Improving Out-of-Distribution Generalization of Learned Dynamics by Learning Pseudometrics and Constraint Manifolds

Yating Lin[1], Glen Chou[2], and Dmitry Berenson[1]

***Abstract*— We propose a method for improving the prediction accuracy of learned robot dynamics models on out-of-distribution (OOD) states. We achieve this by leveraging two key sources of structure often present in robot dynamics: 1) sparsity, i.e., some components of the state may not affect the dynamics, and 2) physical limits on the set of possible motions, in the form of nonholonomic constraints. Crucially, we do not assume this structure is known *a priori*, and instead learn it from data. We use contrastive learning to obtain a distance pseudometric that uncovers the sparsity pattern in the dynamics, and use it to reduce the input space when learning the dynamics. We then learn the unknown constraint manifold by approximating the normal space of possible motions from the data, which we use to train a Gaussian process (GP) representation of the constraint manifold. We evaluate our approach on a physical differential-drive robot and a simulated quadrotor, showing improved prediction accuracy on OOD data relative to baselines.**

I. Introduction

A key component of the robot autonomy stack is the dynamics model, which predicts how the robot state changes given the current state and control. Since the dynamics of many systems are difficult to hand-model, a popular choice is to learn them from data using, e.g., neural networks (NNs) [1] or Gaussian processes (GPs) [2]. This learned model is then used for planning and control. A key assumption for these models to work well is that the training data is drawn in an independently and identically distributed (i.i.d.) fashion from the same distribution where the model will be deployed. This is often violated in robotics, where we may need to visit states and controls that are significantly different from the training data, referred to as out-of-distribution (OOD) inputs [3]. In OOD domains, the model accuracy can degrade and hamper the robot's performance, revealing a critical need for learned dynamics that generalize better on OOD inputs.

To work towards this goal, we explore two insights: that many robots 1) have *sparse* dynamics, i.e., not all state variables affect the dynamics, and 2) satisfy nonholonomic *constraints* arising from physics and design, e.g., a car cannot translate sideways. Enforcing that our learned model conforms to this information can improve accuracy, as naïvely-trained dynamics may predict highly non-physical behavior, especially in OOD domains. However, without copious *a priori* knowledge, it is difficult to know what sparsity pattern and constraints to prescribe. Moreover, idealized sparsity and hand-coded constraints can fail to hold due to hardware imperfections, or if the robot undergoes faults.

This work was supported in part by the Office of Naval Research Grant N00014-21-1-2118 and NSF grants IIS-1750489, IIS-2113401, and IIS-2220876. [1]University of Michigan {yatinlin, dmitryb}@umich.edu [2]Massachusetts Institute of Technology, gchou@mit.edu

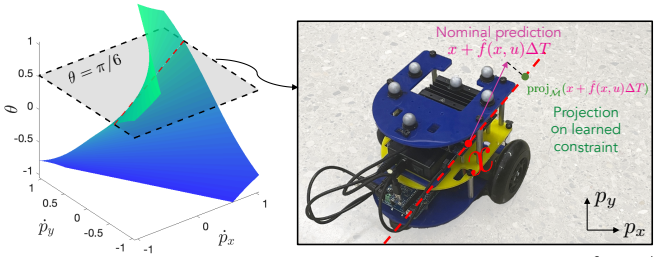

Fig. 1. **Left**: Visualization of the constraint manifold $\mathcal{M} = \{x, \dot{x} \mid -\dot{p}_x \sin(\theta) + \dot{p}_y \cos(\theta) = 0\}$ satisfied by the unicycle (17). **Right**: The unicycle dynamics are sparse, i.e., the set of possible velocities (red dotted line) is the same for all x with the same orientation. We learn the sparsity pattern and constraints (on a physical differential-drive robot), and *project* the predictions of our learned dynamics $\hat{f}(x,u)$ to be consistent with them.

In this paper, we strike a middle ground: by learning any sparsity and constraints from a small dataset, we can use them to adjust the learned dynamics to be more accurate than a naïve model, while also avoiding *a priori* prescription of structure. In particular, we hypothesize that learned constraints can generalize better than learned dynamics in OOD domains, as they are often defined in a lower-dimensional space where data-density is easier to achieve. Our method learns a distance pseudometric that uncovers the sparsity pattern in the dynamics, which we use to perform dimensionality reduction on the input of the learned dynamics. We then identify an implicit constraint manifold in the space of state and state derivatives satisfied by feasible transitions. We restrict attention to nonholonomic constraints in this work. Finally, to make predictions, we project the output of our learned dynamics onto the learned constraint. We contribute:

- a method for contrastive learning of distance pseudometrics to identify sparsity in dynamics and constraints
- a framework for approximate learning of nonholonomic constraints satisfied by the robot from trajectory data
- a method for improving the accuracy of learned dynamics by projecting its output to satisfy learned constraints
- evaluation on a physical differential-drive robot and simulated quadrotor, improving accuracy over baselines

II. Related Work

Many methods that use learned dynamics for control [2], [4] often fail far from training data. Some methods aim to mitigate this unreliability, e.g., [5], [6] bias planners away from unreliable model transitions. Other work [7]–[9] plans safely with learned dynamics by enforcing that the system remains within a bound of the training data. This bound explicitly limits model extrapolation; in contrast, our goal is to use the model far from data, by empirically maintaining accuracy via sparsification and learned constraints.

Other methods improve OOD generalization of learned dynamics via symmetry and physical constraints. For instance,

rotational symmetry of visual dynamics can be encoded via data augmentation (e.g., random crops) [10] or equivariant networks [11]. Other methods improve generalization with respect to task-irrelevant features of the input data (e.g., [12], [13]). While similar in motivation, these methods exploit the structure in image observations. Instead, our method directly learns structure in the dynamics via a sparsity pattern and constraints, complementing the above methods. Physics-informed learning has shown that enforcing Lagrange and Hamilton's equations can improve generalization in learning unconstrained [14]–[16] and constrained dynamics [17]–[19]. In the latter work, the functional form of the constraint is assumed to be known *a priori*, with parametric uncertainty. Our work sits between physics-constrained and unstructured learning, in that we discover constraints and sparsity directly from data, without assuming their existence or form *a priori*.

