NeuralTPS: Learning Signed Distance Functions Without Priors From Single Sparse Point Clouds

Chao Chen, Yu-Shen Liu, *Member, IEEE*, and Zhizhong Han

***Abstract*—Surface reconstruction for point clouds is one of the important tasks in 3D computer vision. The latest methods rely on generalizing the priors learned from large scale supervision. However, the learned priors usually do not generalize well to various geometric variations that are unseen during training, especially for extremely sparse point clouds. To resolve this issue, we present a neural network to directly infer SDFs from single sparse point clouds without using signed distance supervision, learned priors or even normals. Our insight here is to learn surface parameterization and SDFs inference in an end-to-end manner. To make up the sparsity, we leverage parameterized surfaces as a coarse surface sampler to provide many coarse surface estimations in training iterations, according to which we mine supervision for our thin plate splines (TPS) based network to infer smooth SDFs in a statistical way. Our method significantly improves the generalization ability and accuracy on unseen point clouds. Our experimental results show our advantages over the state-of-the-art methods in surface reconstruction for sparse point clouds under synthetic datasets and real scans.**

***Index Terms*—Scene reconstruction, signed distance functions, sparse point clouds, surface reconstruction.**

I. Introduction

SIGNED distance functions (SDFs) have been a popular 3D representation that shows impressive performance in various tasks [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]. An SDF describes a signed distance field as a mapping from a coordinate to a signed distance, and represents a surface as a level set of the field. We can learn SDFs from signed distance supervision using coordinate-based neural networks. However, obtaining the signed distance supervision requires continuous surfaces such as water-tight manifolds, hence it is still challenging to infer signed distance supervision from raw point clouds due to the discrete character of points.

Received 3 September 2023; revised 3 September 2024; accepted 2 October 2024. Date of publication 7 October 2024; date of current version 4 December 2024. This work was supported in part by National Key R&D Program of China under Grant 2022YFC3800600, and in part by the National Natural Science Foundation of China under Grant 62272263 and Grant 62072268. Recommended for acceptance by P. Mordohai. *(Corresponding author: Yu-Shen Liu.)*

Chao Chen and Yu-Shen Liu are with the School of Software, Tsinghua University, Beijing 100190, China (e-mail: chenchao19@tsinghua.org.cn; liuyushen@tsinghua.edu.cn).

Zhizhong Han is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: h312h@wayne.edu).

Digital Object Identifier 10.1109/TPAMI.2024.3476349

Current methods [17], [18], [19], [20], [21], [22], [23], [34], [35], [36], [37], [38], [39], [40], [41] mainly leverage priors to infer SDFs for point clouds. They learn priors from well established signed distance supervision around point clouds during training, and then generalize the learned priors to infer SDFs for unseen point clouds during testing. Although local priors learned at a part level [8], [10], [42], [43], [44], [45] improve the generalization of global priors learned at a shape level [17], [18], [19], [20], [21], [22], [23], [46], the geometric variations that local priors can cover are still limited. Hence, some methods [5], [7], [9], [12], [13], [14], [15], [47] try to directly infer SDFs from single point clouds using various strategies [5], [7], [9], [12], [13], [16]. However, these methods require dense point clouds to assure the inference performance, which drastically limits their performance with sparse point clouds in real scans. Therefore, how to achieve better generalization of inferring SDFs from single sparse point clouds is still a challenge.

To overcome this challenge, we introduce a neural network to infer SDFs from single sparse point clouds. Our novelty lies in the way of inferring SDFs without signed distance supervision, learned priors or even normals, which significantly improves the generalization ability and accuracy on unseen point clouds. We achieve this by learning surface parameterization and SDF inference in an end-to-end manner using a neural network that overfits a single sparse point cloud. To make up the sparsity, the end-to-end learning turns parameterized surfaces into a coarse surface sampler which produces many coarse surface estimations on the fly to statistically infer the SDF. To target extremely sparse point clouds, we parameterize the surface of a point cloud as a single patch on a 2D plane, where 2D samples can be mapped to 3D points that lead to a coarse surface estimation. We further leverage the estimated coarse surface as a reference to infer the SDF based on thin plate splines (TPS) [48] in the feature space, which produces smooth signed distance fields. Our method can statistically infer the SDFs from the permutation of coarse surfaces in different iterations, which reduces the effect of inaccuracy brought by each single coarse surface.

We initially reported our results in our paper published at CVPR 2023 [49], and presented more details, results and analysis in this extension. Our method outperforms the latest methods under the widely used benchmarks. Our contributions are listed below.

- We introduce a neural network to infer SDFs from single sparse point clouds without using signed distance supervision, learned priors or even normals. We propose the

Structure-Aware Chamfer Distance for sparse point clouds with large missing parts or structures.
- We justify the feasibility of learning surface parameterization and inferring SDFs from sparse point clouds in an end-to-end manner. We provide a novel perspective to use surface parameterization to mine supervision.
- Our method outperforms the state-of-the-art methods in surface reconstruction for sparse point clouds under the widely used benchmarks.

II. Related Works

Neural implicit representations have achieved promising performance in various tasks [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64]. We can learn neural implicit representations from different supervision including 3D supervision [35], [36], [39], [65], [66], multi-view [1], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], and point clouds [17], [18], [42], [83]. Here, we mainly focus on reviewing methods related to point clouds below.

A. Data-Driven Based Methods

With 3D supervision, most methods adopted data-driven strategy to learn priors, and generalized the learned priors to infer implicit representations for unseen point clouds. Some methods learned global priors [17], [18], [19], [20], [21], [22], [23], [84] at a shape level. To improve the generalization of learned priors, some methods learned local priors [8], [10], [42], [43], [44], [45], [85], [86] at a part or patch level. With the learned priors, we can infer implicit representations for unseen point clouds, and then leverage the marching cubes algorithm [87] to reconstruct surfaces.

These methods rely on a large scale dataset to learn priors while they may not generalize well to unseen point clouds that have large geometric variations from the samples in the large scale dataset.

