Experimental Techniques for Topology Control on DeepSDFs

Stephanie Atherton, Marina Levay, Ualibyek Nurgulan, Erendiro Pedro, Shree Singhi

July 2025

1 Introduction

In the Topology Control project mentored by Professor Paul Kry and project assistants Daria Nogina and Yuanyuan Tao, we sought to explore preserving topological invariants of meshes within the framework of DeepSDFs. Deep Signed Distance Functions are a neural implicit representation used for shapes in geometry processing, but they don't come with the promise of respecting topology. After finishing our ML pipeline, we explored various topology-preserving techniques through our simple, initial case of deforming a "donut" (a torus) into a mug.

2 DeepSDF

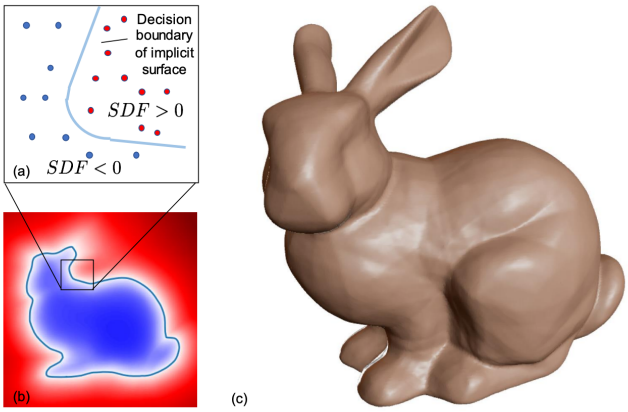


Figure 1: Signed Distance Field (SDF) representation of a 3D bunny. The network predicts the signed distance from each spatial point to the surface.

Signed Distance Functions (SDFs) return the shortest distance from any point in 3D space to the surface of an object. Their sign indicates spatial relation: negative if the point lies inside, positive if outside. The surface itself is defined implicitly as the zero-level set: the locus where $\mathrm{SDF}(x) = 0$.

In 2019, Park et al. introduced *DeepSDF*, the first method to learn a continuous SDF directly using a deep neural network (Park et al., 2019). Given a shape-specific latent code $z \in \mathbb{R}^d$ and a 3D point $x \in \mathbb{R}^3$, the network learns a continuous mapping

$$f_\theta(z_i, x) \approx \mathrm{SDF}^i(x),$$

where f_θ takes a latent code z_i and a 3D query point x and returns an approximate signed distance. The training set $X := \{(x, s) : \mathrm{SDF}(x) = s\}$. Training minimizes the clamped L1 loss between predicted and true distances

$$\mathcal{L}(f_\theta(x), s) = |\mathrm{clamp}(f_\theta(x), \delta) - \mathrm{clamp}(s, \delta)|$$

with

$$\mathrm{clamp}(x, \delta) = \min(\delta, \max(-\delta, x)).$$

Clamping focuses the loss near the surface, where accuracy matters most. The parameter δ sets the active range.

This is trained on a dataset of 3D point samples and corresponding signed distances. Each shape in the training set is assigned a unique latent vector z_i, allowing the model to generalize across multiple shapes.

Once trained, the network defines an implicit surface through its decision boundary, precisely where $f_\theta(z, x) = 0$. This continuous representation allows smooth shape interpolation, high-resolution reconstruction, and editing directly in latent space.

2.0.1 Training Field Notes

We sampled training data from two meshes, torus.obj and mug.obj using a mix of blue-noise points near the surface and uniform samples within a unit cube. All shapes were volume-normalized to ensure consistent interpolation.

DeepSDF is designed to intentionally overfit. Validation is typically skipped. Effective training depends on a few factors: point sample density, network size, shape complexity, and sufficient epochs.

After training, the implicit surface can be extracted using Marching Cubes or Marching Tetrahedra to obtain a polygonal mesh from the zero-level set.

Training Parameters	
SDF Delta	1.0
Latent Mean	0.0
Latent SD	0.01
Loss Function	Clamped L1
Optimizer	Adam
Network Learning Rate	0.001
Latent Learning Rate	0.01
Batch Size	2
Epochs	5000
Max Points per Shape	3000
Network Architecture	
Latent Dimension	16
Hidden Layer Size	124
Number of Layers	8
Input Coordinate Dim	3
Dropout	0.0
Point Cloud Sampling	
Radius	0.02
Sigma	0.02
Mu	0.0
Number of Gaussians	10
Uniform Samples	5000

For higher shape complexity, increasing the latent dimension or training duration improves reconstruction fidelity.

