UAB
Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Árboles i Taulés, Albert. *Equality Learning Constraints from Demonstrations*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Grau en Enginyeria de Dades)

This version is available at https://ddd.uab.cat/record/337348

under the terms of the  license.

Equality Learning Constraints from Demonstrations

Albert Árboles

Resum– Aprendre restriccions d'igualtat a partir de demostracions permet que els robots aprofitin l'estructura geomètrica subjacent al moviment expert. Aquest treball proposa un marc per aprendre restriccions de superfície explícites a partir de trajectòries demostrades i utilitzar-les per generar moviments consistents amb la restricció. Un model neuronal explícit aprèn la superfície com el conjunt de nivell zero d'una funció escalar, mentre que un controlador neuronal d'imitació prediu moviments locals. Per garantir el compliment de la restricció, els estats predits es projecten sobre la superfície apresa mitjançant correccions basades en Newton. La restricció apresa també s'integra en control predictiu de model (MPC) per a tasques d'assoliment d'objectius i transferència de restriccions. Els resultats mostren que la projecció redueix la desviació mitjana un 91% en l'esfera, un 71% en l'el·lipsoide i un 66% en el cilindre, mentre que l'MPC manté errors inferiors al 2% de l'escala geomètrica característica. Aquests resultats mostren que combinar aprenentatge explícit de restriccions i MPC és efectiu per a la planificació de moviment sobre superfícies.

Paraules clau– Aprenentatge per demostració, aprenentatge de restriccions geomètriques, representacions neuronals explícites, control predictiu de model, planificació de moviment.

Abstract– Learning equality constraints from demonstrations enables robots to exploit the geometric structure underlying expert motion. This work proposes a framework for learning explicit surface constraints from demonstrated trajectories and using them for constraint-consistent motion generation. A neural explicit model learns the surface as the zero level set of a scalar function, while a neural imitation controller predicts local motion commands. To enforce constraint satisfaction, predicted states are projected onto the learned surface using Newton-based corrections. The learned constraint is also integrated into Model Predictive Control (MPC) for goal-reaching and constraint-transfer tasks. Experimental results show that projection reduces mean surface deviation by 91% on the sphere, 71% on the ellipsoid, and 66% on the cylinder, while MPC maintains surface errors below 2% of the characteristic geometric scale. These results show that explicit constraint learning combined with MPC is effective for surface-constrained motion planning.

Keywords– Learning from Demonstration, Geometric Constraint Learning, Neural Explicit Representations, Model Predictive Control, Motion Planning .

1 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 has successfully reproduced demonstrated trajectories but lacked the ability to infer the equality constraint—the surface—that generated them [1].

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 con-

• Contact Email: albert.arbolesi@autonoma.cat

• Supervised by: Lluís Ribas (Universitat Autònoma de Barcelona), Baiyu Peng and Aude Billard (Learning Algorithms and Systems Laboratory, EPFL)

• Academic year 2025/26

troller 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.

1.1 Project Context and Execution

This thesis was developed within the broader research context of equality-constrained motion learning at LASA. In particular, the work was intended to provide experimental and implementation support for the research line developed by Baiyu Peng, by exploring how equality constraints can be learned from demonstrations and exploited in downstream control tasks.

The project was executed as a progressive experimental pipeline composed of four main stages: data generation, constraint learning, controller evaluation, and constraint transfer through Model Predictive Control.

First, synthetic demonstration trajectories were generated on three analytical surfaces: a sphere, an ellipsoid, and a cylinder. These trajectories were used as the only source of positive on-surface samples. Since the objective was to learn the equality constraint from demonstrations, negative samples were generated by displacing trajectory points along locally estimated normals, and Gaussian noise was added to simulate uncertainty in the demonstrations.

Second, a neural explicit constraint model was implemented and trained to approximate each surface as the zero level set of a scalar function. Different training objectives were evaluated under the same data generation and validation protocol, allowing a controlled comparison between Margin Loss, k-NN Loss, and Neural Pull. Regularization strategies were then analyzed to study their effect on both geometric reconstruction and gradient stability.

Third, the learned constraint was integrated into a projected imitation controller. A behavioral cloning policy was trained from the demonstrations to predict local displacements, and each predicted state was corrected using Newton-based projection onto the learned surface. This stage was used to compare pure trajectory imitation against constraint-aware execution.

Finally, the learned constraint was reused in new control scenarios through a receding-horizon optimal control formulation implemented as Model Predictive Control (MPC). Goal-reaching, obstacle avoidance, and reference tracking tasks were evaluated without retraining the constraint model. This final stage tested whether the learned surface could be transferred to new objectives beyond the original demonstrations.