B. Overfitting Based Methods

For better generalization, some methods focus on learning implicit functions by overfitting neural networks on single point clouds. These methods introduce novel constraints [5], [7], [12], [13], [14], [15], ways of leveraging gradients [9], [47], differentiable poisson solver [11] or specially designed priors [45], [88] to learn signed [5], [7], [9], [12], [13], [16], [49], [89], [90] or unsigned distance functions [47], [91], [92], [93], [94], [95]. Although these methods have made great progress without learning priors, they require dense point clouds to infer the distance or occupancy fields around point clouds.

C. Learning From Sparse Point Clouds

With sparsity, the gap between points on surfaces makes it hard to accurately infer implicit functions. Some methods learned priors [8], [45], and conducted test-stage optimization on unseen sparse point clouds [45]. Without priors, NeedleDrop [96] was proposed to infer occupancy fields by learning whether a dropped needle goes across the surface or not. However, this self-supervision is not accurate at any point on a surface and heavily relies on the length of the needle. VIPSS [97] learns an implicit function from an unoriented point set based on Hermite interpolation, which is sensitive to parameter settings.

Our method falls in this category, but we aim to infer SDFs without learning priors or supervision. We achieve this by learning surface parameterization and SDFs inference in an end-to-end manner for capture a better sense on surfaces.

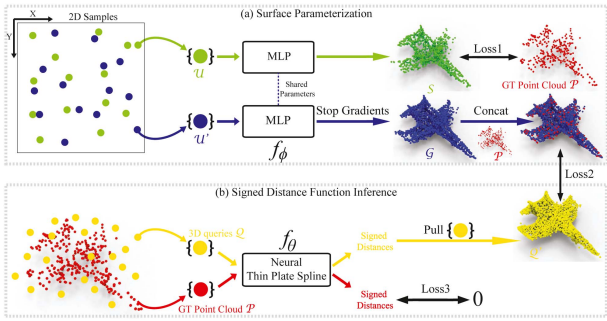

Fig. 1. The overview of our method. (a) The upper branch learns parameterization by covering 3D points $\mathcal{S}$ generated from 2D samples $\mathcal{U}$ onto the sparse input $\mathcal{P}$ like a sheet. The lower branch generates denser 3D points $\mathcal{G}$ from 2D samples $\mathcal{U}'$ to guide SDF learning. And the two branches share parameters. (b) We learn an SDF by sampling 3D queries $\mathcal{Q}$ and pulling them into $\mathcal{Q}'$ to cover the sparse input $\mathcal{P}$, and constrains signed distances at points in $\mathcal{P}$.

D. Neural Splines

Splines have been widely used in image manipulation [98] or generation [99]. NeuralSpline was proposed to fit point clouds with normals using implicit functions [100]. With normals, it simply infers occupancy of points on the normals, hence it focuses on fitting rather than inference. Instead, we target a more challenging scenario where we focus on inferring SDF without normals or learned priors.

III. Method

A. Overview

We aim to infer an SDF f_θ from a single sparse point cloud $\mathcal{P} = \{p_i | i \in [1, I]\}$, where the SDF is parameterized by a network with parameters θ. At any location q in 3D space, SDF f_θ predicts signed distance $d = f_\theta(q)$. Our method learns surface parameterization and SDF inference in an end-to-end manner, where we aim to use surface parameterization as a coarse surface estimation sampler to provide supervision for SDF inference, as illustrated in Fig. 1.

During surface parameterization in Fig. 1(a), we randomly sample two sets of 2D points $\mathcal{U}$ and $\mathcal{U}'$ in a unit square in each iteration. We map each 2D sample into a 3D point using a neural network f_ϕ. This mapping leads to two sets of 3D points $\mathcal{S} = \{s_j | j \in [1, J]\}$ and $\mathcal{G} = \{g_k | k \in [1, K]\}$, each of which forms a chart covering the whole shape. We use point cloud $\mathcal{P}$ to regulate

$\{s_j\}$, and use $\{g_k|k \in [1, K]\}$ to mine supervision to infer SDF f_θ in the following.

We leverage a thin plate splines (TPS) based network (NeuralTPS) to infer SDF f_θ in Fig. 1(b). Our network learns a feature space, where we leverage TPS interpolation to produce features of arbitrary queries $q \in \mathcal{Q}$ in 3D space based on the features of sparse point cloud $\mathcal{P}$. Our network predicts signed distances $f_\theta(q)$ at q from its interpolated feature. We infer SDF f_θ by minimizing the difference between the surface $\mathcal{Q}'$ produced with the current inferred signed distance field and the chart $\mathcal{G}$ from the parameterized surface. Moreover, we also regulate f_θ by constraining sparse point cloud $\mathcal{P}$ to locate on the zero level of the SDF f_θ.

To remedy the inaccuracy in $\mathcal{G}$, we regard $\mathcal{G}$ in each iteration as a sample of coarse surface estimation, and infer f_θ by minimizing the loss expectation in a statistical manner. Moreover, we introduce a confidence weight to consider the confidence of each point in $\mathcal{G}$.

In short, our method is formed by two modules. The first module is for surface parameterization, the second is for learning SDF. One branch in the first module generates denser 3D points $\mathcal{G}$ from 2D samples $\mathcal{U}'$ to guide SDF learning and the other branch learns parameterization by covering 3D points $\mathcal{S}$ generated from 2D samples $\mathcal{U}$ onto the sparse input $\mathcal{P}$ like a sheet. The second module learns an SDF by sampling 3D queries $\mathcal{Q}$ and pulling them into $\mathcal{Q}'$ to cover the sparse input $\mathcal{P}$, and constrains signed distances at points in $\mathcal{P}$.

