

Equality learning constraints from demonstrations

Albert Árboles

Supervisor: Baiyu Peng

Learning Algorithms and Systems Laboratory (LASA)

École Polytechnique Fédérale de Lausanne (EPFL)

Lausanne, Switzerland

Abstract—Learning equality constraints from demonstration data enables robots to understand and exploit the geometric structure underlying expert motion. In this work, we propose a framework for learning explicit surfaces from demonstrated trajectories and using them to generate robust, constraint-consistent robot motions. A neural explicit model is trained from demonstration data to capture the surface by minimizing a task-specific training loss, while Euclidean and Chamfer distance metrics are used exclusively to evaluate how accurately the learned surface fits the demonstrations. These same trajectories are then used to train a neural motion controller capable of predicting feasible incremental movements toward arbitrary goals. To guarantee strict adherence to the learned constraint, each predicted state is projected onto the surface through Newton-based corrections. Finally, an optimal control formulation is employed to optimize motion under the learned equality constraint, enabling robust and constraint-aware planning across multiple tasks. Experimental results show that the learned surfaces accurately capture the underlying geometry and that both the projected controller and the optimal control formulation achieve low surface-deviation errors while maintaining smooth and reliable trajectories. This demonstrates the feasibility of combining explicit constraint learning with optimal control for autonomous manipulation and motion planning.

Index Terms—Learning from Demonstration, Geometric Constraint Learning, Neural Explicit Representations, Model Predictive Control, Motion Planning

I. INTRODUCTION

Trajectory generation and motion planning are fundamental for autonomous robots, particularly in contact-rich tasks that require motion constrained to a surface. Without an explicit surface model, policies may drift from the workspace or violate contact constraints.

Traditional learning-from-demonstration approaches focus on directly imitating trajectories observed in expert demonstrations. However, such methods focus on reproducing demonstrated trajectories rather than explicitly identifying the geometric constraint, leading to poor generalization to new goals or configurations. Previous work in our laboratory successfully reproduced demonstrated trajectories but lacked the ability to infer the equality constraint—the surface—that generated them.

In this work, we address this limitation by learning an explicit representation of the constraint directly from demonstration data. Demonstrated trajectories are used to train a neural network that represents the constraint as the zero level set of a scalar function. The model is trained using a control-

oriented loss that enforces consistency between the learned constraint geometry and its gradients.

Subsequently, the learned constraint is exploited for constraint-aware planning and control. The differentiable surface representation enables local correction of constraint violations through projection, while trajectory generation can be formulated as an optimal control problem subject to the learned equality constraint. This formulation enforces adherence to the underlying constraint and allows the controller to generate feasible motions that generalize beyond the demonstrated trajectories.

Overall, the proposed approach learns an explicit surface representation from demonstrations and enables constraint-aware planning and control through projection and optimization.

II. RELATED WORKS

Learning Inequality Constraints from Demonstration

The majority of constraint-learning research has focused on *inequality constraints*, typically associated with safety, collision avoidance, or defining feasible workspaces. Early works utilized Inverse Reinforcement Learning (IRL) to recover cost functions that penalize restricted regions. Scobee and Sastry formulated this as a maximum likelihood problem to infer constraints from demonstrations [1]. This was further extended by Maximum Entropy Constraint Learning (MECL), which jointly trains a policy and a constraint network to identify infeasible states [2].

Alternative approaches follow the Inverse Optimal Control (IOC) framework, which relies on the assumption that demonstrated trajectories optimize a cost subject to constraints. Chou et al. leveraged Karush-Kuhn-Tucker (KKT) conditions to recover parametric constraints [3] and later extended this to non-parametric Gaussian Process representations [4]. While effective for defining “safe vs. unsafe” zones (inequalities), these methods require analytical model of the environment and are generally not designed to learn equality constraints.

Learning Equality Constraints from Demonstration

Learning *equality constraints*—manifolds where $h(x) = 0$ —is substantially less explored than standard trajectory imitation. In the context of Manifold Learning, early approaches applied dimensionality reduction techniques to capture the structure of data. Linear methods such as Principal Component

Analysis (PCA) are simple but often lack the complexity to represent intricate constraint manifolds [5].

Within the Learning from Demonstration (LfD) framework, constraints have also been inferred using Inverse Optimal Control (IOC). Approaches like *Inverse KKT* aim to recover a cost function and associated constraints that rationalize the demonstrated behavior [6]. However, these methods are typically tied to time-parameterized optimal control formulations. Alternatively, some works perform motion planning in learned latent spaces [7] or learn adaptive sampling distributions [8] to bias planning towards valid regions.

To address these limitations, recent deep learning approaches propose learning implicit representations of the constraint manifold. The Equality Constraint Manifold Neural Network (ECoMaNN) learns a level-set function directly from demonstrations by aligning the subspaces of the neural network with the tangent and normal spaces of the data. Unlike previous methods, ECoMaNN allows for a projection-based strategy that can be directly integrated into sampling-based motion planners to find valid configurations on the manifold [9].

