Learning Equality Constraints From Trajectory Demonstrations

Baiyu Peng, Albert Árboles, and Aude Billard, *Fellow, IEEE*

***Abstract*—Equality constraints are common in robotics, arising in applications such as pose-constrained manipulation, coordinated motion, and task-space planning. Classical constrained planners typically assume an analytic constraint function, whereas in practice only feasible demonstrations may be available. We study how to learn explicit equality constraints directly from such demonstrations. We propose Orthonormal Neural Constraint Learning (ONCL), a method that learns multiple scalar constraint functions jointly and represents the equality constraint by their common zero set. Rather than estimating local normal directions and generating large amounts of supervised off-constraint data as related works, ONCL regularizes the gradients of different constraint functions to be normalized and mutually orthogonal, inducing a structured local normal space during learning. This yields an explicit differentiable constraint model in the original state space for evaluation and downstream planning. We evaluate ONCL on ten synthetic and robotic datasets spanning dimensions from 2 to 12. ONCL significantly outperforms each of three baselines on six datasets, while remaining competitive on the other datasets. We further validate ONCL on a physical KUKA iiwa 14 using kinesthetic demonstrations collected by a human operator. The learned pose constraint is subsequently used on the same platform for downstream trajectory planning under a modified task objective and an additional known obstacle-avoidance constraint.**

***Index Terms*—Learning from demonstration, constrained motion planning, transfer learning.**

I. Introduction

LEARNING from demonstration (LfD) is a standard paradigm for teaching robots from human examples, especially when the desired behavior is difficult to specify analytically [1], [2]. Beyond reproducing motions, demonstrations can also reveal the geometric structure that governs them.

Different learning problems can be posed from the same demonstrations. Some approaches focus on imitation, while others, as in inverse reinforcement learning, seek to recover a reward or objective function. In this letter, we focus on a different learning object: task constraints, which specify what must remain feasible throughout execution. This is particularly important in constrained planning and control, where downstream methods often require an explicit constraint function that can be evaluated on candidate states during motion generation [3], [4]. Compared with a policy tied to a particular task instance, an explicit constraint model can be queried on arbitrary states and reused across different goals, optimizers, and planners.

Many existing works on learning constraints from demonstrations focus on inequality constraints [5], [6], [7], [8], [9], [10], [11]. In contrast, many robotics tasks are governed by equality constraints: examples include pose-constrained manipulation, coordinated multi-arm motions, and other task-space relations where feasible states lie on a constraint manifold. Learning such constraints is fundamentally different from learning an inequality boundary, and the goal is to recover an explicit equality function whose zero set captures the underlying constraint satisfied by these demonstrations.

Earlier approaches to learning equality constraints from demonstrations often relied on predefined libraries of geometric relations, which limit flexibility when the true constraint is not contained in the library [12], [13]. A more recent method, most notably ECoMaNN [14], moves toward learning a continuous equality constraint function from data. However, it typically relies on first estimating local geometric directions and then generating many off-constraint samples for supervision. This strategy is natural for uniformly scattered feasible points, but is less suitable for trajectory demonstrations, where local neighborhoods are shaped by temporal continuity and explicit off-constraint augmentation becomes increasingly expensive in higher-dimensional spaces.

Our work starts from a different motivation. Rather than treating normal estimation and off-constraint sample generation as prerequisites, we ask whether the desired geometric structure can be induced directly within the constraint model itself. This leads to a simple alternative: learn multiple scalar constraint functions jointly, define the equality constraint through their common zero set, and regularize their gradients to be normalized and mutually orthogonal. In this way, the network is encouraged to recover a coherent local normal structure implicitly during training, without explicit normal labels or off-constraint supervision.

Based on this idea, we propose Orthonormal Neural Constraint Learning (ONCL), a method for learning explicit equality constraints directly from feasible trajectory demonstrations. The learned representation is differentiable, can be evaluated on arbitrary candidate states, and can be integrated into downstream optimization-based planning. We validate ONCL on ten

Received 29 April 2026; accepted 18 August 2026. Date of publication 14 September 2026; date of current version 30 September 2026. This article was recommended for publication by Associate Editor Y. Huang and Editor J. Kober upon evaluation of the reviewers' comments. This work was supported by euROBIN Grant. *(Corresponding author: Baiyu Peng.)*

Baiyu Peng and Aude Billard are with the LASA, School of Engineering, EPFL (Swiss Federal Institute of Technology in Lausanne), 1015 Lausanne, Switzerland (e-mail: baiyu.peng@epfl.ch; aude.billard@epfl.ch).

Albert Árboles is with the School of Engineering, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain (e-mail: albert.arbolesi@autonoma.cat).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2026.3733514, provided by the authors.

Digital Object Identifier 10.1109/LRA.2026.3733514

2377-3766 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.

synthetic and robotic datasets, together with additional studies on trajectory-structured data, training efficiency, data scaling, robustness to imperfect demonstrations, and real-robot learning and planning.

II. Related Work

A. Learning Constraints From Demonstrations

Learning constraints from demonstrations has attracted growing attention across robotics, reinforcement learning, and imitation learning. Most of this literature focuses on *inequality* constraints rather than equality constraints. Representative examples exploit KKT conditions together with a known dynamics model to recover constraints from locally optimal demonstrations, including approaches for learning general locally optimal constraints [5], parametric constraints in high-dimensional spaces [6], and Gaussian-process-based constraints [7]. Another line, often framed as inverse constrained reinforcement learning, combines a known reward function with policy optimization to separate demonstrations from constraint-violating rollouts [8], [9], [10]. More recently, positive-unlabeled formulations have been proposed to learn inequality constraints as binary feasibility classifiers [11].

