Stochastic Preconditioning for Neural Field Optimization: An Experimental Study

Michael Belyaev
Supervised by Amir Vaxman

Abstract

This report explores and evaluates the benefits of stochastic preconditioning (SP) as presented in "Stochastic Preconditioning for Neural Field Optimization" [1]. This report focuses on reconstructing a signed distance function (SDF) from surfaces represented as point clouds.

Background

Neural Fields and Coordinate-based Neural Networks

Coordinate-based neural networks have recently attracted significant attention due to their ability to achieve accurate approximations of fields, which are quantities defined for all spatial and/or temporal coordinates. A field, typically represented as $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x})$, is simply a function that maps a coordinate $\boldsymbol{x}$ to some quantity, typically a scalar, vector or tensor.

A typical field $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x})$ isn't defined by a simple analytical expression or formula, like the gravitational field generated by a solid ball. Instead, it may be described as a set of parameters, $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$ where the parameters are either hand-crafted, optimised or learned. Fields learned using coordinate-based neural networks are called NEURAL FIELDS.

Implicit Neural Representations

We say that a field, $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x})$, is defined implicitly if its values are not given directly, but are instead described by one or several equations. These equations relate the field's value at a given coordinate $\boldsymbol{x}$ to properties like its slope or curvature, as shown in the general form

$$\mathcal{C}(\boldsymbol{x}, \boldsymbol{\Phi}(\boldsymbol{x}), \nabla_{\boldsymbol{x}}\boldsymbol{\Phi}(\boldsymbol{x}), \nabla^2_{\boldsymbol{x}}\boldsymbol{\Phi}(\boldsymbol{x}), \ldots) = 0.$$

Following the definition used in [3], we say that a neural field $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{\theta})$ provides an IMPLICIT NEURAL REPRESENTATION if it is defined implicitly.

Representing Shapes with Distance Functions

Consider a surface $\mathcal{S}$ in three-dimensional space. Let $d_{\mathcal{S}}(\boldsymbol{x})$ represent the minimal distance from a point $\boldsymbol{x}$ to the surface. The equation $d_{\mathcal{S}}(\boldsymbol{x}) = 0$ then implicitly represents the surface $\mathcal{S}$. This is useful when $\mathcal{S}$ is given as a set of points from real-world measurements. In such cases, a properly trained neural field $\Phi(\boldsymbol{x}, \boldsymbol{\theta})$ can approximate the distance function $d_{\mathcal{S}}(\boldsymbol{x})$ and the zero-level set, where $\Phi(\boldsymbol{x}, \boldsymbol{\theta}) = 0$, approximates the surface $\mathcal{S}$.

Let's assume that the surface $\mathcal{S}$ is two-sided and $d_{\mathcal{S}}(\boldsymbol{x})$ is its signed distance function (SDF). The SDF not only measures distance but also indicates which side of the surface a point is on (e.g., inside or outside). While the SDF may have singularities where its gradient is not defined, at non-singular points it is a well-behaved function that satisfies a set of fundamental mathematical equations:

$$|\nabla_{\boldsymbol{x}} d_{\mathcal{S}}(\boldsymbol{x})| = 1 \text{ outside } \mathcal{S}, \quad d_{\mathcal{S}}(\boldsymbol{x}) = 0 \text{ on } \mathcal{S}, \quad \nabla_{\boldsymbol{x}} d_{\mathcal{S}}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) = 1 \text{ on } \mathcal{S}. \tag{1}$$

Here, $\boldsymbol{n}(\boldsymbol{x})$ is the unit normal vector of the surface $\mathcal{S}$. The first equation, $|\nabla_{\boldsymbol{x}} d_{\mathcal{S}}(\boldsymbol{x})| = 1$, is the eikonal equation. Unfortunately, these equations do not uniquely define the distance function, as shown in Fig. 1.