Neural Manifold and Surface Modeling

In parallel, the computer vision and graphics communities have developed powerful Neural Explicit Representations for continuous surfaces. Neural implicit models, such as Signed Distance Functions (SDFs) [10], learn high-resolution geometry from point clouds. Recently, methods incorporating geometric inductive biases have emerged to improve surface consistency. For instance, "Neural Pull" trains signed distance functions by pulling space onto the surface using gradient constraints, resulting in high-fidelity representations even from sparse data [11].

Integrating these expressive neural surface models into robotic control loops offers a promising direction. By combining the representational power of neural networks with the rigor of projection-based control, we aim to bridge the gap between surface modeling and constraint-consistent trajectory execution.

III. METHODS

A. Problem Setup

We consider the problem of learning an explicit equality constraint from demonstration trajectories. Let $\mathcal{D} = \{\mathcal{T}_i\}_{i=1}^M$ denote a set of demonstration trajectories, where each trajectory $\mathcal{T}_i = \{\mathbf{x}_t\}_{t=1}^{T_i}$ consists of state observations evolving on an unknown constraint surface.

We assume that demonstrations are feasible with respect to an unknown true constraint $h^* : \mathbb{R}^N \rightarrow \mathbb{R}$, such that all demonstrated states satisfy

$$\forall \mathbf{x} \in \mathcal{T}_i \in \mathcal{D}, \quad h^*(\mathbf{x}) = 0.$$

The true constraint h^* is not available and must be inferred from data.

1) *Explicit Constraint Parameterization*: We represent the constraint explicitly as the zero level set of a differentiable scalar function $h_\theta : \mathbb{R}^N \rightarrow \mathbb{R}$, parameterized by θ :

$$\mathcal{S}_\theta = \{\mathbf{x} \in \mathbb{R}^N \mid h_\theta(\mathbf{x}) = 0\}.$$

This explicit representation provides direct access to the gradient $\nabla h_\theta(\mathbf{x})$, which is exploited for projection-based correction and constraint-aware control. The objective of learning is to estimate parameters θ such that $\mathcal{S}_\theta$ approximates the unknown surface defined by h^* using only on-surface demonstration data.

Two representation families are possible for our methods: (i) analytic function and (ii) neural networks. Analytic models provide stronger inductive biases but only suitable for the case where the true constraint parameterization is known, while neural networks are more flexible and more expressible for general application. This work focuses on neural representations capable of modeling arbitrary shapes.

a) *Illustrative Example*: For intuition, consider the case of a spherical surface embedded in $\mathbb{R}^3$. The corresponding constraint function can be written as

$$h(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_0\| - r,$$

where $\mathbf{x}_0 \in \mathbb{R}^3$ denotes the center of the sphere and $r > 0$ its radius, both of which are learnable parameters. In this case, the gradient

$$\nabla h(\mathbf{x}) = \frac{\mathbf{x} - \mathbf{x}_0}{\|\mathbf{x} - \mathbf{x}_0\|}$$

corresponds to the outward surface normal for all $\mathbf{x} \neq \mathbf{x}_0$. While our experiments focus on three-dimensional examples for visualization, the proposed formulation and learning approach are not limited to $\mathbb{R}^3$.

2) *Demonstration Trajectories*: Let $\mathcal{D} = \{\mathcal{T}_i\}_{i=1}^M$ denote a set of demonstration trajectories, where each trajectory $\mathcal{T}_i = \{\mathbf{x}_t\}_{t=1}^{T_i}$ consists of state observations in $\mathbb{R}^N$. The demonstrations are assumed to be feasible with respect to an unknown true equality constraint $h^* : \mathbb{R}^N \rightarrow \mathbb{R}$.

Accordingly, all demonstrated states are assumed to lie on the corresponding constraint surface,

$$\mathcal{S}^* = \{\mathbf{x} \in \mathbb{R}^N \mid h^*(\mathbf{x}) = 0\},$$

such that

$$\forall \mathbf{x} \in \mathcal{D}, \quad \mathbf{x} \in \mathcal{S}^*.$$

The demonstration data therefore provide samples from the feasible region of the constraint, without requiring explicit knowledge of the constraint function or its derivatives.

B. Constraint Learning

1) *Neural Constraint Representation*: The constraint function $h_\theta : \mathbb{R}^N \rightarrow \mathbb{R}$ is parameterized by a fully connected neural network with two hidden layers of 64 units each and ReLU activations. The network outputs a scalar value whose zero level set represents the learned surface.

2) *Training Data Construction*: Each training sample is labeled $(\mathbf{x}, y)$, where $y = 1$ denotes on-surface points and $y = 0$ denotes off-surface samples.

a) *Positive samples*: All points from demonstration trajectories are treated as positive samples and assumed to lie on the constraint surface.

b) *Negative samples*: Negative samples are generated by perturbing positive samples along the outward surface normal:

$$\mathbf{x}_{\text{neg}} = \mathbf{x}_{\text{pos}} + \varepsilon \mathbf{n}(\mathbf{x}_{\text{pos}}), \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\text{neg}}^2),$$

with $\sigma_{\text{neg}} = 0.15$. Samples satisfying $|h(\mathbf{x}_{\text{neg}})| \leq \tau$ ($\tau = 10^{-3}$) are discarded to ensure strictly off-surface negatives. A balanced 1:1 ratio of positive to negative samples is maintained.

As illustrated in Fig. 1, negative samples are generated by perturbing on-surface points along the outward normal, resulting in a balanced training set.

