Neural Signed Distance Functions for 3D Shape Representation

Honglin Chen
Department of Computer Science
Columbia University
New York, NY 10025
hc3284@columbia.edu

Duowen Chen
Department of Computer Science
Columbia University
New York, NY 10025
dc3505@columbia.edu

Abstract

Neural signed distance functions (SDFs) have emerged as a powerful representation for 3D shapes. A neural SDF encodes a shape as the signed distance function to its surface, providing a continuous, compact and differentiable representation for the geometry. In this paper, we explore and implement three different neural SDF techniques: *DeepSDF* [4], *SIREN* [7] and *NGLoD* [8]. We further provide a detailed analysis on their strength and weakness and compare their performance on surface reconstruction task. [1]

1 Introduction

Effective 3D shape representations have long been sought by the 3D vision and graphics community to represent high-quality shapes with complex details and various topology for learning-based approaches. In recent years, neural signed distance functions (or neural implicit functions) have emerged as a powerful tool to represent 3D shapes with infinite resolution and arbitrary topology. It has served as an effective parameterization for 3D shapes [8, 11], view-dependent appearance [4, 6] and human bodies and faces [1]. For 3D tasks, a multi-layer perceptron (MLP) is commonly adopted to encode the shortest signed distance d from a point $\mathbf{x}$ to a surface $\mathcal{S}$:

$$f_\theta(\mathbf{x}; z) = d, \mathbf{x} \in \mathbb{R}^3, \tag{1}$$

where θ are the network weights and z is an input latent vector encoding a particular shape. In contrast to the *de facto* standard explicit polygonal mesh discretization, the surface $\mathcal{S}$ is implicitly encoded as the zero level-set of the function f:

$$\mathcal{S} = \left\{ \mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}; z) = 0 \right\}. \tag{2}$$

Our goal is to explore and analyze different neural SDF techniques for efficient 3D shape representation. We implement three neural SDF techniques: *DeepSDF* [4], *SIREN* [7] and *NGLoD* [8]. We further compare their strength and weakness by applying these techniques to geometry reconstruction task.

2 Related Works

2.1 Neural Implicit Representation

Neural implicit representation for reconstructing continuous 3D shape as a level-set, where a MLP is used to map a 3D coordinate to a SDF [4] or a occupancy field [3], has been shown effective. This

[1] We have switched our project topic given the feedback to our project proposal.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

representation is superior for it's agnostic to resolution, scales with shape complexity and admits effective learning of priors. DISN [10] improved the quality of reconstructing 3D scene by adding a local feature extraction module by projecting the 3D point to the image plane. Sitzmann [7] shows that with periodic activation functions, neural implicit network can represent complex scene with just 5 layers of fully connected layers. Recent work [8] reached real time rendering by attaching feature vector to octree nodes and capturing different level of details. We choose to leave out the comparison for a recent implicit SDF representation [11] that combines low frequency and high frequency representations to represent the SDF such that high frequency representation serves as a displacement field that is performed on the low frequency SDF representation to achieve an even better reconstruction for details. For a comprehensive survey of the recent advance in neural implicit representation, see [9].

3 Methods

3.1 DeepSDF

As the pioneer work that uses neural network to represent 3D shapes, DeepSDF choose to present the level set of a 3D shape as a SDF function. The training is performed on directly regress the continuous SDF from point samples. Therefore, ground truth SDF need to be provided. Then for each sample points, a simple L_1 loss is used that is presented as [4]:

$$\mathcal{L}\left(f_\theta(x), s\right) = \left|clamp\left(f_\theta(x), \delta\right) - clamp(s, \delta)\right| \tag{3}$$

Here, f_θ is the implicitly fitted neural SDF representation and s is the ground truth SDF. We minimize the summed loss over all points. And δ here is a metric used to allow fast ray-tracing.
The training is conducted using $ReLU$ activation with MLP having 8 hidden-layer dimension as 512 in the paper [4].