While highly relevant, these methods are not directly applicable to learning general equality constraints. Conceptually, most are designed to recover a feasible region, whereas an equality constraint defines a thin surface that must be satisfied exactly.

Learning equality constraints from demonstrations has been explored less. Earlier approaches often assumed predefined geometric relations or task-specific constraint classes, such as move-in-line or fixed-point constraints [12], [13]. These methods are effective only when the true constraint belongs to the prescribed family, and cannot recover constraints outside it. The closest prior work to ours is ECoMaNN [14], which learns an explicit equality constraint function for motion planning. Its training pipeline combines local geometric estimation, off-manifold data augmentation, and several geometric consistency losses. This design is natural for dense point-cloud-like feasible samples, where local neighborhoods provide surface-like coverage. For trajectory demonstrations, however, local neighborhoods are dominated by temporal continuity and often resemble short curve segments rather than local surface patches, making normal estimation less reliable. Moreover, ECoMaNN relies on explicit off-manifold supervision: it generates perturbed samples around demonstrations and assigns them geometric targets derived from local estimation. Since the number of such augmented samples grows exponentially with the constraint dimension m [14], this strategy becomes increasingly expensive in higher-dimensional settings.

By contrast, our method uses only a small number of nearby perturbations[1] as unlabeled probe points for regularization, without assigning explicit target distances, signs, or normals.

[1]In our experiments, the number of perturbed points is set to be equal to the number of demonstrated states and does not grow with the constraint dimension.

B. Surface Modeling and Distance Functions

Our work is also related to neural implicit surface modeling and distance-function learning, since surface learning can be viewed as a special case of equality-constraint learning. Methods such as Occupancy Networks and DeepSDF learn scalar implicit functions whose level sets represent 3D surfaces, typically from dense point clouds or image-based supervision [15], [16]. Other approaches introduce geometric regularization so that the learned scalar field behaves like a distance function, for example through Eikonal-style constraints [17], [18]. These methods are related to ours in that they also regularize gradients, but they use a scalar function and do not impose inter-output orthogonality. Some additionally assume oriented points or normal supervision [19]. Similar ideas have also been adopted in robotics, for example in iSDF and configuration-space distance fields [20], [21].

Despite this similarity, most such methods target 3D surface reconstruction rather than general equality constraints in state spaces of arbitrary dimension. As our experiments show, methods effective for surface reconstruction [17], [18] do not necessarily transfer well to general equality-constraint learning. Moreover, they typically assume dense geometry-oriented samples and may rely on normals, signed-distance targets, or image-derived observations, whereas we consider trajectory demonstrations without such geometric supervision.

C. Manifold Learning

Our work is also related to manifold learning, which seeks lower-dimensional structure in high-dimensional data. Classical methods such as Isomap, Locally Linear Embedding, and Laplacian Eigenmaps recover latent embeddings or coordinates of the data manifold [22], [23], [24]. More recent neural approaches extend this perspective using autoencoders, implicit models, or latent-space planning [25], [26].

The key difference lies in the learned representation. Manifold-learning methods typically recover latent coordinates, local charts, or on-manifold samplers, but generally do not provide an explicit equality function in the original space. In contrast, ONCL learns a differentiable constraint function whose zero set represents the feasible manifold and whose nonzero values indicate constraint violation, making it directly applicable to evaluation and optimization-based constrained motion planning.

III. Preliminary and Problem Formulation

We consider equality-constraint learning from feasible trajectory demonstrations. Let $\mathcal{D}_\tau = \{\tau_k\}_{k=1}^{K}$ denote a dataset of K demonstrated trajectories, where the k-th trajectory is written as $\tau_k = \{x_{k,t}\}_{t=1}^{T_k}$. Here, T_k is the length of trajectory τ_k, $x_{k,t} \in \mathbb{R}^n$ denotes the t-th state in this trajectory, e.g., a robot configuration, a task-space state, or a feature state derived from state-action variables. We further denote by $\mathcal{D} = \{x_{k,t} \mid k = 1, \ldots, K,\ t = 1, \ldots, T_k\}$ the set of all demonstrated states.

Many robotic tasks are governed by several equality relations that must hold simultaneously, for instance, when both position and orientation are constrained, or when multiple end-effectors

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.

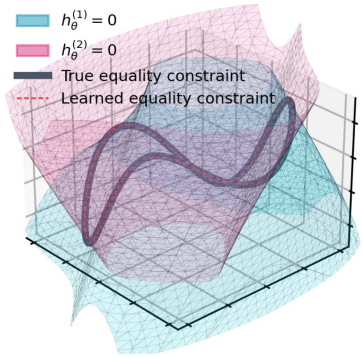


Fig. 1. Illustrative example of equality-constraint learning with ONCL on the `3DTwistEllipse` dataset, where the ground-truth constraint manifold is shown as a thick gray curve. ONCL approximates the constraint by learning two scalar functions $h_\theta^{(1)}$ and $h_\theta^{(2)}$ whose gradients are encouraged to have unit norm and to be mutually orthogonal everywhere. Their zero-level sets are shown as translucent surfaces. Their intersection (red dashed curve) defines the learned constraint and closely matches the ground-truth constraint.

are coupled through a shared object. We therefore represent the true equality constraint as a single vector-valued function

$$h_\star : \mathbb{R}^n \to \mathbb{R}^m, \qquad h_\star(x) = \left[h_\star^{(1)}(x), \ldots, h_\star^{(m)}(x)\right]^\top. \quad (1)$$

where the m components are m independent scalar equality constraints. Throughout this letter we assume $m \leq n$. Here, m should be understood as the constraint dimension of feasible set, not necessarily the number of semantic task constraints. For example, in the `3DTwistEllipse` dataset studied in the experimental section, the motion in $\mathbb{R}^3$ is restricted to a 1D curve. Then the true constraint $h(x)$ has $m = 2$ components instead of 1. To see this, Fig. 1 shows the learned constraints $h^{(1)} = 0$ and $h^{(2)} = 0$, where their intersection represents the feasible region, i.e., that 1D curve.