Finally, our work relates to constraint learning. [20] learns equality constraints, but is limited to holonomic constraints. Other methods [21]–[25] learn inequality constraints; but, unlike ours, require near-optimal demonstrations, and cannot learn from unstructured trajectory data, which is critical for dynamics-learning to cheaply improve data-density. The most similar method [23] uses GPs to learn nonlinear inequality constraints, but supervises the GP via constraint gradient data that is hard to obtain for dynamics-learning. Our focus also differs: instead of safety, we explore how learned constraints can empirically improve dynamics prediction.

III. Preliminaries and Problem Statement

Definitions: In this paper, we consider deterministic systems

$$\dot{x} = f(x, u), \tag{1}$$

where $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ and $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are the state and control space. The trajectories of (1) may implicitly satisfy constraints, e.g., a car cannot instantaneously move sideways. In this paper, we consider vector-valued *nonholonomic equality* constraints, i.e., $\mathbf{g}(x, \dot{x}) = \mathbf{0}$, which are implied by (1). Here, $\mathbf{g} : \mathcal{X} \times T_x(\mathcal{X}) \to \mathbb{R}^c$, where c is the number of constraints and $T_x(\mathcal{X})$ is an n-dimensional vector space at every x. We also assume the implicit constraint $\mathbf{g}(x, \dot{x}) = \mathbf{0}$ can be approximated as an affine function of $\dot{x}$:

$$G(x)\dot{x} + g(x) = 0 \iff \underbrace{\begin{bmatrix} G(x) & g(x) \end{bmatrix}}_{\doteq \Gamma(x)} \begin{bmatrix} \dot{x} \\ 1 \end{bmatrix} = 0, \tag{2}$$

where

$$G(x) = \begin{bmatrix} g_1^{(1)}(x) & \cdots & g_n^{(1)}(x) \\ & \vdots & \\ g_1^{(c)}(x) & \cdots & g_n^{(c)}(x) \end{bmatrix}; \; g(x) = \begin{bmatrix} g_0^{(1)}(x) \\ \vdots \\ g_0^{(c)}(x) \end{bmatrix}. \tag{3}$$

Constraints of the form (2) generalize Pfaffian constraints [26], which omit the bias terms $g_0^{(i)}(x)$, and includes a class of nonholonomic constraints [26]. These implicit constraints define a feasible manifold in the space of states/derivatives

$$\mathcal{M} \doteq \{(x, \dot{x}) \in \mathcal{X} \times T_x(\mathcal{X}) \mid \mathbf{g}(x, \dot{x}) = \mathbf{0}\}. \tag{4}$$

At a fixed x, we define $T_x\mathcal{M}$, of dimension $n - c$, as the set of admissible derivatives $\{\dot{x} \mid \exists u \in \mathcal{U}, f(x, u) = \dot{x}\}$. Similarly, we define the normal space $N_x\mathcal{M}$ [27], of dimension c, as $\{\dot{x} \mid \neg\exists u \in \mathcal{U}, f(x, u) = \dot{x}\}$.

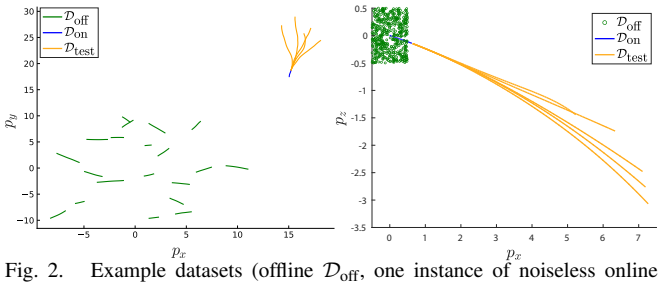

Fig. 2. Example datasets (offline $\mathcal{D}_{\text{off}}$, one instance of noiseless online $\mathcal{D}_{\text{on}}$, and the associated test $\mathcal{D}_{\text{test}}$) used for training and evaluation for the unicycle (17) (**left**) and quadrotor (18) (**right**). The test data $\mathcal{D}_{\text{test}}$ is far from the training data $\mathcal{D}$, and can be considered OOD w.r.t. that data (this is confirmed by decreased prediction accuracy on these inputs; see Sec. V).

Gaussian processes: In this paper, we use Gaussian processes (GPs) [28] for dynamics learning (though our method is also compatible with NNs or other function approximators). We give a brief overview of GPs below. Consider a training set $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$ are the training inputs and labels, and labels y are corrupted with additive Gaussian noise $\mathcal{N}(0, \sigma^2)$. Given test inputs $\mathbf{x}_*$, we can infer the posterior distribution of test labels using a GP (see [28] for details).

This process relies on a positive-definite kernel $k(\mathbf{x}, \mathbf{x})$ with hyper-parameters $\Theta = \{\sigma_1, l_1, ..., \sigma_d, l_d\}$. Here, σ_i, l_i are the variances and length-scales of dimension i, which can be learned by maximizing the log marginal likelihood of D. For multi-dimensional labels y, we model each output dimension independently with a scalar-output GP, which we refer to as independent GPs (IGPs).

Problem statement: We assume the dynamics (1) and constraint (4) are unknown, but that we are given a dataset $\mathcal{D}$ generated by the system. We divide the dataset $\mathcal{D}$ into offline (online) datasets $\mathcal{D}_{\text{off}}$ ($\mathcal{D}_{\text{on}}$). $\mathcal{D}_{\text{off}} = \{\xi_{xu}^{(n)}, \dot{\xi}^{(n)}\}_{n=1}^N$ contains N state-control trajectories $\xi_{xu}^{(n)} \doteq (x_1^{(n)}, u_1^{(n)}, \ldots, x_T^{(n)})$, each sampled at T time indices, and corresponding derivatives $\dot{\xi}^{(n)} \doteq (\dot{x}_1^{(n)}, \ldots, \dot{x}_T^{(n)})$ collected offline. During online deployment, we execute one control trajectory $u_{\text{on}}(t)$ and collect the data observed online at regular time instants, which constitutes the online dataset $\mathcal{D}_{\text{on}} = (\xi_{xu}^{\text{on}}, \dot{\xi}^{\text{on}})$.

Using $\mathcal{D}$, we train a dynamics model $\dot{x} = f(x, u)$. Inputs far from $\mathcal{D}$ are referred to as out of distribution (OOD). Our goal is to improve prediction accuracy on states that are OOD with respect to $\mathcal{D}$; we refer to these inputs as $\mathcal{D}_{\text{test}}$. See Fig. 2 for examples of $\mathcal{D}_{\text{off}}$, $\mathcal{D}_{\text{on}}$, and $\mathcal{D}_{\text{test}}$ used in Sec. V.

