Gaussian Process Constraint Learning for Scalable Safe Motion Planning from Demonstrations

Hao Wang*[1,2], Glen Chou*[2], Dmitry Berenson[2]
[1]ME, [2]EECS, University of Michigan, Ann Arbor, MI, 48109, USA. *equal contribution

***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 of the demonstrations to determine the location and shape of the constraints, and uses these to train a GP which is consistent with this information. We demonstrate our method on a 12D quadrotor constraint learning problem, showing that the learned constraint is accurate and can be used within a kinodynamic RRT to plan probabilistically-safe trajectories.**

I. Introduction and Related Work

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) [3, 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. To address safety in LfD, recent work has represented tasks as *constrained* optimization problems, and learns the unknown cost function and constraints from demonstrations [9, 7, 6, 17, 11], enabling the learning of complex tasks in manipulation and mobile robotics. However, these methods require that the unknown constraints can be described by a known representation (i.e. as a union of axis-aligned boxes), restricting these methods to learning highly-structured tasks. Moreover, such representations can be highly inefficient (e.g. many boxes may be required to approximate complex constraints), leading to a computational burden which makes it challenging to scale these methods up to learn realistic constraints. We address these issues via the insight that the Karush-Kuhn-Tucker (KKT) optimality conditions provide information on the location and shape of the unknown constraint which can be embedded in a Gaussian process (GP) constraint representation, requiring minimal *a priori* knowledge on the underlying constraint structure. Our contributions are:

- We show how to learn a GP constraint that is consistent with the location/shape data given by the KKT conditions.
- We show how the learned GP constraint and its uncertainty can be used to compute probabilistically-safe plans.
- We evaluate our method on a complex nonlinear constraint learning problem on a 12D quadrotor system, showing that our method outperforms baselines.

II. Preliminaries and Problem Statement

A. Demonstrator's problem and KKT optimality conditions

We represent a demonstration of a task σ performed on a system $x_{t+1} = f(x_t, u_t, t)$, $x \in \mathcal{X}$, $u \in \mathcal{U}$ as a constrained optimization over state/control trajectories $\xi_{xu} \doteq (\xi_x, \xi_u)$:

Problem 1 (Forward (demonstrator's) problem / task σ)**.**

$$\begin{aligned} \underset{\xi_{xu}}{\text{minimize}} \quad & c(\xi_{xu}) \\ \text{subject to} \quad & \phi(\xi_{xu}) \in \mathcal{S}(\theta) \subseteq \mathcal{C} \Leftrightarrow \quad \mathbf{g}_{\neg k}(\xi_{xu}, \theta) \leq \mathbf{0} \\ & \bar{\phi}(\xi_{xu}) \in \bar{\mathcal{S}} \subseteq \bar{\mathcal{C}}, \quad \phi_\sigma(\xi_{xu}) \in \mathcal{S}_\sigma \subseteq \mathcal{C}_\sigma \\ & \Leftrightarrow \quad \mathbf{h}_k(\xi_{xu}) = \mathbf{0}, \quad \mathbf{g}_k(\xi_{xu}) \leq \mathbf{0} \end{aligned}$$

where $c(\cdot)$ is known, $\phi(\cdot)$ maps from trajectories to a constraint space $\mathcal{C}$ (where the constraint is evaluated) containing *constraint states* $\kappa \in \mathcal{C}$. The safe set $\mathcal{S}(\theta)$, defined by $\theta \in \Theta$, is unknown to the learner. $\bar{\phi}(\cdot)$ and $\phi_\sigma(\cdot)$ map to constraint spaces $\bar{\mathcal{C}}$ and $\mathcal{C}_\sigma$, containing a known shared safe set $\bar{\mathcal{S}}$ and task-dependent safe set $\mathcal{S}_\sigma$. We encode the dynamics in $\bar{\mathcal{S}}$ and the start/goal constraints in $\mathcal{S}_\sigma$. We group the constraints of Prob. 1 as (in)equality (ineq/eq) and (un)known ($\neg k/k$), 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 $\mathbf{g}_{\neg k}(\xi_{xu}, \theta) \in \mathbb{R}^{N_{\neg k}^{\text{ineq}}}$. We focus on scalar time and parameter-separable constraints