The true feasible set is therefore $\mathcal{C}^\star = \{x \in \mathbb{R}^n \mid h_\star(x) = 0\}$, that is, the set of states that simultaneously satisfy all m scalar equality constraints.

Our objective is to learn a differentiable function $h_\theta : \mathbb{R}^n \to \mathbb{R}^m$ parameterized with θ from demonstrations to approximate the unknown vector-valued equality constraint $h_\star$. The learned feasible set is therefore defined as $\mathcal{C}_\theta = \{x \in \mathbb{R}^n \mid h_\theta(x) = 0\}$.

Accordingly, the learned constraint is written as

$$h_\theta(x) = \left[h_\theta^{(1)}(x), \ldots, h_\theta^{(m)}(x)\right]^\top. \quad (2)$$

In practice, we implement h_θ as a single neural network with shared hidden layers but m different output units, where each output corresponds to one scalar constraint function $h_\theta^{(i)}(x)$. This shared parameterization is adopted for efficiency; in principle, one could also use m separate scalar-output networks.

IV. Method

A. Motivation

Our motivation comes from a limitation of existing equality-constraint learning pipelines. Methods such as ECoMaNN [14] typically proceed in two stages: they first estimate local geometric directions, such as tangent or normal directions, from neighborhoods of the training data, and then generate many off-constraint samples by perturbing the data along these directions and their combinations. The network is subsequently trained using these estimated directions and augmented samples as supervision. This strategy is natural when the training data are provided as uniformly scattered feasible points. However, it is less suitable for trajectory demonstrations. Along a trajectory, nearby samples mainly reflect temporal continuity and often do not reveal the full local geometry of the underlying equality constraints, making normal estimation less reliable. Moreover, off-constraint data augmentation becomes increasingly expensive in higher-dimensional spaces.

Our goal is to avoid this pipeline altogether. Instead of explicitly estimating local normals and explicitly supervising off-constraint samples, we ask whether the constraint network itself can organize a meaningful local normal space during learning.

Since all demonstrations are feasible, the most basic requirement is that every demonstrated state should satisfy the learned equality constraint. This gives the data-fitting term

$$\mathcal{L}_{\text{data}} = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \|h_\theta(x)\|_2^2 . \quad (3)$$

However, fitting feasible demonstrations alone is not sufficient for learning a useful equality-constraint model, since many different functions can fit the same data equally well, including trivial solutions such as the constant all-zero function. Moreover, different outputs may collapse to redundant constraints, and their gradients may differ only by arbitrary rescaling. To address this issue, ONCL introduces additional structure directly on the gradients of the learned scalar constraint functions.

B. Orthonormal Neural Constraint Learning

The core idea of ONCL is to regularize the gradients of different scalar constraint functions so that they form a structured local normal frame. To this end, we evaluate the regularization terms on a set $\mathcal{O} = \mathcal{D} \cup \widetilde{\mathcal{D}}$, where $\widetilde{\mathcal{D}}$ denotes random samples generated around demonstrated states (discussed later).

The first regularization term encourages each constraint gradient to have unit norm:

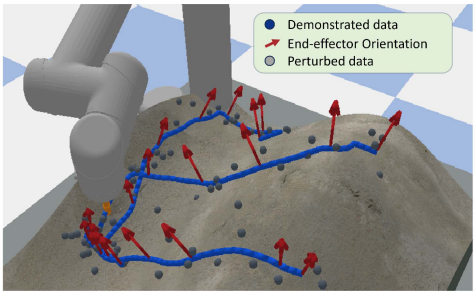
$$\mathcal{L}_{\text{norm}} = \frac{1}{|\mathcal{O}|} \sum_{x \in \mathcal{O}} \sum_{i=1}^{m} \left(\|\nabla h_\theta^{(i)}(x)\|_2^2 - 1\right)^2 . \quad (4)$$

This prevents arbitrary scaling of different outputs and encourages each scalar constraint to behave locally like a distance function. The second regularization term encourages gradients of different outputs to be mutually orthogonal:

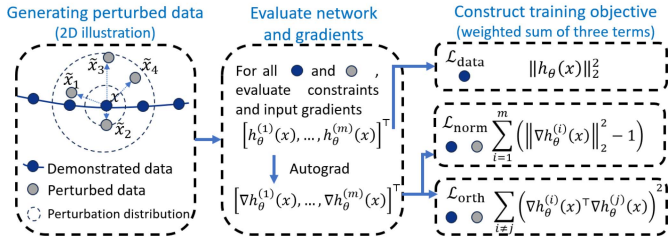
$$\mathcal{L}_{\text{orth}} = \frac{1}{|\mathcal{O}|} \sum_{x \in \mathcal{O}} \sum_{i \neq j} \left(\nabla h_\theta^{(i)}(x)^\top \nabla h_\theta^{(j)}(x)\right)^2 . \quad (5)$$

This discourages different outputs from collapsing to redundant constraints and encourages them to span distinct local normal directions. This orthonormal structure is a modeling choice rather than an assumption on the ground-truth constraint: it gives a simpler objective and a better-conditioned local representation. We discuss this in Section IV-C.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.



