sum_{t=1}^{T-1} \gamma_r (r_{t+1} - r_t)^2$, where $R = \{x, y, z, \dot{\alpha}, \dot{\beta}, \dot{\gamma}\}$ and $\gamma_r = 1$. Our algorithm assumes parametric cost uncertainty of $\gamma_r \in [0.01, 3]$, and we assume the cost function is known exactly for the baseline [6]. This problem is especially challenging for the baseline, since having many unknown constraint parameters can lead to non-identifiability of the constraint from the sampled trajectories. Furthermore, the nonlinearity of the quadrotor dynamics (we use the dynamics in [6]) makes sampling difficult. We compute $\mathcal{G}_s/\mathcal{G}_{\neg s}$ via Problems 5/6, taking 26.6/43.0 seconds on average over 10 different queries. Fig. 7 compares the coverage and accuracy of $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ between our approach and the baseline [6] for each of the constraint spaces (position, pose, velocity, and control). The baseline and our approach perform comparably for some of the "simpler" convex constraints (e.g. the angular velocity/control safe sets). However, the baseline struggles to learn the unsafe sets (due to the simultaneous identification of so many constraints from poor trajectory samples) and position constraints (as the quadrotor has second order dynamics, it is difficult to sample combinations of trajectories which uniquely imply a single state is unsafe). We also note that the baseline accuracies fluctuate greatly due to imperfect trajectory sampling and the difficulty of distinguishing between multiple constraints: different data may cause the optimization to switch unsafeness assignments from one constraint to another active constraint). By extracting volumes of $\mathcal{G}_s$ using the method in Sec. IV-C, we pass $\mathcal{G}_s$ to a trajectory optimizer [26] to generate novel safe trajectories (Fig. 6, right). This experiment suggests that by avoiding trajectory sampling, our KKT-based approach performs better on high-dimensional nonlinear systems.

VII. DISCUSSION AND CONCLUSION

We present an algorithm which uses the KKT optimality conditions to determine constraints that make observed demonstrations appear locally-optimal with respect to an uncertain cost function. As the KKT conditions are an *implicit* condition on the set of constraints that can possibly explain the demonstrations, we sidestep the shortcomings of previous methods [5], [6] which rely on sampling lower-cost trajectories as explicit certificates of unsafeness. In future work, we aim to address two shortcomings of our method: first, we require the dynamics to be known in closed form, while [5], [6] just need a simulator; second, the number of decision variables in our method scales linearly with the number of demonstrations, making it important that the demonstrations are informative with respect to the unknown constraint. To address these issues, we plan to extend our method to handle uncertain dynamics and develop an active learning method to obtain informative demonstrations.

REFERENCES

[1] N. D. Ratliff, J. A. Bagnell, and M. Zinkevich, "Maximum margin planning," in *ICML*, 2006, pp. 729–736.

[2] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.

[3] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *RAS*, vol. 57, pp. 469–483, 2009.

[4] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *ICML*, San Francisco, CA, USA, 2000, pp. 663–670.

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

[6] G. Chou, N. Ozay, and D. Berenson, "Learning parametric constraints in high dimensions from demonstrations," *3rd Conference on Robot Learning (CoRL)*, 2019. [Online]. Available: arxiv.org/abs/1910.03477

[7] A. Keshavarz, Y. Wang, and S. P. Boyd, "Imputing a convex objective function," in *ISIC*. IEEE, 2011, pp. 613–619.

[8] P. Englert, N. A. Vien, and M. Toussaint, "Inverse kkt: Learning cost functions of manipulation tasks from demonstrations," *IJRR*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[9] M. Johnson, N. Aghasadeghi, and T. Bretl, "Inverse optimal control for deterministic continuous-time nonlinear systems," in *CDC*, 2013.

[10] S. Singh, J. Lacotte, A. Majumdar, and M. Pavone, "Risk-sensitive inverse reinforcement learning via semi- and non-parametric methods," *IJRR*, vol. 37, no. 13-14, 2018.

[11] M. Menner, P. Worsnop, and M. N. Zeilinger, "Constrained inverse optimal control with application to a human manipulation task," *IEEE Transactions on Control Systems Technology*, 2019.

[12] L. Armesto, J. Bosga, V. Ivan, and S. Vijayakumar, "Efficient learning of constraints and generic null space policies," in *ICRA*, 2017.

[13] C. Pérez-D'Arpino and J. A. Shah, "C-LEARN: learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *ICRA*, 2017.

[14] C. Li and D. Berenson, "Learning object orientation constraints and guiding constraints for narrow passages from one demonstration," in *ISER*, 2016.

[15] N. Mehr, R. Horowitz, and A. D. Dragan, "Inferring and assisting with constraints in shared autonomy," in *CDC*, Dec 2016, pp. 6689–6696.

[16] A. L. Pais, K. Umezawa, Y. Nakamura, and A. Billard, "Learning robot skills through motion segmentation and constraints extraction," HRI, 2013.

[17] G. Ye and R. Alterovitz, "Demonstration-guided motion planning," in *ISRR*, 2011.

[18] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *HRI*, 2007, pp. 255–262.

[19] ——, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *IROS*, 2008.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[21] T. Tao, *Analysis II*, 3rd ed. Springer, 2016.

[22] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*, 1st ed. Athena Scientific, 1997.

[23] L. Liberti and C. C. Pantelides, "An exact reformulation algorithm for large nonconvex nlps involving bilinear terms," *J. Global Optimization*, vol. 36, no. 2, pp. 161–189, 2006.

[24] D. Berenson, S. S. Srinivasa, and J. J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *IJRR*, vol. 30, no. 12, pp. 1435–1460, 2011.

[25] M. Kalakrishnan, S. Chitta, E. A. Theodorou, P. Pastor, and S. Schaal, "STOMP: stochastic trajectory optimization for motion planning," in *ICRA*, 2011.

[26] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.