1.2 Organization of the Paper

The remainder of the paper is organized as follows. Section 2 reviews related work on constraint learning, equality constraints from demonstrations, and neural surface representations. Section 3 describes the proposed methodology, including the neural constraint model, training losses, projection mechanism, and Model Predictive Control formulation. Section 4 presents the experimental setup, evaluation metrics, and quantitative results. Section 5 concludes the paper by summarizing the main findings, limitations, and future directions.

2 Related Work

2.1 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 [2]. This was further extended by Maximum Entropy Constraint Learning (MECL), which jointly trains a policy and a constraint network to identify infeasible states [3].

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 [4] and later extended this to non-parametric Gaussian Process representations [5]. While effective for defining "safe vs. unsafe" zones (inequalities), these methods require an analytical model of the environment and are generally not designed to learn equality constraints.

2.2 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 [6].

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 [7]. However, these methods are typically tied to time-parameterized optimal control formulations. Alternatively, some works perform motion planning in learned latent spaces [8] or learn adaptive sampling distributions [9] 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 [10].

2.3 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) [11], 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 [12].

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.

3 METHODS

3.1 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 in $\mathbb{R}^N$.

We assume that all demonstrations are feasible with respect to an unknown true equality constraint $h^\star : \mathbb{R}^N \to \mathbb{R}$, such that all demonstrated states lie on the corresponding constraint surface

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

Accordingly,

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

The true constraint function $h^\star$ is not available and must be inferred solely from these on-surface samples. No explicit information about the constraint form or its derivatives is assumed.

3.1.1 Explicit Constraint Parameterization

We represent the constraint explicitly as the zero level set of a differentiable scalar function $h_\theta : \mathbb{R}^N \to \mathbb{R}$, 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 $\mathcal{S}_\theta$ accurately approximates the unknown surface $\mathcal{S}^\star$ using only demonstration data.

Two representation families are possible: (i) analytic functions and (ii) neural networks. Analytic models provide strong inductive biases but are only applicable when the true constraint parameterization is known a priori. Neural networks, in contrast, are more flexible and expressive, making them suitable for general scenarios involving unknown or complex geometries. This work therefore focuses on neural representations capable of modeling arbitrary constraint surfaces.

Illustrative Example For intuition, consider 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. 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 naturally extend to higher dimensions.

3.2 Constraint Learning

3.2.1 Neural Constraint Representation

The constraint function $h_\theta : \mathbb{R}^N \to \mathbb{R}$ 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.

3.2.2 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.

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

Negative samples Negative samples are generated by displacing each positive sample along a local normal estimated with PCA.

For each point $\mathbf{p}_i$, we first build a neighborhood

$$\mathcal{N}_i = \mathrm{kNN}(\mathbf{p}_i, k),$$

then compute its local mean and covariance:

$$\boldsymbol{\mu}_i = \frac{1}{k} \sum_{\mathbf{x} \in \mathcal{N}_i} \mathbf{x}, \qquad C_i = \sum_{\mathbf{x} \in \mathcal{N}_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^\top.$$

Let $(\lambda_{i,j}, \mathbf{v}_{i,j})$ be the eigenpairs of C_i ordered as $\lambda_{i,1} \leq \lambda_{i,2} \leq \lambda_{i,3}$. The local normal is chosen as

$$\hat{\mathbf{n}}_i = \mathbf{v}_{i,1}, \qquad \hat{\mathbf{n}}_i \leftarrow \frac{\hat{\mathbf{n}}_i}{\|\hat{\mathbf{n}}_i\| + \varepsilon}.$$

The neighborhood size k is selected adaptively from a global PCA of the full demonstration set. With centered data $X_c = X - \bar{X}$,

$$\Sigma = \frac{X_c^\top X_c}{N},$$

and global eigenvalues ordered as $\lambda_1^{(g)} \geq \lambda_2^{(g)} \geq \lambda_3^{(g)}$, we define

$$r = \frac{\lambda_3^{(g)}}{\lambda_1^{(g)} + \varepsilon}.$$

Then:

- if $r < 0.01$, set $k = 25$;
- if $0.01 \leq r < 0.05$, set $k = 15$;
- if $0.05 \leq r < 0.3$, set $k = 12$;
- if $r \geq 0.3$, set $k = 10$.