(a) `6DSurfNormal` dataset.



(b) ONCL training objectives.

Fig. 2. Illustration of the ONCL learning objective. (a) In `6DSurfNormal`, the demonstrated end-effector states (blue) lie on a curved surface with surface-normal orientations (red), while nearby perturbed data are shown in gray. (b) Perturbed data are sampled around the demonstrations. The shared network evaluates m scalar constraint functions and their gradients with respect to the input. The data-fitting loss $\mathcal{L}_{\text{data}}$ uses only the demonstrations, whereas the normalization loss $\mathcal{L}_{\text{norm}}$ and orthogonality loss $\mathcal{L}_{\text{orth}}$ use both demonstrations and perturbed data.

Together, (4) and (5) encourage the rows of the Jacobian $J_\theta(x)$ to be normalized and mutually orthogonal. In this way, the network is encouraged to discover the constraint structure implicitly, without explicit normal supervision.

The complete training objective is therefore

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_{\text{norm}}\mathcal{L}_{\text{norm}} + \lambda_{\text{orth}}\mathcal{L}_{\text{orth}}, \tag{6}$$

where $\lambda_{\text{norm}} > 0$ and $\lambda_{\text{orth}} > 0$ are weighting coefficients.

The set $\mathcal{O}$ contains not only demonstrated states but also nearby perturbed states. Concretely, we generate nearby probe points by adding Gaussian perturbations,

$$\tilde{x} = x + \xi, \qquad \xi \sim \mathcal{N}(0, \Sigma_{\text{near}}), \tag{7}$$

where $x \in \mathcal{D}$ is a demonstrated state. In our implementation, the standard deviation in each dimension is set to 0.05 times the corresponding span of the training data in that dimension, so that most probe points remain close to the demonstrated equality constraint. In each epoch, we generate the same number of perturbed points as demonstrated states, and use the resulting set as $\widetilde{\mathcal{D}}$. These perturbed states are used only to probe local gradient geometry. Unlike prior pipelines based on explicit off-constraint supervision, we do not assign target distances, target signs, or target normals to them.

Fig. 2 summarizes the ONCL training workflow and the data used by each objective term.

C. Why Orthonormality is a Modeling Choice

Orthonormality is a representation choice rather than an assumption on the ground-truth constraint. For any invertible $A \in \mathbb{R}^{m\times m}$, h and $\tilde{h} = Ah$ define the same zero set. ONCL therefore favors, among equivalent representations, one whose gradients form an approximately orthonormal local normal frame.

Proposition 1: Let $\mathcal{C} = \{x \mid h(x) = 0\}$, and suppose that $J_h(x)$ has full row rank at $x \in \mathcal{C}$. Then there exists a locally equivalent representation $\tilde{h}$ satisfying

$$J_{\tilde{h}}(x)J_{\tilde{h}}(x)^\top = I_m. \tag{8}$$

Proof: Define

$$B(x) = \left(J_h(x)J_h(x)^\top\right)^{-1/2}, \qquad \tilde{h}(x) = B(x)h(x). \tag{9}$$

Since $B(x)$ is invertible, h and $\tilde{h}$ have the same local zero set. By the product rule,

$$J_{\tilde{h}}(x) = B(x)J_h(x) + \frac{\partial B(x)}{\partial x}h(x). \tag{10}$$

At $x \in \mathcal{C}$, $h(x) = 0$, and hence

$$J_{\tilde{h}}(x)J_{\tilde{h}}(x)^\top = B(x)J_h(x)J_h(x)^\top B(x)^\top = I_m. \tag{11}$$

D. Determining the Number of Constraint Functions

We assume that the constraint dimension m is known. This is a practical necessity because constraint learning from demonstrations is inherently ill-posed [10]: different output dimensions and constraint functions may explain the same demonstrations equally well. When available, prior task knowledge therefore provides the most reliable choice of m.

Two heuristic alternatives are model selection over candidate output dimensions and local dimension estimation, such as local PCA with eigengap analysis as in ECoMaNN [14]. However, both can be unreliable for trajectory demonstrations, whose local neighborhoods are strongly influenced by temporal continuity.

Accordingly, all methods in our experiments are given the true constraint dimension for fair comparison. Jointly estimating m and the constraint function remains future work.

V. Experiments

This section first evaluates ONCL on ten synthetic datasets against three baselines, and then presents a physical-robot experiment in which the constraint is learned from real-robot demonstrations and validated on the robot. All code and datasets are available at https://github.com/epfl-lasa/Learn_Equality_Constraints_from_Demonstrations and the experiment video is online at https://www.youtube.com/watch?v=S43TTF1arno.

A. Experimental Setup

1) Datasets:

The datasets are introduced and summarized in Table I. All 3D and 4D trajectory datasets are generated by first constructing a dense feasible sample set on the ground-truth constraint and then performing a local neighbor walk on this set to obtain smooth demonstrations.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.

TABLE I
DATASETS USED IN THE BENCHMARK. n DENOTES THE DATA DIMENSION AND m THE TRUE NUMBER OF EQUALITY CONSTRAINT FUNCTIONS. N_{TRAJ} IS THE NUMBER OF DEMONSTRATION TRAJECTORIES AND N_{TRAIN} IS THE NUMBER OF TOTAL TRAINING SAMPLES IN ALL DEMONSTRATIONS.