3.2 SIREN

In DeepSDF paper [4], the representation of 3D shapes are mainly simple ones and this technique weren't adopted to images, audio signals or other natural signals(PDEs). It suffers from the limitation of neural-networks that the learned function is not shift-invariant and piecewise linear(for ReLU) [7]. Inspired by Klcek et al. [2], they propose substituting a sin function as activation instead of using $ReLU$ which leverage the fact that sin is both shift-invariant and has well-defined second derivative. Different from DeepSDF paper, SIREN doesn't compute the ground-truth SDF information, instead they implicitly build the SDF function ($\Phi(\mathbf{x})$) by using the loss they proposed based on solving a Eikonal boundary value problem that constrains the norm of spatial gradient $|\nabla_x \Phi(\mathbf{x})|$ to 1 almost everywhere. And formally presented as [7]:

$$\mathcal{L}_{\text{sdf}} = \int_\Omega \left\| |\nabla_{\mathbf{x}} \Phi(\mathbf{x})| - 1 \right\| d\mathbf{x} + \int_{\Omega_0} \|\Phi(\mathbf{x})\| + \left(1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle\right) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi(\Phi(\mathbf{x})) d\mathbf{x} \tag{4}$$

Here, the first term is showing the constraint for Eikonal equation and the second term penalizes for the value of $\Phi(\mathbf{x})$ where $\mathbf{x}$ are surface points since SDF on surface is 0 and aligning the gradient of SDF(here will be surface points) with the surface normal which are properties of a SDF function.
The training is performed directly on 3D pointcloud data and implicitly build the SDF function without knowing the ground-truth SDF using the loss presented above. The reconstruction process is then same as DeepSDF that we use a marching cube algorithm performed on discretely sampled learned SDF to reconstruct the 3D mesh.

3.3 NGLoD

Level of detail is the techniques that has been widely used to improve performance of rendering with limited memory budget. In the NGLoD paper, they utilize this idea that the feature vectors of a geometry is represented as an Octree where the feature vectors are stored on the nodes of the tree. For each geometry, a sparse voxel octree is created spanning the bounding volume $[-1, 1]^3$. Each volume will hold feature vectors at its eight corners. The voxel will only exist if it contains a surface. This structure also implicitly represents the LOD structure as the levels of Octree. And to achieve a continuous representation of LOD and SDF, the feature vector for a point at an arbitrary position is

retrieve through linear interpolation at different levels of LODs and summed together. The resulting surface extracting network serves as a decoder network that takes the interpolated feature vector and outputs an SDF value. The decoder MLP will have a compact representation since as the point gets closer to the surface, the octree volume covers less area which results in a simpler local surface patch. Then during training, for an arbitrary point, the SDF value is computed through querying its feature vector from the octree's feature volumn by interpolation and passed to the decoder network. NGLoD also assumes we have the ground-truth SDF value, so the loss is then simply [8]:

$$J(\theta, \mathcal{Z}) = \mathbb{E}_{\mathbf{x},d} \sum_{L=1}^{L_{\max}} \left\| f_{\theta_L}([\mathbf{x}, \mathbf{z}(\mathbf{x}; L, \mathcal{Z})]) - d \right\|^2 \tag{5}$$

Where L is the level of detail, f_θ is the neural representation of SDF, d is the ground truth SDF. $\mathbf{x}$ and $\mathbf{z}$ are the 3D point location and its feature vector.

3.4 Comparison

We provide a detailed comparison about the strength and weakness of the three approaches in table 1. More comparisons can be found in the result and conclusion section.

Table 1: Comparison

	Input	Activation	Network	Memory Cost	Accuracy	Implementation
DeepSDF [4]	Point Cloud	ReLU	MLP	High	Low	Easy
SIREN [7]	Point Cloud	Sine	MLP	Medium	Medium	Easy
NGLoD [8]	Mesh	ReLU	Octree MLP	Low	High	Hard

4 Results

We choose to perform the reconstruction task on all of the three papers and merge the three implementations to the same framework. The tasks are performed by first overfitting each paper's network to a specific shape and then using marching cube to extract the 3D mesh reconstructed from the three learned neural representation of SDF. All shape are being normalized to the unit cube and therefore the reconstructed mesh and computed Chamfer Distance are comparable among the three methods.