Finally, negatives are sampled as

$$\mathbf{x}_i^- = \mathbf{x}_i + s_i\,\delta\,\hat{\mathbf{n}}_i, \qquad s_i \in \{-1, +1\},$$

where $\delta = \texttt{std}$ is either constant or randomly sampled, depending on the selected configuration. A balanced 1:1 ratio of positive and negative samples is maintained.

As illustrated in Fig. 1, negative samples are generated by PCA-based normal displacement of on-surface points, resulting in a balanced training set.

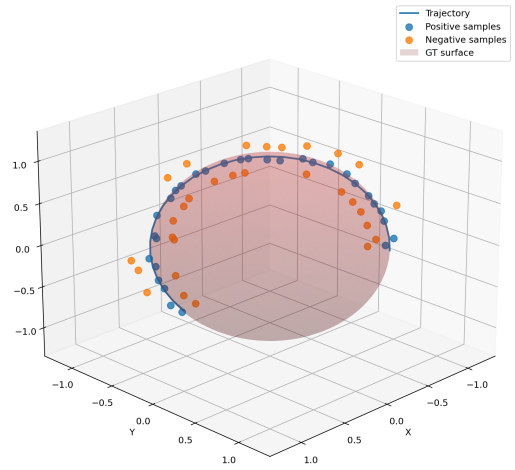

Fig. 1: Example demonstration trajectory and the resulting positive and negative samples generated for explicit surface learning.

3.2.3 *Training Objectives (Loss Functions)*

To learn a zero-level set constraint function, this work introduces and compares three loss functions.

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

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

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

This loss encourages the network to predict zero values on demonstrated states, while pushing off-surface samples to have function values with magnitude at least m. The Softplus term provides a smooth penalty for violations of this margin. Importantly, this objective supervises only the *scalar value* of the constraint function and does not explicitly constrain the orientation or smoothness of its gradients away from the surface.

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

$$\mathcal{L}_{\text{kNN}} = \mathbb{E}_{\mathbf{x}_{\text{pos}}}[|h_\theta(\mathbf{x}_{\text{pos}})|] \tag{2}$$

$$+\,\alpha\,\mathbb{E}_{\mathbf{x}_{\text{neg}}}\left[\text{softplus}(d_k(\mathbf{x}_{\text{neg}}) - |h_\theta(\mathbf{x}_{\text{neg}})|)\right], \tag{3}$$

where $d_k(\mathbf{x}_{\text{neg}})$ denotes the mean Euclidean distance from a negative sample to its k nearest positive samples.

In this formulation, positive samples are explicitly constrained to lie on the zero-level set through the first term. For negative samples, the effective margin varies spatially according to the local density of demonstrations: negatives close to the surface are penalized more strongly, while negatives farther away are allowed larger values of $|h_\theta|$. This results in a locally adaptive constraint that better reflects the geometry of the demonstrations. However, the loss still relies purely on scalar supervision and does not impose any explicit regularization on the gradient of h_θ.

Adaptive Neural Pull Loss The Neural Pull loss enforces geometric consistency by explicitly projecting query points toward the zero-level set using the gradient of the learned constraint. Given a query point $\mathbf{x}$, a projected point $\hat{\mathbf{x}}$ is obtained via a gradient-based correction with an adaptive step size $\eta(\mathbf{x})$, inversely related to $\|\nabla h_\theta(\mathbf{x})\|$:

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

This projection is applied to *all* samples, including both positive and negative points. For positive samples, the update is near-identity since $h_\theta(\mathbf{x}_{\text{pos}}) \approx 0$, whereas off-surface points are actively pulled toward the learned surface.

Let $\mathcal{N}_k(\hat{\mathbf{x}})$ denote the set of indices of the k nearest positive samples to the projected point $\hat{\mathbf{x}}$ in Euclidean distance. The loss minimizes the distance between projected points and nearby demonstrations, together with an explicit surface consistency term for positive samples:

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

Unlike the previous losses, Neural Pull directly leverages the gradient of h_θ during training, encouraging projection directions that align with true surface normals. As a result, the learned constraint not only approximates the zero-level set accurately, but also yields well-conditioned gradients that are particularly suitable for Newton-based projection and downstream control.

3.2.4 *Regularization and Optimization*

We include two regularization terms to improve geometric fidelity and training stability.

Eikonal Regularization To encourage locally well-scaled gradients and stable projection behavior, we penalize deviations from unit gradient norm:

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

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.

3.3 Motion Control with Learned Constraint

After learning the geometric constraint $h_\theta(x)$, we consider two alternative control strategies that exploit the learned surface in different settings: (i) a Projected Imitation Controller, and (ii) Model Predictive Control (MPC).