Dataset	n	m	N_{train}	N_{traj}	Description
2DPlanarArmLine	2	1	128	-	A 2-DoF planar arm whose end-effector is constrained to move along a horizontal line in task space. The constraint is learned in joint space.
3DTwistEllipse	3	2	128	15	A 1D ellipse curve embedded in 3D space by a twisting transformation.
3DPlanarArmLine	3	2	2048	30	A 3-DoF planar arm whose end-effector is constrained to move along a horizontal line in task space. The constraint is learned in joint space.
3DArmEllipse	3	2	512	20	A 3-DoF spatial arm whose end-effector follows an elliptical curve in a plane. The constraint is learned in joint space.
3DTwoSphere	3	1	512	15	Two partially overlapping spherical surfaces, where demonstrations move along the surfaces but do not enter their interior.
3DLinkedCurves	3	2	512	16	See Fig. 4a.
4DMultiHandle	4	2	2752	64	See Fig. 4b.
6DSurfNormal	6	3	3000	20	A 6-DoF UR5 end-effector pose space, where position lies on a curved surface and orientation remains perpendicular to it (surface visualized in Fig. 2a)
6DArmUp	6	3	15000	500	A 6-DoF UR5 joint space with an upward-facing end-effector orientation constraint and unconstrained position.
12DDualArm	12	10	1500	15	See Fig. 3.

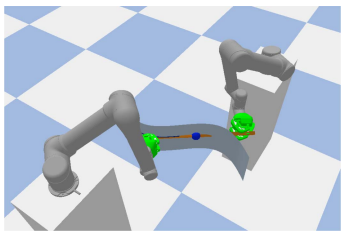
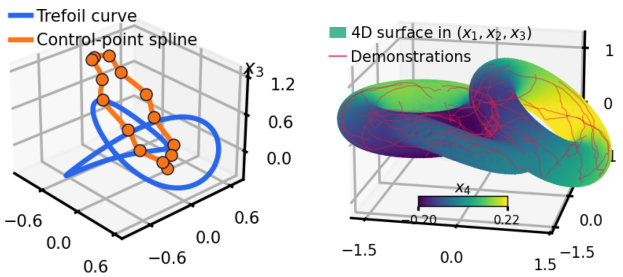


Fig. 3. Illustration of the `12DDualArm` dataset. A 12D dual-arm pose constraint in which two end-effectors rigidly grasp the same object with fixed span, while the object center follows a sinusoidal vertical ribbon and its principal axis aligns with the local guide tangent.



(a) `3DLinkedCurves` (b) `4DMultiHandle`

Fig. 4. Two complex-topology datasets. `3DLinkedCurves` is the union of a trefoil curve and a control-point spline curve in $\mathbb{R}^3$. `4DMultiHandle` is constructed from two intersecting tori with an additional connecting tube; the plot shows its first three coordinates, with color indicating the fourth coordinate.

For `6DSurfNormal`, trajectories are first generated on the curved surface and then augmented with orientations derived from the local surface frame and a smoothly varying free spin. For `6DArmUp`, trajectories are generated directly in UR5 joint space by perturbing the first four joints and solving the wrist joints to maintain the upward-facing constraint under step-size limits. For `12DDualArm`, trajectories are generated analytically from the guide-curve task parameterization and the dual-arm object geometry.

2) Baselines: We compare ONCL with three baselines.

DataAug is an off-constraint data augmentation baseline inspired by prior work on learning implicit surfaces and constraint manifolds [14], [19]. For each on-manifold training sample x, it estimates a local normal subspace using KNN-PCA and generates an off-manifold sample $x_{\text{off}} = x + \delta$, where δ is sampled in the estimated normal subspace. We train a scalar implicit function h_θ to output zero on the original samples and to match the perturbation magnitude on the augmented samples:

$$\mathcal{L}_{\text{DataAug}} = \mathbb{E}_x \left[|h_\theta(x)|^2 \right] + \lambda_{\text{off}} \mathbb{E}_{x_{\text{off}}} \left[\left(|h_\theta(x_{\text{off}})| - \|\delta\| \right)^2 \right].$$

VAE [27] is a manifold-learning baseline that learns a latent generative model of feasible data. Although it does not produce an explicit equality constraint function, we include it as a relevant latent-variable alternative.

ECoMaNN [14] is the closest prior equality-constraint learning method. It combines a learned constraint function with local tangent/normal alignment, off-constraint augmentation, and contrastive losses. We use the authors' implementation for fair comparison.

3) Evaluation Metrics: We use a distance-based metric to evaluate the geometric accuracy of the learned constraint. Since the learned feasible region is represented as $\mathcal{C}_\theta = \{x \in \mathbb{R}^n \mid h_\theta(x) = 0\}$, we first obtain samples in $\mathcal{C}_\theta$ by projecting a set of evaluation query points Q onto it and then compute their distance to the true constraint region.

For each dataset, the evaluation query set contains two types of query points: 75% near-manifold queries and 25% broader-range queries. The former are generated by adding Gaussian perturbations[2] to demonstrated states, and are used to evaluate local geometric accuracy around the feasible region. The latter are sampled uniformly from the whole state space, and are used to reveal possible spurious zero-level branches away from the true manifold.

[2]For each coordinate d, we first compute the range of the demonstrated states as $s_d = \max_{x \in \mathcal{D}} x_d - \min_{x \in \mathcal{D}} x_d$. The Gaussian noise is added independently to each coordinate, with standard deviation $0.16 s_d$.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.