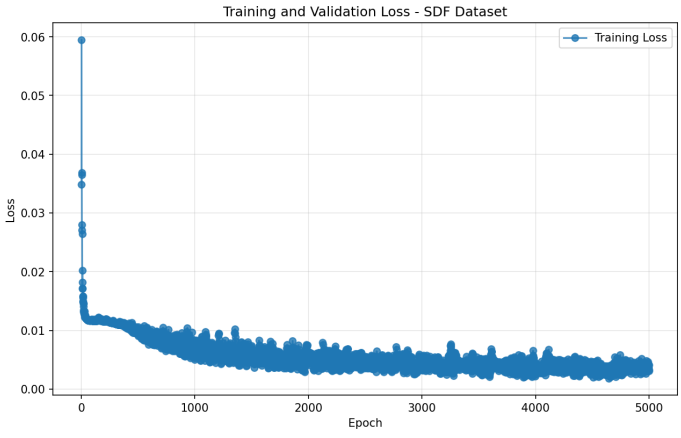


Figure 2: Training loss of deepsdf

Latent Space Interpolation

One compelling application is interpolation in latent space. By linearly blending between two shape codes z_a and z_b, we generate new shapes along the path

$$z(t) = (1 - t) \cdot z_a + t \cdot z_b, \quad t \in [0, 1].$$

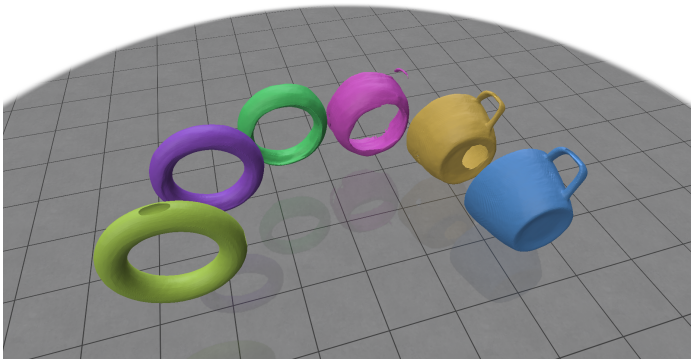


Figure 3: Latent space interpolation between mug and torus.

While DeepSDF enables smooth morphing between shapes, it exposes a core limitation: a lack of topological consistency. Even when the source and target shapes share the same genus, interpolated shapes can exhibit unintended holes, handles, or disconnected components. These are not artifacts, they reveal that the model has no built-in notion of topology as reported by Liu et al, 2022.

2.0.2 Preserving Genus

A torus and a mug are considered the same shape up to *homotopy*. Homotopy is the mathematical way of saying we can deform one shape into another without having to cut or glue any parts of our shape. The torus and the mug are both closed, orientable forms that are homotopic to one another, so they also have the same *genus*, or number of holes, as a topological invariant. For the torus

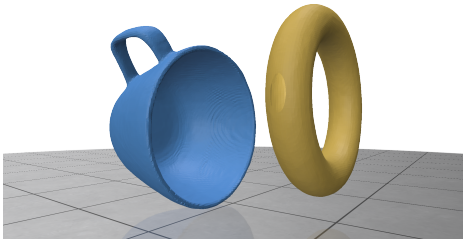


Figure 4: The torus has genus $g = 1$ and is homotopic to the mug.

we have 1 hole in the center and for the mug we have 1 hole between the cup part and the handle, so both forms have genus $g = 1$. When passing our shapes through our model, we want to preserve their genus. Finding a smooth path in latent space for our shapes will help us to do so.

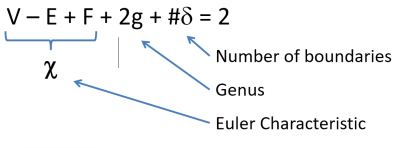


Figure 5: The Euler characteristic χ holds a relationship with the genus g and boundary components b of a surface.

This becomes especially clear when examined in low-dimensional latent spaces (e.g., $z_{\text{dim}} = 2$), where linear interpolation between two topologically similar shapes often crosses regions of different genus. In higher-dimensional spaces, the manifold of valid shapes may be more forgiving, potentially containing topology-preserving paths, but they're unlikely to be straight lines. This motivates a broader goal: to enable smoother transitions by introducing regularization techniques, such as those proposed by Liu et al., 2022, which encourage structural preservation during interpolation. And a second idea is to rather than relying on naive linear paths, we shift our focus to discovering homotopic trajectories in latent space, paths that traverse the learned shape manifold while maintaining geometric and topological coherence. In effect, interpolation becomes a trajectory planning problem: navigating from one latent point to another without straying into regions of invalid or unstable topology.