In the table below, we show the L1 Chamfer Distance between the reconstructed mesh and the ground truth mesh.

Table 2: **Results: Chamfer L1 Distance**

	DeepSDF	SIREN	NGLoD / LoD2	NGLoD / LoD4	NGLoD / LoD6
Squirrel	0.0486	0.0320	0.0291	0.0273	**0.0271**
Spot	0.1078	0.1648	0.0317	0.0306	**0.0304**
QueenAnneChair	0.5278	0.1751	0.0860	**0.0763**	0.0766
Petmonster	0.1578	0.0786	0.0354	0.0305	**0.0297**
Roy	0.1506	0.5944	0.1048	0.0951	**0.0936**
SapphosHead	OutOfMemory	Didn't Converge	0.0115	0.0080	**0.0077**
CatWithHat	0.1458	0.3262	**0.1181**	0.3725	0.1917
Yoda	0.2869	0.1972	0.0503	**0.0457**	0.0609

We see NGLoD out performs in all cases. The reason for some of the lower number level of detail out performs the larger ones is due our insufficient sampling (limited by our computational resources and time) that we will show examples below. This cause some artifacts outside the surface where the network can't predict SDF value correctly. The NGLoD [8] also used IoU to evaluate performance but for simplicity we omit that result here. From figures below, we also show that because of the sampling technique used in NGLoD, it usually out performs when meshes don't contain hollow parts 1 and artifacts will show up due to low occupancy rate in the unit cube where the mesh locates 7.

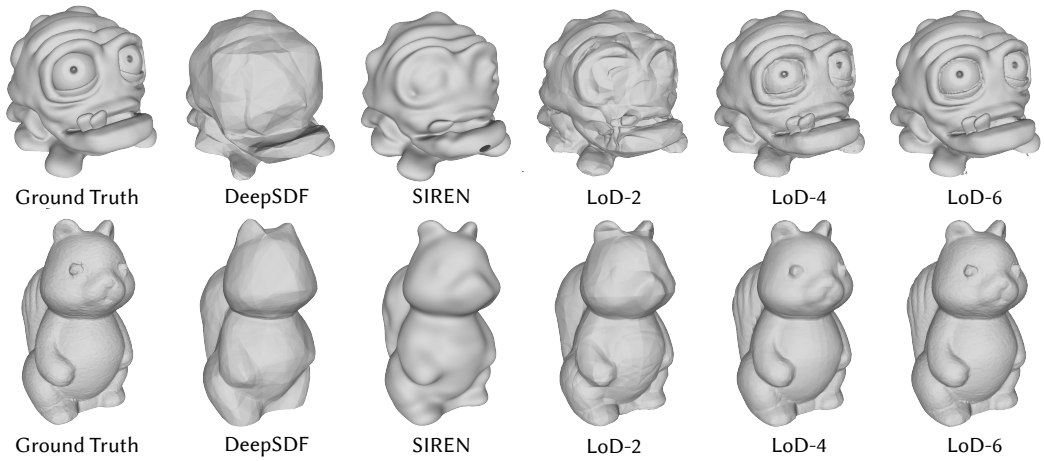

Figure 1: NGLoD with 4 and 6 levels of detail outperforms DeepSDF and SIREN in reconstructing the geometry details. SIREN usually produces over-smooth shapes while DeepSDF produces more piece-wise linear shapes due to their activation functions.

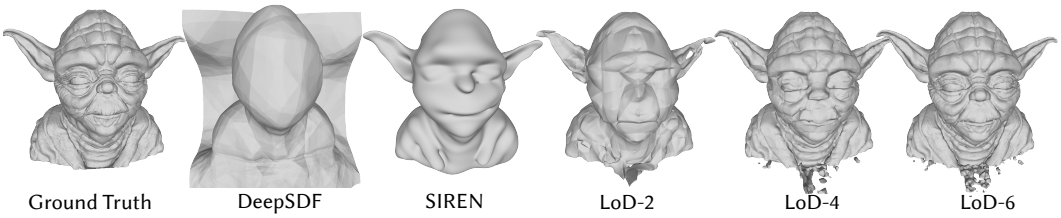

Figure 2: Despite some false positive SDF values outside of the surface, NGLoD is able to better reconstruct the high-resolution details in the Yoda model compared to DeepSDF and SIREN.