$$\mathbf{g}_{\neg k}(\xi_j^{\text{loc}}, \theta) \leq \mathbf{0} \;\Leftrightarrow\; g_{\neg k}(\phi(x_t)) \leq \theta, \quad \forall t = 1, \ldots, T. \quad (1)$$

where we abuse notation and refer to $\phi(\cdot)$ as its time-separable counterpart. Handling time-dependent constraints (e.g. temporal logic formulas [10]) is a direction for future work. As in [9, 8], θ is an offset that defines a safe sublevel set. However, unlike previous work, we crucially do not assume the nonlinear part of the constraint $g_{\neg k}(\phi(x_t))$ is known; we instead represent it as a GP and learn it jointly with θ. We assume each state-control demonstration ξ^{loc} solves Prob. 1 to local optimality, satisfying Prob. 1's KKT conditions [4] (see Sec. V for discussion on the case where this fails to hold). With Lagrange multipliers λ, ν, the relevant KKT conditions for the jth demonstration ξ_j^{loc}, denoted KKT(ξ_j^{loc}), are:

$$\begin{aligned} &\text{Primal feasibility:} && \mathbf{g}_{\neg k}(\xi_j^{\text{loc}}, \theta) \leq \mathbf{0}, && \text{(2a)} \\ &\text{Lagrange mult.} && \lambda_{i,k}^j \geq 0, \;\; i = 1, ..., N_k^{\text{ineq}} \;\Leftrightarrow\; \boldsymbol{\lambda}_k^j \geq \mathbf{0} && \text{(2b)} \\ &\text{nonnegativity:} && \lambda_{i,\neg k}^j \geq 0, \;\; i = 1, ..., N_{\neg k}^{\text{ineq}} \;\Leftrightarrow\; \boldsymbol{\lambda}_{\neg k}^j \geq \mathbf{0} && \text{(2c)} \\ &\text{Complementary} && \boldsymbol{\lambda}_k^j \odot \mathbf{g}_k(\xi_j^{\text{loc}}) = \mathbf{0} && \text{(2d)} \\ &\text{slackness:} && \boldsymbol{\lambda}_{\neg k}^j \odot \mathbf{g}_{\neg k}(\xi_j^{\text{loc}}, \theta) = \mathbf{0} && \text{(2e)} \\ &\text{Stationarity:} && \nabla_{\xi_{xu}} c(\xi_j^{\text{loc}}) + \boldsymbol{\lambda}_k^{j\top} \nabla_{\xi_{xu}} \mathbf{g}_k(\xi_j^{\text{loc}}) \\ & && \quad + \boldsymbol{\lambda}_{\neg k}^{j\top} \nabla_{\xi_{xu}} \mathbf{g}_{\neg k}(\xi_j^{\text{loc}}, \theta) && \text{(2f)} \\ & && \quad + \boldsymbol{\nu}_k^{j\top} \nabla_{\xi_{xu}} \mathbf{h}_k(\xi_j^{\text{loc}}) = \mathbf{0} \end{aligned}$$

where $\nabla_{\xi_{xu}}(\cdot)$ takes the gradient with respect to a flattened ξ_{xu} and $\odot$ is elementwise multiplication. We denote the vectorized multipliers as $\boldsymbol{\lambda}_k^j \in \mathbb{R}^{N_k^{\text{ineq}}}$, $\boldsymbol{\lambda}_{\neg k}^j \in \mathbb{R}^{N_{\neg k}^{\text{ineq}}}$, and $\boldsymbol{\nu}_k^j \in \mathbb{R}^{N_k^{\text{eq}}}$. Quantities unknown to the learner are in blue. Intuitively, (2a) enforces that ξ_j^{loc} is feasible for Prob. 1 (it is in the safe set $\mathcal{S}(\theta)$ and satisfies the known constraints), that the multipliers are zero unless the corresponding constraints are active (2b)-(2e), and that its cost cannot be locally improved (2f).