IV. Method

Our method is shown in Fig. 3. We first learn nominal GP dynamics, where the GP input space is sparsified by a learned pseudometric (Sec. IV-A). Next, we learn a sparsified GP for the constraints (Sec. IV-B), using approximate data (Sec. IV-B.2). Finally, we predict by projecting the nominal dynamics onto the learned constraint (Sec. IV-C).

A. Learning sparse dynamics

We describe a framework for obtaining a pseudometric via contrastive learning (Sec. IV-A.1), how to specialize the

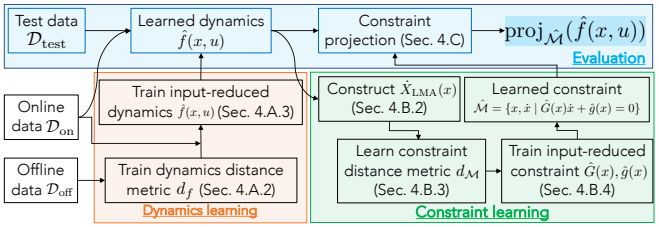

Fig. 3. Method. **Dynamics learning**: Given offline data $\mathcal{D}_{\text{off}}$, we learn a dynamics pseudometric d_f, which we use to reduce the input space dimension of the dynamics model $\hat{f}(x,u)$. We train $\hat{f}(x,u)$ on the input-reduced offline data. **Constraint learning**: We use approximate normal space data to learn a constraint distance pseudometric $d_{\mathcal{M}}$, which we use to reduce the constraint input space. We train the constraint $\hat{\mathcal{M}}$ using input-reduced offline and online data, paired with the normal space data. **Evaluation**: For prediction, we evaluate $\hat{f}(x,u)$ and project it onto $\hat{\mathcal{M}}$.

method to identify the sparsity pattern of the dynamics (Sec. IV-A.2), and how we use this knowledge to train dynamics which are sparse in their input space (Sec. IV-A.3).

1) Contrastive learning of pseudometrics: We discuss a framework for learning pseudometrics, and specialize it in Sec. IV-A.2 and IV-B.3 to dynamics and constraint learning. Pseudometrics are distance metrics without the restriction that unique elements of the input space have nonzero distance, i.e., for all z, z', $z'' \in \mathcal{Z}$, a pseudometric $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}$ satisfies: 1) $d(z,z) = 0$, 2) $d(z,z') = d(z',z)$, and 3) $d(z,z'') \leq d(z,z') + d(z',z'')$ [29]. For a generic dataset $D = \{z_i\}_{i=1}^{Z}$, $z_i \in \mathcal{Z}$, we parameterize a pseudometric $d_\theta : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}$ with parameters $P_\theta \in \mathbb{S}_+^{(n+m)\times(n+m)}$, where $\mathbb{S}_+$ is the space of positive semi-definite matrices,

$$d_\theta(z_1, z_2) = (z_1 - z_2)^\top P_\theta (z_1 - z_2), \tag{5}$$

In general, we can search over input-dependent metrics (i.e., P_θ is a function of z); for simplicity, we restrict attention to the case where P_θ is constant. To learn the pseudometric $d_\theta(\cdot,\cdot)$, we use contrastive learning [30]. In the context of metric learning, contrastive learning takes positive and negative pairs as inputs. Here, positive pairs are close in the output space and are above a distance threshold in the input space, and negative pairs are all others (see Sec. IV-A.2). Contrastive learning encourages $d_\theta(\cdot,\cdot)$ to evaluate to small (large) values for positive (negative) pairs. We specifically use the popular InfoNCE loss [31], defined as

$$\mathcal{L}_{\text{con}} = -\mathbb{E}_{z\sim D}\left[\log \frac{\exp(\text{sim}(z, z^p))}{\sum_{k=1}^{N_b} \exp(\text{sim}(z, z_k^n))}\right] \tag{6}$$

where (z, z^p) is a positive pair, (z, z_k^n) is one of N_b negative pairs, and $\text{sim}(\cdot,\cdot)$ is a similarity metric (we use cosine similarity, commonly used in contrastive learning [32], which is a discriminative distance metric in high-dimensional spaces),

$$\text{sim}(z, z') = \frac{z^T P_\theta z'}{\sqrt{z^\top P_\theta z}\sqrt{z'^\top P_\theta z'}} \tag{7}$$

Then, we can train such a pseudometric by minimizing $\mathcal{L}_{\text{con}}$.

2) Learning a pseudometric for sparsifying the dynamics: We employ the framework of Sec. IV-A.1 to learn a pseudometric that extracts the sparsity pattern of (1) from data. Consider a pseudometric that assigns distances between (x,u) and (x',u') to be small (large) if they have (dis)similar derivatives $f(x,u)$ and $f(x',u')$. Then, if the dynamics are invariant along any directions v, i.e., $f(x,u) = f(x+\alpha v, u)$, $\forall \alpha \in \mathbb{R}$, an ideal pseudometric would assign $d((x,u),(x+\alpha v,u)) = 0$. For a pseudometric of the form (5), we can recover these directions v as the eigenvectors of P_θ corresponding to zero eigenvalues. When P_θ is diagonal (as in the results), each eigenvector corresponds to a specific dimension of the input. We discuss how to use this knowledge for dynamics learning in Sec. IV-A.3. To actually obtain such a pseudometric, we can leverage the contrastive learning framework of Sec. IV-A.1. In particular, we select the input z as the state-control space (x,u). We define the positive pair corresponding to (x_i,u_i) as (x_{i^*},u_{i^*}), where $i^* = \arg\min_{j\in\mathcal{D}} \|f(x_i,u_i) - f(x_j,u_j)\|$ such that $\|(x_i,u_i) - (x_{i*},u_{i*})\| \geq \epsilon_1$. Negative pairs are selected as the other non-minimum pairs in the batch. Training the dynamics pseudometric d_f is done by minimizing (6).