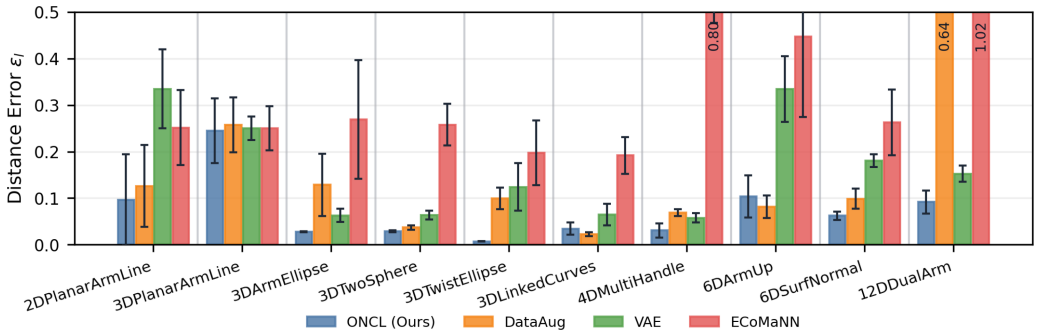


Fig. 5. Main benchmark on ten datasets using distance error ε_l (lower is better). Error bars indicate standard deviation across 15 runs.

Let Q denote the resulting evaluation query set. For ONCL, DataAug, and ECoMaNN, each query point $q \in Q$ is projected onto the learned feasible set by Newton-style iteration on the learned constraint function until convergence. For VAE, projection is approximated by encoder-decoder reconstruction. Let Q_p denote the resulting projected samples on the learned feasible set. The distance error is then defined as

$$\varepsilon_l = \frac{1}{|Q_p|} \sum_{x \in Q_p} \min_{y \in G} \|x - y\|_2, \tag{12}$$

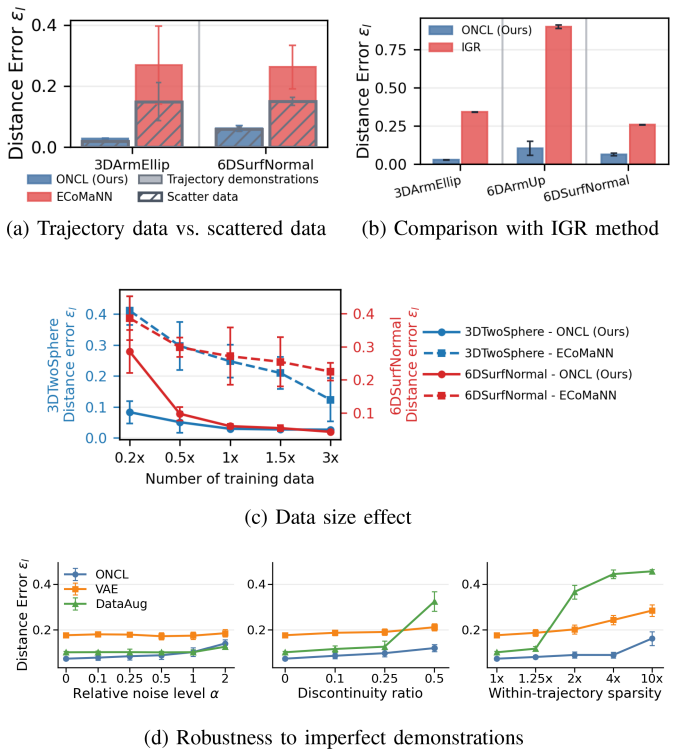
where G is a dense reference set sampled from the ground-truth feasible set. This metric measures the geometric distance from the learned feasible set to the true feasible set. Lower ε_l indicates better geometric accuracy.

4) Implementation Protocol: To ensure a fair comparison, all methods use networks with comparable capacity. ONCL, ECoMaNN, and DataAug use fully connected networks with three hidden layers of 128 units and SiLU activations [28]. For VAE, both encoder and decoder use the same architecture, and the latent dimension is set to the intrinsic manifold dimension $n - m$. Hyperparameters for all methods are tuned by grid search.[3] All methods are trained on the same demonstration set for each dataset, and all experiments are repeated with 15 random seeds.

B. Main Benchmark Across All Datasets

Fig. 5 summarizes the benchmark results of distance error ε_l. Since the main comparison concerns ONCL against the baselines, we further performed one-sided paired t-tests between ONCL and each baseline on each dataset using the 15 independent runs, with the alternative hypothesis that ONCL has a lower distance error. ONCL is significantly better than ECoMaNN and VAE on 9 out of 10 datasets, with `3DPlanarArmLine` being the only exception in both comparisons. Compared with DataAug, ONCL is significantly better on 6 datasets (`3DArmEllipse`, `3DTwoSphere`, `3DTwistEllipse`, `4DMultiHandle`, `6DSurfNormal` and `12DDualArm`). Overall, these results indicate that ONCL provides a clear advantage over ECoMaNN and VAE, and remains favorable compared with DataAug.

[3]Due to space limitations, detailed hyperparameter settings are omitted and only provided in the code repository.



(a) Trajectory data vs. scattered data (b) Comparison with IGR method

(c) Data size effect

(d) Robustness to imperfect demonstrations

Fig. 6. Additional analyses using distance error ε_l (lower is better). (a) ONCL maintains similar performance when trained with trajectory demonstrations and uniformly scattered feasible samples, whereas ECoMaNN degrades substantially under trajectory data. (b) ONCL consistently outperforms the surface-modeling baseline IGR. (c) Distance error decreases as the number of training samples increases. (d) Robustness on `12DDualArm` under Gaussian perturbations, discontinuous trajectory observations, and within-trajectory point sparsification, shown from left to right.