B. Overview of Gaussian processes

A GP is defined as a set of (potentially infinitely many) random variables, any finite number of which have a joint Gaussian distribution [19]. It is fully parameterized 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 a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, and assuming a noisy observation model $y_i \sim \mathcal{N}(f(x_i), \sigma^2)$, the predictive conditional posterior $\tilde{f}|\mathcal{D}$ is also a Gaussian if a GP is used as the prior. Performing inference at a finite set of points $\{z_j\}_{j=1}^k$, the posterior mean and covariance evaluated at the points are given by

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

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

where $\boldsymbol{Z}$, $\boldsymbol{X}$, and $\boldsymbol{Y}$ are vectors containing all elements in $\{z_j\}_{j=1}^k$, $\{x_j\}_{i=1}^n$, and $\{y_i\}_{i=1}^n$, respectively [13].

C. Problem statement

Given locally-optimal demonstrations $\{\xi_{xu}^j\}_{j=1}^{N_{\text{dem}}}$, we wish to learn an unknown constraint $\mathcal{S}(\theta) = \{\phi(x) \mid g_{\neg k}(\phi(x)) \leq \theta\}$, where $g_{\neg k} \sim \mathcal{GP}(m, k)$ is represented as a GP, that is consistent with the demonstrations' KKT conditions. Moreover, we wish to use the learned constraint to plan probabilistically-safe trajectories which connect novel start/goal states.

III. Method

Our method extracts salient information on the location and shape of the unknown constraint from the KKT conditions (Sec. III-A), uses this information to train a GP representation of the constraint (Sec. III-B), and plans novel probabilistically-safe trajectories using the learned constraint (Sec. III-C).

A. Obtaining constraint value and gradient information

For a demonstration, the KKT conditions (2) provide information on A) if and when a constraint is *tight* (i.e. at which time-steps the demonstration is touching the boundary of the unknown constraint) via complementary slackness (2e), and B) the shape of constraint locally around the active demonstration points (i.e. in the form of the gradient of the constraint at that point) via stationarity (2f). Combining both sources of information is crucial in recovering an accurate constraint.

We first describe a method for inferring when the constraint is tight. As shorthand, we denote $\mathbf{s}_j(\boldsymbol{\lambda}_k^j, \boldsymbol{\lambda}_{\neg k}^j, \boldsymbol{\nu}_k^j, \theta) \in \mathbb{R}^{|\xi_{xu}|}$ as the LHS of the stationarity condition (2f) for the jth demonstration ξ_j^{loc}. Recall that complementary slackness (2e) enforces that at each timestep, the Lagrange multiplier corresponding to the unknown inequality constraint must be zero unless the constraint is tight. Moreover, as any locally-optimal trajectory ξ^{loc} must satisfy the stationarity condition (2f), we can identify where the unknown constraint $g_{\neg k}(\cdot)$ must be tight on ξ_j^{loc} by finding a stationarity residual vector $\hat{\mathbf{s}}_j$ of minimal norm, under the restriction that $g_{\neg k}(\cdot)$ is never tight (see Fig. 1.A) and that the KKT conditions corresponding to the known constraints are satisfied. This is achieved by solving Prob. 2, which is a linear program (LP):