B. Surface Parameterization

We learn to parameterize a surface represented by a sparse point cloud $\mathcal{P}$ on a 2D plane in Fig. 1(a), which is a 2D-to-3D mapping. We adopt this classic surface parameterization since it is easy to do sampling on a 2D parameterization space and transform 2D samples to points on a 3D surface. Specifically, we leverage an MLP f_ϕ with five layers to learn a mapping from a 2D sample to a 3D point. This mapping produces a 3D chart $\mathcal{S}$ using a set of randomly sampled 2D points $\mathcal{U}$, i.e.

$$\mathcal{S} = f_\phi(\mathcal{U}). \tag{1}$$

We regulate this mapping by covering $\mathcal{S}$ onto the ground truth points $\mathcal{P}$, which maximizes the overlapping between $\mathcal{S}$ and $\mathcal{P}$ using a Chamfer Distance (CD) loss,

$$L_{CD} = \frac{1}{J}\sum_{s\in\mathcal{S}} \min_{p\in\mathcal{P}} ||s-p||_2^2 + \frac{1}{I}\sum_{p\in\mathcal{P}} \min_{s\in\mathcal{S}} ||p-s||_2^2. \tag{2}$$

Our surface parameterization is similar to previous methods such as FoldingNet [101], AtlasNet [102], and PCN [103] that apply the folding operation, i.e., mapping a 2D sample to a 3D point via MLP. FoldingNet and PCN employ regularly sampled 2D points on a grid as the start points, while AtlasNet uses randomly sampled points. Our method also uses randomly sampled 2D points as input to increase the point density, but we only leverage one patch to cover the shape rather than multiple patches. so that we can better fill the gaps between sparse points using generated points.

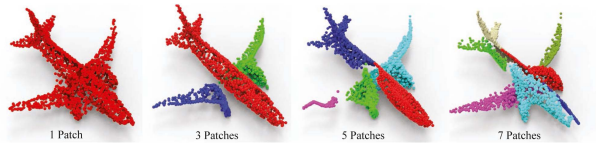

Fig. 2. The effect of patch numbers. More patches means more gaps between patches.

We visualize the effect of patch numbers in Fig. 2, where each subfigure shows a point cloud with the same number of points but different number of patches. The comparison indicates that more patches can not fill the gaps among points, while our single chart reveals a more compact surface.

With surface parameterization, we regard our MLP as a coarse surface sampler which predicts an additional coarse surface estimation $\mathcal{G}$ using another set of 2D samples $\mathcal{U}'$, i.e.

$$\mathcal{G} = f_\phi(\mathcal{U}'). \tag{3}$$

In each iteration during training, we leverage $\mathcal{G}$ to infer SDF f_θ in Fig. 1(b). We stop the gradients that can be back-propagated from the loss on $\mathcal{G}$, which avoids the impact of SDF inference on surface parameterization. This is also the reason why we design a two-branch structure for surface parameterization, which differs our method from AtlasNet a lot. Notice that the two branches are not separate since they share the same MLP. The branch below with blue arrows in Fig. 1 does not work since gradients are stopped during training.

C. Signed Distance Function Inference

We introduce NeuralTPS to infer SDF f_θ from sparse point cloud $\mathcal{P}$. We sample 3D queries q using a Gaussian function centered at each point in $\mathcal{P}$. The inferred SDF predicts signed distances $f_\theta(p_i)$ and $f_\theta(q)$ at each point $p_i \in \mathcal{P}$ and each sampled query q. Here, we impose two different constraints to signed distances $f_\theta(p_i)$ on the surface and signed distances $f_\theta(q)$ in 3D space.

For points p on surface of $\mathcal{P}$, we expect them on the zero level set of SDF f_θ, hence we leverage a MSE loss,

$$L_{Surf} = \sum_{p\in\mathcal{P}} (f_\theta(p))^2. \tag{4}$$

For points $q \in \mathcal{Q}$ in 3D space, we expect the signed distance field could provide the correct signed distances and gradients which can be used to pull q onto the nearest points on the coarse surface $\mathcal{G}$. Here, we use a pulling operation introduced in [9] to pull q to q',

$$q' = q - f_\theta(q)\nabla f_\theta(q)/||\nabla f_\theta(q)||_2. \tag{5}$$

Different from [9], we introduce a novel confidence-weighted loss to optimize the set $\mathcal{Q}' = \{q'\}$ to cover the coarse surface $\mathcal{G}$ with considering the confidence of each point in $\mathcal{G}$,

$$L_{Pull}(\mathcal{Q}', \mathcal{G}) = \sum_{q'\in\mathcal{Q}', g\in\mathcal{G}} w||q'-g||_2^2, \tag{6}$$

where g is the nearest point of q on $\mathcal{G}$. In practice, we find the nearest point from the union of $\mathcal{G}$ and sparse points $\mathcal{P}$. We use

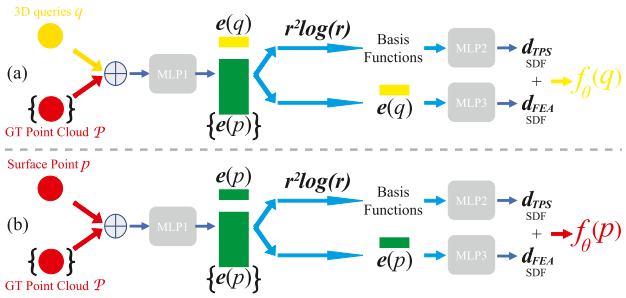

Fig. 3. The illustration of NeuralTPS with one query, such as (a) a surface point or (b) a sampled point.

w to model the confidence of each point $g \in \mathcal{G}$ to remedy the inaccuracy in $\mathcal{G}$. g has higher confidence if it is nearer to sparse point cloud $\mathcal{P}$ and vice versa. Hence, we formulate w as,

$$w = exp\left(-\delta * ||g - p||_2^2\right), \tag{7}$$

where p is the nearest point of g on the sparse point cloud $\mathcal{P}$ and the decay parameter δ is set to 50 in our experiments.