These two approaches are independent and address complementary use cases.

3.3.1 Projected Imitation Controller

We consider a data-driven controller that combines imitation learning with explicit enforcement of a learned equality constraint. The controller is defined by a policy

$$\pi_\phi : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N,$$

parameterized by a feedforward neural network and trained via behavioral cloning on demonstration trajectories.

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

$$\Delta x_t = \pi_\phi([x_t, g]), \qquad \hat{x}_{t+1} = x_t + \Delta x_t, \tag{4}$$

where $[\cdot, \cdot]$ denotes concatenation. The policy operates on state–goal pairs and follows a shallow residual architecture with ReLU activations and layer normalization. Residual connections promote stable training and encourage locally consistent updates. The output displacement is bounded using a scaled hyperbolic tangent activation.

Training minimizes the mean squared error between predicted and demonstrated displacements,

$$\Delta x_t^{\text{demo}} = x_{t+1} - x_t,$$

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

Projection onto the learned constraint The displacement predicted by the imitation policy does not, in general, guarantee satisfaction of the learned constraint. Therefore, each unconstrained next state $\hat{x}_{t+1}$ is projected onto the zero-level set of the learned constraint function h_θ.

Starting from $x^0 = \hat{x}_{t+1}$, we apply an iterative Newton-style update

$$x^{k+1} = x^k - \frac{h_\theta(x^k)}{\|\nabla h_\theta(x^k)\|^2} \nabla h_\theta(x^k), \tag{5}$$

which locally moves the state along the surface normal to reduce the constraint violation. A backtracking line search ensures monotonic decrease of $|h_\theta(x)|$ and improves numerical stability. The iteration terminates once the constraint residual falls below a fixed tolerance or a maximum number of iterations is reached.

To complement the update equations, we analyze trajectories before and after projection to make this correction step explicit: raw policy predictions can drift away from the surface, while projected states are pulled back onto the learned manifold (Fig. 4).

If the projection fails to converge, the controller falls back to the unconstrained prediction for that step. This procedure enforces geometric consistency while preserving the structure of the demonstrated motion.

3.3.2 Model Predictive Control with Learned Surface Constraint

When a system model and task-specific cost function are available, the learned surface can be incorporated into a receding-horizon optimal control formulation, implemented here as Model Predictive Control (MPC). We consider a discrete-time system with first-order dynamics

$$x_{k+1} = x_k + \Delta t \, u_k, \tag{6}$$

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

$$h_\theta(x_k) = 0, \qquad \forall k = 1, \ldots, N. \tag{7}$$

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,

$$\min_{\{u_k\}} \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N) \quad \text{s.t.} \quad h_\theta(x_k) = 0, \tag{8}$$

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 formulation does not rely on additional demonstration data and is particularly suitable for transferring the learned constraint across different tasks, goals, or cost functions.

4 Experiments

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

We first compare the learning accuracy of the constraint model under different loss functions and regularization schemes. Based on this comparison, we select a loss–regularization configuration and evaluate a projected imitation controller against a raw imitation baseline in terms of true constraint satisfaction.

Finally, to evaluate the transferability of the learned constraints, we design new scenarios that share the same underlying surfaces but differ in their task objectives. In these scenarios, MPC is deployed using the learned constraints, and its performance is quantitatively evaluated on unseen tasks.

4.1 Experimental Setup

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

Sphere The spherical surface is represented by the signed-distance function

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

Ellipsoid The ellipsoidal surface is defined by the implicit equation

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

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 using PCA-based local normals with adaptive neighborhood size, and Gaussian noise is added to simulate sensor uncertainty.

4.1.1 Noise Injection

To improve robustness and simulate sensor noise, Gaussian noise is added to positive samples during training:

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

All sampling procedures are seeded for reproducibility.

4.2 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.

4.2.1 Evaluation Protocol (Learned Surface)

We evaluate the learned explicit surface $h_\theta : \mathbb{R}^3 \to \mathbb{R}$ 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.

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.

