Gaussian Process Constraint Learning for Scalable Chance-Constrained Motion Planning from Demonstrations

Glen Chou*, Hao Wang*, and Dmitry Berenson

Abstract—We propose a method for learning constraints represented as Gaussian processes (GPs) from locally-optimal demonstrations. Our approach uses the Karush-Kuhn-Tucker (KKT) optimality conditions to determine where on the demonstrations the constraint is tight, and a scaling of the constraint gradient at those states. We then train a GP representation of the constraint which is consistent with and which generalizes this information. We further show that the GP uncertainty can be used within a kinodynamic RRT to plan probabilistically-safe trajectories, and that we can exploit the GP structure within the planner to exactly achieve a specified safety probability. We demonstrate our method can learn complex, nonlinear constraints demonstrated on a 5D nonholonomic car, a 12D quadrotor, and a 3-link planar arm, all while requiring minimal prior information on the constraint. Our results suggest the learned GP constraint is accurate, outperforming previous constraint learning methods that require more *a priori* knowledge.

Index Terms—Learning from Demonstration, Planning under Uncertainty, Machine Learning for Robot Control, Robot Safety

I. INTRODUCTION

THE need for robots that can safely perform tasks in unstructured environments has increased as robots are deployed in the real world. One popular paradigm for teaching robots tasks is learning from demonstration (LfD) [1], [2] via inverse optimal control (IOC), which assumes the demonstrator is solving an *unconstrained* optimization, and learns the underlying reward/cost function. However, hard constraints are crucial for safety-critical applications and are not well-enforced by these methods [3]. To address safety in LfD, recent work has represented tasks as *constrained* optimization problems, and learns the unknown cost function and constraints from demonstrations [4], [5], [6], [7], [8] via the Karush-Kuhn-Tucker (KKT) optimality conditions, enabling constraint learning for complex manipulation and mobile robotics tasks. However, these methods require that the unknown constraints can be described by an *a priori* known representation or parameterization (e.g. as a union

of axis-aligned boxes [4], [9]), restricting these methods to the learning of highly-structured constraints. Moreover, such representations can be highly inefficient (e.g. many boxes may be required to approximate complex constraints), leading to a computational burden that makes it challenging to scale these methods up to learn realistic constraints. Consider a demonstrator steering a quadrotor to avoid collisions with a tree (Fig. 1). On the one hand, we are unlikely to obtain an efficient constraint representation for learning trees *a priori* unless we can learn one (e.g. via deep learning) using an enormous number of demonstrations, and on the other hand, a prohibitive number of boxes is needed to represent the tree.
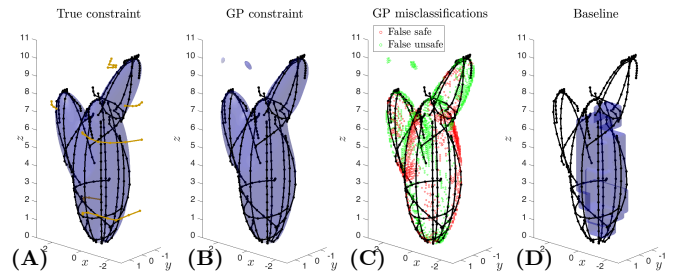

Fig. 1. Demonstrations (black) avoiding a tree-like obstacle on a 12D quadrotor. **(A)** True constraint (blue); plans using the GP constraint (gold). **(B)** Posterior mean of the GP constraint (blue). **(C)** Errors of GP posterior mean w.r.t. the true constraint. **(D)** Constraint learned via [4] using 6 boxes.

We address these issues via the insight that the demonstrations' Karush-Kuhn-Tucker (KKT) optimality conditions provide information on 1) where the unknown constraint is tight on the demonstrations, and 2) the gradient of the constraint (i.e. the surface normal on the constraint boundary) at those points. Crucially, we show that this information can be extracted in a way that is *agnostic* to the chosen constraint representation. This is in sharp contrast to prior work [4], which uses the KKT conditions to directly determine a set of constraint parameters, for a fixed constraint parameterization, which make the demonstrations satisfy their KKT conditions. This representation-agnostic constraint information can be incorporated into a flexible non-parametric Gaussian process (GP) function approximator, which enables constraint learning while requiring minimal *a priori* knowledge on the underlying constraint structure. Our contributions are:

- We show how to use the demonstrations' KKT conditions to extract information on the values and gradients of the unknown constraint, how to ensure it is robust to the ill-posedness of the constraint learning problem, and how it can be jointly incorporated into a GP.

- We show how the uncertainty of the learned GP constraint can be used to plan chance-constrained trajectories which satisfy the unknown constraint with a specified probability, and how the Gaussian structure of the uncertainty can be exploited in the planner to exactly compute trajectory safety probabilities.
- We evaluate our method on complex nonlinear constraint learning problems demonstrated on a 5D nonholonomic car, a 12D quadrotor and a three-link planar arm, showing that our method outperforms baselines.

II. RELATED WORK

Our method is related to prior work in IOC [8], [10], [11] that uses the KKT conditions to learn an unknown cost function, assuming the constraints are known. Other IOC methods [12], [13], [14] use flexible function approximators to learn unknown cost functions without using known features, again assuming known constraints. Our approach is complementary to these methods, as it seeks to learn the constraints.

Our work also builds upon constraint learning methods [4], [5], [6], [15], which often assume a known constraint parameterization to simplify the inverse problem which recovers the unknown constraint [4], [9]. When this assumption is removed [6], the unsafe set defined by the unknown constraint is assumed to be well-approximated by a finite union of simple unsafe sets, e.g. axis-aligned boxes [6]. However, the inverse problem scales exponentially with the number of simple sets, as each set adds binary decision variables to the optimization. This renders complex constraints infeasible to learn unless the true constraint representation is known, restricting previous methods (e.g. [4], [9]) to learn simple constraints, e.g. unions of a few boxes [4], [9], or to know the parameterization [5, Fig. 6] [4, Fig. 2]. Our work is also related to methods that plan using the uncertainty in the learned constraint, e.g. [9]. However [9] scales exponentially in the number of simple unsafe sets; in contrast, we use the uncertainty of the GP constraint to scale more gracefully.

Finally, our work relates to planning under uncertainty, where the uncertainty may arise from sensing [16], state estimation [17], motion [18], and the environment/obstacles; our work relates most closely to this final case. [19] plans with uncertain obstacles via chance-constrained optimization and requires polytopic obstacles and linear Gaussian dynamics. Under similar assumptions, [20] embeds chance constraints in a Rapidly-Exploring Random Tree (RRT) [21]. In contrast, we assume deterministic dynamics but can handle GP-representable constraints and nonlinear dynamics.

III. PRELIMINARIES AND PROBLEM STATEMENT

A. Demonstrator's problem and KKT optimality conditions

We represent a length T demonstration of a task σ performed on a deterministic discrete-time nonlinear system $x_{t+1} = f(x_t, u_t, t)$, $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, $u \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ as a constrained optimization over state/control trajectories $\xi_{xu} \doteq (\xi_x, \xi_u) \doteq [x_1, \ldots, x_T, u_1, \ldots, u_{T-1}]$:

Problem 1 (Forward (demonstrator's) problem / task σ).
$$\underset{\xi_{xu}}{\text{minimize}} \quad c_\sigma(\xi_{xu})$$
$$\text{subject to} \quad \phi(\xi_{xu}) \in \mathcal{S} \subseteq \mathcal{C} \Leftrightarrow \mathbf{g}^*_{\neg k}(\phi(\xi_{xu})) \leq \mathbf{0}$$
$$\bar{\phi}(\xi_{xu}) \in \bar{\mathcal{S}} \subseteq \bar{\mathcal{C}}, \quad \phi_\sigma(\xi_{xu}) \in \mathcal{S}_\sigma \subseteq \mathcal{C}_\sigma$$
$$\Leftrightarrow \quad \mathbf{h}_k(\xi_{xu}) = \mathbf{0}, \quad \mathbf{g}_k(\xi_{xu}) \leq \mathbf{0}$$

where $c_\sigma(\cdot)$ is a known, possibly task-dependent cost function, and $\phi(\cdot)$ maps state/control trajectories to *constraint states* κ in *constraint space* $\mathcal{C}$ (i.e. $\kappa \in \mathcal{C}$), where the constraint is evaluated. For example, for an obstacle constraint, $\phi(\cdot)$ would select the position components of the states. The safe set $\mathcal{S} \subseteq \mathcal{C} \subseteq \mathbb{R}^{n_c}$ is defined by the unknown inequality constraint $\mathbf{g}^*_{\neg k}(\phi(\xi_{xu})) \leq \mathbf{0}$ and is unknown to the learner. $\bar{\phi}(\cdot)$ and $\phi_\sigma(\cdot)$ map to spaces $\bar{\mathcal{C}}$ and $\mathcal{C}_\sigma$, containing a known task-shared safe set $\bar{\mathcal{S}}$ and task-dependent safe set $\mathcal{S}_\sigma$, defined by known equality and inequality constraints $\mathbf{h}_k(\xi_{xu}) = \mathbf{0}$, $\mathbf{g}_k(\xi_{xu}) \leq \mathbf{0}$. We embed the dynamics in $\bar{\mathcal{S}}$ and the start/goal constraints in $\mathcal{S}_\sigma$. We focus on unknown scalar, state-dependent, time-separable inequality constraints

$$\mathbf{g}^*_{\neg k}(\phi(\xi_{xu})) \leq \mathbf{0} \Leftrightarrow g^*_{\neg k}(\phi_{\text{sep}}(x_t)) \leq 0, \quad \forall t = 1, ..., T, \quad (1)$$

where $\phi_{\text{sep}} : \mathcal{X} \mapsto \mathcal{C}$ is the time-separable counterpart of $\phi(\cdot)$, $g^*_{\neg k} : \mathcal{C} \mapsto \mathbb{R}$, and $\kappa_t = \phi_{\text{sep}}(x_t)$. We note that extending to control-dependent constraints is straightforward. Moreover, we can learn the (un)safe set for an M-dimensional vector-valued constraint by learning $g^*_{\neg k}(\cdot) = \max_{i=1,...,M} g^*_{i,\neg k}(\cdot)$. We assume each demonstration ξ_{loc} solves Prob. 1 to local optimality, satisfying Prob. 1's KKT conditions [22]. With Lagrange multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\nu}$, the relevant KKT conditions for the jth demonstration ξ_{loc}^j, denoted KKT(ξ_{loc}^j), are:

Primal feasibility:	$\mathbf{g}^*_{\neg k}(\phi(\xi_{\text{loc}}^j)) \leq \mathbf{0},$	(2a)
Lagrange mult. nonnegativity:	$\boldsymbol{\lambda}_k^j \geq \mathbf{0}$	(2b)
	$\lambda_{\neg k}^{j,t} \geq 0, \ t = 1, ..., T^j \Leftrightarrow \boldsymbol{\lambda}_{\neg k}^j \geq \mathbf{0}$	(2c)
Complementary slackness:	$\boldsymbol{\lambda}_k^j \odot \mathbf{g}_k(\xi_{\text{loc}}^j) = \mathbf{0}$	(2d)
	$\boldsymbol{\lambda}_{\neg k}^j \odot \mathbf{g}^*_{\neg k}(\phi(\xi_{\text{loc}}^j)) = \mathbf{0}$	(2e)
Stationarity:	$\nabla_{\xi_{xu}} c_\sigma(\xi_{\text{loc}}^j) + \boldsymbol{\lambda}_k^{j\top} \nabla_{\xi_{xu}} \mathbf{g}_k(\xi_{\text{loc}}^j)$	
	$+ \boldsymbol{\lambda}_{\neg k}^{j\top} \nabla_{\xi_{xu}} \mathbf{g}^*_{\neg k}(\phi(\xi_{\text{loc}}^j))$	(2f)
	$+ \boldsymbol{\nu}_k^{j\top} \nabla_{\xi_{xu}} \mathbf{h}_k(\xi_{\text{loc}}^j) = \mathbf{0}$	

where $\odot$ denotes element-wise multiplication. Here, $\boldsymbol{\lambda}_k^j \in \mathbb{R}^{N_{\text{ineq}}^j}$, $\boldsymbol{\nu}_k^j \in \mathbb{R}^{N_{\text{eq}}^j}$, and $\boldsymbol{\lambda}_{\neg k}^j \in \mathbb{R}^{T^j}$ are vectorized Lagrange multipliers for the known inequality, known equality, and unknown inequality constraints for ξ_{loc}^j, i.e. $\boldsymbol{\lambda}_{\neg k}^j = [\boldsymbol{\lambda}_{\neg k}^{j,1}, \ldots \boldsymbol{\lambda}_{\neg k}^{j,T^j}]$. The blue quantities are unknown to the learner. Intuitively, (2a) enforces that ξ_{loc}^j is feasible for Prob. 1 (it lies in $\mathcal{S}$ and satisfies the known constraints), that a multiplier is zero unless its associated constraint is tight (2b)-(2e), and that its cost cannot be locally improved (2f).

In previous work [4], [9], the unknown constraint is modeled as $g^*_{\neg k}(z, \theta)$, where θ are parameters for a known representation of $g^*_{\neg k}$ with a low-order dependence on θ, e.g. linear $g^*_{\neg k}(z, \theta) = \theta^\top g(z)$, where $g(z)$ are known features; the constraint learning problem then reduces to finding θ. In contrast, we do not require a known parameterization for $g^*_{\neg k}(\cdot)$, instead approximating $g^*_{\neg k}(\cdot)$ as a GP to be learned.

B. Overview of Gaussian processes

A GP is a set of (potentially infinitely many) random variables, any finite number of which have a joint Gaussian
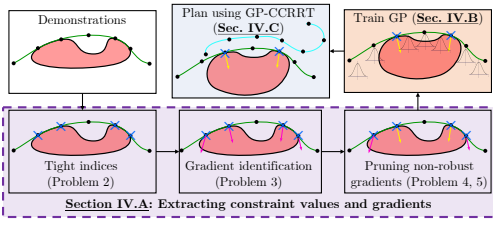
Fig. 2. We obtain constraint values/gradients (Sec. IV-A) to train a GP constraint (Sec. IV-B) for planning (Sec. IV-C)

distribution [23]. It is defined by a mean function $m(x)$ and a covariance function $k(x, x')$. In regression, GPs are often used as the prior distribution for an unknown function $f(x)$ of interest, i.e. $f \sim \mathcal{GP}(m, k)$. Given an input-output dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N_d}$, and assuming a noisy observation model $y_n \sim \mathcal{N}(f(x_n), \sigma^2)$, the predictive conditional posterior $\tilde{f}|\mathcal{D}$ is also a Gaussian if a GP is used as the prior. In performing inference at a set of points $\{z_m\}_{m=1}^{N_q}$, the posterior mean and covariance on these points are given by

$$\mathbb{E}[\tilde{f}(\mathbf{Z})|\mathcal{D}] = k(\mathbf{Z}, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma^2 I)^{-1}\mathbf{Y}, \quad (3)$$

$$\text{cov}(\tilde{f}(\mathbf{Z})|\mathcal{D}) = k(\mathbf{Z}, \mathbf{Z}) - k(\mathbf{Z}, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma^2 I)^{-1}k(\mathbf{X}, \mathbf{Z}). \quad (4)$$

where $\mathbf{Z}$, $\mathbf{X}$, and $\mathbf{Y}$ are vectors containing all elements in $\{z_m\}_{m=1}^{N_q}$, $\{x_n\}_{n=1}^{N_d}$, and $\{y_n\}_{n=1}^{N_d}$, respectively [23].