3) Learning the sparse dynamics model: By following Sec. IV-A.2, we can obtain a pseudometric that sets distances along invariant directions to zero. For diagonal P_θ, we can drop out input dimensions corresponding to diagonal entries equal to zero when learning the dynamics. We refer to the reduced input as (x_{red}, u); note that we only reduce the input, not the output, which remains in $\mathbb{R}^n$. If we have noisy data or an imperfect pseudometric, we can drop out inputs if the diagonal entries in P_θ lie below some threshold. Intuitively, this performs principal component analysis on the input data, removing input "features" that are not predictive for the dynamics. If the removed states globally do not affect the dynamics (reasonable for smooth systems with enough data), we can expect the reduced model to generalize better in OOD scenarios, as it has fewer inputs upon which to be OOD. We show in Sec. V that this reduction improves predictions in practice. Similar logic applies for non-diagonal P_θ, e.g., we can eigen-decompose P_θ, rotate the data using the eigenbasis, and remove dimensions of the rotated data with small eigenvalues. We learn the sparsified dynamics with a GP on the reduced space, where the data $(x_i, u_i, f(x_i,u_i))$ is used to train an IGP by maximizing log-likelihood, giving the learned model

$$\dot{x} = \hat{f}(x_{\text{red}}, u). \tag{8}$$

We make predictions via the posterior mean of the GP.

B. Learning the constraint manifold

We describe our method for learning the constraint manifold (4). First, we detail our constraint formulation (Sec. IV-B.1) and discuss challenges in identifying the constraints from finite data. We discuss how to overcome these challenges using synthetic data (Sec. IV-B.2), and how we can improve generalization of the learned constraint by learning a constraint pseudometric (Sec. IV-B.3). Finally, given this, we show how to learn the constraint manifold (Sec. IV-B.4).

1) Constraint formulation: As discussed in Sec. III, we assume the unknown constraint can be represented as in (2) as an affine function of $\dot{x}$. Thus, the constraint learning problem reduces to learning $\Gamma : \mathcal{X} \to \mathbb{R}^{c\times n+1}$. For systems of interest, $c < n$, and controls can be taken to generate

state derivatives $\dot{x}$ which satisfy (2). In principle, to recover $\Gamma(x)$, we can apply different control actions u_i at a fixed state x until we obtain $n-c$ linearly independent augmented derivatives $\{[f(x,u_i)^\top \ 1]\}_{i=1}^{n-c}$. For concreteness, we denote the concatenation of these vectors as $\dot{X}(x) \in \mathbb{R}^{(n-c)\times(n+1)}$:

$$\dot{X}(x) = \begin{bmatrix} f(x,u_1) & \cdots & f(x,u_{n-c}) \\ 1 & \cdots & 1 \end{bmatrix}^\top. \tag{9}$$

Then, we can compute a basis for the null space of $\dot{X}(x)$ to recover $G(x)$, $g(x)$. However, it is difficult to reset the state of a robot with non-trivial dynamics in a way that different actions can be taken at the exact same x. Thus, we propose a method to approximate $G(x)$ and $g(x)$ using synthetic data.

2) Approximating the normal space: To estimate $\dot{X}(x)$, we use the learned $\hat{f}$ as a proxy for f to compute a synthetic Learned Multi-Action (LMA) dataset, for $K \geq n-c$:

$$\dot{X}_{\text{LMA}}(x) = \begin{bmatrix} \hat{f}(x,u_1) & \cdots & \hat{f}(x,u_K) \\ 1 & \cdots & 1 \end{bmatrix}^\top. \tag{10}$$

Given an approximation of $\dot{X}(x)$, we can compute its null space to approximate $\Gamma(x)$. We do this by applying the singular value decomposition (SVD) to $\dot{X}_{\text{LMA}}(x)$,

$$\dot{X}_{\text{LMA}}(x) = U\hat{\Sigma}V^\top = [U_1 \ U_2] \begin{bmatrix} \hat{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \hat{\Sigma}_2 \end{bmatrix} \begin{bmatrix} V_1^\top \\ V_2^\top \end{bmatrix}, \tag{11}$$

where Σ_2 contains all singular values less than a threshold ϵ, and V_2 is an approximate basis for the null space of $\dot{X}(x)$, i.e., $\Gamma(x) = V_2^\top$. In general, ϵ should be positive to handle any approximation error in $\dot{X}(x)$ on $\Gamma(x)$, and can also be used to identify the number of constraints c. However, since the SVD is not unique [33], directly using V_2 to represent $\Gamma(x)$ can cause challenges for learning, as V_2 may not change smoothly with x. While in general, no smoothly-varying basis exists [34], the following method is empirically effective. We first convert $V_2^\top$ to reduced row echelon form (rref); and define $\tilde{\Gamma}^{(i)}(x)$ as the ith row of $\text{rref}(V_2^\top(x))$, which smoothly standardizes the basis up to a sign-change. To keep signs consistent, at a given x, we find the nearest datapoint x_{ref}, and negate the vectors in $\text{rref}(V_2^\top(x))$ if their dot products with $\text{rref}(V_2^\top(x_{\text{ref}}))$ lie below zero. We call this approximate normal space basis $\Gamma_{\text{approx}}(x)$, where the ith row is:

$$\Gamma^{(i)}_{\text{approx}}(x) \doteq \begin{cases} \tilde{\Gamma}^{(i)}(x), & \text{if } \tilde{\Gamma}^{(i)}(x)\tilde{\Gamma}^{(i)}(x_{\text{ref}})^T \geq 0 \\ -\tilde{\Gamma}^{(i)}(x), & \text{if } \tilde{\Gamma}^{(i)}(x)\tilde{\Gamma}^{(i)}(x_{\text{ref}})^T < 0 \end{cases} \tag{12}$$

We can then form a dataset $\{x_i, \Gamma_{\text{approx}}(x_i)\}_{i=1}^{|\mathcal{D}|}$ to train a GP constraint in Sec. IV-B.4. Before training, we discuss how to learn a pseudometric (similar to Sec. IV-A.2) to reduce the input space of the constraint, improving generalization.