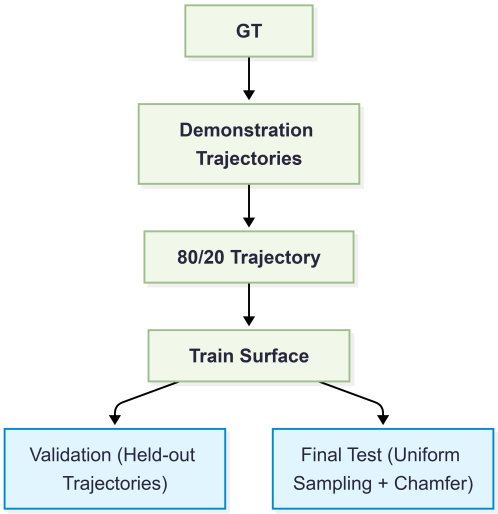

Fig. 2: Overview of the data generation, training, and evaluation pipeline.

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.

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 5,000 uniformly sampled points covering the entire 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.

4.2.2 Geometric Metrics

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 $\mathcal{S}$:

$$\text{CD}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{1}{|\hat{\mathcal{S}}|}\sum_{\hat{\mathbf{x}}\in\hat{\mathcal{S}}} \min_{\mathbf{x}\in\mathcal{S}} \|\hat{\mathbf{x}}-\mathbf{x}\|_2 + \frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}\in\mathcal{S}} \min_{\hat{\mathbf{x}}\in\hat{\mathcal{S}}} \|\mathbf{x}-\hat{\mathbf{x}}\|_2. \tag{9}$$

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

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.

4.3 Comparison of Loss Functions

We compare three training losses—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 1) show all losses achieve similar reconstruction errors, with no single loss dominating across all geometries.

TABLE 1: SYMMETRIC CHAMFER DISTANCE COMPARISON. LOWER IS BETTER. RESULTS ARE REPORTED FOR NOISY DEMONSTRATIONS ($\sigma = 0.03$).

Loss Function	Sphere	Ellipsoid	Cylinder
Margin Loss	0.0689	0.0750	0.1471
k-NN Loss	0.0635	**0.0722**	**0.1440**
Neural Pull	**0.0612**	0.0735	0.1456

Neural Pull is chosen for downstream control because it explicitly uses the gradient of the learned constraint during training, improving the stability of Newton-based projection. Additional qualitative reconstruction results are provided in Appendix A.

4.4 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 2 shows that Eikonal improves accuracy on geometries with anisotropic curvature or open boundaries.

TABLE 2: ABLATION ON REGULARIZATION TERMS USING NEURAL PULL LOSS (↓ 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**

4.5 Evaluation of the Projected Imitation Controller

We evaluate the imitation controller alone, and combined with the Newton-based projection introduced in Sec. 3.3.1. As shown in Fig. 3, the raw imitation controller reproduces the overall motion pattern on the sphere but suffers from drift and fails to remain on the surface.

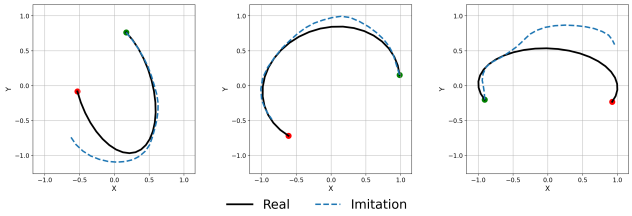

Fig. 3: Trajectory comparison on the sphere for the raw imitation controller without projection. The policy reproduces the general motion pattern but exhibits drift and fails to remain on the surface.

4.5.1 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.

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 3 shows that projection substantially reduces the mean distance to the surface, with the largest improvement observed on the Sphere. Fig. 4 illustrates this effect qualitatively on Sphere trajectories.

To contextualize these values, all geometries are normalized to unit-order scale. 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 a characteristic length of approximately 1.1. Under this normalization, a mean deviation of 0.01 corresponds to approximately **1% relative geometric error**, and 0.015–0.02 on the Ellipsoid corresponds to **1.3–1.8%**.

TABLE 3: MEAN EUCLIDEAN DISTANCE TO THE SURFACE (↓ BETTER). PROJECTION ENFORCES GEOMETRIC CONSISTENCY.

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

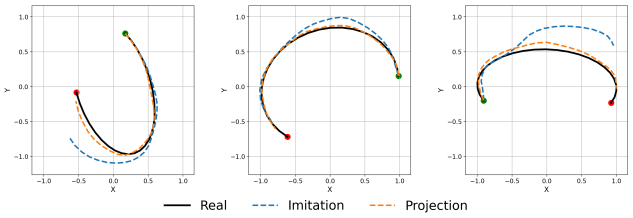

Fig. 4: Trajectory comparison on the sphere between the real trajectory, the raw imitation controller, and the projection-enhanced controller. The raw imitation trajectory drifts away from the surface, while Newton-based projection improves geometric consistency.