C. Problem statement

Given locally-optimal demonstrations $\{\xi_{\text{loc}}^j\}_{j=1}^{N_{\text{dem}}}$, we want an estimate $g_{\neg k}(\cdot)$ of the unknown constraint $g_{\neg k}^*(\cdot)$, defining

$$\mathcal{S} = \{\phi_{\text{sep}}(x) \mid g_{\neg k}(\phi_{\text{sep}}(x)) \leq 0\} = \{\kappa \mid g_{\neg k}(\kappa) \leq 0\} \quad (5)$$

as a safe set that is consistent with the demonstrations' KKT conditions, where the true constraint $g_{\neg k}^*(\cdot)$ is assumed to be a sample from $\mathcal{GP}(m, k)$. Moreover, we wish to use the learned constraint to plan trajectories ξ_{xu}^{plan} that connect novel start/goal states while satisfying $g_{\neg k}^*(\cdot)$ with at least some specified probability $1 - \delta$, i.e. $\text{Pr}(\mathbf{g}_{\neg k}^*(\phi(\xi_{xu}^{\text{plan}})) \leq \mathbf{0}) \geq 1 - \delta$.

IV. METHOD

Our method determines where the unknown constraint is tight and its gradient at those points from the KKT conditions (Sec. IV-A), uses this information to train a GP representation of the constraint (Sec. IV-B), and plans novel probabilistically-safe trajectories using the learned constraint (Sec. IV-C). We overview the flow of our method in Fig. 2.

A. Obtaining constraint value and gradient information

For a locally-optimal demonstration, the KKT conditions (2) provide information on the following:

A) If/when the unknown constraint $g_{\neg k}^*(\cdot)$ is *tight* (i.e. at which time steps of the demonstration $g_{\neg k}^*(\phi_{\text{sep}}(x_t)) = 0$) via complementary slackness (2e) and stationarity (2f).

B) How the constraint changes locally around these tight demonstration points, in the form of the constraint gradient at that point, $\nabla_{x_t} g_{\neg k}^*(\phi_{\text{sep}}(x_t))$, via stationarity (2f).

Combining both sources of information is crucial in recovering an accurate constraint that is KKT-consistent.

1) Constraint value information: We first describe a method for inferring when the unknown constraint $g_{\neg k}^*(\cdot)$ is tight. As shorthand, let the stationarity residual $\mathbf{s}^j(\boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j) = [\mathbf{s}_{x_1}^j(\boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j), ..., \mathbf{s}_{x_T}^j(\boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j),$ $\mathbf{s}_{u_1}^j(\boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j), ..., \mathbf{s}_{u_t}^j(\boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j)]^\top \in \mathbb{R}^{|\xi_{xu}|}$ be the LHS of the stationarity condition (2f) for the jth demonstration ξ_{loc}^j, where $\mathbf{s}_{x_t/u_t}^j(\boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j) \in \mathbb{R}^{n_x}/\mathbb{R}^{n_u}$ is the sub-vector containing the residual terms for x_t / u_t.

Recall that complementary slackness (2e) enforces that at each timestep, the Lagrange multiplier for the the unknown constraint is zero unless the constraint is tight. Moreover, as any locally-optimal trajectory ξ_{loc} must satisfy the stationarity condition (2f), we can determine that the unknown constraint $g_{\neg k}^*(\cdot)$ must be tight on ξ_{loc}^j at timestep t if we cannot force the norm of the stationarity residual at that timestep to be zero, i.e. $\|\mathbf{s}_{x_t}^j\| > 0$, while also enforcing that $g_{\neg k}^*(\phi_{\text{sep}}(x_t))$ is not tight (cf. Fig. 3.A for intuition) and that the KKT conditions for the known constraints are satisfied. This is achieved by solving Prob. 2 – a rapidly-solvable linear program (LP):

Problem 2 (Tightness check at time t on demonstration j).

$$\begin{aligned} \underset{\boldsymbol{\lambda}_k^j, \boldsymbol{\nu}_k^j}{\text{minimize}} \quad & \|\mathbf{s}_{x_t}^j(\boldsymbol{\lambda}_k^j, \mathbf{0}, \boldsymbol{\nu}_k^j)\|_1 \\ \text{subject to} \quad & \text{(2b)}, \text{(2d)}, \end{aligned}$$

where the effect of the unknown inequality constraint on the residual is erased by zeroing out its corresponding Lagrange multipliers $\boldsymbol{\lambda}_{\neg k}^j$. Then, the following result holds:

Corollary 1. If the optimal value of Prob. 2, denoted $p_2^{t,j,*}$, is greater than 0, then the true constraint is tight: $g_{\neg k}^*(\kappa_t^j) = 0$.

Proof. Suppose for contradiction that $g_{\neg k}^*(\kappa_t^j) < 0$. Then, since ξ_{loc}^j satisfies (2), $g_{\neg k}^*(\kappa_t^j) < 0$ implies via (2e)-(2f) that there exists $\boldsymbol{\lambda}_k^j$, $\boldsymbol{\lambda}_{\neg k}^j = \mathbf{0}$, $\boldsymbol{\nu}_k^j$ such that $p_2^{t,j,*} = 0$. However, by the theorem statement, $p_2^{t,j,*} > 0$. Contradiction. $\quad\square$

By solving Prob. 2 and checking if $p_2^{t,j,*} > 0$ for $t = 1, \ldots, T^j$, we can find a set of timesteps where $g_{\neg k}^*(\kappa_t^j) = 0$; call these identified tight timesteps $\mathbf{t}_{\text{tight}}^j$. Intuitively, Prob. 2 checks if we can ensure $g_{\neg k}^*(\kappa_t^j) = 0$ despite the known constraints, e.g. dynamics, control constraints, which may be simultaneously tight. By solving Prob. 2 $\sum_{j=1}^{N_{\text{dem}}} T^j$ times (once for each timestep), we can check tightness over the entire dataset. We close with two important remarks. First, complementary slackness and stationarity do not provide information on $g_{\neg k}(\kappa_t^j)$ for timesteps $\mathbf{t}_{\neg \text{tight}}^j \doteq \{1, \ldots, T^j\} \setminus \mathbf{t}_{\text{tight}}^j$; we can only deduce using primal feasibility that $g_{\neg k}^*(\kappa_t^j) \leq 0$ for $t \in \mathbf{t}_{\neg \text{tight}}^j$. Second, the estimated set of tight timesteps $\mathbf{t}_{\text{tight}}$ may only be a subset of the true set of tight timesteps, i.e. $\mathbf{t}_{\text{tight}}^j \subseteq \{t \mid g_{\neg k}^*(\kappa_t^j) = 0\}$; this is because the system may lie on the constraint boundary but the cost cannot be improved by crossing it (Fig. 3.C), i.e. there are multipliers such that $\|\mathbf{s}_{x_t}^j(\boldsymbol{\lambda}_k^j, \mathbf{0}, \boldsymbol{\nu}_k^j)\|_1 = 0$ despite $\boldsymbol{\lambda}_{\neg k}^{j,t} = 0$; KKT cannot guarantee that these points are tight.