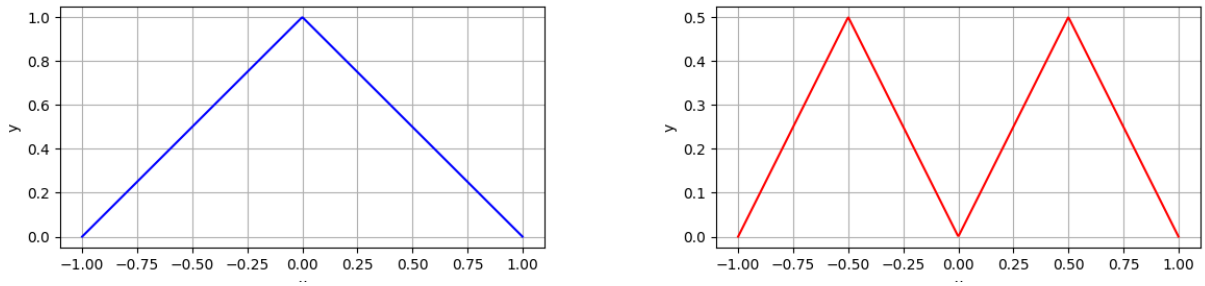


Figure 1: Left: the graph of the distance function for $\mathcal{S} = \{-1, +1\}$. Right: graph of another 1D function which satisfies (1) almost everywhere.

Stochastic Preconditioning

Stochastic preconditioning (SP) is a method [1] to improve neural field optimization by adding Gaussian noise to the input coordinates during training. The process works by querying the neural field, $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$, at a perturbed location, $\boldsymbol{x} + \delta$, instead of the original location, $\boldsymbol{x}$, where δ is a Gaussian-distributed offset with a standard deviation of α. This technique is extremely easy to apply to existing neural field-based methods.

Neural Surface Reconstruction from Oriented Point Clouds

Point clouds are a common way to represent 3D surface geometry, where a collection of data points each has a corresponding 3D coordinate $\boldsymbol{x}$ and a unit normal vector $\boldsymbol{n}(\boldsymbol{x})$ that indicates the surface's orientation at that point. The goal of this task is to train a neural field to approximate the signed distance function (SDF) of the surface. This way, the surface can be reconstructed as the zero-level set of the neural field, where $\Phi(\boldsymbol{x}, \boldsymbol{\theta}) = 0$.

The loss function used for these experiments follows the recommendation of [3], which are also followed by the stochastic preconditioning paper. The four main components are:

$$\mathcal{L}_{\text{surface}} = \int_{\Omega_0} |\Phi(\boldsymbol{x})| \, d\boldsymbol{x} \approx \frac{1}{n} \sum_{i=1}^{n} |\Phi(\boldsymbol{x}_i, \boldsymbol{\theta})|, \tag{2}$$

$$\mathcal{L}_{\text{normal}} = \int_{\Omega_0} \left[1 - \nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})\right] \, d\boldsymbol{x} \approx \frac{1}{n} \sum_{i=1}^{n} \left[1 - \nabla_{\boldsymbol{x}_i} \Phi(\boldsymbol{x}_i, \boldsymbol{\theta}) \cdot \boldsymbol{n}(\boldsymbol{x}_i)\right], \tag{3}$$

$$\mathcal{L}_{\text{eik}} = \int_{\Omega} \left| |\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}, \boldsymbol{\theta})| - 1 \right| \, d\boldsymbol{x}, \tag{4}$$

$$\mathcal{L}_{\text{offsurface}} = \int_{\Omega \setminus \Omega_0} e^{-\gamma |\Phi(\boldsymbol{x}, \boldsymbol{\theta})|} \, d\boldsymbol{x}. \tag{5}$$

Here, Ω_0 represents the surface $\mathcal{S}$ to be reconstructed, and Ω is the 3D domain containing it. The total loss function is then a weighted sum of these terms:

$$\mathcal{L}_{\text{total}} = \lambda_0 \cdot \mathcal{L}_{\text{surface}} + \lambda_1 \cdot \mathcal{L}_{\text{normal}} + \lambda_2 \cdot \mathcal{L}_{\text{eik}} + \lambda_3 \cdot \mathcal{L}_{\text{offsurface}}. \tag{6}$$

Experiments

The experiments in this report focus on reconstructing signed distance functions (SDFs) from oriented point clouds. The goal is to evaluate the robustness and effectiveness of stochastic preconditioning.