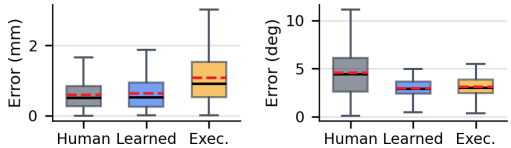
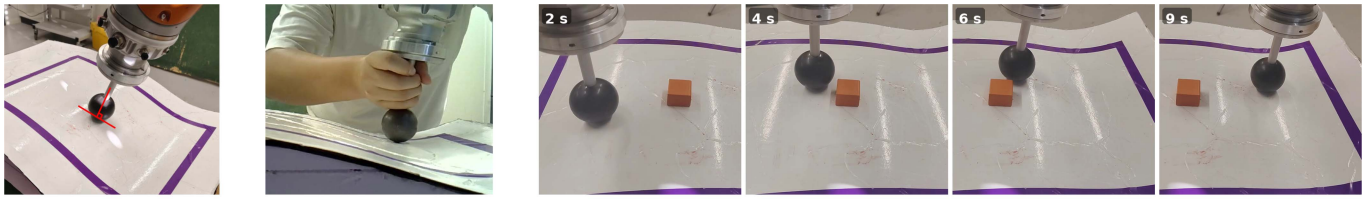


Fig. 7. Real-robot `SurfNormal` position (left) and orientation (right) errors relative to the true constraint. *Human* is the kinesthetic demonstration. *Learned* is the learned constraint error, evaluated at execution-path states by first projecting each state to the learned constraint manifold and comparing the projection with the true constraint. *Exec.* is the measured robot execution error relative to the true constraint. Black lines indicate medians and red dashed lines indicate means.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.



(a) (b) (c)

Fig. 8. Real-robot constraint learning task using a KUKA iiwa 14 robot. The end-effector pose is constrained such that its position lies on an unknown curved surface and its orientation remains normal to the surface. (a) Experimental setup. (b) Kinesthetic demonstrations collected by a human operator. (c) Execution snapshots showing the robot moving along the learned constraint surface while avoiding a known obstacle.

TABLE II
TRAINING TIME IN SECONDS (MEAN ± STANDARD DEVIATION)

Method	3DArmEllipse	6DSurfNormal	6DArmUp
ONCL (Ours)	6.8 ± 0.3	85.4 ± 1.8	123.0 ± 2.4
DataAug	7.2 ± 0.2	92.5 ± 0.6	479.9 ± 3.4
VAE	2.1 ± 0.1	24.6 ± 0.6	119.1 ± 0.6
ECoMaNN	47.5 ± 0.3	903.7 ± 29.7	2118.2 ± 111.9

Other methods still achieve the lowest mean error on some datasets. In particular, despite its simple formulation, DataAug obtains the best mean performance on `3DLinkedCurves` and `6DArmUp`. Interestingly, DataAug performs worse than ONCL on `3DTwistEllipse`, a dataset with a seemingly simpler structure than `3DLinkedCurves`. This difference may be explained by the substantially smaller number of training samples available in `3DTwistEllipse`, which makes the local normal estimation used by DataAug less reliable. ECoMaNN, in contrast, performs poorly on most high-dimensional datasets, likely because its normal-alignment objective depends on local normal estimates whose accuracy deteriorates as the ambient dimension increases.

C. Additional Analysis

1) Trajectory Demonstrations Vs. Uniform Scatter Samples: Trajectory demonstrations considered in this letter are structurally different from uniformly scattered samples used in prior works [14], and are more challenging to learn from. We test ONCL and ECoMaNN on additional datasets where the same number of feasible points are uniformly sampled from the true constraints. As shown in Fig. 6(a), ECoMaNN exhibits a large degradation when trained with trajectory demonstrations rather than with uniformly scattered samples, while ONCL shows similar performance.

2) Training Time: Besides learning accuracy, training efficiency is also important in practice. Table II reports the training time in seconds for four methods on three representative datasets. Among the explicit constraint-learning methods, ONCL is consistently faster than both DataAug and ECoMaNN. This advantage becomes particularly pronounced on the larger dataset `6DArmUp`.

3) Effect of Training Data Size: Fig. 6(c) shows that the distance error on `3DTwoSphere` and `6DSurfNormal` generally decreases as the training-set size increases from $0.2\times$ to $3.0\times$ its default value.

4) Comparison With Surface Modeling Methods: We compare our method to a surface modeling method Implicit Geometry Regularization (IGR) [17], [18], which learns a 3D surface from point clouds with gradient regularization. We compare ONCL with IGR on three representative datasets. The results are shown in Fig. 6(b). ONCL consistently achieves significantly lower distance error. This suggests that surface modeling methods cannot be readily applied to learn higher-dimensional constraints and supports the need for a structured multi-output constraint formulation as in ONCL.

5) Robustness to Imperfect Demonstrations: We also evaluate methods on imperfect demonstration sets. Fig. 6(d) reports the distance error ε_l of ONCL, DataAug and VAE on `12DDualArm` under three types of training-data perturbations. First, we add isotropic Gaussian noise to the training samples. The relative noise level α controls the perturbation scale relative to the typical distance between nearby clean training samples. Second, we create discontinuous trajectories by removing three contiguous segments from each trajectory, where the discontinuity ratio denotes the fraction of points removed. Third, we sparsify points within each trajectory by retaining approximately one point every k original samples, where the within-trajectory point sparsity is reported as $k\times$. Overall, ONCL degrades smoothly as the perturbation level increases.