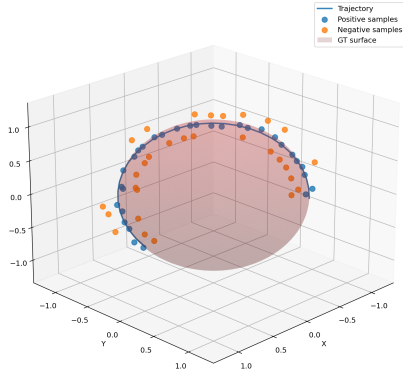


Fig. 1: Example demonstration trajectory and the resulting positive and negative samples generated for explicit surface learning.

c) *Noise injection*: To improve robustness and simulate sensor noise, Gaussian noise is added to positive samples:

$$\tilde{\mathbf{x}}_{\text{pos}} = \mathbf{x}_{\text{pos}} + \delta, \quad \delta \sim \mathcal{N}(0, 0.03^2 I).$$

All sampling procedures are seeded for reproducibility.

3) *Training Objectives (Loss Functions)*: To learn a zero-level set constraint function, this work introduces and compares three loss functions.

a) *Margin-based Loss*: This baseline enforces surface consistency for positives and a margin for negatives using Softplus:

$$\mathcal{L}_{\text{margin}} = \mathbb{E}_{\mathbf{x}_{\text{pos}}} [h_{\theta}(\mathbf{x}_{\text{pos}})] + \alpha \mathbb{E}_{\mathbf{x}_{\text{neg}}} [\text{softplus}(m - |h_{\theta}(\mathbf{x}_{\text{neg}})|)]. \quad (1)$$

where m is a fixed margin and α weights the negative term,

b) *k-NN Weighted Loss*: To adapt supervision to local geometry, the fixed margin is replaced by a distance-based bound derived from the k nearest positive neighbors:

$$\mathcal{L}_{\text{kNN}} = \alpha \mathbb{E}_{\mathbf{x}_{\text{neg}}} [\text{softplus}(d_k(\mathbf{x}_{\text{neg}}) - |h_{\theta}(\mathbf{x}_{\text{neg}})|)],$$

where $d_k(\cdot)$ denotes the mean Euclidean distance to the k nearest positive samples.

c) *Adaptive Neural Pull Loss*: The most effective strategy projects query points toward the zero-level set using the network gradient. An adaptive step size $\eta(\mathbf{x})$, inversely related to $\|\nabla h_{\theta}(\mathbf{x})\|$, improves stability:

$$\hat{\mathbf{x}} = \mathbf{x} - \eta(\mathbf{x}) \frac{h_{\theta}(\mathbf{x}) \nabla h_{\theta}(\mathbf{x})}{\|\nabla h_{\theta}(\mathbf{x})\|}.$$

The loss minimizes the distance between projected points and nearby demonstrations, together with a surface consistency term:

$$\mathcal{L}_{\text{pull}} = \mathbb{E}_{\mathbf{x}} \left[\frac{1}{k} \sum_{j \in \mathcal{N}_k(\hat{\mathbf{x}})} \|\hat{\mathbf{x}} - \mathbf{x}_{\text{pos},j}\| \right] + \mathbb{E}_{\mathbf{x}_{\text{pos}}} |h_{\theta}(\mathbf{x}_{\text{pos}})|.$$

4) *Regularization and Optimization*: We include two regularization terms to improve geometric fidelity and training stability.

a) *Eikonal Regularization*: To encourage SDF behavior, we penalize deviations from unit gradient norm:

$$\mathcal{R}_{\text{eik}} = \mathbb{E}_{\mathbf{x}} [(\|\nabla h_{\theta}(\mathbf{x})\| - 1)^2].$$

b) *L2 Regularization*: Standard weight decay is applied to the network parameters:

$$\mathcal{R}_{\text{L2}} = \sum_{p \in \theta} \|p\|^2.$$

In our experiments, we will perform an ablation comparing: (i) no regularization, (ii) L2 regularization, and (iii) Eikonal regularization, isolating their effects on gradient stability and geometric accuracy.

C. Motion Control with Learned Constraint

After learning the geometric constraint $h_{\theta}(\mathbf{x})$, we consider two alternative control strategies that exploit the learned surface in different settings: (i) a *Projected Imitation Controller*, and (ii) an *Optimal Controller*. These two approaches are independent and address complementary use cases.

1) *Projected Imitation Controller*: We consider a data-driven controller that combines imitation learning with explicit enforcement of the learned surface constraint. The controller is defined by a policy $\pi_{\phi} : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^N$, parameterized by a feedforward neural network and trained via behavioral cloning on demonstration data.

Given the current state $x_t \in \mathbb{R}^N$ and a goal g , the policy predicts a bounded incremental displacement

$$\Delta x_t = \pi_{\phi}([x_t, g]), \quad \hat{x}_{t+1} = x_t + \Delta x_t. \quad (2)$$

The network operates on the concatenated state-goal input and follows a shallow residual architecture with ReLU activations and layer normalization. Residual connections promote stable training and encourage identity mappings when appropriate. The output is bounded using a scaled hyperbolic tangent activation.