Experimental Setup

To replicate these experiments, here is the exact setup and parameters I used. Most of the experiments were carried out using the official implementation of stochastic preconditioning, available on GitHub. I used the

provided **INGP Hashgrid MLP** [2] as the neural field representation. Unless otherwise noted, each model was trained for 500 iterations.

The loss function followed (6) with the following weights:

$$\lambda_0 = 3 \times 10^3 \quad \text{(surface)}, \quad \lambda_1 = 1 \times 10^2 \quad \text{(normal)}, \quad \lambda_2 = 5 \quad \text{(eikonal)}, \quad \lambda_3 = 1 \times 10^2 \quad \text{(off-surface)},$$

and the off-surface decay parameter was fixed to $\gamma = 100$. For stochastic preconditioning, the input noise was applied with α initialized to 0.02 times the bounding box diagonal. This value was exponentially decayed to zero over the first third of training iterations, as suggested in [1].

For comparison with SP, I also ran the official implementation of HotSpot [4], available on GitHub, using the authors' recommended settings.

The ground-truth and reconstruction model files generated for these experiments can be found in this Google Drive folder.

Metrics. Meshes were scaled to a unit bounding-box diagonal before evaluating Chamfer and Hausdorff distances.

System details. All experiments were run on an NVIDIA RTX A4500 GPU. I followed the installation instructions provided on GitHub, except that I used PyTorch with CUDA 12.8, installed via:

```
pip install torch torchvision --extra-index-url https://download.pytorch.org/whl/cu128
```

Standard Benchmark Models

Armadillo

On the armadillo model, stochastic preconditioning (SP) shows a clear improvement over the baseline. The baseline Hashgrid MLP reconstruction Fig. 2a exhibits artifacts on the arms and between the legs, while the SP version Fig. 2b removes these artifacts and produces smoother fine details such as ears, fingers, and toes. My replication of the paper's experiment calculating the mean Chamfer distance achieves comparable and slightly better results (5.01×10^{-4} vs. 7.10×10^{-4}).

(a) Hashgrid MLP (b) Hashgrid MLP + SP (c) Increased Normal Loss

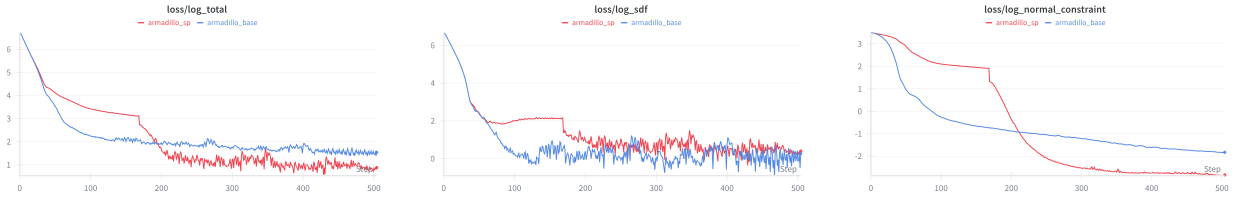
Figure 2: Armadillo

I also investigated whether adjusting the loss coefficients could replicate the effect of SP, as shown in Fig. 3c. Increasing the normal loss coefficient from 1×10^2 to 1×10^4, I was able to produce a reconstructed surface that was visually comparable to that obtained with SP. However, experiments with changing the other loss coefficients led to significantly worse reconstructions.

The loss curves Fig. 3 show an interesting behaviour. Initially, the baseline model converges more rapidly. However, once the stochastic preconditioning noise, α, decays to zero, the SP model quickly catches up and overtakes the baseline model.