2) Gradient value information: Next, we obtain a set of KKT-consistent gradients of the unknown constraint at each identified tight timestep $t \in \mathbf{t}_{\text{tight}}^j$. In Prob. 3, we set the Lagrange multipliers $\boldsymbol{\lambda}_{\neg k}^{j,t} = 1$, for all $t \in \mathbf{t}_{\text{tight}}^j$, and set the non-tight Lagrange multipliers as $\boldsymbol{\lambda}_{\neg k}^{j,t} = 0$, for all $t \in \mathbf{t}_{\neg \text{tight}}^j$;
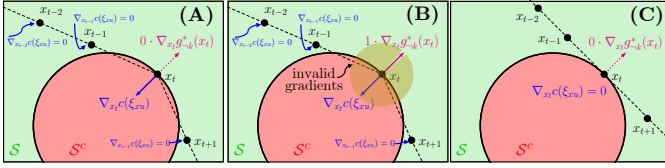
Fig. 3. Consider a demonstrator minimizing path length on a kinematic system; $\phi_{\text{sep}}(x_t) = x_t \in \mathbb{R}^2$. In this simplified setting, we can interpret (2f) as balancing between vectors ∇c and $\lambda \nabla g_{\neg k}$; if they cancel to $\mathbf{0}$, stationarity holds. We visualize this for Prob. 2-3. **(A)** Prob. 2: $\|\mathbf{s}^{x_t}\|$ can only go to zero if $\boldsymbol{\lambda}_k^t > 0$; thus, we detect $g_{\neg k}^*(x_t) = 0$. **(B)** Prob. 3: only a scaling of the magenta constraint normal can make $\|\mathbf{s}^{x_t}\| = 0$; all gradients in gold are not anti-parallel to ∇c and cannot cancel it. **(C)**: sometimes if $g_{\neg k}^*(x_t) = 0$, it's still possible for $\|\mathbf{s}^{x_t}\| = 0$ with $\boldsymbol{\lambda}_{\neg k}^t = 0$.

denote the concatenation of the multipliers as $\mathbf{1}_{\text{tight}}(\xi_{\text{loc}}^j)$. We then solve for gradients $\nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))$, for all $t \in \mathbf{t}_{\text{tight}}^j$, which make the demonstration KKT-consistent along with the Lagrange multipliers of the known constraints:

Problem 3 (Gradient identification on demonstration j)**.**

find $\quad \boldsymbol{\lambda}_k^j, \boldsymbol{\nu}_k^j, \nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j)), \forall t \in \mathbf{t}_{\text{tight}}^j$
subject to $\quad$ (2a), (2b), (2d)
$\quad\quad\quad \mathbf{s}^j(\boldsymbol{\lambda}_k^j, \mathbf{1}_{\text{tight}}(\xi_{\text{loc}}^j), \boldsymbol{\nu}_k^j) = \mathbf{0}.$

Prob. 3 remains an LP as we fix $\boldsymbol{\lambda}_{\neg k}^j$ to avoid bilinearity. To show this does not overly restrict the set of KKT-consistent gradients, we show that while the true gradient may be not be feasible for Prob. 3, a *positive scaling* of it will be. A scaled gradient is acceptable for two reasons. First, it can be impossible to uniquely identify an unscaled gradient via KKT alone: by letting the tight multipliers $\boldsymbol{\lambda}_{\neg k}^{j,t}, t \in \mathbf{t}_{\text{tight}}^j$ take positive non-unit values, they can scale their values to satisfy KKT for different scalings of $\nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))$. Second, while a scaled gradient affects how quickly the constraint changes away from the tight point, it does not affect the shape of the constraint (i.e. it does not rotate the unit surface normal vector at the boundary of the unsafe set). Let $\mathcal{F}$ be the feasible set of Prob. 3 and $\text{proj}_{\nabla g_{\neg k}}(\mathcal{F}) \doteq \{\nabla g_{\neg k} \mid \exists(\boldsymbol{\lambda}_k, \boldsymbol{\nu}_k, \nabla g_{\neg k}) \in \mathcal{F}\}$. Then, we have the following result:

Theorem 1. A positive scaling of the true constraint gradient $\alpha_t^j \nabla_{x_t} g_{\neg k}^*(x_t^j)$, for $\alpha_t^j > 0$, for all $t \in \mathbf{t}_{\text{tight}}^j$, is contained in $\text{proj}_{\nabla g_{\neg k}}(\mathcal{F})$.

Proof. Since ξ_{loc}^j is locally-optimal, it satisfies its KKT conditions; i.e. there exists $\boldsymbol{\lambda}_k^j \geq \mathbf{0}$, $\boldsymbol{\nu}_k^j$, and $\boldsymbol{\lambda}_{\neg k}^j \geq \mathbf{0}$, where $\boldsymbol{\lambda}_{\neg k}^{j,t} = \mathbf{0}$, for all $t \in \mathbf{t}_{\neg\text{tight}}^j$. This is via Prob. 2: if $t \in \mathbf{t}_{\neg\text{tight}}^j$, the KKT conditions can be satisfied if $\boldsymbol{\lambda}_{\neg k}^{j,t} = \mathbf{0}$. Denote one such KKT-consistent set of multipliers as $\boldsymbol{\lambda}_k^{j,*}$, $\boldsymbol{\nu}_k^{j,*}$, and $\boldsymbol{\lambda}_{\neg k}^{j,*}$. As $g_{\neg k}^*(\cdot)$ is state-dependent and time-separable, we can write $\boldsymbol{\lambda}_{\neg k}^{j\top} \nabla_{\xi_{xu}} \mathbf{g}_{\neg k}^*(\phi(\xi_{\text{loc}}^j)) = [\boldsymbol{\lambda}_{\neg k}^{j,1} \nabla_{x_1} g_{\neg k}^*(\phi_{\text{sep}}(x_1^j)), \dots, \boldsymbol{\lambda}_{\neg k}^{j,T} \nabla_{x_T} g_{\neg k}^*(\phi_{\text{sep}}(x_T^j)), \mathbf{0}_{1 \times n_u(T^j - 1)}]$. Then, a feasible solution for Prob. 3 is $\boldsymbol{\lambda}_k^j = \boldsymbol{\lambda}_k^{j,*}$, $\boldsymbol{\nu}_k^j = \boldsymbol{\nu}_k^{j,*}$, and $\nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j)) = \boldsymbol{\lambda}_{\neg k}^{j,*,t} \nabla_{x_t} g_{\neg k}^*(\phi_{\text{sep}}(x_t^j))$, for all $t \in \mathbf{t}_{\text{tight}}^j$. Thus, the theorem holds by setting $\alpha_t^j = \boldsymbol{\lambda}_{\neg k}^{j,*,t}$. $\qquad\square$

While fixing $\boldsymbol{\lambda}_{\neg k}^j$ restricts $\text{proj}_{\nabla g_{\neg k}}(\mathcal{F})$ and reduces scaling ambiguity, due to other active constraints, these recovered KKT-consistent gradients may still not be unique. While scaled gradients are tolerable, a rotation of the true gradient can also lie in $\text{proj}_{\nabla g_{\neg k}}(\mathcal{F})$, complicating the learning as: 1) the unsafe
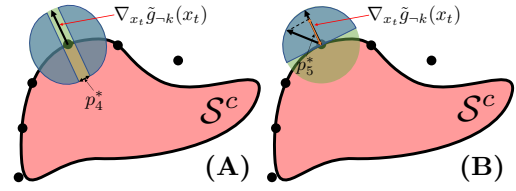

Fig. 4. Prob. 4 and 5 intuition. **(A):** Prob. 4 searches for a new gradient orthogonal to the original gradient by maximizing the distance from the origin as measured in the coordinates of $\nabla_{x_t} \tilde{g}_{\neg k}^\perp$. If $p_4^* = 0$ (i.e. the gradient remains in the gap between the blue areas as the gap $\to 0$), the new gradient must remain in the span of the original gradient. **(B):** Prob. 5 searches for a new gradient with minimal dot product w.r.t. the original gradient; if the result remains in the blue semicircle (i.e. $p_5^* > 0$) and $p_4^* = 0$, the gradient from Prob. 3 is unique up to a scaling.

set shape becomes uncertain, 2) modeling gradient uncertainty is challenging, as determining the set of all consistent gradient vectors is computationally intensive [9], and 3) the gradient uncertainty cannot be well-modeled by a Gaussian distribution, as required by our GP representation.