Training minimizes the mean squared error between predicted and demonstrated displacements, $\Delta x_t = x_{t+1} - x_t$,

rather than absolute states, which encourages locally consistent motion generation.

The projected imitation controller does not require access to an explicit system model or cost function, and is therefore suitable when only demonstration data are available.

2) Optimal Control with Learned Surface Constraint:

When a system model and task-specific cost function are available, the learned surface can be incorporated directly into an optimal control formulation. We consider a discrete-time system with first-order dynamics

$$x_{k+1} = x_k + \Delta t u_k, \quad (3)$$

and formulate trajectory generation as a finite-horizon optimal control problem subject to the learned equality constraint.

The surface is represented implicitly by the learned function $h_\theta(x)$, and enforced as

$$h_\theta(x_k) = 0, \quad \forall k = 1, \dots, N. \quad (4)$$

This constraint restricts the motion to the learned surface while allowing tangential movement along it.

The resulting problem minimizes a task-dependent cost, such as goal-reaching or reference tracking,

$$\min_{\{u_k\}} \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N) \quad \text{s.t.} \quad h_\theta(x_k) = 0, \quad (5)$$

where $\ell(\cdot)$ and $\ell_f(\cdot)$ denote stage and terminal costs, respectively.

The equality constraint is handled using standard techniques from constrained optimal control, such as sequential quadratic programming or projection-based methods. This controller does not rely on demonstration data and is particularly suitable for constraint transferring across different tasks, goals, or cost functions.

IV. EXPERIMENTS

This section evaluates the proposed method on learning three analytic constraint surfaces: a **sphere**, an **ellipsoid**, and a **cylinder**.

We first compare the learning accuracy of our approach under different loss functions and regularization schemes. Based on this comparison, we select the best-performing loss-regularization combination and evaluate a **projected imitation controller** synthesized from the trained network, in terms of its **true constraint satisfaction**, against a baseline imitation controller.

Finally, to demonstrate the **transferability** of the learned constraints, we design several **new scenarios** that share the same underlying constraints but differ in their task goals. In these scenarios, an **optimal controller** is deployed using the learned constraints, and its performance is quantitatively evaluated to assess generalization to previously unseen tasks.

A. Experimental Setup

We evaluate the proposed method on three ground-truth analytic constraint surfaces defined by explicit equations.

a) *Sphere*: The spherical surface is represented by the signed-distance function

$$h_{\text{sphere}}(x) = \|x - x_0\| - r = 0.$$

b) *Ellipsoid*: The ellipsoidal surface is defined by the implicit equation

$$h_{\text{ellipsoid}}(x) = \left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 + \left(\frac{x_3}{c}\right)^2 - 1 = 0.$$

c) *Cylinder*: The cylindrical surface is represented by the signed-distance formulation

$$h_{\text{cylinder}}(x) = \sqrt{x_1^2 + x_2^2} - R = 0,$$

which defines an open surface extending along the x_3 axis.

The neural network is trained with **Adam** (learning rate 10^{-3} , batch size 128) for 600 epochs, which suffices for validation loss convergence. Demonstration trajectories are split 80/20 at the trajectory level. Negative samples are generated along the outward normal of positive samples, and Gaussian noise is added to simulate sensor uncertainty.

B. Evaluation and Metrics

This section describes the evaluation protocol and metrics used to assess both the quality of the learned constraint surface and its suitability for downstream control tasks.

1) *Evaluation Protocol (Learned Surface)*: We evaluate the learned explicit surface $h_\theta : \mathbb{R}^3 \rightarrow \mathbb{R}$ using a two-stage protocol consisting of a trajectory-level data split and a final global geometric assessment based on uniform sampling.

As illustrated in Fig. 2, the protocol combines trajectory-level validation with a global test of geometric reconstruction quality.

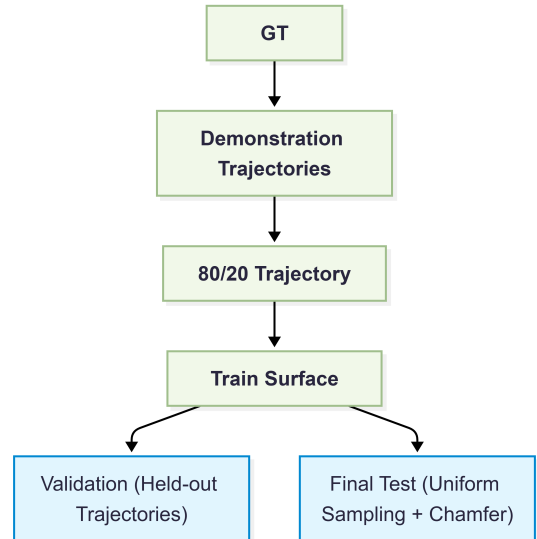


Fig. 2: Overview of the data generation, training, and evaluation pipeline.

a) *Trajectory-level 80/20 split*: Given demonstration trajectories $\{\mathcal{T}_i\}_{i=1}^M$, we perform a split at the *trajectory level* with fraction $p = 0.2$ (80/20). The held-out trajectories $\{\mathcal{T}_i\}_{i \in \mathcal{H}}$ (20%) are reserved strictly for **validation**, while the remaining trajectories $\{\mathcal{T}_i\}_{i \in \mathcal{S}}$ (80%) form the training set. Training samples are obtained by concatenating all points from the training trajectories; negative samples are generated *only* from these points to prevent data leakage.