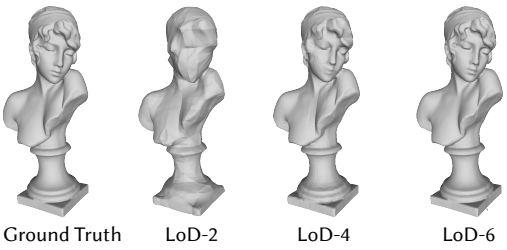

Figure 3: Accuracy improves as LoD increases.

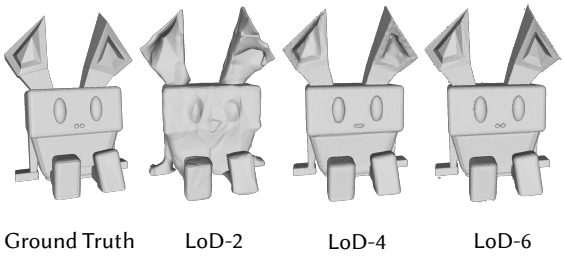

Figure 4: Adding more LoDs can improve the smoothness of the shape.

We trained our models on a Linux server with an Intel Dual 14 Core 2.2Ghz processor, 394GB of RAM and 4 GeForce RTX 3090 GPU and a machine with GeForce RTX 2080 GPU with 47G RAM. We randomly sample several shapes from the Thingi10K dataset [12] and use them as the training data.

We use a 5-layer MLP with ReLU activation function for the DeepSDF experiment and a 3-layer MLP with sine activation function for the SIREN experiment. Here the hidden dimensions are both 256 and we omit the latent vector for simplicity.

In the NGLoD experiment, we perform experiment using different numbers of level of details (LoD = 2, 4, 6) and show that the accuracy improves as the LoD increases. All the MLP in the NGLoD experiment have only a single hidden layer with dimension h = 128 and a ReLU activation in the intermediate layer, thereby being significantly smaller and faster to run than the other two papers.

In the SIREN experiment, we initialize the distribution of activations and their frequencies ($\omega_0 = 30.0$)

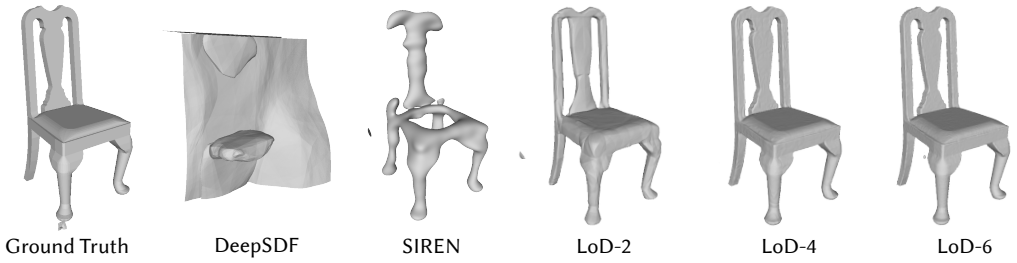

Figure 5: NGLoD is able to faithfully reconstruct the shapes where both DeepSDF and SIREN struggle to converge.

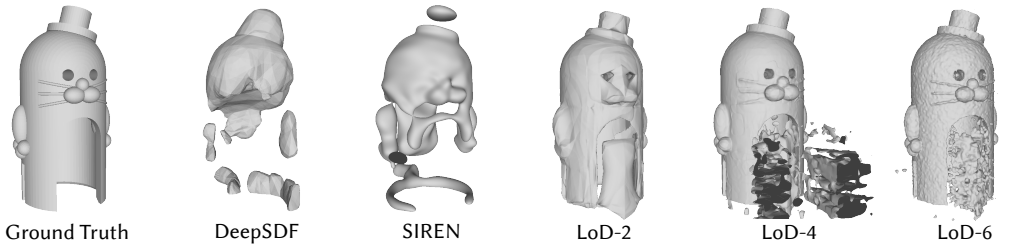

Figure 6: Sometimes NGLoD will predict wrong positive SDF outside of surface. We believe that it may be due to insufficient random samples outside of the surface in our implementation.

according to the discussion in Sec. 3.2 of the SIREN paper [7]. We set the initial learning rate to be 5×10^{-4} for the SIREN experiment and use 1×10^{-3} for all the other experiments.