3) Training the constraint distance pseudometric: For the learned constraint to be useful in aiding OOD generalization of the dynamics, the constraints must also be accurate OOD. To improve generalization as is done in Sec. IV-A.2, we learn c distance pseudometrics, one for each of the c rows of $\Gamma(x)$, with the aim of reducing the input space when learning the constraint. Here, we take the contrastive learning framework of Sec. IV-A.1, and let $z = x$. Denote the distance pseudometric for to the ith row of $\Gamma(x)$, $\Gamma^{(i)}(x)$, as

$$d_{\mathcal{M}^{(i)}}(x,x') = (x-x')^\top P_{\mathcal{M}^{(i)}}(x-x'), \tag{13}$$

For each datapoint x_i, we select its positive pair as the point x_i^p in $\mathcal{D}$ minimizing $\|\Gamma^{(i)}_{\text{approx}}(x_i) - \Gamma^{(i)}_{\text{approx}}(x_i^p)\|$ satisfied $\|x_i - x_i^p\| \geq \epsilon_2$. Negative pairs are selected as the remaining non-minimum states in the batch. We train each pseudometric by minimizing (6).

4) Learning the constraint on reduced input space: Similar to Sec. IV-A.3, for each row of $\Gamma(x)$, we drop out the inputs that correspond to a zero eigenvalue. We refer to the reduced input state for row i as x^i_{red}, $i = 1, \ldots c$. We model each entry of the normal space matrix using an IGP, where each is a function of its corresponding reduced input space. Specifically, we follow the framework of Sec. III, where the training data and labels is comprised of states x_i and approximate normal space bases $\Gamma_{\text{approx}}(x_i)$ (see (12)), respectively, where x_i are drawn from the dataset $\mathcal{D}$ described in Sec. III. Using this data, we train an IGP by maximizing log-likelihood.We call this learned constraint $\hat{\Gamma}(x) = [\hat{G}(x) \ \ \hat{g}(x)]$, and the corresponding manifold,

$$\hat{\mathcal{M}} \doteq \{(x,\dot{x}) \in \mathcal{X} \times T_x(\mathcal{X}) \mid \hat{G}(x)\dot{x} + \hat{g}(x) = 0\}, \tag{14}$$

which is evaluated via the trained GP's posterior mean.

C. Generating predictions at evaluation time

At runtime, we wish to make predictions with the learned dynamics $\dot{x}_{\text{pred}} = \hat{f}(x,u)$ (8) that conform with the learned constraint $\hat{\mathcal{M}}$ (14). Since for a fixed x, the constraint (14) is affine in $\dot{x}$, we can *project* the prediction $\dot{x}_{\text{pred}}$ from the nominal dynamics (8) to be consistent with $\hat{\mathcal{M}}$ by solving:

$$\begin{aligned} \text{proj}_{\mathcal{M}}(\dot{x}_{\text{pred}}) \doteq \underset{\dot{x}}{\text{argmin}} \quad & \|\dot{x} - \dot{x}_{\text{pred}}\|_2^2 \\ \text{subject to} \quad & \hat{G}(x)\dot{x} + \hat{g}(x) = 0, \end{aligned} \tag{15}$$

which can be efficiently done online by solving a quadratic program (QP). We use (15) for prediction in Sec. V.

V. Results

We evaluate our method on a simulated unicycle, physical differential-drive robot (DDR), and simulated quadrotor. First, we describe our baselines, ablations, method variants, and test settings. Our baseline is a standard GP trained on $\mathcal{D}$ ("GP" in Tab. I-III) (i.e., no sparsity or constraint). We refer to our sparsified GP (i.e., with dropped inputs) as "DGP". Our method, which also projects onto the constraint learned via (11), is "DGP + Approx.", the ideal version of our full method (which projects onto the constraint learned via $\dot{X}(x)$ (9)) is "DGP + Ideal", and an ablation which sparsifies but does not project is "DGP + None". "ID" evaluates each method on in-distribution (ID) test data; "OOD" on OOD test data. We consider test data as ID if it lies in the support of the distribution that the offline data was sampled from, and OOD otherwise. For the OOD experiments, we evaluate all methods when given both offline and online data, i.e., where $\mathcal{D} = \mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{on}}$. Finally, to show robustness to noise, we evaluate on OOD data where $\mathcal{D}_{\text{off}}$ is noiseless but $\mathcal{D}_{\text{on}}$ is corrupted with Gaussian measurement noise $N(0, \eta I)$. We test with $\eta = 0.05$ and $\eta = 0.1$. All dynamics GPs are trained using a radial basis function (RBF) kernel. For the corrupted dataset we also employed low-pass filtering to enhance

		None	Ideal	Approx.
ID	GP	4.79 ± 4.17	3.63 ± 2.53	4.05 ± 3.40
	DGP	2.96 ± 2.99	2.56 ± 2.21	3.03 ± 3.19
OOD, $\eta = 0.0$	GP	9.08 ± 1.14	7.70 ± 1.01	7.82 ± 1.25
	DGP	3.94 ± 0.48	3.51 ± 0.44	3.76 ± 0.71
OOD, $\eta = .05$	GP	43.58 ± 25.27	41.66 ± 26.47	43.42 ± 24.80
	DGP	35.52 ± 25.10	33.46 ± 25.90	35.88 ± 25.03
OOD, $\eta = .10$	GP	56.05 ± 37.13	55.41 ± 35.85	54.05± 31.73
	DGP	45.44 ± 30.51	44.34 ± 29.79	45.29 ± 28.87

TABLE I

100-STEP PREDICTION ERROR (UNICYCLE). DATA: $\mathcal{D}_{\text{ON}} \cup \mathcal{D}_{\text{OFF}}$.

robustness against high-frequency noise. We simulate with timestep $\Delta T = 0.01$ for the quadrotor and $\Delta T = 0.1$ otherwise, and report cumulative prediction error,

$$\mathcal{L}_{\text{err}} = \sum_{t=0}^{T} \|x_t^{\text{true}} - x_t^{\text{pred}}\|_2. \tag{16}$$

Nonholonomic unicycle ($x \in \mathbb{R}^3$, $u \in \mathbb{R}^2$): The dynamics are