Problem 2 (Tightness identification on demonstration j)**.**

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

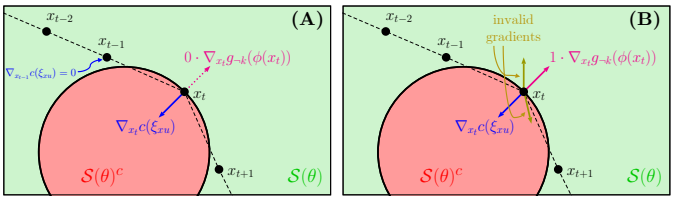

Fig. 1. Cartoon: demonstrator minimizes path length; system is kinematic. In this simplified setting, we can interpret (2f) as balancing between the vectors ∇c and $\lambda \nabla g_{\neg k}$; if they cancel out to $\mathbf{0}$, stationarity holds. We visualize these terms for Prob. 2-3. **(A)** Prob. 2: indices of $\mathbf{s}$ for x_t can only go to zero if $\lambda_{\neg k,t} \neq 0$; thus, we detect $g_{\neg k}(x_t) = \theta$. **(B)** Prob. 3: only a scaling of the magenta gradient can make $\mathbf{s}$ zero; the gold gradients cannot cancel ∇c.

where the effect of the unknown constraint on the residual is erased by zeroing out its corresponding Lagrange multipliers and parameters. By recovering the corresponding stationarity vector $\hat{\mathbf{s}}_j$ from the solution of Prob. 2, we can determine the timesteps $\hat{\mathbf{t}}$ at which $\mathbf{s}_j$ cannot be made zero. Note that as the unknown constraint is time-independent, taking the form (1), there is no trade-off between reducing the stationarity residual corresponding to one timestep instead of another. By the assumption that demonstration j exactly satisfies (2), $\mathbf{s}_j$ must be made identically zero; hence, the unknown constraint must be tight on timesteps $\hat{\mathbf{t}}$ corresponding to nonzero entries in $\hat{\mathbf{s}}_j$, i.e. $g_{\neg k}(\phi(x_t)) = \theta$ for all t in $\hat{\mathbf{t}}$. In contrast, we note that the KKT conditions do not provide information on the value of $g_{\neg k}(\phi(x_t))$ for nontight timesteps $\check{\mathbf{t}} \doteq \{1, \ldots, T\} \setminus \hat{\mathbf{t}}$; we can only infer that $g_{\neg k}(\phi(x_t)) \leq \theta$ for $t \in \check{\mathbf{t}}$.

Next, we introduce a method for obtaining a set of KKT-consistent gradients of the unknown constraint at each timestep where the constraint is tight. In Prob. 3, we fix the Lagrange mutipliers $\boldsymbol{\lambda}_{\neg k}^j = \mathbf{1}$, drop θ as it is removed by the gradient, and solve for $\nabla_{\xi_{xu}} g_{\neg k}$ as a decision variable:

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

$$\begin{aligned} \underset{\boldsymbol{\lambda}_k^j, \boldsymbol{\nu}_k^j, \nabla_{\xi_{xu}} g_{\neg k}}{\text{minimize}} \quad & \|\mathbf{s}_j(\boldsymbol{\lambda}_k^j, \mathbf{1}, \boldsymbol{\nu}_k^j, 0)\|_1 \\ \text{subject to} \quad & \text{(2a) – (2e).} \end{aligned}$$

Fixing $\boldsymbol{\lambda}_{\neg k}^j$ is required to avoid bilinearity in Prob. 3, which remains an LP. The returned $\nabla_{\xi_{xu}} g_{\neg k}$ are consistent with the KKT conditions (2) (see Fig. 1.B). Note that the recovered $\nabla_{\xi_{xu}} g_{\neg k}$ from Prob. 3 are *not* the only consistent gradients, because $\boldsymbol{\lambda}_{\neg k}^j$ can take values other than 1, and the resulting $\boldsymbol{\lambda}_{\neg k}^{j\top} \nabla_{\xi_{xu}} \mathbf{g}_{\neg k}(\xi_j^{\text{loc}})$ can still satisfy (2f) (e.g. resulting in a scaling of the returned gradient). We will address this non-uniqueness in future work (see Sec. V for discussion).