Though quantifying the uncertainty in the constraint gradients is challenging, we can efficiently check if a given KKT-consistent normal vector is unique, modulo a positive scaling. This can be done by checking that there does not exist another KKT-consistent normal vector that either A) lies in the orthogonal complement of the given normal vector or B) points in directly the opposite direction (see Fig. 4). Let $\nabla_{x_t} \tilde{g}_{\neg k}^j$ be the gradient returned by Prob. 3 for timestep t on ξ_{loc}^j and $\nabla_{x_t} \tilde{g}_{\neg k}^{j,\perp} \in \mathbb{R}^{n_c \times (n_c - 1)}$ as a basis for its orthogonal complement. Then, condition A) can be checked by solving:

Problem 4 (Orthogonal check at time t on demonstration j)**.**

$\underset{\boldsymbol{\lambda}_k^j, \boldsymbol{\nu}_k^j, \nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))}{\text{maximize}} \quad \left\| \nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))^\top \nabla_{x_t} \tilde{g}_{\neg k}^{j,\perp} \right\|_1$

subject to $\quad$ (2a), (2b), (2d)
$\quad\quad\quad \mathbf{s}^j(\boldsymbol{\lambda}_k^j, \mathbf{1}_{\text{tight}}(\xi_{\text{loc}}^j), \boldsymbol{\nu}_k^j) = \mathbf{0}.$

Intuitively, Prob. 4 searches for an alternate gradient in the orthogonal complement of the gradient obtained via Prob. 3 such that some assignment of multipliers also exists to satisfy the KKT conditions. Due to the non-convex objective, Prob. 4 can be modeled as a mixed integer linear program (MILP) with only a small number of binary variables, thus remaining rapidly-solvable. Next, condition B) can be checked via:

Problem 5 (Anti-parallel check; time t on demonstration j)**.**

$\underset{\boldsymbol{\lambda}_k^j, \boldsymbol{\nu}_k^j, \nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))}{\text{minimize}} \quad \nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))^\top \nabla_{x_t} \tilde{g}_{\neg k}^j$

subject to $\quad$ (2a), (2b), (2d)
$\quad\quad\quad \mathbf{s}^j(\boldsymbol{\lambda}_k^j, \mathbf{1}_{\text{tight}}(\xi_{\text{loc}}^j), \boldsymbol{\nu}_k^j) = \mathbf{0}.$

Prob. 5, an LP, searches for a KKT-consistent gradient minimizing the dot product with $\nabla_{x_t} \tilde{g}_{\neg k}$, i.e. pointing as anti-parallel to the original gradient as possible. Thm. 2 shows how Probs. 4-5 can check gradient uniqueness. Denote the optimal values of Prob. 4 and 5 as p_4^* and p_5^*. We have:

Theorem 2. If $p_4^* = 0$ and $p_5^* > 0$, the true gradient $\nabla_{x_t} g_{\neg k}^*(\phi_{\text{sep}}(x_t^j))$ is a positive scaling of the recovered gradient $\nabla_{x_t} \tilde{g}_{\neg k}(\phi_{\text{sep}}(x_t^j))$, i.e. there exists $\alpha > 0$ such that $\nabla_{x_t} g_{\neg k}^*(\phi_{\text{sep}}(x_t^j)) = \alpha \nabla_{x_t} \tilde{g}_{\neg k}(\phi_{\text{sep}}(x_t^j))$.

Proof. First, $p_4^* = 0$ iff all feasible $\nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))$ lie in

span$(\nabla_{x_t} \tilde{g}_{\neg k})$, as the objective of Prob. 4 is just the norm of the coordinates of $\nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j))$ in the basis of the orthogonal complement, i.e. there exists $\beta \in \mathbb{R}$ such that $\nabla_{x_t} g_{\neg k}^j = \beta \nabla_{x_t} \tilde{g}_{\neg k}^j$. Second, if $p_5^* > 0$, then $\nabla_{x_t} g_{\neg k}^{j,\top} \nabla_{x_t} \tilde{g}_{\neg k}^j > 0$, for all $\nabla_{x_t} g_{\neg k}^j \in \text{proj}_{\nabla_{g_{\neg k}}}(\mathcal{F})$. Third, by combining these two results, we have that $\beta \nabla_{x_t} \tilde{g}_{\neg k}^{j,\top} \nabla_{x_t} \tilde{g}_{\neg k}^j > 0$, implying $\beta > 0$, as $\|\nabla_{x_t} \tilde{g}_{\neg k}^j\| > 0$ in order for $t \in \mathbf{t}_{\text{tight}}^j$. Finally, from Thm. 1, we know $\nabla_{x_t} g_{\neg k}^* = \gamma \nabla_{x_t} g_{\neg k}$, for some $\gamma > 0$ and $\nabla_{x_t} g_{\neg k} \in \text{proj}_{\nabla_{g_{\neg k}}}(\mathcal{F})$; we recover the theorem statement by setting $\alpha = \beta\gamma$. $\qquad\square$

Our approach is to use the tight points with a unique KKT-consistent unit normal vector to train our GP constraint (see Sec. IV-B); we call these gradients *robustly-identified* and their timesteps as $\mathbf{t}_{\text{rob}}^j \subseteq \mathbf{t}_{\text{tight}}^j$, for all $j = 1, \ldots, N_{\text{dem}}$.

B. Embedding KKT-based information in a Gaussian process

Let the number of robustly-identified points over all demonstrations be N_{robust}. We collect the constraint states corresponding to the robustly-identified gradients and denote it as $\mathcal{D}_\kappa \doteq \{\phi_{\text{sep}}(x_t^j) \mid t \in \mathbf{t}_{\text{rob}}^j, j \in \{1, \ldots, N_{\text{dem}}\}\} \in \mathbb{R}^{N_{\text{robust}} \times n_c}$. We also collect the robustly-identified gradients $\mathcal{D}_\nabla \doteq \{\nabla_{x_t} g_{\neg k}(\phi_{\text{sep}}(x_t^j)) \mid t \in \mathbf{t}_{\text{rob}}^j, j \in \{1, \ldots, N_{\text{dem}}\}\} \in \mathbb{R}^{N_{\text{robust}} \times n_c}$. Moreover, as the value of the unknown constraint equals zero at all robustly-identified points, we can define a third set $\mathcal{D}_g \doteq \mathbf{0}_{N_{\text{robust}}}$, i.e. the zero vector of size N_{robust}. We wish to learn a GP which is consistent with both the constraint values $\mathcal{D}_g$ as well as the constraint gradients $\mathcal{D}_\nabla$. Note that derivative of a GP is a GP, and the joint distribution of a GP and its derivative is also a GP [24]; forming this joint GP provides us an avenue for incorporating both the constraint value and gradient information. Like the derivation of the GP posterior without derivative observations (Sec. III-B), one can derive the posterior distribution conditioned on the training inputs, their derivatives, and the outputs. For brevity, please refer to [25] for detailed derivations. For this joint GP, we can define the training inputs and outputs as $\mathbf{X} = \mathcal{D}_\kappa$ and $\mathbf{Y} = [\mathcal{D}_g, \mathcal{D}_\nabla]$, comprising the dataset $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$, and use the negative marginal log likelihood $-\mathcal{L}_{\text{MLL}}$ [23, Eq. 2.30] to optimize the GP hyperparameters.

A key subtlety is that as the learned constraint is a GP, its constraint value at any given query point is not deterministic; rather, it is sampled from a Gaussian distribution whose mean and variance is determined by the training data and the location of the query point (i.e. $g_{\neg k}(\kappa_m) \sim \mathcal{N}(\mu_m, \sigma_m^2 \mid \mathcal{D}, \kappa_m)$). Moreover, while the demonstrations are guaranteed safe by assumption (i.e. $g_{\neg k}(\kappa_t) \leq 0$ for all t), the stochasticity of the GP values prevents us from enforcing that the demonstrations are safe with probability 1, as the Gaussian has infinite support. Instead, we select a standard deviation threshold ρ for which we want the demonstrations to be safe and add a hinge loss on its violation, where $R = \sum_{j=1}^{N_{\text{dem}}} T^j$:

$$\mathcal{L}_{\text{feas}} = (1/R) \sum_{n=1}^{R} \max(\mu(x_n \mid \mathcal{D}) + \rho\sigma(x_n \mid \mathcal{D}), 0). \quad (6)$$

Then the full training loss is $\mathcal{L} = -\mathcal{L}_{\text{MLL}} + \mathcal{L}_{\text{feas}}$.