Table 1: Armadillo Metrics

Method	Chamfer	Hausdorff	IoU
Hashgrid MLP	0.003977	0.291914	0.066706
Hashgrid MLP + SP	0.000501	0.038156	**0.283356**
Hashgrid MLP (Increased Normal Loss)	**0.000485**	**0.037472**	0.216273



(a) Total Loss (b) Surface Loss (c) Normal Loss

Figure 3: Logarithmic loss curves for the Armadillo model, comparing the baseline model (blue) with the model trained with stochastic preconditioning (red).

Dragon

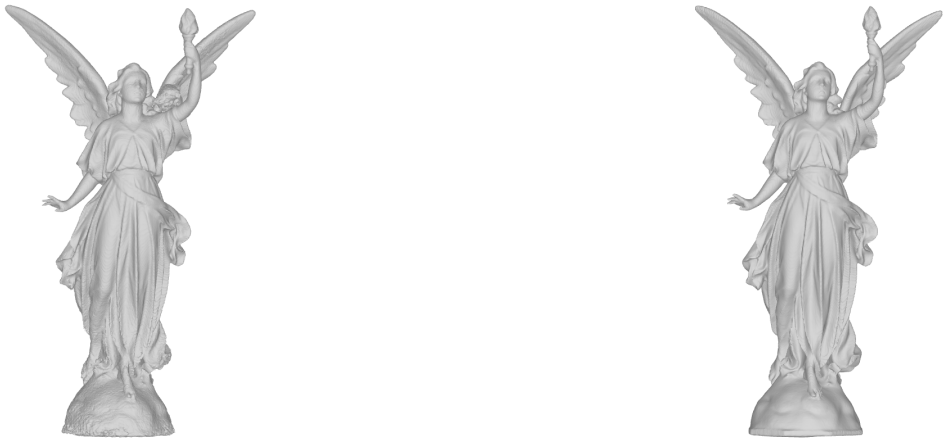
The application of SP improves the reconstruction quality on the Dragon model. In the baseline Hashgrid MLP (Fig. 4a), artifacts appear along the tail and the surface is relatively rough. With SP (Fig. 4b), these tail artifacts are removed and the reconstruction is smoother and less grainy.

(a) Hashgrid MLP (b) Hashgrid MLP + SP

Figure 4: Dragon

Lucy

The Lucy model also benefits from stochastic preconditioning. In the baseline reconstruction (Fig. 5a), there is a noticeable artifact on the shoulder, and the feet and base appear noisy. With SP (Fig. 5b), the shoulder artifact is removed and the surface at the feet and base is cleaner and less grainy.



(a) Hashgrid MLP

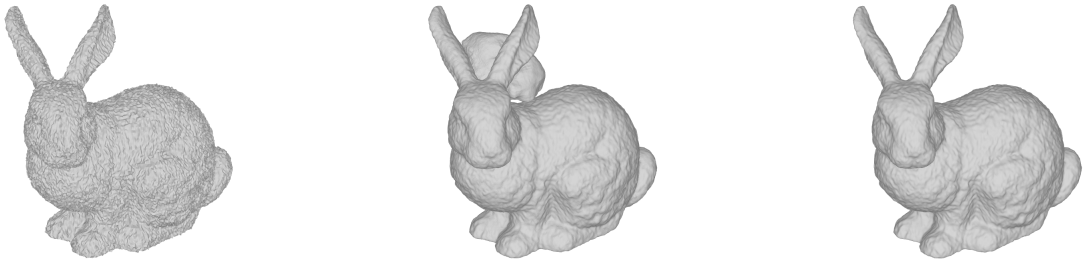
(b) Hashgrid MLP + SP

Figure 5: Lucy

Noisy Bunny

To test robustness, I added a small amount of Gaussian noise to the ground-truth point cloud. The specific noise parameters used are a standard deviation of 1% of the bounding-box diagonal.

Both the baseline model and the SP model were able to reconstruct the overall surface despite the noisy input. However, the baseline reconstruction shows an artifact between the ears, while the SP version removes it and produces a cleaner result (Fig. 6).