To further reduce the impact of inaccuracy in $\mathcal{G}$, we minimize $L_{Pull}(\mathcal{Q}', \mathcal{G})$ in a statistical manner over $\{\mathcal{G}\}$ obtained in different iterations rather than $\mathcal{G}$ in a single iteration. Hence, we aim to find an optimal $\mathcal{Q}'$ that has the smallest average deviation over $\{\mathcal{G}\}$. We reformulate $L_{Pull}(\mathcal{Q}', \mathcal{G})$ into $\mathbb{E}_{\{\mathcal{G}\}}\{L_{Pull}(\mathcal{Q}', \mathcal{G})\}$.

D. Loss Function

We optimize surface parameterization and SDF inference in an end-to-end manner by adjusting parameters θ and ϕ using the following objective function,

$$\min_{\theta,\phi}\left(L_{CD} + \alpha L_{Surf} + \beta \mathbb{E}_{\{\mathcal{G}\}}\{L_{Pull}(\mathcal{Q}', \mathcal{G})\}\right), \tag{8}$$

where α and β are balance weights, and we set $\alpha = 0.1$ and $\beta = 0.1$ in our experiments.

E. Neural Thin Plate Splines

We introduce NeuralTPS to infer SDF as a smooth function. Our key idea is to learn an optimal feature space that can be further mapped to signed distances, where we regress signed distances at queries using features of surface points by TPS interpolation.

We illustrate NeuralTPS in Fig. 3(a). We start from concatenating surface point $p_i \in \{\mathcal{P}\}$ with sampled queries $q \in \mathcal{Q}$ in each iteration. This aims to extract point features $\mathbf{e}(p_i)$ and $\mathbf{e}(q)$ using the same parameters for TPS interpolation. So, we leverage an MLP (denoted as MLP1) to learn features of points p_i and query q, i.e., $\mathbf{e}(p_i) = MLP(p_i)$ and $\mathbf{e}(q) = MLP(q)$. Then, we regard the features of surface points as control nodes to regress signed distances d_{TPS} at queries q using TPS interpolation below,

$$d_{TPS} = \sum_{i=1}^{I} c_i \psi\left(||\mathbf{e}(p_i) - \mathbf{e}(q)||_2^2\right), \tag{9}$$

and

$$\psi(r) = r^2 log(r), \tag{10}$$

where $\psi(r)$ is known as the thin plate radial basis function, and we will report results with other basis functions in our experiments. $\{c_i\}$ are weights for integrating basis functions, which are learnable parameters in another MLP (denoted as MLP2).

To complement the potential interpolation error in the linear summation, we predict a displacement d_{FEA} for signed distances at queries using point features $\mathbf{e}(q)$ through an MLP (denoted as MLP3). In summary, we formulate signed distances at queries q as,

$$f_\theta(q) = d_{TPS} + d_{FEA}. \tag{11}$$

We use the same way to predict signed distance $f_\theta(p)$ of surface points $p \in \{\mathcal{P}\}$, as illustrated in Fig. 3(b).

F. Structure-Aware Chamfer Distance

In this paper, we extend our original method to handle large missing parts or structures. Without learned priors, reconstruction from point clouds with missing parts remains a challenge for most previous overfitting-based surface reconstruction methods, such as NeuralPull [9], SAP [11], and NeuralSplines [100]. To address this issue, we additionally propose the Structure-Aware Chamfer Distance (SACD). The core concept behind SACD is to capture as much structural information as possible during surface parameterization. By considering the point density, we are able to better align the parameter space with the surface represented by the sparse points. Hence, we can use L_{SACD} to replace the Chamfer Distance L_{CD} used in (8).

Structure-Aware Constraint: We learn a mapping from a set of 2D samples $\mathcal{U}$ to a 3D chart $\mathcal{S}$ through surface parameterization, as shown in Fig. 1.

Similar to the original CD, our SACD constrains the surface parameterization through a bi-directional distance loss that includes the distances from the 3D chart $\mathcal{S}$ to the ground truth points $\mathcal{P}$, and vice versa. The difference lies in the design of capturing as much structural information as possible. To this end, it is not enough for each point in $\mathcal{S}$ to be only related to its nearest point in $\mathcal{P}$. Instead, we connected each point to its r nearest points, weighted according to their proximity,

$$L_{\mathcal{S}2\mathcal{P}} = \frac{1}{Jr}\sum_{s\in\mathcal{S}}\sum_{i=1}^{r} w_i^s * ||s - \mathcal{N}_i(s, \mathcal{P})||_2^2, \tag{12}$$

where $\mathcal{N}_i(s, \mathcal{P})$ denotes the ith nearest points of s in $\mathcal{P}$. r is a perceptual parameter which is set to 3. We use w_i^s to model the correlation between $\mathcal{N}_i(s, \mathcal{P})$ and s. If $\mathcal{N}_i(s, \mathcal{P})$ is nearer to s, it has a higher correlation, and vice versa. Thus, w_i^s is denoted as,

$$w_i^s = exp\left(-\epsilon * ||\mathcal{N}_i(s, \mathcal{P}) - s||_2^2\right), \tag{13}$$

where the decay parameter ϵ is set to 10in our experiments. Under $L_{\mathcal{S}2\mathcal{P}}$, s will be matched to a location near the point with the highest correlation. As it is not feasible to reconstruct the original structure directly from the sparse ground truth points $\mathcal{P}$, we approximate the original structure by enhancing the points in $\mathcal{P}$ with structural information through $L_{\mathcal{S}2\mathcal{P}}$.