4.6 Constraint Transferring

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

We consider three representative new tasks: (1) goal reaching, (2) obstacle avoidance, and (3) reference trajectory tracking.

TABLE 4: 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

4.6.1 Goal Reaching

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 4 and visualized in Fig. 5.

4.6.2 Obstacle Avoidance

We next evaluate whether the learned constraint can be reused in environments with additional task-specific requirements. In 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 5 show that introducing obstacle avoidance does not significantly degrade surface adherence. Fig. 6 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 5: SURFACE ADHERENCE WITH AND WITHOUT OBSTACLE AVOIDANCE. THE LEARNED CONSTRAINT REMAINS EFFECTIVE WHEN COMBINED WITH ADDITIONAL TASK CONSTRAINTS.

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

4.6.3 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 6. These results indicate that the learned constraint can be reused across diverse control objectives without modification.

TABLE 6: REFERENCE TRAJECTORY TRACKING PERFORMANCE USING THE LEARNED CONSTRAINT (SPHERE GEOMETRY).

Geometry	Distance to Surface	Tracking Error
Sphere	0.0115	0.0117

5 CONCLUSION

This section concludes the work by summarizing the main findings of the proposed framework for learning equality constraints from demonstrations and using them for constraint-aware motion generation. The results show that explicitly learning a geometric constraint, and integrating it into projection-based and optimization-based controllers, provides a reliable way to generate motions that remain consistent with the underlying surface. The main conclusion is that surface-constrained motion planning should not be evaluated only in terms of geometric reconstruction accuracy, but also in terms of gradient quality, projection stability, and downstream control performance.

Although the three evaluated loss functions, Margin, k-NN, and Neural Pull, achieve comparable Chamfer Distances, the results highlight that pure geometric accuracy is not the sole criterion for downstream control performance. In particular, the small numerical differences observed in the reconstruction metrics do not translate into meaningful distinctions in surface reconstruction quality alone.

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 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 the pipeline. In contrast, losses that only enforce scalar consistency, such as $h(\mathbf{x}) \approx 0$, provide no direct guarantee on gradient behavior, which can lead to unstable projections despite similar Chamfer errors.

To contextualize the reported surface errors, they can be related to the characteristic scale of each geometry. All surfaces are normalized to unit-order dimensions: the Sphere and Cylinder have unit radius, while the Ellipsoid has semi-axes $(1.5, 1.0, 0.8)$, yielding an average characteristic length of approximately 1.1. Under this normalization, a mean surface deviation of 0.01 corresponds to roughly 1% relative geometric error on the Sphere and Cylinder. For the Ellipsoid, observed deviations in the range 0.015–0.02 correspond to approximately 1.3%–1.8% of the characteristic scale. These results indicate that both the projection-enhanced controller and the MPC maintain surface adherence with better than 98% geometric accuracy across all tested geometries, despite differences in curvature and topology.

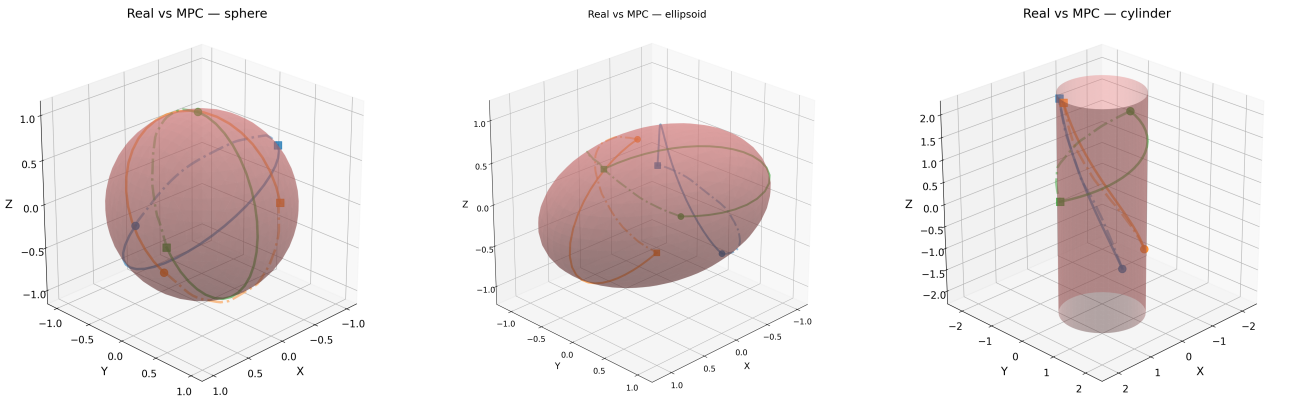

Fig. 5: Goal-reaching trajectories generated using the learned constraint. The controller adapts to new goals while maintaining adherence to the surface. Solid lines represent reference trajectories used for visual comparison, while dashed lines represent the MPC-generated trajectories.