(a) Ground-Truth

(b) Hashgrid MLP

(c) Hashgrid MLP + SP

Figure 6: Noisy Bunny

Activation Functions

Robotdog

The official Hashgrid MLP implementation of [1] uses the Softplus activation function, defined as:

$$\text{Softplus}(x) = \ln(1 + e^x)\,. \tag{7}$$

I replaced Softplus with ReLU, defined as

$$\mathrm{ReLU}(x) = \max(0, x). \tag{8}$$

As shown in Fig. 7, the ReLU version produces a noticeably lower-quality reconstruction. The ears and hind legs have holes and are less well defined, and the overall surface detail is degraded compared to the Softplus model.

High-Genus Shapes

vbunny (high-genus model)

The stochastic preconditioning (SP) technique, while effective on many models, shows a clear limitation when applied to high-genus shapes. On the vbunny model, SP successfully removes exterior artifacts but fails to eliminate those on the interior of the surface.

I compared this result with another very recent method, HotSpot [4]. HotSpot introduces a new loss function called heat loss $\mathcal{L}_{\text{heat}}$, defined as

$$\mathcal{L}_{\text{heat}} = \frac{1}{2} \int_{\Omega} e^{-2\lambda|\Phi(\boldsymbol{x},\boldsymbol{\theta})|} \left(\|\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}, \boldsymbol{\theta})\|^2 + 1 \right) d\mathbf{x}. \tag{9}$$

The HotSpot paper argues that the eikonal loss alone is necessary but insufficient for recovering a true signed distance function. Their full loss combines the surface, eikonal, and heat terms:

$$\mathcal{L} = \lambda_0 \cdot \mathcal{L}_{\text{surface}} + \lambda_1 \cdot \mathcal{L}_{\text{eik}} + \lambda_2 \cdot \mathcal{L}_{\text{heat}}, \tag{10}$$

where λ_0, λ_1, and λ_2 are the weights for each loss term. Importantly, HotSpot does not require surface normals. As shown in Fig. 8 and Table 2, HotSpot reconstructs the vbunny exceptionally well, resolving both interior and exterior artifacts that SP could not. Although the HotSpot results shown here were trained for more than 500 iterations, the method is considerably faster per iteration, so the overall runtime remains competitive. By contrast, extending the Hashgrid MLP to 10,000 iterations did not lead to any meaningful improvement in reconstruction quality.

Approximation of the Eikonal Equation with the Heat Equation

The motivation for the heat loss can be understood by relating it to the eikonal equation. Here is a brief and informal derivation. We start with the screened Poisson equation,

$$\nabla^2 h - \lambda^2 h = 0, \tag{11}$$

and assume a solution of the form

$$h(x) = e^{-\lambda \Phi(\boldsymbol{x})}, \tag{12}$$

where $\Phi(\boldsymbol{x})$ is the implicit function we want to recover. Taking derivatives of (12) gives

$$\nabla h = -\lambda e^{-\lambda \Phi(\boldsymbol{x})} \nabla \Phi(\boldsymbol{x}) \quad \text{and} \quad \nabla^2 h = \lambda^2 e^{-\lambda \Phi(\boldsymbol{x})} \|\nabla \Phi(\boldsymbol{x})\|^2 - \lambda e^{-\lambda \Phi(\boldsymbol{x})} \nabla^2 \Phi(\boldsymbol{x}). \tag{13}$$

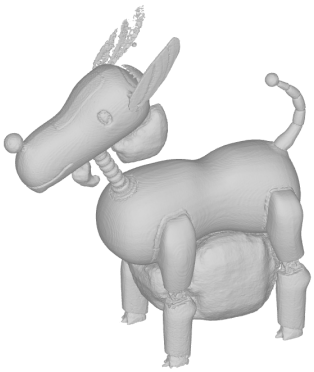
Plugging this into the screened Poisson equation (11) gives us

$$\|\nabla \Phi(\boldsymbol{x})\|^2 - 1 - \tfrac{1}{\lambda} \nabla^2 \Phi(\boldsymbol{x}) = 0. \tag{14}$$