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & \cos\theta \\ 0 & \sin\theta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix}, \tag{17}$$

where $u = (\omega, v)$ are controls (angular, linear velocity), p_x, p_y are the position, and θ is the heading. The unicycle satisfies the constraint $-\dot{p}_x \sin(\theta) + \dot{p}_y \cos(\theta) = 0$, i.e., $G(x) = [-\sin(\theta) \quad \cos(\theta) \quad 0]$ and $c = 1$. We learn a dynamics model with input $[x,\ u] = [p_x, p_y, \theta, \omega, v]^\top \in \mathbb{R}^5$ and output $y = [\dot{p}_x, \dot{p}_y, \dot{\theta}]^\top \in \mathbb{R}^3$. As the dynamics are only affected by θ and u, we learn a dynamics pseudometric (5) with zero ϵ_1, giving $P_f = \text{diag}([0, 0, 1.15, 1.66, 1.63])$ (cf. Sec. IV-A.2). We use P_f to remove p_x and p_y from the input when learning the dynamics (8) . We also learn the constraint pseudometric (13) with zero ϵ_2 , where $P_{\mathcal{M}} = \text{diag}([0, 0., 1.59])$. Thus, we remove p_x and p_y from the input when learning the constraint (14), for which we use an RBF kernel. The offline dataset $\mathcal{D}_{\text{off}}$ contains 20 trajectories, generated by applying 25 random controls in $[\omega, v] \in [-0.2, 0.2] \times [0, 1.5]$, starting from initial conditions in $[p_x, p_y, \theta] \in [-10, 10]^2 \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. For the online dataset $\mathcal{D}_{\text{on}}$, we sample an initial state from $[p_x, p_y, \theta] \in [15, 20]^2 \times [\frac{\pi}{3}, \frac{\pi}{2}]$, and roll out by sampling 5 controls in $[\omega, v] \in [-0.2, 0.2] \times [0, 1.5]$, and holding each for 5 steps, giving 25 total datapoints. Finally, starting from the end of the online trajectory, we test the $20 \times 5 = 100$ multi-step prediction error (20 actions, sampled from $[\omega, v] \in [-0.2, 0.2] \times [0, 1.5]$, each held for 5 steps), and generate 5 such trajectories for evaluation. $\mathcal{D}_{\text{on}}$ is generated 10 times, each starting from a unique initial state; test data is also regenerated. See Fig. 2 (left) for $\mathcal{D}_{\text{off}}$, one of $\mathcal{D}_{\text{on}}$, and the associated $\mathcal{D}_{\text{test}}$. The prediction accuracies, averaged over all online datasets and test trajectories, are in Table I.

When given both $\mathcal{D}_{\text{off}}$ and $\mathcal{D}_{\text{on}}$ to train the dynamics and constraint, we see for ID predictions that all method variants have relatively similar performance, with some small improvements from DGP and constraint projection. This is not surprising, as the baseline GP is already quite accurate and confirms that projection or sparsification does not hurt performance where data is plentiful. For OOD predictions, the sparse DGP is uniformly more accurate than the GP baseline, for all the noise levels tested. When further combining DGP with projection, "DGP + Ideal" performs best on the noiseless case. There is minor degradation for "DGP + Approx." due to the error in the training data, but it still is more accurate than the variants with no projection. With larger magnitudes of measurement noise, DGP still performs uniformly better than the GP variants, while the improvement due to projection shrinks for higher noise. This is likely due to the long prediction horizon in this problem, which makes it critical to estimate the constraint with high accuracy, which is more difficult to achieve with noisy data, whereas the sparsity pattern can be more easily estimated. We visualize the true and predicted trajectories in Fig. 4 (left). Overall, these results suggest that both sparsifying and projection can improve prediction accuracy, in spite of noisy data.

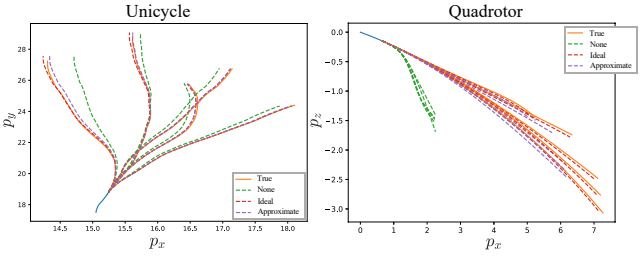

Fig. 4. Some predicted trajectories for the unicycle (**left**) and quadrotor (**right**) on OOD data. The given (noiseless) online data $\mathcal{D}_{\text{on}}$ is in blue. "True": ground truth trajectory. The predicted trajectories are: "None": GP, no projection. "Ideal": DGP, with projection on ideal learned constraint. "Approximate": DGP, with projection on approximate learned constraint.

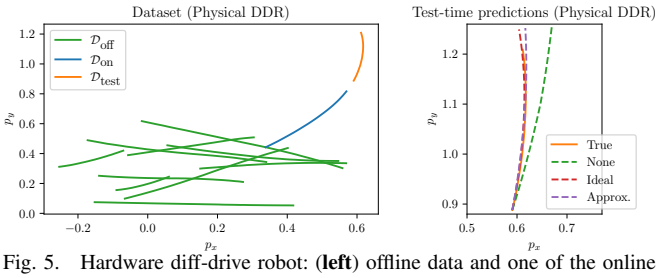

Fig. 5. Hardware diff-drive robot: (**left**) offline data and one of the online and associated test datasets; (**right**) examples of predicted trajectories.

Hardware DDR: To show our method's robustness to non-idealized constraints, we evaluate on a physical DDR (cf. Fig. 1, right), where we assume the same state/control space as the unicycle and data is collected using a Vicon mocap system. We learn a dynamics pseudometric $P_f = \text{diag}([0, 0.04, 1.72, 0.43, 0.44])$ and constraint pseudometric $P_{\mathcal{M}} = \text{diag}([0, 0, 1.60])$ with ϵ_1 and ϵ_2 set to 0.5, so we drop out p_x, p_y from the input in both cases as their diagonal values fall below a threshold. We use a cosine kernel when learning the constraint. The offline dataset $\mathcal{D}_{\text{off}}$ contains 10 trajectories, generated by applying 2 random controls from $(w, v) \in [-0.2, 0.2] \times [0, 0.2]$ and with initial conditions from $(p_x, p_y, \theta) \in [-0.2, 0.6] \times [0.05, 0.6] \times [-0.6, 0.8]$. For the online dataset $\mathcal{D}_{\text{on}}$, we sample from the same u range, but with initial conditions from $(p_x, p_y, \theta) \in [-0.4, 0.6] \times [0.4, 1.2] \times [-1.5, 1.7]$. We generate $\mathcal{D}_{\text{on}}$ 8 times, and generate 1 associated $\mathcal{D}_{\text{test}}$ per online dataset. See Fig. 5 for the datasets and predicted trajectories, and Tab. II for prediction accuracies, averaged over all online datasets and test trajectories. We use both $\mathcal{D}_{\text{off}}$ and $\mathcal{D}_{\text{on}}$ to train the dynamics and constraint. For the ID range, all methods are comparable; this is not surprising, as the baseline model is already quite accurate, moreover supporting that our method does not degrade predictions near data. With evaluation in the OOD