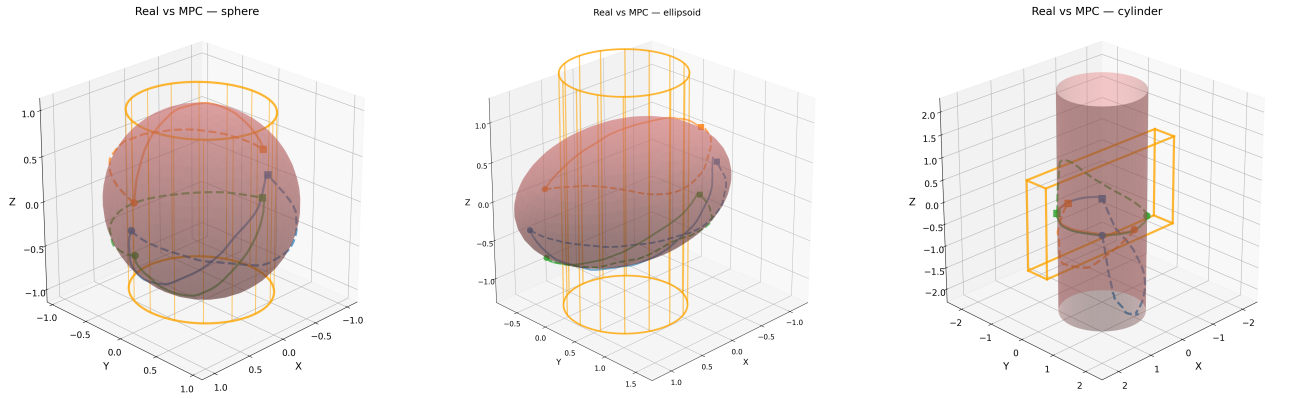

Fig. 6: Obstacle avoidance under the learned surface constraint. Trajectories adapt to obstacles while remaining on the surface. **Obstacles are depicted as orange wireframe shapes**, while the learned surface is shown in red. Solid lines represent reference trajectories used for visual comparison, while dashed lines represent the MPC-generated trajectories.

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 important for maintaining a valid Signed Distance Function approximation. More importantly, Eikonal regularization directly benefits both projection and MPC, 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.

The imitation controller successfully captures the qualitative motion patterns of the demonstrations but lacks any mechanism to enforce geometric constraints. As shown by the comparison between the raw and projected controllers, 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 correcting the state back toward the learned zero-level set. 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.

The MPC results demonstrate that the learned explicit constraint can be embedded into a receding-horizon optimization framework. In goal-reaching scenarios, MPC generates smooth and feasible trajectories while respecting the surface constraint. When obstacle avoidance is introduced, the controller successfully balances equality constraints, associated with surface adherence, with additional task constraints for collision avoidance, without substantially degrading geometric consistency. However, in these cases, MPC does not necessarily select the shortest or most efficient path; the current evaluation focuses primarily on constraint satisfaction rather than trajectory optimality. Investigating how to jointly optimize for surface adherence, obstacle avoidance, and path efficiency remains an interesting direction for future work.

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

Despite these encouraging results, several limitations re-

main. First, projection relies on local Newton updates and may fail when initialized far from the surface or when gradients are poorly conditioned. Second, the experiments were limited to analytic geometries. Extending the approach to complex, noisy real-world surfaces and physical robotic systems remains an important direction for future work. Future research should also explore adaptive linearization strategies, tighter coupling between learning and control objectives, and deployment on real robotic systems performing contact-rich manipulation tasks.

Overall, these findings suggest that learning control-oriented explicit constraints, rather than relying on imitation alone or optimizing only for geometric accuracy, is a promising direction for surface-constrained motion planning. Beyond the scope of this thesis, the work carried out under the supervision of Baiyu Peng has also contributed to a broader research project that has resulted in a manuscript currently under review in *IEEE Robotics and Automation Letters*, in which the author is included as a co-author. The source code developed for this work, including examples of the experiments, is publicly available in the project repository [13].

REFERENCES

[1] 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.

[2] D. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2020.