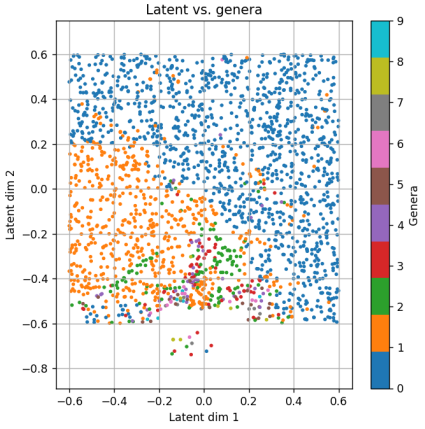


Figure 6: Visualizing genera count in 2D latent space

3 Lipschitz Regularization

Our goal is to achieve a smooth transition, and smoothness in the latent space is described by a metric called the Lipschitz bound. Liu et al, 2022, proposed a smoothness regularizer to minimize a learned Lipschitz bound on the latent vector of a neural field (the implicit function encoded as the NN from the DeepSDF which maps input 3D coordinates to the SDF values).

To use this method, we added a normalization layer and augmented the DeepSDF loss function with a simple regularization term encouraging small Lipschitz.

Topology Control: Pathfinding for Genus Preservation

Recall that while DeepSDFs can be an advantageous shape presentation, they are not known for preserving topology. When we "preserve topology", what we really want to do is preserve a *topological invariant* in deforming shapes. In our case, we wish to learn a path between two latent codes guided by a genus classifier (as well as a volume regressor) with the hopes that the topological guidance provided will help us to meaningfully interpolate between codes.

4 Pathfinding in Latent Space

Suppose you are a hiker hoping to a reach a point A from a point B amidst a chaotic mountain range. Now your goal is to plan your journey so that there will be minimal height change in your trajectory, i.e., you are a hiker that hates going up and down much! Fortunately, an oracle gives us a magical device, let us call it f, that can give us the exact height of any point we choose. In other words, we can query $f(x)$ for any position x, and this device is differentiable - $f'(x)$ exists!

Metaphors aside, the problem of planning a path from a particular latent vector A to another B in the learned latent space would greatly benefit from another auxilary network that learns the mapping from the latent vectors to the desired topological or geometric features. We will introduce a few ways to use this magical device - a simple neural network.

4.1 Gradient as the Compass

Now the best case scenario would be to stay at the same height for the whole of the journey, no? This greedy approach puts a hard constraint on the problem, but it also greatly reduces the possible space of paths to take. For our case, we would love to move toward the direction that does not change our height, and this set of directions precisely forms the nullspace of the gradient.

No height change in mathematical terms means that we look for directions where the derivatives equal zero, as in

$$D_v f(x) = \nabla f(x) \cdot v = 0,$$

where the first equality is a fact of the calculus and the second shows that any desirable direction $v \in \text{null}(\nabla f(x))$.

This does not give any definitive directions for a position x but a set of possible good directions. Then a greedy approach is to take the direction that aligns most with our relative position against the goal - the point B.

Almost done, but the final question is what would we do if the direction we want to take toward B is orthogonal to the gradient vector? That is, we should also take some steps toward the gradient, but how much of a stride we

take is also important. Toward this, let x be the current position and $\Delta x = \alpha \nabla f(x) + n$, where n is the projection of $B - x$ to $\text{null}(\nabla f(x))$. Then we want $f(x + \Delta x) = f(B)$. Note that

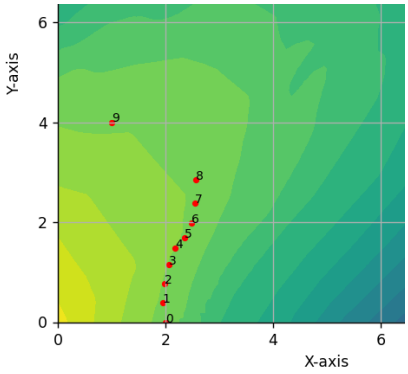
$$f(B) = f(x + \Delta x) \approx f(x) + \nabla f(x) \cdot \Delta x \tag{1}$$

$$= f(x) + \nabla f(x) \cdot (\alpha \nabla f(x) + n) \tag{2}$$

$$\iff \alpha \approx \frac{f(B) - f(x)}{|\nabla f(x)|^2}. \tag{3}$$

Now we just take $\Delta x = \frac{f(B)-f(x)}{|\nabla f(x)|^2} \nabla f(x) + \text{proj}_{\text{null}(\nabla f(x))}(B - x)$ for each position x on the path.