		None	Ideal	Approx.
ID	GP	0.60 ± 8e-4	0.61 ± 1e-7	0.61 ± 8e-4
	DGP	0.50 ± 5e-3	0.52 ± 6e-4	0.51 ± 4e-3
OOD, $\eta = 0.0$	GP	1.33 ± 0.62	1.03 ± 0.37	1.04 ± 0.37
	DGP	0.75 ± 0.32	0.68 ± 0.27	0.70 ± 0.25
OOD, $\eta = .05$	GP	1.68 ± 0.93	0.90 ± 0.35	0.95 ± 0.33
	DGP	1.58 ± 0.78	0.83 ± 0.27	0.90 ± 0.25
OOD, $\eta = .10$	GP	2.15 ± 0.88	1.42 ± 0.57	1.37 ± 0.46
	DGP	1.85 ± 0.57	1.38 ± 0.42	1.33 ± 0.34

TABLE II

40-STEP PREDICTION ERROR (HARDWARE DDR). DATA: $\mathcal{D}_{\text{ON}} \cup \mathcal{D}_{\text{OFF}}$.

		None	Ideal	Approx.
ID	GP	17.58 ± 16.11	8.75 ± 7.27	10.75 ± 9.99
	DGP	9.43 ± 8.25	3.15 ± 2.29	6.20 ± 5.85
OOD, $\eta = 0.0$	GP	87.91 ± 77.34	35.21 ± 29.30	71.64 ± 69.15
	DGP	66.93 ± 68.50	23.79 ± 28.19	64.78 ± 72.08
OOD, $\eta = .05$	GP	187.58 ± 92.03	73.21 ± 34.43	113.98 ± 78.24
	DGP	156.14 ± 97.32	41.48 ± 37.08	85.36 ± 86.06
OOD, $\eta = .10$	GP	216.14 ± 95.72	93.91 ± 51.90	122.68 ± 73.83
	DGP	181.81 ± 104.44	58.85 ± 58.95	90.15 ± 82.83

TABLE III

100-STEP PREDICTION ERROR (QUADROTOR). DATA: $\mathcal{D}_{\text{ON}} \cup \mathcal{D}_{\text{OFF}}$.

range, DGP is uniformly more accurate than the GP baseline (cf. Tab. II), again supporting that sparsity improves OOD accuracy. Constraint projection further improves prediction accuracy, where in Tab. II, the "Ideal" column is computed by projecting onto the unicycle constraint $-\dot{p}_x \sin(\theta) + \dot{p}_y \cos(\theta) = 0$. Crucially, as the hardware can violate this ideal constraint, this shrinks the accuracy gap between the ideal and approximate projection. Finally, as more noise is injected into $\mathcal{D}_{\text{on}}$, all methods degrade, but DGP with projection (our full method) remains more accurate than baselines without sparsity or projection, and for $\eta = 0.10$ slightly outperforms the ideal case. Overall, this suggests that in spite of the approximate data, constraint-learning can be valuable compared to pre-specifying constraints that may fail to hold in reality.

Planar quadrotor ($x \in \mathbb{R}^6$, $u \in \mathbb{R}^2$): The quadrotor satisfies

$$\begin{bmatrix} \dot{p_x} \\ \dot{p_z} \\ \dot{\phi} \\ \dot{v_x} \\ \dot{v_z} \\ \dot{\omega_\phi} \end{bmatrix} = \begin{bmatrix} v_x \\ v_z \\ \omega_\phi \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}\sin(\phi) & -\frac{1}{m}\sin(\phi) \\ \frac{1}{m}\cos(\phi) & \frac{1}{m}\cos(\phi) \\ \frac{1}{J} & -\frac{1}{J} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad (18)$$

with positions, p_x, p_z, velocities v_x, v_z, orientation ϕ, and angular velocity ω_ϕ, and $m = 0.486$, $l = 0.25$, and $J = 0.0383$. There are $c = 4$ constraints, analytically derived as:

$$\Gamma(x) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -v_x \\ 0 & 1 & 0 & 0 & 0 & 0 & -v_z \\ 0 & 0 & 1 & 0 & 0 & 0 & -\omega_\phi \\ 0 & 0 & 0 & 1 & \tan(\phi) & 0 & g\tan(\phi) \end{bmatrix}. \quad (19)$$

We learn $P_f = \text{diag}([0, 0, 1.2, 1.4, 1.4, 1.3, 1.1, 1.0])$ with zero ϵ_1. Thus, we remove p_x and p_z from the input when learning the dynamics (8). We also learn c constraint pseudometrics (13) with zero ϵ_2, where $P_{\mathcal{M}^{(i)}}$ recover the sparsity patterns of x ($P_{\mathcal{M}^{(i)}}, i = 1, ..., 4$, which only have nonzero entries on their 4th, 5th, 6th, and 3rd diagonals, respectively). We use this to drop out the associated inputs when learning the constraint (14), where we use a linear kernel. The offline dataset $\mathcal{D}_{\text{off}}$ contains 5000 trajectories, generated by applying 2-step random u from $[\frac{mg}{2}, 2.5] \times [\frac{mg}{2}, 2.5]$, with initial conditions sampled from $[p_x, p_y, \phi, v_x, v_z, \omega_\phi] \in [-1, 1] \times [-1, 1] \times [-\frac{\pi}{3}, \frac{\pi}{3}] \times [0, 5] \times [-5, 0] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. For $\mathcal{D}_{\text{on}}$, we sample 5 control actions from the same set, each held for 10 steps, which typically takes the system far from $\mathcal{D}_{\text{off}}$ (cf. Fig. 2). We test the $10 \times 10 = 100$ multi-step prediction error (10 random actions held for 10 steps each) on 5 such trajectories (this is $\mathcal{D}_{\text{test}}$). We regenerate $\mathcal{D}_{\text{on}}$ and the associated $\mathcal{D}_{\text{test}}$ for 10 initial conditions, and give statistics in Tab. III.