b) *Validation on held-out trajectories*: After training on the 80% split, we evaluate h_θ on the held-out trajectories $\{\mathcal{T}_i\}_{i \in \mathcal{H}}$. This validation set is used to monitor generalization performance during training and to select the best-performing model checkpoint, including hyperparameter tuning.

c) *Final global geometric evaluation (Test)*: To assess the final quality of the learned surface, we perform a global geometric evaluation using a fresh set of uniformly sampled points covering the analytic surface and its vicinity. The zero level set of h_θ is extracted via Marching Cubes at a fixed resolution, and points are uniformly sampled from the resulting mesh for quantitative evaluation.

2) Geometric Metrics:

a) *Chamfer Distance*: To quantify the geometric accuracy of the learned surface, we compute the symmetric Chamfer Distance (CD) between the predicted surface $\hat{\mathcal{S}}$ and the ground-truth surface $\mathcal{S}$:

$$\text{CD}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{1}{|\hat{\mathcal{S}}|} \sum_{\hat{\mathbf{x}} \in \hat{\mathcal{S}}} \min_{\mathbf{x} \in \mathcal{S}} \|\hat{\mathbf{x}} - \mathbf{x}\|_2 + \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \min_{\hat{\mathbf{x}} \in \hat{\mathcal{S}}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2. \quad (6)$$

This metric captures both surface coverage and reconstruction accuracy and is widely used for evaluating implicit and explicit surface representations.

b) *Surface Deviation*: For control-related evaluations, we additionally report the mean Euclidean distance of trajectory points to the ground-truth surface, measuring true constraint satisfaction during execution.

C. Comparison of Loss Functions

We compare three training loss—Margin, k-NN, and Neural Pull—using 10 demonstration trajectories per surface (25 points each). Geometric accuracy is measured via symmetric Chamfer Distance (CD). Results (Table I) show all losses achieve similar reconstruction errors, with Neural Pull slightly better on Cylinder.

TABLE I: Symmetric Chamfer Distance comparison. Lower is better. The Neural Pull objective yields the most accurate geometric reconstruction across all test geometries.

Loss Function	Sphere	Ellipsoid	Cylinder
Margin Loss	0.0649	0.0704	0.1433
k-NN Loss	0.0639	0.0707	0.1439
Neural Pull	0.0668	0.0709	0.1426

Qualitative results are shown in Fig. 3. Neural Pull is chosen for downstream control because it explicitly aligns the network gradient with surface normals, improving the stability

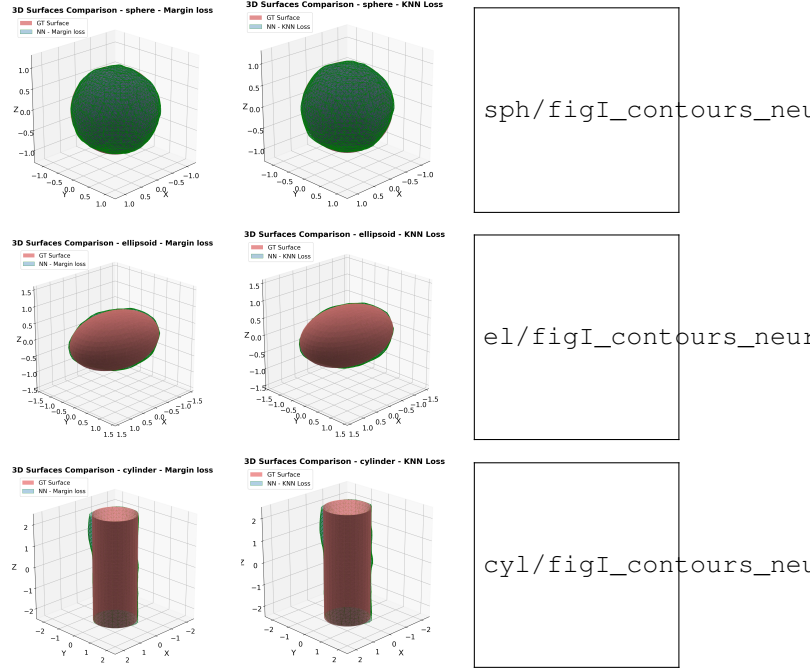


Fig. 3: Qualitative reconstruction results. While Chamfer distances are comparable, Neural Pull (right) provides a robust SDF approximation suitable for control.

of Newton-based projection. It also synergizes with Eikonal regularization, ensuring unit gradient norm.

D. Comparison of Regularizations

Using 35 trajectories per surface, we evaluate Neural Pull with: (i) no regularization, (ii) L2 weight decay, and (iii) Eikonal regularization. Table II shows that Eikonal improves accuracy on geometries with anisotropic curvature or open boundaries. Fig. 4 provides a qualitative comparison of the learned explicit surfaces.

TABLE II: Ablation on regularization terms using Neural Pull loss ($\downarrow$ better).