Similarly, for any point $p \in \mathcal{P}$, we approximate its density using the sum of distances to its k-nearest neighbors, that is,

$$\rho_p = exp\left(-\eta * \sum_{i=1}^{k} ||p - \mathcal{N}_i(p, \mathcal{P})||_2^2\right), \tag{14}$$

where η is another decay parameter and k is another perceptual parameter. We set η to 10 and k to 3 in our experiments. A larger sum of these distances indicates a lower density at that point. We aim to pay more attention to these low-density points within $\mathcal{P}$. Hence, we formulate $L_{\mathcal{P}2\mathcal{S}}$ as,

$$L_{\mathcal{P}2\mathcal{S}} = \frac{1}{I}\sum_{p\in\mathcal{P}} exp(-\rho_p) * min_{s\in\mathcal{S}}||p - s||_2^2. \tag{15}$$

We adopt this approach because we observed that sparse point clouds, particularly those with large missing parts or structures, usually get ignored by the neural network in some area with extremely low density points, leading to degraded results. As previously mentioned, these points lose their structural information because they are too far away from their neighboring points. Therefore, we increase the weight of these points in the loss function to make the network focus more on the extremely sparse points. Ultimately, our structure-aware constraint is a summation of losses calculated in both directions,

$$L_{SACD} = L_{\mathcal{P}2\mathcal{S}} + L_{\mathcal{S}2\mathcal{P}}. \tag{16}$$

Training Strategy: We also modified the training strategy of NeuralTPS with L_{SACD}. Our goal is for the surface parameterization to map a set of 2D samples $\mathcal{U}$ to a 3D chart $\mathcal{S}$, capturing the coarse structure as quickly as possible. This early structuring provides a roughly correct basis for inferring the signed distance function. At the beginning stage of training, we use a higher value for r to capture more structural information for each point, despite the risk of accumulating redundant or incorrect data. As training progresses, we gradually reduce r to refine the learning process, allowing for convergence and optimization of results.

L2 Parameter Regularization: Additionally, we incorporate an L2 parameter regularization constraint on the network parameters ϕ and θ during training. This constraint is commonly employed to prevent overfitting, which in our case, helps avoid overfitting both the 3D chart $\mathcal{S}$ in the surface parameterization and the points pulled to the surface during signed distance function inference. This regularization promotes the development of smooth and complete surfaces, even in cases of extreme sparsity. We will validate this in our experiments. The constraint is denoted as,

$$L_{Reg} = \frac{1}{||\phi||}\sum_{\phi}\phi^2 + \frac{1}{||\theta||}\sum_{\theta}\theta^2. \tag{17}$$

In summary, to deal with sparse point clouds with large missing parts or structures, we optimize surface parameterization and SDF inference using the following objective function,

$$\min_{\theta,\phi}\left(L_{SACD} + \gamma L_{Surf} + \delta\mathbb{E}_{\{\mathcal{G}\}}\{L_{Pull}(\mathcal{Q}', \mathcal{G})\} + \lambda L_{Reg}\right), \tag{18}$$

where γ, δ and λ are balance weights, and we set $\gamma = 0.1$, $\delta = 0.1$ and $\lambda = 1e-4$ in our experiments.

G. Motivation of NeuralTPS

One of the challenges for SDF inference from sparse point clouds is to produce a smooth field. We adopt TPS [48] in the learned feature space, since TPS is a unique solution to scattered data interpolation with maximum smoothness evaluated by the second order partial derivatives [104]. The smoothness is a measurement of the aggregate curvature of f_θ over the region of the surface. Since the smoothness may filter out sharp edges, we conduct TPS interpolation in the learned feature space rather than 3D space. In the next subsection, we will specifically describe the principles of TPS in higher dimensional spaces, which is the theoretical basis that our NeuralTPS relies on.

H. The Principles of NeuralTPS

Let $\Omega \subset \mathbb{R}^d$, $d \subset \mathbb{N}$, be a closed and bounded domain. Given a set $\mathcal{M} = \{p_i\}_{i=0}^N$ of scattered points in Ω and function values at $\mathcal{G}$ as $\{z_i\}_{i=0}^N$, a TPS is a smooth function $u : \Omega \to \mathbb{R}$ which minimises the loss function,

$$J_\alpha(u) = J_1(u) + \alpha J_2(u), \tag{19}$$

where $J_1(u)$ is a fitting term which measures the distances between the corresponding point-sets $\{p_i\}$ and $\{z_i\}$, and $J_2(u)$ is a deformation term which measures the smooth variant. α is a tuning parameter to control the rigidity of the deformation. Small values of α will cause TPS to follow the data closely, but may be sensitive to errors in the data, while large values of α will give a smooth fit, but may not fully represent the data. The definitions of $J_1(u)$ and $J_2(u)$ are formulated as,

$$J_1(u) = \frac{1}{N}\sum_{i=1}^{N}(u(p_i) - z_i)^2, \tag{20}$$

and

$$J_2(u) = \int_\Omega L(u)dx, \tag{21}$$

where L is a differential operator and it supervises the smoothness of TPS. In 2D cases, L can be expressed as the square of the Hessian matrix, thus $J_2(u)$ is,

$$J_2(u) = \iint \left[\left(\frac{\partial^2 u(x)}{\partial x_1^2}\right)^2 + 2\left(\frac{\partial^2 u(x)}{\partial x_1 x_2}\right)^2 + \left(\frac{\partial^2 u(x)}{\partial x_2^2}\right)^2\right] dx_1 dx_2, \tag{22}$$

where $x = (x_1, x_2)$ represents a general point in the domain Ω at which the smoothness or deformation of the function u is being evaluated. For cases in higher dimensional spaces, such as the feature space applied in NeuralTPS, we can learn TPS in higher dimensional spaces with the same fitting term $J_1(u)$. The only adjustment needed is the deformation term $J_2(u)$. In general, $J_2(u)$ is,

$$J_2(u) = \int_\Omega \sum \frac{m!}{t_1! \dots t_d!}\left(\frac{\partial u(x)}{\partial x_1^{t_1} \dots \partial x_d^{t_d}}\right)^2 dx, \tag{23}$$

where m is the order of the derivatives we take and d is the dimension of x. $\{t_i\}_{i=1}^{d}$ are nonnegative integer vectors which satisfies $\sum t_i = m$.