B. Embedding KKT-based information in a Gaussian process

After obtaining a set of tight points $\{x_i \in \mathcal{X}\}_{i=1}^k$ and the gradients of the unknown constraint at each tight point, we can construct the following dataset $\mathcal{D} = \{(x_i, \nabla g_{\neg k}(x_i)\}$ which we wish our learned GP to be consistent with. As the deriviative of a GP is also a GP [20], we can form a joint GP over the function values and their derivatives to perform inference using value $g_{\neg k}(x_i) = 0$ and gradient data $\nabla g_{\neg k}(x_i)$. For brevity, we refer to [22] for more details.

Since the demonstrations are safe (i.e. $g_{\neg k}(\phi(x_t)) \leq \theta$ for all t), we enforce this in the learned GP by calculating the posterior mean at every demonstration point (i.e. their constraint values $g_{\neg k}(\phi(x_t))$) and find θ by solving Prob. 4 with these fixed constraint values, which also is an LP:

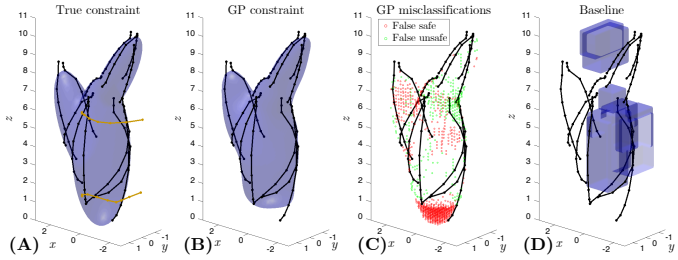

Fig. 2. 12D quadrotor obstacle example. Demonstrations (black). **(A)** True constraint (blue), novel plans using the GP constraint (gold). **(B)** Posterior mean of the learned GP constraint (blue). **(C)** GP misclassifications compared to the true constraint. **(D)** Baseline recovered constraint.

Problem 4 (θ calculation via inverse KKT)**.**

$$\begin{aligned} \underset{\theta,\boldsymbol{\lambda}_k^j,\boldsymbol{\lambda}_{\neg k}^j,\boldsymbol{\nu}_k^j}{\text{minimize}} \quad & \textstyle\sum_{j=1}^{N_{\text{dem}}} \left\|\mathbf{s}_j(\boldsymbol{\lambda}_k^j,\boldsymbol{\lambda}_{\neg k}^j,\boldsymbol{\nu}_k^j,\theta)\right\|_1 \\ \text{subject to} \quad & \text{(2a)} - \text{(2e)},\ \forall \xi_j^{\text{loc}},\ j = 1,\ldots,N_{\text{dem}} \end{aligned}$$

C. Planning with the learned constraint

We wish to find a trajectory connecting a pair of novel start/goal states which satisfies the learned constraint with probability $1-\delta$. We use a constrained kinodynamic RRT [16] to plan with the learned constraint, though it can also be used in an optimization-based planner. Given the posterior $\tilde{g}|\mathcal{D}$ and a candidate state x_c in the RRT, we can obtain $\mathbb{E}[\tilde{g}_{\neg k}(\phi(x_c))|\mathcal{D}]$ and $\text{Var}(\tilde{g}_{\neg k}(\phi(x_c))|\mathcal{D})$. As the posterior $\tilde{g}|\mathcal{D}$ is Gaussian, we can compute confidence intervals (CI) on $\tilde{g}_{\neg k}(\phi(x_c))|\mathcal{D}$ by buffering $\mathbb{E}[\tilde{g}_{\neg k}(\phi(x_c))|\mathcal{D}]$ by an appropriate scaling of the posterior standard deviation $\sigma_p \doteq \sqrt{\text{Var}(\tilde{g}_{\neg k}(\phi(x_c))|\mathcal{D})}$ (e.g. the 95% CI is $E[\tilde{g}_{\neg k}(\phi(x_c))|\mathcal{D}] \pm 1.96\sigma_p$). The probability of a plan being contained in safe set $\mathcal{S}(\theta)$, as defined by $g_{\neg k}(\cdot)$, with probability $1-\delta$ is given by

$$\Pr\Big(\textstyle\bigwedge_{i=1}^{T}(x_i \in \mathcal{S}(\theta))\Big) > 1-\delta. \tag{5}$$