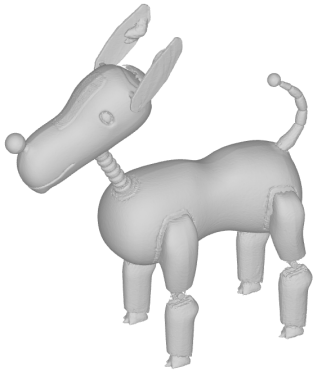
In the limit as $\lambda \to \infty$, the Laplacian term vanishes and we recover the eikonal equation

$$\|\nabla \Phi(\boldsymbol{x})\| = 1. \tag{15}$$

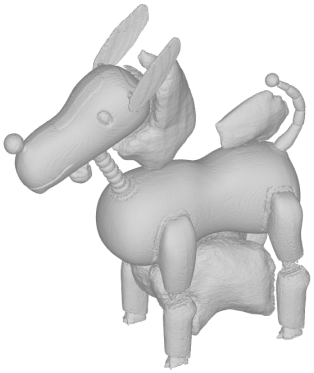
The heat loss is designed so that minimizing it corresponds to solving the screened Poisson equation (11). In fact, it can be derived directly as the energy functional whose minimizer gives the expression in (9).



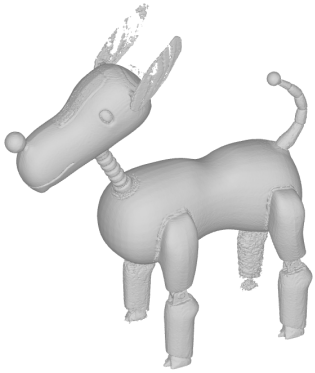
(a) Hashgrid MLP (Softplus)



(b) Hashgrid MLP + SP (Softplus)



(c) Hashgrid MLP (ReLU)



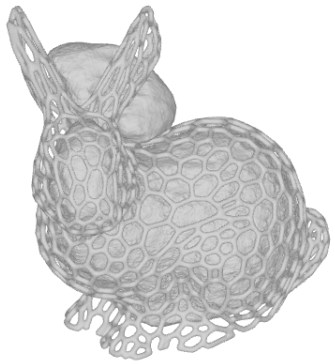
(d) Hashgrid MLP + SP (ReLU)

Figure 7: Robotdog

Removing the Eikonal Loss

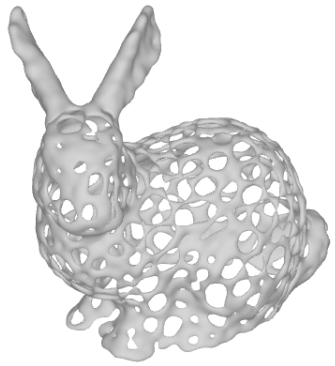
The HotSpot paper also emphasizes that, while the eikonal loss is insufficient on its own, it remains a necessary component. In the official implementation, the default weight for the heat loss is set to 3. To test its role, I ran several variants:

- **No Eikonal Loss(Fig. 9a):** only surface and heat losses.
- **High Heat Loss (Fig. 9b):** eikonal removed, heat weight increased tenfold to 30.
- **Very High Heat Loss (Fig. 9c):** eikonal removed, heat weight increased 100-fold to 300.
- **Normal Loss (Fig. 9d):** replacing the eikonal loss with the normal loss.

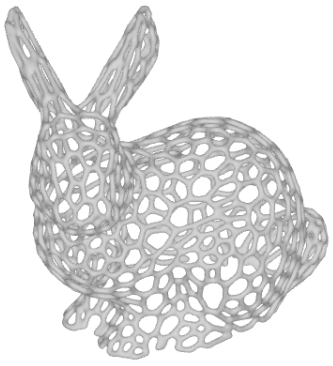


(a) Hashgrid MLP

(b) Hashgrid MLP + SP



(c) HotSpot (1k iter)



(d) HotSpot (10k iter)

Figure 8: vbunny

As shown in Fig. 9, removing the eikonal loss leads to poor reconstructions, even with larger heat weights. Replacing it with the normal loss produces results on par with the original HotSpot setup, but at the cost of requiring surface normals.