Regularizer	Sphere	Ellipsoid	Cylinder
None	0.0572	0.0647	0.1352
L2 Only	0.0562	0.0642	0.1339
Eikonal	0.0581	0.0634	0.1318

E. Evaluation of the Projected Imitation Controller

We evaluate the imitation controller alone, and combined with the Newton-based projection introduced in Sec. III. As shown in Fig. 5, the raw imitation controller reproduces the overall motion pattern but suffers from drift and fails to remain on the surface.

1) *Projection-Enhanced Controller*: We compare the **Raw Imitation Controller** (open-loop) with the **Projected Controller**, which applies Newton corrections to keep states on the constraint surface.

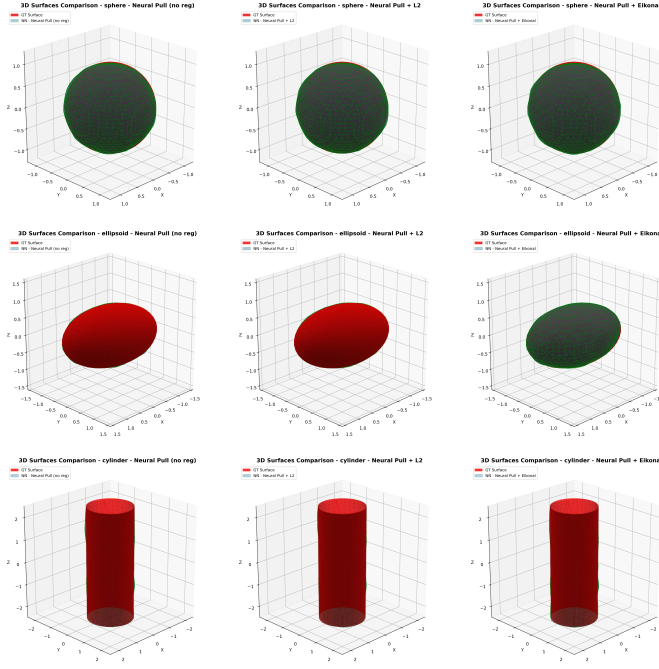


Fig. 4: Visual ablation of regularization terms. While L2 is effective for the uniform curvature of the Sphere, Eikonal regularization (right column) provides superior surface definition on the Ellipsoid and Cylinder, supporting the hypothesis that unit gradient enforcement aids in learning complex curvatures.

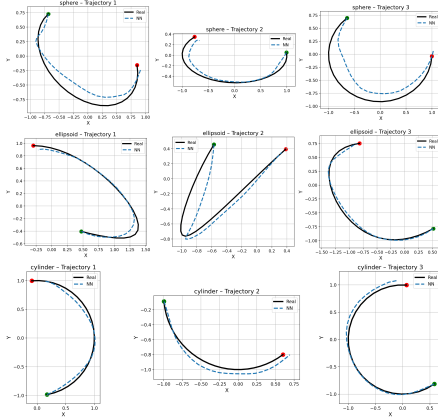


Fig. 5: Trajectories generated by the raw imitation controller (without projection). The policy reproduces the general motion pattern but exhibits drift and cannot remain on the surface.

a) *Geometric Consistency vs. Trajectory Tracking*: Trajectory tracking aims to follow the exact demonstration, while geometric consistency ensures states remain on $h_\theta(\mathbf{x}) = 0$.

The raw policy **does not enforce the constraint**, so small errors accumulate, causing drift off the surface. The Newton projection corrects this by moving predicted states back to the zero-level set along the surface normal. **Tangential drift is not corrected**, so the trajectory may deviate from the demonstration but remains geometrically valid.

Table III shows that projection reduces the mean distance to the surface by nearly an order of magnitude. Fig. 6 illustrates this effect qualitatively on Sphere trajectories.

TABLE III: Mean Euclidean distance to the surface ($\downarrow$ better). Projection enforces geometric consistency.

Geometry	Raw	Projected
Sphere	0.108	0.010
Ellipsoid	0.069	0.020
Cylinder	0.038	0.013

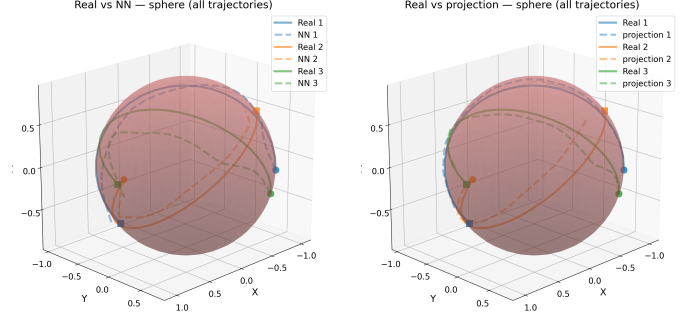


Fig. 6: Sphere trajectories. **Left**: Raw imitation drifts off surface. **Right**: Projected controller maintains proximity to surface using Newton corrections. Small residual violations may remain due to local updates and model errors, but geometric consistency is preserved.

F. Constraint Transferring

To evaluate the **transferability** of the learned constraint, we deploy an optimal controller in a set of scenarios that share the same underlying surface geometry but differ in task objectives. Importantly, no additional demonstrations are provided for these tasks: the controller relies solely on the learned constraint $h_\theta(\mathbf{x})$ and task-specific cost functions. This evaluation assesses whether the learned surface representation can be reused across qualitatively different planning problems.