Results. In figure 7, we present some results regarding how the algorithm fared in learned latent spaces.



(a) On a latent to volume space

(b) On a latent to ——— space

Figure 7: Path finding via gradients in different contexts

4.2 Optimizing Vertical Laziness

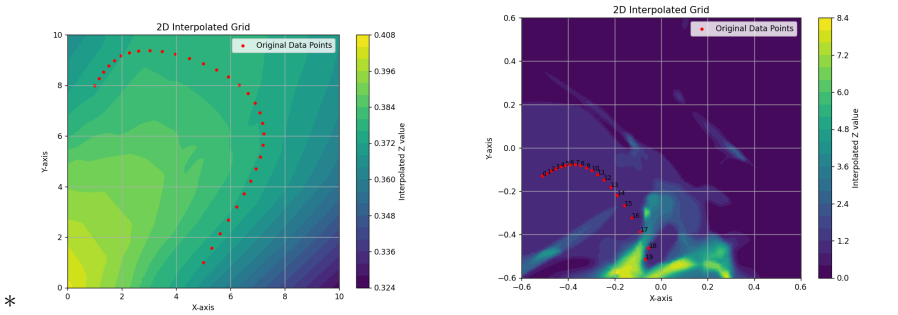
Sometimes it is impossible to hope for the best case scenario. Even when $f(A) = f(B)$, it might happen that the latent space is structured such that A and B are in different components of the level set of the function f. Then there is no hope for a smooth hike without ups and downs! But the situation is not completely hopeless if we are fine with taking detours as long as such undesirables are minimized. We frame the problem as

$$\underset{x_i \in L, \forall i}{\text{argmin}} \sum_{i=1}^{n} |f(x_i) - f(B)|^2 + \lambda \sum_{i=1}^{n-2} |(x_{i+2} - x_{i+1}) - (x_{i+1} - x_i)|^2,$$

where L is the latent space and $\{x_i\}_{i=1}^n$ is the sequence defining the path such that $x_1 = A$ and $x_n = B$. The first term is to encourage consistency in the

function values, while the second term discourages sudden changes in curvature, thereby ensuring smoothness. Once defined, various gradient-based optimization algorithms can be used on this problem, which is now imposed with a relatively *soft* constraint.

Results. In figure 8, we present some results regarding how the algorithm fared in learned latent spaces.



(a) On a latent to volume space (b) On a latent to expected genus space

Figure 8: Path finding via optimization in different contexts

4.3 A Geodesic Path

One alternative attempt of note to optimize pathfinding used the concept of *geodesics*, or the shortest, smoothest distance between two points on a Riemannian manifold. In the latent space, one is really working over a latent data manifold, so thinking geometrically, we experimented with a geodesic path algorithm. In writing this algorithm, we set our two latent endpoints and optimized the points in between by minimizing the energy of our path on the output manifold. This alternative method worked similarly well to the optimized pathfinding algorithm posed above!

5 Future Work

Genus is a commonly known and widely applied shape feature in 3D computer graphics, but there's a whole mathematical menu of topological invariants to explore along with their preservation techniques! Throughout our research, we also considered:

- How can certain regularization techniques (e.g. Lipschitz, eikonel, Laplacian) serve to "wrangle" topological features?
- To what extent can alternative approaches like diffeomorphic flows and persistent homology provide topological guarantees?

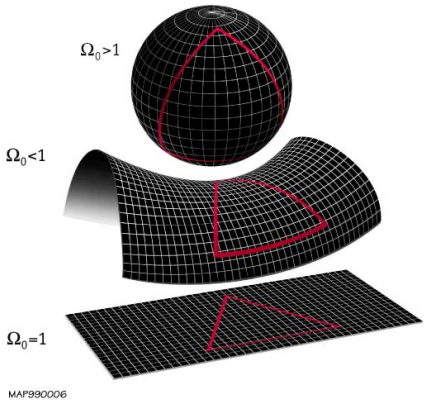


Figure 9: Geodesic triangles on manifolds of postive, negative, and zero curvature (respectfully, from top to bottom)

- Would topological accuracy benefit from a non-differentiable algorithm (e.g. RRT) that can directly encode genus as a discrete property? How would we go about certifying such an algorithm?
- How can homotopy continuation be used for latent space pathfinding? How will this method complement continuous deformations of homotopic shapes?
- What are some use cases to preserve topology, and what choice of topological invariant should we pair with those cases?

We invite all readers and researchers to consider the above questions as well.

6 References and Codebase

GitHub: https://github.com/paulkry/topology-control