Stochastic Preconditioning Hyperparameters

As mentioned earlier, the stochastic preconditioning (SP) paper [1] recommends initializing the parameter α to 1–2% of the bounding-box diagonal and decaying it exponentially to zero over the first third of training. To test how sensitive the method is to this choice, I experimented with alternative noise schedules.

Table 2: vbunny

Method	Chamfer	Hausdorff	IoU
Hashgrid MLP	0.000422	0.084007	0.143944
Hashgrid MLP + SP	0.000572	0.079678	0.247838
HotSpot (1k iter)	0.000854	0.082362	0.280314
HotSpot (10k iter)	**0.000102**	**0.027738**	**0.465142**

(a) No Eikonal Loss

(b) High Heat Loss

(c) Very High Heat Loss

(d) Normal Loss

Figure 9: vbunny with no eikonal loss

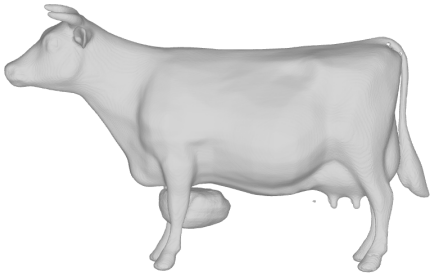
Cow

On the Cow model, the specific way in which the SP noise is decayed, and the number of iterations over which it is applied, appears to have little effect on the final result. The quality of the reconstructed surface

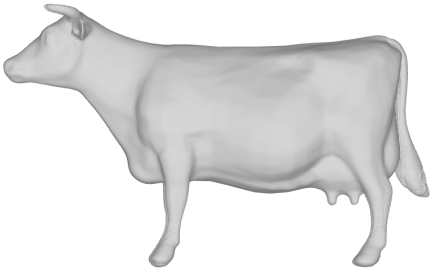
remains similar across different schedules. To demonstrate this, I compared three versions:

- **Constant (Fig. 10b):** fixed noise magnitude for 167 of the 500 iterations, then dropped to zero.
- **Long Decay (Fig. 10c):** exponential decay over the first 400 iterations.
- **Short Decay (Fig. 10d):** exponential decay over the first 100 iterations.

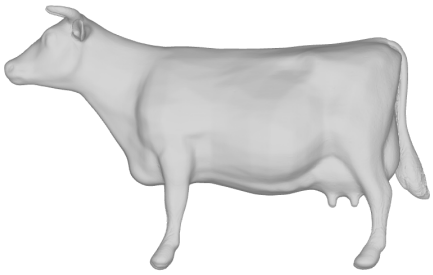
As shown in Fig. 10, all three schedules successfully removed the artifacts between the legs of the cow. The only difference was a slightly degraded tail in the model trained with the shortest decay schedule.



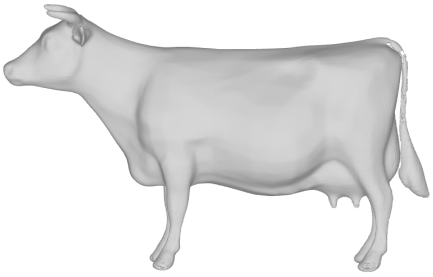
(a) Hashgrid MLP



(b) Hashgrid MLP + SP (Constant)



(c) Hashgrid MLP + SP (Long Decay)



(d) Hashgrid MLP + SP (Short Decay)

Figure 10: Cow

Sensitivity to Initial Noise

Stochastic preconditioning seems to be sensitive to the initial magnitude of noise. I compared the recommended setting of $\alpha = 0.02$ (2% of the bounding-box diagonal) with a doubled value of $\alpha = 0.04$. As shown

in Fig. 11, increasing the initial noise amount led to a slightly worse reconstruction. The surface contained significantly more artifacts inside the model.