We consider three representative scenarios: (1) goal reaching on the surface, (2) obstacle avoidance while remaining on the surface, and (3) reference trajectory tracking.

1) *Goal Reaching on the Surface*: In the first scenario, the controller is tasked with reaching goals placed arbitrarily on the learned surface. These goals differ from those observed in the demonstrations used for constraint learning.

The controller successfully generates trajectories that converge to the target while remaining close to the learned surface. Compared to demonstration trajectories, the resulting paths are typically smoother and shorter, reflecting the global optimization over the planning horizon. Geometric consistency is quantified by the mean distance to the ground-truth surface, reported in Table IV and visualized in Fig. 7.

2) *Obstacle Avoidance Under Shared Constraints*: We next evaluate whether the learned constraint can be reused in environments with additional task-specific requirements. In

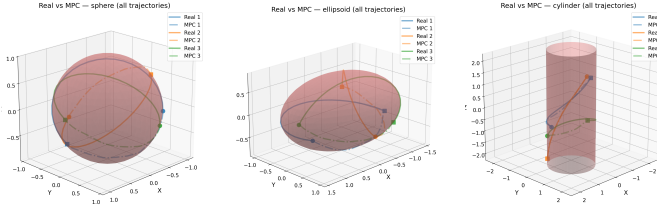


Fig. 7: Goal-reaching trajectories generated using the learned constraint. The controller adapts to new goals while maintaining adherence to the surface.

TABLE IV: Goal-reaching performance using the learned constraint. Mean distance to the surface indicates geometric consistency.

Geometry	Mean Distance to Surface
Sphere	0.0142
Ellipsoid	0.0214
Cylinder	0.0117

this scenario, the controller must avoid obstacles while simultaneously satisfying the surface constraint. Obstacle avoidance is encoded through inequality constraints, while the learned surface remains enforced as an equality constraint.

Results in Table V show that introducing obstacle avoidance does not significantly degrade surface adherence. Fig. 8 illustrates how the controller deforms the trajectory to bypass obstacles while remaining on the learned surface. This demonstrates that the constraint representation can be composed with new task constraints without retraining.

TABLE V: Surface adherence with and without obstacle avoidance. The learned constraint remains effective when combined with additional task constraints.

Geometry	Mean Distance to Surface	
	No Obstacles	With Obstacles
Sphere	0.0111	0.0126
Ellipsoid	0.0176	0.0202
Cylinder	0.0120	0.0114

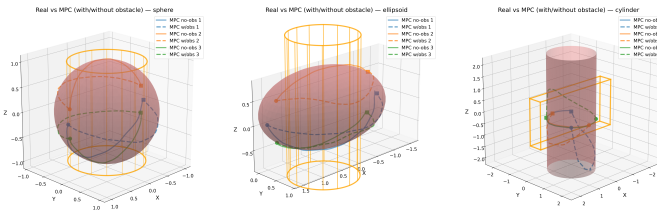


Fig. 8: Obstacle avoidance under the learned surface constraint. Trajectories adapt to obstacles while remaining on the surface.

3) *Reference Trajectory Tracking*: Finally, we evaluate constraint transfer in a reference tracking task, where the controller is required to follow a predefined trajectory lying on

the surface. This scenario tests whether the learned constraint supports tasks that prioritize tracking accuracy rather than goal convergence.

For the spherical geometry, the controller maintains both low surface deviation and low tracking error, as reported in Table VI. These results indicate that the learned constraint can be reused across diverse control objectives without modification.

TABLE VI: Reference trajectory tracking performance using the learned constraint (Sphere geometry).

Geometry	Distance to Surface	Tracking Error
Sphere	0.0115	0.0117

V. DISCUSSION

The results demonstrate that learning an explicit geometric constraint and integrating it tightly with control yields robust and geometrically consistent motion across multiple planning regimes.

a) *Loss Function Selection Beyond Geometric Accuracy*: Although the three evaluated loss functions (Margin, k-NN, and Neural Pull) achieve comparable Chamfer Distances, our results highlight that *pure geometric accuracy is not the sole criterion* for downstream control performance. In particular, the small numerical differences observed in Table I do not translate into meaningful distinctions in surface reconstruction quality.

Instead, Neural Pull emerges as the most suitable objective due to its *structural alignment* with the requirements of control. By explicitly leveraging the gradient of the learned implicit function during training, Neural Pull produces a vector field that is naturally consistent with surface normals. This property is critical for the stability and convergence of the Newton-based projection step used throughout our framework. In contrast, losses that only enforce scalar consistency ($h(\mathbf{x}) \approx 0$) provide no guarantees on gradient behavior, which can lead to unstable projections despite similar Chamfer errors.

b) *Interpreting Error Magnitudes*: To contextualize the reported surface errors, we relate them to the characteristic scale of each geometry. All surfaces are normalized to unit-order dimensions: the Sphere and Cylinder have unit radius ($R = 1$), while the Ellipsoid has semi-axes $(a, b, c) = (1.5, 1.0, 0.8)$, yielding an average characteristic length of approximately 1.1.