It can be shown that a unique minimizer u exists in (19). It defines a spatial mapping that maps any location x in space to a new location $u(x)$, which can be defined as,

$$u(x) = \Gamma(x) + \alpha \sum_{i=1}^{N} w_i U(p_i, x), \tag{24}$$

where $\{w_i\}_{i=1}^{N}$ is a set of mapping coefficients. $U(p_i, x)$ is a suitable radial basis function centered at the point $p_i \in \Omega$, represented by,

$$u(p_i, x) = -||x - p_i||^2 log(||x - p_i||). \tag{25}$$

Γ in (24) is a appropriate polynomial, which is formulated as,

$$\Gamma(x) = \sum_{j=1}^{M} a_j \gamma_j(x), \tag{26}$$

where $\gamma_j(x)$ are monomials of order up to 1. The coefficients $\{a_j\}_{j=1}^{M}$ and $\{w_i\}_{i=1}^{N}$ can be solved by the following linear system,

$$\begin{cases} (K + n\alpha I)\boldsymbol{w} + P\boldsymbol{a} = \boldsymbol{y}, \\ P^T \boldsymbol{w} = 0, \end{cases} \tag{27}$$

where $K_{ij} = U(p_i, p_j)$, I is the identity matrix and $P_{ij} = \gamma_j(p_i)$. By solving the linear system, the targeted TPS can smooth the scattered points in any dimension.

Unlike solving the above linear system to obtain the minimizer u in classic methods, our NeuralTPS resolves this problem from the excellent fitting ability of the neural network and interpolates TPS in the high-dimensional feature space. Based on the smoothing effect of TPS in the high-dimensional space, we can reconstruct smoother and more complete surfaces from sparse point clouds than other methods.

I. Details

Network and Training: In surface parameterization in Fig. 1, our MLP is formed by 5 fully connected layers of size 128, 256, 512, 256, and 3. We have ReLU on the first two layers, leaky ReLU on the third and fourth layers, and tanh on the final output layer. In each iteration during training, we sample 2D points to generate $\mathcal{S}$ with 2000 3D points to calculate L_{CD} with the ground truth $\mathcal{P}$, and generate $\mathcal{G}$ with 5000 3D points to calculate L_{Pull}.

In Fig. 3, MLP1 is formed by 10 fully connected layers. All of the first 9 layers have a dimension of 512, and the last layer has a dimension of 128. We leverage the ReLU after each layer. Both MLP2 and MLP3 are formed by 1 fully connected layer with a size of 1, and do not leverage any activation function. We establish $\mathcal{Q}$ by sampling queries around each point in $\mathcal{S}$ with a Gaussian distribution, which gets more queries around the surface.

For each shape, we train our network in 40 000 iterations on an NVIDIA GTX 1080Ti GPU using an ADAM optimizer with a batch size of 5,000 and an initial learning rate of 0.0001.

Initialization Scheme: We employ the same initialization scheme as NeuralPull [9] to determine the inside and outside of the surface. We initialize the parameters in our network using Geometric Network Initialization (GNI) [7] to approximate the signed distance function of a sphere. Here, the sign of the signed distance outside of the shape is positive and negative inside. This initialization scheme enables NeuralTPS to gradually refine the implicit field from a standard sphere to the implicit field of the target object without normals during training, preserving the correct inside-outside relationship.

Sampling Ground Truth Points: We employ random sampling to select 300 points on the mesh surface, creating a realistically sparse point cloud. Given the small sample size, random sampling may result in denser concentrations of points in some areas and sparser distributions in others. It is important to note that we do not attempt to ensure a uniform distribution of samples, mirroring the inherent non-uniformity of real-world LIDAR-scanned point clouds. This approach allows us to assess our method's performance on sparse and non-uniform point clouds.

Sampling 3D Queries: We leverage a method introduced by NeuralPull [9] to sample queries around each point on the point cloud. We use Gaussian distribution centered at each point and set the standard deviation as the distance to the 51th nearest neighbor in the point cloud. This allows queries to be not too far from the surface. We sample 5000 queries around the point cloud in each iteration.

Sampling 2D Points for Surface Parameterization: We use uniform distribution in a range of $[0, 1]$ to sample 2D points. In each iteration, we sample 2,000 points to generate a 3D point cloud to supervise the surface parameterization and sample 5,000 points to generate a coarse surface estimation for SDF inference.

Number of Sparse Points: We sample only 300 points on each shape as the input of our method for training and evaluation. The reason why we sample extremely sparse points is that the communication bandwidth is a critical resource for many real-time and safety-related applications in the real world. Extremely sparse points can reduce the communication burden, especially among a group of units. Hence, these applications value more on complete and correct structures than geometry details. We push the sparsity limit in this paper and use points that are as sparse as we know to reconstruct surfaces. Note that NeedleDrop [96] also uses 300 points as input, but samples 300 different points in each iteration. Moreover, our method is not only applicable to extremely sparse point clouds but also to working with dense point clouds. We perform well with both sparse and dense point clouds. We will describe it in detail in the following experiments.

IV. Experiments

We evaluate our method in surface reconstruction from synthetic point clouds and real scans. The point clouds represent shapes and scenes. For each point cloud, we predict signed distances at grid locations using the inferred SDF f_θ, and then run the marching cubes algorithm [87] to extract a surface.