(a) Initial $\alpha = 0.02$ (b) Initial $\alpha = 0.04$

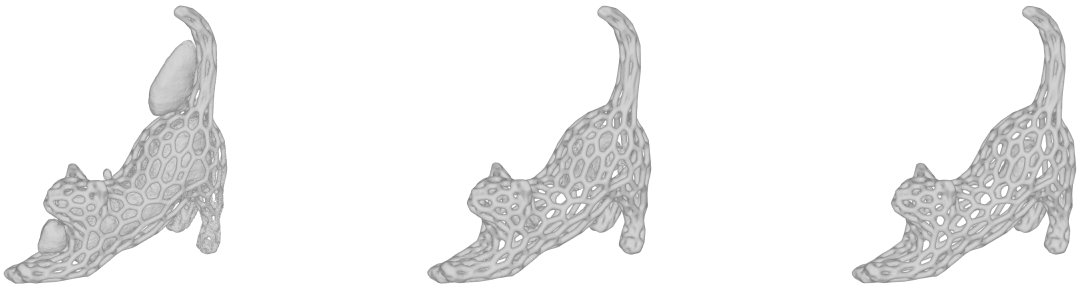
Figure 11: Catstretch

Sinusoidal Noise Schedule

I also experimented with an alternative schedule where the SP noise was decayed sinusoidally rather than exponentially. The noise was set to oscillate and gradually decay to zero according to the function:

$$\alpha_{\text{noise}}(t) = \alpha_{\text{final}} + (\alpha_{\text{init}} - \alpha_{\text{final}}) \cdot (1 - t/T) \cdot \frac{1 + \cos(2\pi N_{\text{osc}}\, t/T)}{2}.$$

For these experiments, the total decay time was $T = 250$ iterations, with the alpha parameter starting at $\alpha_{\text{init}} = 0.4$ and decaying to $\alpha_{\text{final}} = 0$. The noise completed $N_{\text{osc}} = 5$ full oscillations. My choice of parameters was arbitrary and not tuned.



(a) Hashgrid MLP (b) SP (Exponential Decay) (c) SP (Sinusoidal Decay)

Figure 12: Catstretch with varying decay schedules

As shown in Fig. 12, the baseline model produces artifacts both outside the object and across the cat's surface. The exponential SP schedule removes most exterior artifacts, but leaves floating artifacts in the

torso and back regions. In contrast, the sinusoidal schedule eliminates these torso artifacts, further reduces those on the back, and achieves lower total and surface loss.

Table 3: Catstretch Losses

Model	**Total Loss**	**Surface Loss**	**Normal Loss**
Hashgrid MLP	4.63	2.41	**0.33**
Hashgrid MLP + SP (Exp)	3.53	1.53	0.49
Hashgrid MLP + SP (Sine)	**2.61**	**0.83**	0.67

Conclusion

This report sets out to evaluate stochastic preconditioning (SP) for neural field optimization in the context of signed distance function (SDF) reconstruction from point clouds. The experiments show that SP consistently improves reconstruction quality across a range of models and, even when the benefit is limited, it does not degrade results compared to the baseline. In my opinion, the choice of noise parameters does not seem to have a significant impact on the outcome. However, the results on high-genus shapes revealed that changes to the loss formulation, such as the heat loss in HotSpot [4], yield far more significant improvements than SP. A key advantage of SP, however, is its practicality: it is incredibly easy to integrate it into any neural field-based method.

References

[1] Selena Ling, Merlin Nimier-David, Alec Jacobson, and Nicholas Sharp. Stochastic preconditioning for neural field optimization. *ACM Transactions on Graphics*, 44(4):84:1–84:10, 2025.

[2] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[3] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[4] Zimo Wang, Cheng Wang, Taiki Yoshino, Sirui Tao, Ziyang Fu, and Tzu-Mao Li. HotSpot: Signed distance function optimization with an asymptotically sufficient condition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2025)*, pages 1276–1286, 2025.