Under this normalization, a mean surface deviation of 0.01 corresponds to approximately **1% relative geometric error** on the Sphere and Cylinder. For the Ellipsoid, observed deviations in the range 0.015–0.02 correspond to roughly **1.3–1.8%** of the characteristic scale. These results indicate that both the projection-enhanced imitation controller and the optimal control formulation maintain surface adherence with better than **98% geometric accuracy** across all tested geometries, despite differences in curvature and topology.

c) *Role of Regularization in Control-Oriented Surface Learning:* The regularization ablation further reinforces this perspective. While L2 regularization performs well on the Sphere due to its uniform curvature, Eikonal regularization consistently improves reconstruction quality on the Ellipsoid and Cylinder. These geometries exhibit anisotropic curvature and open boundaries, where enforcing a unit gradient norm is essential for maintaining a valid signed distance function.

More importantly, Eikonal regularization directly benefits both projection and optimal control, as it stabilizes the Jacobians used in linearization and Newton updates. This confirms that regularization should be viewed not merely as a geometric prior, but as a control-critical design choice.

d) *Imitation Control and the Necessity of Projection:* The imitation controller successfully captures the qualitative motion patterns of the demonstrations but lacks any mechanism to enforce geometric constraints. As shown in Fig. 5 and Table III, small prediction errors accumulate over time, causing the trajectory to drift away from the surface.

The Newton-based projection step effectively resolves this issue by removing normal-direction errors and snapping the state back onto the learned surface. While projection does not correct tangential drift—and therefore does not guarantee trajectory-level imitation—it reliably enforces geometric feasibility. This distinction highlights an important conceptual separation between *trajectory tracking* and *constraint satisfaction*, with the latter being the primary requirement for contact-rich and surface-following tasks.

e) *Optimal Control with Learned Explicit Constraints:* The optimal control results demonstrate that the learned explicit constraint can be seamlessly embedded into standard constrained optimization frameworks. In goal-reaching scenarios, the controller generates smoother and shorter trajectories than the demonstrations, reflecting its ability to optimize motion globally while respecting the surface constraint.

When obstacle avoidance is introduced, the controller successfully balances equality constraints (surface adherence) with inequality constraints (collision avoidance), without sacrificing geometric consistency. In these cases, the resulting trajectories are not necessarily shortest-path solutions, as our evaluation prioritizes *constraint satisfaction* over path optimality. Jointly optimizing surface adherence, obstacle avoidance, and efficiency remains an interesting direction for future work.

Finally, in reference tracking mode, the controller maintains low error both relative to the surface and to the reference trajectory, even along high-curvature segments. This confirms that linearized surface constraints, combined with iterative projection, are effective for enforcing contact-consistent motion over extended horizons.

f) *Limitations and Future Work:* Despite these encouraging results, several limitations remain. First, projection relies on local Newton updates and may fail when initialized far from the surface or when gradients are poorly conditioned. Second, tracking experiments were limited to simple analytic geometries; extending the approach to complex, noisy real-world surfaces remains an open challenge.

Future work will explore adaptive linearization strategies, tighter coupling between learning and control objectives, and deployment on real robotic systems performing contact-rich manipulation tasks.

Overall, these results suggest that learning control-oriented explicit surfaces—rather than optimizing for geometric accuracy alone—is a promising direction for surface-constrained motion planning. Unlike previous approaches that primarily focus on trajectory imitation [12], our method explicitly aligns the learned surface gradients with control requirements, resulting in more stable and geometrically consistent motion.

ACKNOWLEDGMENT

The author thanks Baiyu Peng for supervision and guidance, and the LASA Laboratory at EPFL for their support.

REFERENCES

- [1] D. R. Scobee and S. S. Sastry, “Maximum likelihood constraint inference for inverse reinforcement learning,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [2] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, “Inverse constrained reinforcement learning,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, 2021, pp. 7390–7399.
- [3] G. Chou, N. Ozay, and D. Berenson, “Learning constraints from locally-optimal demonstrations under cost function uncertainty,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3682–3690, 2020.
- [4] G. Chou, H. Wang, and D. Berenson, “Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3827–3834, 2022.
- [5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2001.
- [6] P. Englert, N. A. Vien, and M. Toussaint, “Inverse kkt: Learning cost functions of manipulation tasks from demonstrations,” *The International Journal of Robotics Research*, vol. 36, no. 13–14, pp. 1474–1488, 2017.
- [7] B. Ichter and M. Pavone, “Robot motion planning in learned latent spaces,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.
- [8] B. Ichter, J. Harrison, and M. Pavone, “Learning sampling distributions for robot motion planning,” in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.
- [9] G. Sutanto, I. M. Rayas Fernández, P. Englert, R. K. Ramachandran, and G. S. Sukhatme, “Learning equality constraints for motion planning on manifolds,” in *Conference on Robot Learning (CoRL)*. PMLR, 2020, pp. 1665–1678.
- [10] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “Deepsdf: Learning continuous signed distance functions for shape representation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 165–174.
- [11] Y.-S. L. M. Z. Baorui Ma, Zhizhong Han, “Neural pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces,” in *International Conference on Machine Learning (ICML)*. PMLR, 2021.
- [12] N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada, “Ditto: Demonstration imitation by trajectory transformation,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates, 2024, pp. 7565–7572.