We use the point cloud dataset with SDF loss (Eq. 4)in the DeepSDF and SIREN experiment, and mesh dataset with L2 loss (Eq. 5) in the NGLoD experiment. To generate the point cloud dataset, we simply convert the mesh vertices to point cloud. We use the *torchgp* library in NGLoD implementation to perform point sampling on mesh data and *mesh_to_sdf* library to calculate the ground truth SDF. We note that although it is possible to extend NGLoD [8] to accept point cloud input, it requires custom backward gradient implementation for the spatial gradient $\nabla_{\mathbf{x}}\Phi(\mathbf{x})$ in the octree structure. The reason is that we need to use SDF loss in SIREN [7] for the point cloud input. However, the derivative for *grid_sampler_3d_backward*, which is required for computing $\nabla_{\mathbf{x}}\Phi(\mathbf{x})$ in the octree structree, has not been implemented in the current pytorch version. For simplicity, we use the mesh as the input for NGLoD, which is the same as in the original NGLoD paper.

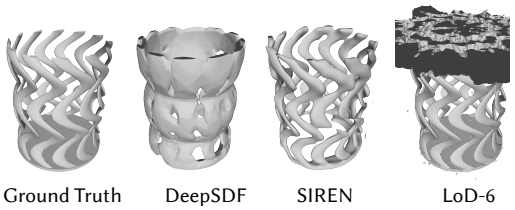

Figure 7: SIREN achieves better results in periodic complex shapes thanks to its periodic activation function.

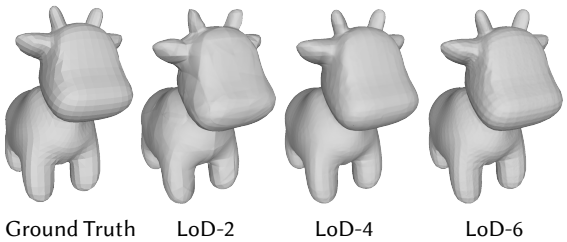

Figure 8: For NGLoD, user can take the tradeoff between the memory cost and reconstruction accuracy into account when choosing the LoD in use for the shape representation.

5 Conclusion & Limitations

From experiments, we see SIREN and DeepSDF are considerably hard to train, which needs large number of epochs and carefully tuned parameters. For NGLoD, it's time consuming for its

resampling of points from mesh that computes the ground truth SDF. However, we also acknowledge that we are not using the CUDA implementation provided by NGLoD paper but using a customized SDF extractor which may hinder performance.
In conclusion, this survey paper compares the three methods DeepSDF, SIREN and NGLoD on mesh reconstruction tasks under the same framework with same dataset, critics and computational resources. We show that DeepSDF are limited by its memory usage and it's hard to learn high frequency detail. We show that SIREN are able to learn high frequency detail but gets hard to train and needs proper weight initialization technique due to the sin activation function. We show NGLoD should be considered the best among the three and can recover high frequency details, but suffers from insufficient point sampling that causes artifacts which should be able to resolve through the resampling technique during training which then get limited by its time consuming nature for building SDF. Recent works are still pushing the boundary for this task and we hope this paper can provide a framework and tool for comparing with future works on this task.

References

[1] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa: Neural articulated shape approximation, 2020.

[2] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. *Lecture Notes in Computer Science*, page 496–510, 2019.

[3] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space, 2019.

[4] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019.

[5] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[6] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[7] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

[8] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. 2021.

[9] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond, 2021.

[10] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction, 2019.

[11] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields, 2021.

[12] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.