TABLE I
ACCURACY OF RECONSTRUCTION USING 300 POINTS UNDER SHAPENET IN TERMS OF $CD_{L1} \times 10$, $CD_{L2} \times 100$ AND NC

	Class	Overfitting-Based Methods						Prior-Based Methods			Ours	Ours+SACD
		NDrop	NPull	SAP	ShpGF	NSpline	VIPSS	OnSurf	P2Skeleton	NKSR		
$CD_{L1} \times 10$	Plane	0.499	0.141	0.141	0.110	0.119	1.193	0.153	0.187	0.157	0.095	**0.088**
	Chair	0.395	0.196	0.363	0.304	0.306	0.851	0.316	0.282	0.251	0.197	**0.195**
	Cabinet	0.229	0.163	0.152	0.604	0.181	0.584	0.244	0.249	0.210	0.138	**0.137**
	Display	0.287	0.145	0.281	0.521	0.193	0.518	0.204	0.287	0.251	0.127	**0.122**
	Vessel	0.488	0.116	0.138	0.367	0.134	0.571	0.128	0.144	0.138	0.104	**0.101**
	Table	0.426	0.400	0.442	0.619	0.318	1.146	0.288	0.401	0.382	0.225	**0.215**
	Lamp	0.554	0.162	0.385	0.446	0.231	0.956	0.229	0.275	0.247	0.120	**0.112**
	Sofa	0.259	0.139	0.151	0.655	0.168	0.451	0.147	0.186	0.178	**0.125**	0.129
	Mean	0.392	0.183	0.257	0.453	0.206	0.784	0.214	0.251	0.227	0.141	**0.137**
$CD_{L2} \times 100$	Plane	0.755	0.036	0.063	0.031	0.127	5.829	0.112	0.137	0.117	0.030	**0.026**
	Chair	0.532	0.174	0.429	0.275	0.247	3.291	0.448	0.355	0.219	0.149	**0.140**
	Cabinet	0.245	0.086	0.062	0.098	0.064	2.336	0.171	0.159	0.120	**0.050**	**0.050**
	Display	0.401	0.099	0.311	0.818	0.095	2.139	0.153	0.402	0.385	0.083	**0.078**
	Vessel	0.844	0.074	0.105	0.439	0.066	2.614	0.066	0.108	0.097	0.051	**0.046**
	Table	0.701	0.892	0.604	1.117	0.312	5.009	0.419	0.894	0.811	0.272	**0.264**
	Lamp	1.071	0.144	0.542	0.591	0.183	4.617	0.351	0.472	0.455	0.051	**0.047**
	Sofa	0.463	0.072	0.073	1.253	**0.053**	1.890	0.066	0.071	0.066	**0.056**	0.062
	Mean	0.627	0.197	0.274	0.578	0.143	3.470	0.223	0.325	0.284	0.093	**0.089**
NC	Plane	0.819	0.897	0.774	0.747	0.895	0.833	0.864	0.849	0.824	0.899	**0.912**
	Chair	0.777	0.861	0.725	0.547	0.759	0.821	0.813	0.804	0.815	0.863	**0.873**
	Cabinet	0.843	0.888	0.824	0.508	0.840	0.851	0.787	0.789	0.800	**0.898**	0.897
	Display	0.873	0.909	0.744	0.643	0.830	0.899	0.855	0.842	0.864	0.924	**0.936**
	Vessel	0.838	0.880	0.813	0.667	0.842	0.867	0.879	0.853	0.801	0.908	**0.913**
	Table	0.795	0.835	0.686	0.601	0.771	0.783	0.827	0.810	0.790	0.877	**0.888**
	Lamp	0.828	0.887	0.777	0.673	0.814	0.848	0.858	0.862	0.841	0.902	**0.910**
	Sofa	0.808	0.905	0.817	0.508	0.828	0.882	0.881	0.872	0.836	**0.919**	0.915
	Mean	0.823	0.883	0.770	0.612	0.822	0.848	0.845	0.835	0.821	0.899	**0.905**

A. *Surface Reconstruction for Shapes*

Dataset and Metrics: We evaluate our method in surface reconstruction for rigid shapes and non-rigid shapes in ShapeNet [105], D-FAUST [106] and MGN [107]. We report our evaluations under the test splitting of ShapeNet from NeuralPull [9], the test set of D-FAUST, and the test set of MGN. We do not learn priors, and train a neural network to overfit to each single point cloud. For each shape, we follow NeedleDrop [96] to randomly sample 300 points on each shape as the input to each method in evaluations. Using the learned implicit functions, we extract meshes as the reconstructed surfaces. We evaluate the reconstructed surfaces using L1 Chamfer Distance (CD_{L1}), L2 Chamfer Distance (CD_{L2}), and normal consistency (NC), where we sample $100\,k$ points on the reconstructed surfaces and ground truth surfaces respectively to measure errors.

Evaluations under ShapeNet with Overfitting-Based Methods: We compare our methods with the state-of-the-art overfitting-based methods including NeedleDrop (NDrop) [96], NeuralPull (NPull) [9], SAP [11], ShapeGF (ShpGF) [108], NeuralSplines (NSpline) [100], VIPSS [97]. Here, we do not compare with SAL [7] or IGR [5], since NDrop and NPull showed better performance over them. All methods do not leverage priors during training, and we train all these methods to overfit to the same sparse point clouds separately. To produce the results of ShapeGF, we use PSR [109] to reconstruct meshes from the predicted point clouds, since the code for reconstruction using gradients is not available. For the normals required by NeuralSplines as input, we provide the normals obtained on the ground truth meshes. We also do not compare the results of NDrop from its original paper, since its official code shows that it samples 300 points from a mesh in each iteration during training, which is equivalent to observing a much denser point cloud rather than a single sparse point cloud with merely 300 during training.

We report numerical comparisons in Table I. The comparisons indicate that we achieve the best results which show our superior performance over the latest overfitting-based methods. The methods without learning priors like NDrop, SAP and NPull can not learn implicit functions from merely 300 points. Our visual comparisons in Fig. 4 highlight our advantages in reconstructing more complete and smooth surfaces. It is worth mentioning that the figures of the sparse points cloud are not clear enough when the number of points is too small. Therefore, we replaced each point in the sparse points as a mesh ball with an appropriate radius to increase its size. In addition, the sparsity of the point clouds results in figures that lack a sense of depth. These are the reasons why some sparse points appear to be uniform.

Evaluations under ShapeNet with Prior-Based Methods: We further compare our methods with the state-of-the-art prior-based methods including OnSurf [45], Point2Skeleton (P2Skeleton) [84] and NKSR [86]. Here, we do not include a comparison with NKF [85], as NKF does not release its source code, and NKSR, which is based on NKF, outperforms it. We reproduce their results using the pre-trained models available in their respective code repositories. Similarly, we provide NKSR with the ground truth normals obtained from the meshes as input.