C. Planning with the learned constraint

We describe a method for planning with the learned GP constraint. As the GP is probabilistic, so is the boundary of the
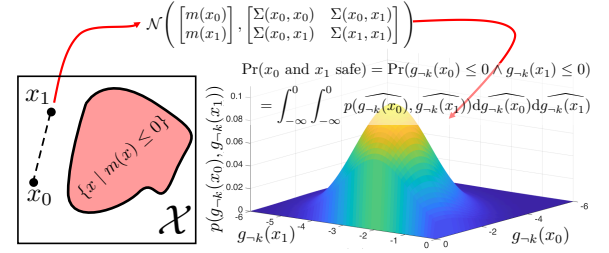

Fig. 5. Illustration of GP-CCRRT. A candidate length 2 trajectory from the root of the RRT induces a bivariate Gaussian; its safety probability can then be calculated by calculating the CDF of the induced Gaussian.

learned safe set $\mathcal{S}$ (5); thus, our planner provides probabilistic, rather than deterministic, safety guarantees. As the dynamics are assumed known, we only consider the uncertainty of the GP constraint in planning. Recall that we wish to connect a start and goal state with a dynamically-feasible trajectory ξ_{xu}^{plan} that satisfies the true constraint $\mathbf{g}_{\neg k}^*(\phi(\xi_{xu}^{\text{plan}}))$ with probability $1 - \delta$. From the assumption (Sec. III-C) that $g_{\neg k}^*$ is drawn from the GP, we achieve this by satisfying the learned constraint $\mathbf{g}_{\neg k}(\phi(\xi_{xu}^{\text{plan}}))$ with probability $1 - \delta$. Unlike previous work in planning under uncertainty [19], [20], [26], we make no structural assumptions on the dynamics or the shape of the constraints.

We modify a constrained kinodynamic RRT [27] to plan with the learned constraint, though our method can be adapted to other sampling or optimization-based planners. Our planner, which we refer to as Gaussian Process-Chance Constrained RRT (GP-CCRRT), is presented in Alg. 1. The main novelty of the proposed planner is its GP constraint-checker: when a new node x_q is sampled, instead of checking if x_q satisfies the timestep-independent chance constraint $\Pr(g_{\neg k}(\phi_{\text{sep}}(x_q)) \leq 0) \geq 1 - \delta$, we check if we can connect x_q to the tree by exactly evaluating the joint probability of safety for the full trajectory from the root to the candidate node x_q (line 7-8). Our ability to efficiently compute this probability relies on the Gaussian structure of our learned GP constraint representation. Let the full trajectory from the root to x_q, denoted ξ_q, be length K. Evaluating the learned GP $g_{\neg k}(\cdot)$ at those K points returns the mean and covariance matrix of the predictive posterior distribution, which is a K-variate Gaussian (line 7). Then, the trajectory safety probability, $p_{\text{safe}}^q \doteq \Pr(\bigwedge_{n=1}^{K}(g_{\neg k}(\kappa_n) \leq 0))$, is obtained by integrating the density of this $|K|$-variate Gaussian from $-\infty$ to 0 in each dimension (i.e. the cumulative distribution function, or cdf), evaluated at $\mathbf{0}_K$ (line 8). Highly-optimized implementations of the multivariate Gaussian CDF [28] enable fast CDF evaluation at planning time. Finally, node x_q is accepted if ξ_q is safe with at least probability $1 - \delta$ (line 8). We visualize the GP constraint check in Fig. 5.

V. RESULTS

We evaluate our method on learning complex, nonlinear constraints demonstrated on a car, quadrotor, and arm. Please see the video for visualizations. We train all GPs using GPyTorch with an RBF kernel using the Adam optimizer. We obtain demonstrations by solving Prob. 1 using IPOPT [29]. We compare with two baselines. The first, [4, Prob. 4], approximates the unknown constraint as a union of B axis-aligned boxes (as in [6, Sec. 4.4]). In the second, we use a

Algorithm 1: GP-CCRRT

Input: x_I, x_G, ϵ, goal_bias

1   $\mathcal{T}_x \leftarrow \{x_I\}, \mathcal{T}_u \leftarrow \{\}$
2   **while** True **do**
3     $x_{\text{desired}} \leftarrow$ SampleState(goal_bias)
4     $x_{\text{near}} \leftarrow$ NearestNeighbor($\mathcal{T}_x, x_{\text{desired}}$)
5     $\xi_{\text{prev}} \leftarrow$ PathFromRoot(x_{near})
6     $x_q, u_q \rightarrow$ ShootToDesired($x_{\text{near}}, x_{\text{desired}}$)
7     $\mu, \Sigma \leftarrow g_{\neg k}(\phi(\xi_{\text{prev}} \cup x_q))$
8   **if** cdf($\mathcal{N}(\mu, \Sigma), \mathbf{0}) \geq 1 - \delta$ **then**
      $(\mathcal{T}_x, \mathcal{T}_u) \leftarrow (\mathcal{T}_x, \mathcal{T}_u) \cup (x_q, u_q)$
9   **if** $\|x_q - x_G\| \leq \epsilon$ **then**
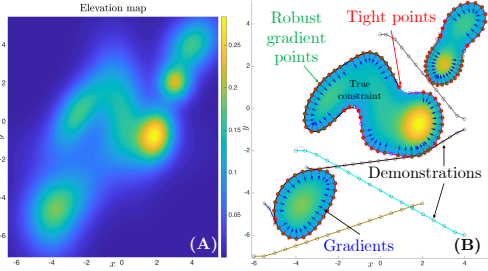10   **return** $\xi_{xu}^{\text{plan}} \leftarrow$ ConstructPath($\mathcal{T}_x, \mathcal{T}_u, x_q$)

Fig. 6. 5D car example. **(A)** Hilly terrain map. **(B)** Demos; identified tight points (red); robustly-identified timesteps (green); robustly-identified gradients (blue).

Fig. 7. 5D car example, learned. **(A)** Learned GP constraint, mean function. **(B)** Mean function misclassifications. **(C)** Constraint learned using baseline [4]. **(D)** Learned GP constraint, buffered by GP uncertainty. **(E)** Buffered misclassifications. **(F)** Plans computed using learned GP constraint.

neural network (NN) instead of a GP to fit the constraint using the same data; in all examples, the NN has 5 hidden layers of size 256, 512, 1024, 512, and 128 and is trained for 200 epochs with learning rate 5×10^{-5}. To train the NN, we use mean squared error losses on the target tight constraint values and gradients, as well a hinge loss that encourages all points to be feasible. We also compute (Table I) how many states are falsely claimed safe ("false safe (FS)") or unsafe ("false unsafe (FU)") by setting $\mathcal{S} = \{\kappa \mid \mu(\kappa \mid \mathcal{D}) + \tau\sigma(\kappa \mid \mathcal{D}) \leq 0\}$ for standard deviations $\tau \in \{0, 1, 2, 2.33\}$. While $\mathcal{S}$ is not used in GP-CCRRT (it uses joint instead of individual safety probabilities), it is a good surrogate for constraint accuracy. Finally, Probs. 2-5 are all solved in 0.5 seconds.

		Our method				Baselines	
		$0\sigma_p$	$1\sigma_p$	$2\sigma_p$	$2.33\sigma_p$	[4]	NN
Car	FS (%)	1.741	0.319	0.071	0.042	5.947	15.555
	FU (%)	0.424	58.761	64.807	66.305	0.117	0.000
Box	FS (%)	3.230	0.462	0.189	0.138	0.000	10.859
	FU (%)	1.648	81.146	86.641	87.190	0.000	0.000
Tree	FS (%)	0.593	0.057	0.004	0.000	14.867	23.427
	FU (%)	0.729	11.108	31.412	37.773	0.160	0.000
Arm	FS (%)	1.403	0.163	0.012	0.003	17.179	15.029
	FU (%)	0.658	57.294	70.644	73.490	0.808	0.151

TABLE I
GP CLASSIFICATION ERRORS (FALSE SAFE (FS); FALSE UNSAFE (FU)).