In the ID case, even though the baseline is already quite accurate, projection onto the ideal and approximate constraints both improve performance ≈2-fold. Moreover, our approximate normal space training data does not overly degrade accuracy relative to the ideal case, as the "Ideal" and "Approx." variants achieve comparable accuracy. On the other hand, sparsification hurts accuracy slightly, likely since with more inputs, the model can overfit better ID.

For the OOD case, we shrink the training range to be $[p_x, p_y, \phi, v_x, v_z, \omega_\phi] \in [-0.5, 0.5] \times [-0.5, 0.5] \times [-\frac{\pi}{4}, \frac{\pi}{4}] \times [0, 2] \times [-2, 0] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ and randomly sample initial conditions for the online data from $[p_x, p_y, \phi, v_x, v_z, \omega_\phi] \in [-1, 1] \times [-1, 1] \times [-\frac{\pi}{3}, \frac{\pi}{3}] \times [0, 5] \times [-5, 0] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, such that the test data is OOD. In this case, both sparsification and constraint projection improves accuracy, with projection onto the ideal constraint improving accuracy ≈3-fold. Moreover, despite errors in the constraint training data, projection onto the approximate constraint still leads to improvement over the no-projection case. As increasing levels of noise are added to $\mathcal{D}_{\text{on}}$, as before, all methods degrade, but our approach (DGP+Approx.) retains an improvement over "GP + None". In fact, the improvement that "DGP+Approx" enables over "DGP+None" increases for larger noise; we hypothesize this is because the nominal dynamics are much more inaccurate with high noise and thus benefit more from the approximate projection. We plot example rollouts for the OOD case in Fig. 4 (right); here, the test data is OOD in all state dimensions. Compared to the baseline GP (green), our method's predictions are more accurate. Overall, these results suggest that both sparsification and projection onto the learned constraint improve prediction accuracy, especially in OOD scenarios, and that the learned constraint generalizes better here than the learned dynamics.

VI. DISCUSSION AND CONCLUSION

We present a method for improving the OOD accuracy of dynamics models from data. We do this by learning a pseudometric to uncover the sparsity in the data and by approximating the normal space with a GP to estimate the constraint manifold that the system must evolve on.

While we our method outperforms baselines in our experiments, there are still limitations and future directions to our work. We wish to scale our approach to higher-dimensional systems, e.g., deformable objects. For such complex systems, it may be easier to find state/control-dependent pseudometrics, rather than the pseudometrics with constant P_θ that we consider here. We also wish to explore identification of sparsity in the output space, which can simplify constraint learning (as $\Gamma(x)$ is often quite sparse), and to use our method in MPC.

REFERENCES

[1] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *ICRA*. IEEE, 2018, pp. 7559–7566.

[2] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *ICML*, 2011, pp. 465–472.

[3] Z. Shen, J. Liu, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui, "Towards out-of-distribution generalization: A survey," *CoRR*, vol. abs/2108.13624, 2021.

[4] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *NeurIPS*, 2018, pp. 4759–4770.

[5] P. Mitrano, D. McConachie, and D. Berenson, "Learning where to trust unreliable models in an unstructured world for deformable object manipulation," *Sci. Robotics*, vol. 6, no. 54, p. 8170, 2021.

[6] J. Guzzi, R. O. Chavez-Garcia, M. Nava, L. M. Gambardella, and A. Giusti, "Path planning with local motion estimations," *IEEE Robotics Autom. Lett.*, vol. 5, no. 2, pp. 2586–2593, 2020.

[7] C. Knuth, G. Chou, J. Reese, and J. Moore, "Statistical safety and robustness guarantees for feedback motion planning of unknown underactuated stochastic systems," in *ICRA*, 2023.

[8] C. Knuth, G. Chou, N. Ozay, and D. Berenson, "Planning with learned dynamics: Probabilistic guarantees on safety and reachability via lipschitz constants," *IEEE Robotics Autom. Lett.*, vol. 6, no. 3, pp. 5129–5136, 2021.

[9] G. Chou, N. Ozay, and D. Berenson, "Model error propagation via learned contraction metrics for safe feedback motion planning of unknown systems," in *CDC*. IEEE, 2021, pp. 3576–3583.

[10] M. Laskin, A. Srinivas, and P. Abbeel, "CURL: contrastive unsupervised representations for reinforcement learning," in *ICML*, 2020.

[11] D. Wang, R. Walters, X. Zhu, and R. P. Jr., "Equivariant Q learning in spatial action spaces," in *CoRL*, 2021.

[12] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *ICLR*, 2021.

[13] V. Pacelli and A. Majumdar, "Learning task-driven control policies via information bottlenecks," in *R:SS*, 2020.

[14] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," in *ICLR*, 2019.

[15] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," in *ICLR*, 2020.

[16] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *NeurIPS*, 2019.

[17] A. R. Geist and S. Trimpe, "Learning constrained dynamics with gauss' principle adhering gaussian processes," in *L4DC*, 2020.

[18] L. Rath, A. R. Geist, and S. Trimpe, "Using physics knowledge for learning rigid-body forward dynamics with gaussian process force priors," in *CoRL*, 2021.

[19] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu, "Neural networks with physics-informed architectures and constraints for dynamical systems modeling," in *L4DC*, 2022.

[20] 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.

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

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

[23] G. Chou, H. Wang, and D. Berenson, "Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations," *IEEE Robotics Autom. Lett.*, vol. 7, no. 2, 2022.

[24] M. Menner, P. Worsnop, and M. N. Zeilinger, "Constrained inverse optimal control with application to a human manipulation task," *IEEE TCST*, 2021.

[25] K. C. Stocking, D. L. McPherson, R. P. Matthew, and C. J. Tomlin, "Maximum likelihood constraint inference on continuous state spaces," in *ICRA*, 2022.

[26] S. M. LaValle, *Planning algorithms*. Cambridge Press, 2006.

[27] W. P. Klingenberg, "Riemannian geometry." De Gruyter, 1995.

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

[29] J. Kelley, *General Topology*. Springer New York, 1975.

[30] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, 2005.

[31] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[32] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.

[33] R. Bro, E. Acar, and T. Kolda, "Resolving the sign ambiguity in the singular value decomposition," *Journal of Chemometrics*, 2008.

[34] R. H. Byrd and R. B. Schnabel, "Continuity of the null space basis and constrained optimization," *Math. Program.*, vol. 35, no. 1, pp. 32–41, 1986. [Online]. Available: https://doi.org/10.1007/BF01589439