[3] 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.

[4] 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.

[5] 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.

[6] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2001.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-Pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces."

[13] A. Árboles, "Equality learning constraint from demonstrations," https://github.com/tinips/Equality-Learning-Constraint-From-Demonstrations, 2026, source code repository.

A ADDITIONAL QUALITATIVE RECONSTRUCTION RESULTS

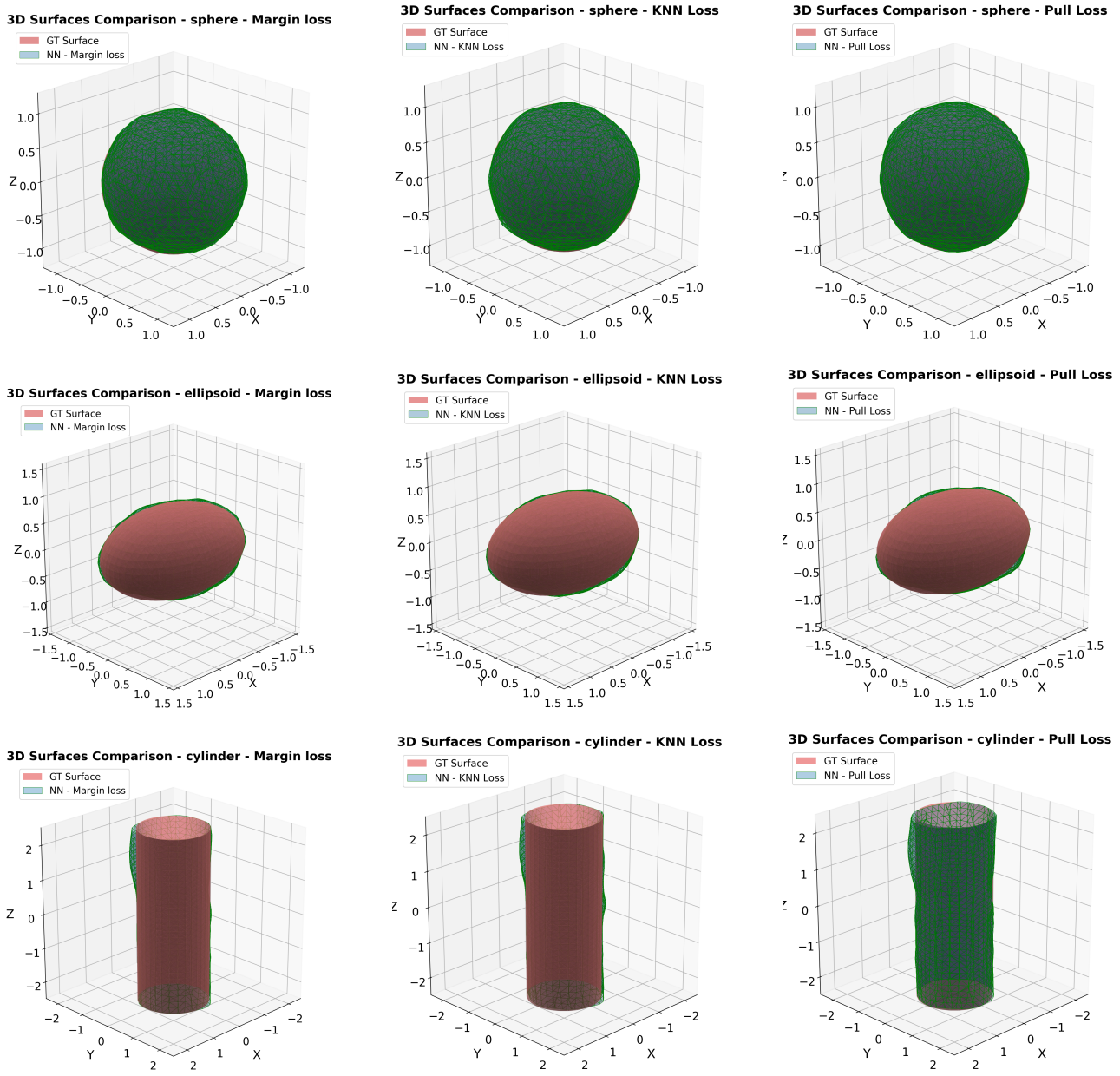

Fig. 7: Additional qualitative reconstruction results for the three evaluated geometries. Rows correspond to Sphere, Ellipsoid, and Cylinder; columns correspond to Margin Loss, k-NN Loss, and Neural Pull.