We derive a sufficient condition for (5) via Boole's inequality:

$$\textstyle\bigwedge_{i=1}^{T}\big(\Pr(x_i \in \mathcal{S}(\theta)) > 1-\delta_i\big) \tag{6}$$

where δ_i is the probability of $x_i \notin \mathcal{S}(\theta)$ and $\sum_{i=1}^{T}\delta_i \leq \delta$. Thus, (5) can be decomposed as a conjunction of probabilities on each timestep of the trajectory [18], simplifying planning.

IV. Evaluation on a 12D Quadrotor

We evaluate our approach on a quadrotor to show that it scales to learn complex nonlinear constraints on high-dimensional systems. We are provided 13 demonstrations (Fig. 2.A, black) avoiding collisions with a treelike obstacle, which is represented as a union of three ellipsoids (Fig. 2.A, blue). We generate the demonstrations by solving trajectory optimization problems using IPOPT [21]. The dynamics and cost function used are as in [8] and [9], respectively. We wish to recover the tree obstacle, which is unknown to the learner and is assumed to be a function of only the position states x, y, z (i.e. $\phi(x)$ is only a function of these states). Crucially, we lack *a priori* knowledge on the structure of the constraint $g_{\neg k}(\cdot)$. To train the GP, we use GPytorch [12] with an RBF kernel and train the model with Adam [14] for 800 epochs at learning rate 0.5. We compare with a baseline method [9, Prob. 4] which handles unknown constraint structure by approximating the unknown constraint as a union of B axis-aligned boxes (as in [7, Sec. 4.4]). As B is unknown, we iteratively add boxes to the representation until the KKT conditions hold. Due to the complexity of the tree obstacle, B is too high for [9, Prob. 4] to be tractably solved to optimality; we use the best solution found within a one-hour time limit. We sweep over B, selecting $B = 10$: $B < 10$ results in a poor fit; $B > 10$ results in poor solutions found in the time limit.

We visualize our results in Fig. 2.B-C. By enforcing the GP to be consistent with the tightness and constraint gradient information, we learn a constraint which is visually accurate (Fig. 2.B). The misclassifications (Fig. 2.C) are in locations where there is a lack of tight demonstration states. This is reasonable, as we cannot expect the GP to be accurate far from the training data. In contrast, the baseline fails to recover a reasonable constraint (Fig. 2.D), failing entirely to cover the upper portion of the obstacle; moreover, the shape is highly inaccurate due to the limitations of axis-aligned boxes. We provide metrics on constraint accuracy in Table I by gridding the position space $x, y, z \in [-3,3] \times [-3,3] \times [0,12]$ and counting the percent of gridpoints which are incorrectly labeled safe (denoted False Safe) and incorrectly labeled unsafe (denoted False Unsafe). The GP mean is the most accurate when combining both metrics. Moreover, the "False Safe" percentage can be made smaller by buffering the GP constraint with the predictive uncertainty, though this is at the cost of conservativeness (higher "False Unsafe"). In contrast, the baseline incorrectly labels many more states as safe, which can lead to safety violations in execution when planning with the learned constraint. Finally, we plan from two novel start/goal pairs using the $2.33\sigma_p$-buffered GP constraint, yielding plans (Fig. 2.A, gold) that are safe with probability at least 0.9 and 0.91 (according to (6)) and are ultimately safe for the true constraint. This example suggests our method scales to complex nonlinear constraints on high-dimensional systems and requires minimal prior information on the unknown constraint.