D. Real Robot Learning and Validation

We conduct a real-robot experiment based on `6DSurfNormal`. The constraint definition is similar but the surface is different. As shown in Fig. 8(a) and (b), the experiment uses a KUKA iiwa 14 robot equipped with a probe tool. A human demonstrator collected a two-minute demonstration directly on the real robot using kinesthetic teaching.

We then train an ONCL constraint model on this demonstration dataset. For downstream execution, the planner discretizes the motion into a sequence of 6D end-effector pose waypoints. It then optimizes all waypoints with a gradient-based trajectory optimization objective that encourages the path to stay on the learned constraint manifold, remain short, stay smooth, and maintain a valid tool-axis representation. The optimized pose waypoints are converted into joint-space waypoints using continuous inverse kinematics. The resulting time-parameterized joint trajectory is sent to the standard ROS joint position trajectory controller for tracking.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.

We evaluate the learned constraint on the real robot using 13 execution trajectories distinct from the demonstrations. Fig. 7 summarizes the position and orientation errors relative to the true constraint for the human demonstrations, the learned constraint, and the executed trajectories. The learned constraint has a position error comparable to that of the human demonstrations and a lower orientation error. Robot execution introduces a slight increase in position error, while largely preserving the orientation accuracy.

The learned constraint can also be reused in related settings. For example, Fig. 8(c) shows an obstacle-avoidance task in which a known obstacle is placed on the surface. We add the obstacle clearance term to the same waypoint optimization problem, so the planned trajectory simultaneously satisfies the learned `SurfNormal` constraint and avoids the obstacle during real-robot execution.

VI. Conclusion

This letter studied equality-constraint learning from feasible trajectory demonstrations. We proposed Orthonormal Neural Constraint Learning (ONCL), a method that learns multiple scalar constraint functions jointly and represents the equality constraint through their common zero set. By regularizing the gradients of different outputs to be normalized and mutually orthogonal, ONCL avoids the explicit normal estimation and explicit off-constraint supervision used in prior works, while still learning an explicit differentiable constraint function.

Experiments on ten synthetic and robotic datasets showed that ONCL achieves significantly lower distance error than other baselines on six datasets. Additional robustness studies demonstrated smooth performance degradation under noisy, discontinuous, and sparse demonstrations. A real-robot experiment on `6DSurfNormal` further showed that the learned constraint can be used for trajectory planning on a physical KUKA iiwa 14 under a modified task objective and an additional known obstacle constraint.

The current method still lacks formal guarantees on the convergence of projection toward the learned zero set and for geometric extrapolation beyond the demonstrated support. Future work will investigate these theoretical limitations, automatic determination of the constraint dimension, and automatic identification of the constrained state variables.

References

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

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbookof Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer, 2008, pp. 1371–1394.

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

[4] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, no. 1, pp. 159–185, 2018.

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

[6] G. Chou, N. Ozay, and D. Berenson, "Learning parametric constraints in high dimensions from demonstrations," in *Proc. Conf. Robot Learn.*, 2020, pp. 1211–1230.

[7] G. Chou, H. Wang, and D. Berenson, "Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3827–3834, Apr. 2022.

[8] D. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[9] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, "Inverse constrained reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 7390–7399.

[10] G. Liu, Y. Luo, A. Gaurav, K. Rezaee, and P. Poupart, "Benchmarking constraint inference in inverse reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2023.

[11] B. Peng and A. Billard, "Positive-unlabeled constraint learning for inferring nonlinear continuous constraints functions from expert demonstrations," *IEEE Robot. Automat. Lett.*, vol. 10, no. 2, pp. 1593–1600, Feb. 2025.

[12] C. Perez-D'Arpino and J. A. Shah, "C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4058–4065.

[13] G. Subramani, M. Zinn, and M. Gleicher, "Inferring geometric constraints in human demonstrations," in *Proc. Conf. Robot Learn.*, 2018, pp. 223–236.

[14] G. Sutanto, I. Rayas Fernández, P. Englert, R. K. Ramachandran, and G. Sukhatme, "Learning equality constraints for motion planning on manifolds," in *Proc. Conf. Robot Learn.*, vol. 155, 2021, pp. 2292–2305.

[15] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3 D reconstruction in function space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.

[16] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.

[17] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3789–3799.

[18] Y. Ben-Shabat, C. Hewa Koneputugodage, and S. Gould, "DiGS: Divergence guided shape implicit neural representation for unoriented point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19301–19310.

[19] M. Atzmon and Y. Lipman, "SALD: Sign agnostic learning with derivatives," in *Proc. Int. Conf. Learn. Representations*, 2021.

[20] J. Ortiz et al., "ISDF: Real-time neural signed distance fields for robot perception," in *Proc. Robot.: Sci. Syst.*, 2022.

[21] Y. Li, X. Chi, A. Razmjoo, and S. Calinon, "Configuration space distance fields for manipulation planning," in *Proc. Robot.: Sci. Syst.*, 2024.

[22] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[23] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[24] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[25] I. M. R. Fernández, G. Sutanto, P. Englert, R. K. Ramachandran, and G. S. Sukhatme, "Learning manifolds for sequential motion planning," 2020, *arXiv:2006.07746*.

[26] S. Park, S. Jeon, and J. Park, "A constrained motion planning method exploiting learned latent space for high-dimensional state and constraint spaces," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 4, pp. 3001–3009, Aug. 2024.

[27] I. Higgins et al., "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Int. Conf. Learn. Representations*, 2017.

[28] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, 2018.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:29 UTC from IEEE Xplore. Restrictions apply.