5D nonholonomic car: We first evaluate our method on a 5D car, showing that can learn a nonlinear, disconnected constraint without prior knowledge. Consider an autonomous vehicle driving on hilly terrain (Fig. 6.A) which must stay below a maximum elevation; the corresponding unsafe set (i.e. the subset of the map above the elevation limit) is the filled-in region in Fig. 6.B. We use the second-order unicycle dynamics from [27, Eq. 13.46] with a discretization time of $\Delta T = 0.5$. Prior work [5] studied a similar example; however, in [5], the map (i.e. the constraint representation) is given, so the only the elevation threshold must be learned. In contrast, we are not given the map, and must learn the representation jointly with the threshold – a much harder problem.
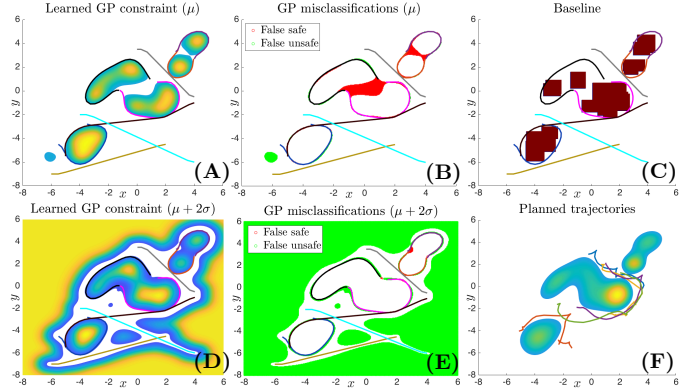
We obtain 9 demonstrations minimizing the xy-space path length, and a control constraint of $\|u_t\|_2^2 \leq 5$ is imposed for all time. Here, ϕ_{sep} maps to the xy-state components. By solving Prob. 2, we identify tight points (Fig. 6.B, red). Next, by solving Probs. 3-5, we find robustly-consistent gradients (Fig. 6.B, blue arrows) at a subset of the tight points (Fig. 6.B, green). Note the accuracy of Prob. 2, which identifies that the cyan trajectory is not tight, despite it curving due to dynamical constraints, and correct identification for the black trajectory, which makes and breaks contact with the constraint boundary. The few tight points that are not identified (e.g. near $[-2, 2]$) are where the constraint boundary is flat; thus, the sub-trajectory is optimal despite being on the boundary (Fig. 3.C). Note that most tight points also have robustly-identifiable gradients; the exceptions are before/after the system leaves the constraint boundary; this is due to the dynamics, as the car may brake/turn to prevent constraint violation, expanding the set of consistent gradients.

We train the GP for 150 epochs at learning rate 0.08. In Fig. 7, we show the GP accuracy and compare with the baseline [4]. Overall, the GP mean faithfully recreates the true unsafe set (Fig. 7.A), though it misclassifies (Fig. 7.B) the center of the middle and top obstacles; this is as there are few tight points in that area. Still, when buffering the GP with its uncertainty (Fig. 7.D-E), the regions which are falsely classified shrink (see Table I), though this is at the cost of conservatively marking much of the map far from the demonstrations as unsafe. This arises from the GP's ability to capture epistemic uncertainty and can actually be desirable as it leads to cautious plans that remain near the data and away from unseen constraints that are inactive on the demonstrations. For the baseline [4], we use 20 boxes and terminate the optimization after 60 minutes. The result has higher error than the learned GP constraint and fails to capture the constraint shape. The NN baseline is also inaccurate, as it drives the value of most tight points to 0 but fails to fit the gradient data (Table I). Finally, we plan with GP-CCRRT with a safety probability of 0.9; five plans are shown in Fig. 7.F, which all satisfy $g_{\neg k}^*(\cdot)$. On average, our planner solves in 3 minutes, with 20 and 50 percent of that time being dedicated to GP posterior and CDF computation, respectively; this can be sped up via lazy checking of the

CDF constraint. Overall, this example suggests we can learn nonconvex constraints with minimal *a priori* knowledge.

12D quadrotor: We evaluate our method on two constraint learning tasks on a 12D quadrotor (see [30, Eq. 19] for the dynamics). We first show our method achieves comparable performance with [4] for learning constraints that can be represented as a union of boxes. We are given 24 short demonstrations (Fig. 8.A) that minimize xyz path length while satisfying a control constraint $\|u_t\|_2^2 \leq 100$, for all t. Moreover, the baseline [4] is also provided the information that the constraint can be represented as a union of two axis-aligned boxes (thus learning the constraint exactly). We train the GP for 600 epochs at learning rate 0.1, and the learned GP (Fig. 8.B) captures the union-of-boxes shape well. Two main inaccuracies are A) the interior of the learned box is hollow (this is expected, as no data can be collected in the obstacle) and B) there are some "ringing effects" (this is caused by the GP, which favors smooth functions, attempting to fit the discontinuous box gradients). Numerically, Table I shows that the GP misclassifications are low, and moreover, the number of states that are falsely claimed to be safe can be driven near zero by buffering with the GP uncertainty. The GP outperforms the NN baseline, which again struggles to fit the gradient data. Overall, this example suggests our method also performs well where previous methods excel.

Next, we evaluate our approach on learning a complex nonlinear constraint which is well beyond the capability of the baseline [4]. We are given 25 demonstrations (Fig. 1.A, black) avoiding collisions with an unknown tree-like obstacle to be learned, which is a union of three ellipsoids (Fig. 1.A, blue). Crucially, we lack *a priori* knowledge on the structure of $g^*_{\neg k}(\cdot)$. The dynamics and cost function used are as in [9] and [4], respectively, and ϕ_{sep} maps to the xyz-state components. We train the GP for 150 epochs at learning rate 0.08. In comparing with the baseline, we use 6 axis-aligned boxes and time out the optimization after 2 hours.

We visualize our results in Fig. 1.B-C. Our learned GP is visually accurate (Fig. 1.B), with minor errors (Fig. 1.C) where there are no tight points. This is reasonable, as we cannot expect the GP to be accurate far from the data. In contrast, the baseline [4] is inaccurate (Fig. 1.D), failing to cover the upper portion of the obstacle; moreover, the shape is inaccurate due to the limitations of axis-aligned boxes. The NN baseline is also inaccurate, failing to fit the constraint gradients (Table I). Numerical results in Table I suggest that the GP mean is the most accurate when considering both metrics. As before, the "False Safe" rate can be made smaller at the cost of conservativeness by buffering with the predictive uncertainty. In contrast, the baseline has a high "False Safe" rate, which can lead to constraint violation in planning. We show six plans computed via GP-CCRRT (Fig. 1.A, gold), which are safe with probability at least 0.9, and which are safe for the true constraint. On average, planning takes 90 seconds; 55 and 20 percent of this is due to GP posterior and CDF calculations. This example suggests our method scales to complex constraints on high-dimensional systems while requiring minimal prior information.