	0 σ_p	1 σ_p	2 σ_p	2.33 σ_p	Baseline
False Safe (%)	1.106	0.061	0.002	0.002	16.781
False Unsafe (%)	0.481	17.149	59.661	67.649	0.386

TABLE I
GP misclassifications w.r.t. the true constraint

V. Discussion and Conclusion

We learn constraints from demonstrations with minimal *a priori* knowledge by leveraging both the informativeness of the KKT conditions and the flexibility of Gaussian processes. An evaluation on a 12D quadrotor shows that our method scales to learn complex constraints which prior methods cannot handle. We wish to address the following in future work:

- While our method can use *approximately* locally-optimal demonstrations by thresholding the stationarity residual, we wish to explore how different suboptimality models [15, 23] can inform the selection of this threshold.
- As Prob. 3 only returns one possible set of KKT-consistent gradients to train the GP, our method may underestimate the posterior uncertainty. While our plans remain safe in practice, we can address this by extracting multiple consistent gradients as in [8] and training the GP with linear constraints on the possible gradients [5].
- Boole's inequality in (5) can be conservative; we will study using the GP posterior to compute the LHS of (5).

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[2] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Ng. An application of reinforcement learning to aerobatic helicopter flight. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

[3] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57 (5):469–483, 2009.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] M.A. Chilenski, M. Greenwald, Y. Marzouk, N.T. Howard, A.E. White, J.E. Rice, and J.R. Walk. Improved profile fitting and quantification of uncertainty in experimental measurements of impurity transport coefficients using gaussian process regression. *Nuclear Fusion*, 55 (2):023012, jan 2015.

[6] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018. URL https://arxiv.org/abs/1812.07084.

[7] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning parametric constraints in high dimensions from demonstrations. In *CoRL 2019, Osaka, Japan*, pages 1211–1230, 2019. URL https://arxiv.org/abs/1910.03477.

[8] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations. In *CoRL 2020, Cambridge, MA, USA*, 2020. URL https://arxiv.org/abs/2011.04141.

[9] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning constraints from locally-optimal demonstrations under cost function uncertainty. *IEEE Robotics Autom. Lett.*, 5(2):3682–3690, 2020. doi: 10.1109/LRA.2020.2974427.

[10] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Explaining Multi-stage Tasks by Learning Temporal Logic Formulas from Suboptimal Demonstrations. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.097.

[11] Peter Englert, Ngo Anh Vien, and Marc Toussaint. Inverse kkt: Learning cost functions of manipulation tasks from demonstrations. *IJRR*, 36(13-14):1474–1488, 2017.

[12] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[13] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition, 2004.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980.

[15] Craig Knuth, Glen Chou, Necmiye Ozay, and Dmitry Berenson. Inferring obstacles and path validity from visibility-constrained demonstrations. *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2020. URL https://arxiv.org/abs/2005.05421.

[16] Steven LaValle. *Planning algorithms*. Cambridge university press, 2006.

[17] Marcel Menner, Peter Worsnop, and Melanie N. Zeilinger. Constrained inverse optimal control with application to a human manipulation task. *IEEE TCST*, 2019.

[18] Masahiro Ono, Lars Blackmore, and Brian C. Williams. Chance constrained finite horizon optimal control with nonconvex constraints. In *Proceedings of the 2010 American Control Conference*, pages 1145–1152, 2010. doi: 10.1109/ACC.2010.5530976.

[19] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[20] E. Solak, Roderick Murray-Smith, William E. Leithead, Douglas J. Leith, and Carl Edward Rasmussen. Derivative observations in gaussian process models of dynamic systems. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 1033–1040, 2002.

[21] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006.

[22] Anqi Wu, Mikio C. Aoi, and Jonathan W. Pillow. Exploiting gradients and hessians in bayesian optimization and bayesian quadrature, 2018.

[23] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1433–1438, 2008.