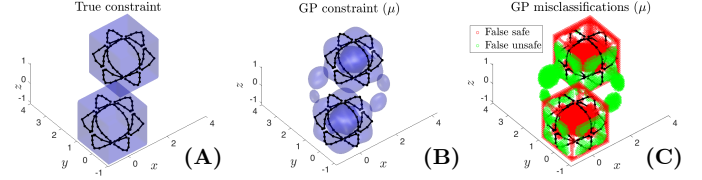**Planar 3-link manipulator**: Finally, we evaluate our method

Fig. 8. 12D quadrotor box example. **(A)** Two box obstacles; demonstrations. **(B)** Learned GP constraint (mean). **(C)** GP misclassifications (mean).
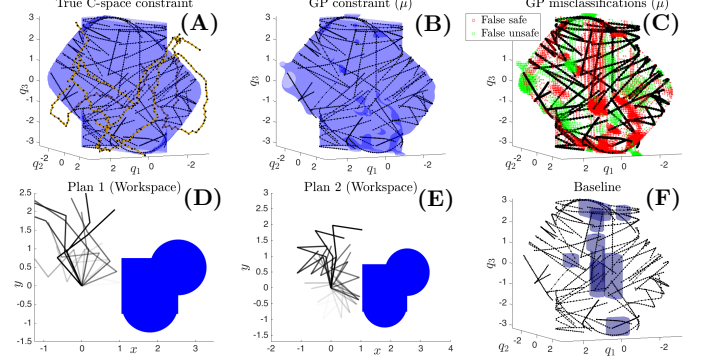

Fig. 9. 3-link planar arm example, learned. **(A)** True C-space constraint; plans found using GP constraint (gold). **(B)** Learned GP constraint, mean function. **(C)** Mean function misclassifications. **(D-E)** Plans computed using learned GP constraint (workspace). **(F)** Constraint learned with baseline [4].

on a kinematic planar 3-link arm. The arm is mounted at the origin and must avoid a blue nonconvex workspace obstacle (Fig. 9.D, E). To show that our method can learn complex constraints on articulated robots, we learn the configuration space (C-space) representation of the obstacle (Fig. 9.A, blue), which is also nonconvex. We obtain 50 demonstrations (Fig. 9.A, black) which minimize the joint-space path length, i.e. $c(\xi_{xu}) = \sum_{t=1}^{T} \|q_{t+1} - q_t\|^2$ subject to a control constraint $\|q_{t+1} - q_t\| \leq 0.1$, for all t. We train the GP for 150 epochs at learning rate 0.08, obtaining a GP whose posterior mean is visually consistent with the true constraint (Fig. 9.B). The posterior mean misclassifications are mostly on the interior of the C-space obstacle (as expected, since no data can be collected there), as well as minor errors on the constraint surface which are further away from the data. We plan via GP-CCRRT, taking two minutes on average, where 40 and 45 percent of the time is due to GP posterior and CDF computations, respectively. We use GP-CCRTT to obtain plans that are safe with probability greater than 0.9; time-lapses of two such plans are shown in Fig. 9.D-E. Finally, we evaluate the baseline [4] with 10 boxes, timing out the optimization after 6 hours. Due to the number of demonstrations and constraint parameters, the baseline struggles (Fig. 9.F), returning boxes that do not adequately satisfy the KKT conditions and fail to capture the features of the true constraint. As before, the posterior mean has low "false safe" and "false unsafe" rates, and the "false safe" rate can be reduced via buffering (see Table I). In contrast, the baseline has a much higher "false safe" rate, since it fails to cover most of the unsafe set, though it has a low "false unsafe" rate, since it marks most of the space as safe. The NN baseline also fails to fit the gradient data, leading to low accuracy (Table I). Overall, this example suggests that we can learn non-convex, non-workspace constraints on articulated robots while requiring

minimal prior information, which is a necessity for C-space obstacles, which can be unintuitive.

VI. DISCUSSION AND CONCLUSION

In this paper, we learn constraints from demonstrations with minimal *a priori* knowledge by finding where the unknown constraint is tight and a scaling of its gradient at those points via the KKT conditions, and then training a GP-represented constraint that is consistent with and generalizes this data. We also show that the Gaussian structure of the GP uncertainty can exploited in an RRT-based planner to compute plans which satisfy the unknown constraint with a specified probability. Our results on a 5D car, 12D quadrotor, and 3-link planar arm show we can learn complex constraints on realistic systems which prior methods cannot handle. We conclude by discussing design choices and future work.

Why use a GP constraint representation? Our learning problem (fitting a function using zero level set data (the tight points) and its gradients at those points) closely relates to fitting manifolds [31] and signed distance functions [32] to data (though our method differs greatly in how it obtains the data, i.e. the KKT conditions). In both [31] and [32], neural networks (NN) are used to fit large datasets on the order of $10^3 \sim 10^4$ and 10^5 for [31] and [32], respectively. In Sec. V, we show that in training the NN with only $\sim 10^2$ tight data points, the NN failed to provide an accurate fit. In contrast, derivative data can be directly embedded via the joint GP, which fits the constraint better with less data (on the order of 10^2). Moreover, GPs handle constraint uncertainty in a principled way, which is crucial for safe planning.

Limitations and future work: First, as GPs are non-parametric, dense coverage of the constraint space with tight points is needed to reduce the predictive uncertainty. This results in cautious plans that stay near the training data, and may needlessly restrict the robot. In future work, we will explore semi-parametric models which combine a known, insufficient parameterization with a non-parametric GP to reduce uncertainty. Second, our method assumes demonstrations are precisely locally-optimal, but this is often untrue due to noisy observations or partial observability [33]; in future work, we will investigate suboptimality models (e.g. [34]) that can be used to adjust the confidence in extracted constraint value/gradient data. Third, we wish to extend our method to time-varying (e.g. temporal logic [35]) constraints. Finally, the scaling of GPs may hamper the learning of high-dimensional constraints; thus, we will explore scalable GP variants (e.g. sparse spectrum GP regression [36]).

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

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

[3] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations with grid and parametric representations," *IJRR*, vol. 40, no. 10-11, pp. 1255–1283, 2021.

[4] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robotics Autom. Lett.*, vol. 5, no. 2, pp. 3682–3690, 2020.

[5] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations," *WAFR*, 2018.

[6] G. Chou, N. Ozay, and D. Berenson, "Learning parametric constraints in high dimensions from demonstrations," in *CoRL*, 2019.

[7] M. Menner, P. Worsnop, and M. N. Zeilinger, "Constrained inverse optimal control with application to a human manipulation task," *IEEE TCST*, 2019.

[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] G. Chou, N. Ozay, and D. Berenson, "Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations," in *CoRL 2020, Cambridge, MA, USA*, 2020.

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

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

[12] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *NIPS*, 2011, pp. 19–27.

[13] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *ICML*, 2016, pp. 49–58.

[14] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *IJRR*, vol. 36, pp. 1073–1087, 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 *ICRA*, 2017.

[16] B. Burns and O. Brock, "Sampling-based motion planning with sensing uncertainty," in *ICRA*, 2007.

[17] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *ICRA*, 2011.

[18] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *AURO*, vol. 35, no. 1, p. 51–76, 2013.

[19] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *ACC*, 2006.

[20] B. Luders, M. Kothari, and J. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *GNC*, 2010.

[21] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, 2001.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. CUP, 2004.

[23] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[24] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen, "Derivative observations in gaussian process models of dynamic systems," in *NIPS 2002*, 2002, pp. 1033–1040.

[25] A. Wu, M. C. Aoi, and J. W. Pillow, "Exploiting gradients and hessians in bayesian optimization and bayesian quadrature," 2018.

[26] B. Luders, S. Karaman, and J. How, "Robust sampling-based motion planning with asymptotic optimality guarantees," in *AIAA GNC*, 2013.

[27] S. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[28] A. Genz and G. Trinh, "Numerical computation of multivariate normal probabilities using bivariate conditioning," in *MCQMC*, 2014.

[29] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[30] G. Chou, N. Ozay, and D. Berenson, "Learning temporal logic formulas from suboptimal demonstrations: theory and experiments," *Autonomous Robots*, July 2021.

[31] G. Sutanto, I. M. R. Fernández, P. Englert, R. K. Ramachandran, and G. S. Sukhatme, "Learning equality constraints for motion planning on manifolds," in *CoRL*, 2020.

[32] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019, pp. 165–174.

[33] C. Knuth, G. Chou, N. Ozay, and D. Berenson, "Inferring obstacles and path validity from visibility-constrained demonstrations," in *WAFR*, vol. 17, 2021, pp. 18–36.

[34] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008.

[35] G. Chou, N. Ozay, and D. Berenson, "Explaining Multi-stage Tasks by Learning Temporal Logic Formulas from Suboptimal Demonstrations," in *Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[36] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *NIPS*, 2007, pp. 1177–1184.