We report numerical comparison in Table I. The comparisons indicate that we achieve the best results which show our superior performance over the latest prior-based methods including OnSurf, P2Skeleton, and NKSR in surface reconstruction on sparse point clouds. The visual comparisons in Fig. 5 highlight our advantages in reconstructing more complete and smooth surfaces. This is because, these prior-based methods do not generalize well to unseen shapes, particularly on extremely

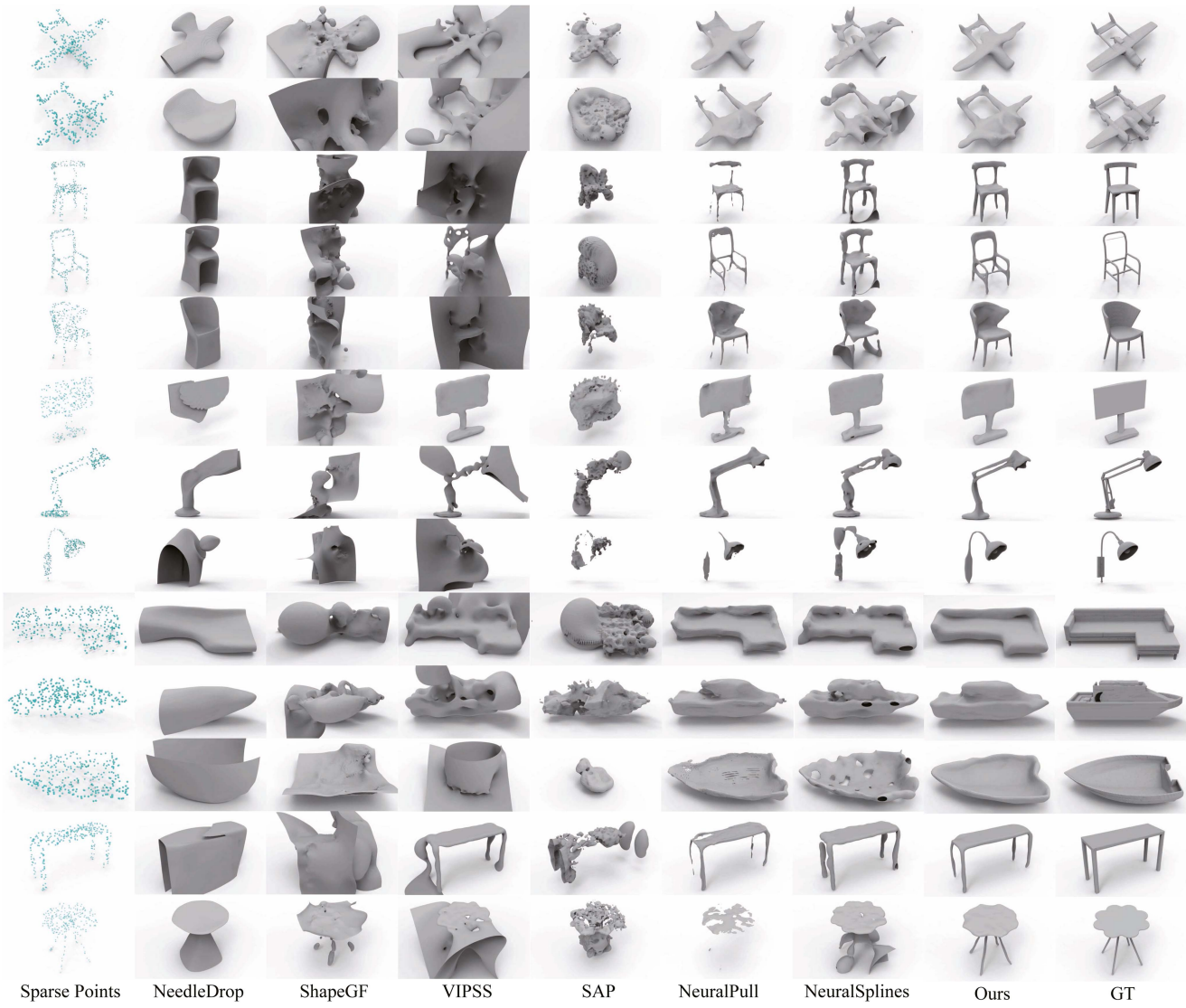

Fig. 4. Visual comparison with the state-of-the-art overfitting-based methods under ShapeNet dataset.

sparse point clouds. For instance, P2Skeleton is designed to handle sparse point clouds of 2000 points, which is significantly more than our 300 points. It also notes that if the input points are too sparse and some points on the convex hull are missing, it struggles to recover the complete geometry. NKSR requires dense point clouds and normal supervision and faces challenges in generalizing to extremely sparse point clouds despite its strong inductive bias toward solving general 3D reconstruction problems.

Evaluations under D-FAUST: We report our evaluations under D-FAUST in Table II. We follow NeedleDrop to report CD_{L2}. We report the 5%, 50%, and 95% percentiles of the CD between the surface reconstructions and the ground truth. Our method learns better SDFs which achieve better accuracy and smoother surfaces. This is also justified by our visual comparisons in Fig. 7. Since the ground truth points are sparse and non-uniform, we can see that some cases have local parts containing few points even missing. For example, the sparse points in the first and third rows of Fig. 7 have extremely sparse or missing points in the chest and arms, which challenges other reconstruction methods to recover these parts accurately. Thanks to the smooth reconstruction capability of NeuralTPS, we can still successfully reconstruct reasonable shapes from most of the randomly sampled sparse and non-uniform points.

TABLE II
ACCURACY OF RECONSTRUCTION WITH 300 POINTS UNDER D-FAUST IN TERMS OF CD_{L2} AND NC

Method	$CD_{L2}\times100$			NC
	5%	50%	95%	
NDrop	0.126	1.000	7.404	0.792
NPull	0.018	0.032	0.283	0.877
SAP	0.014	0.024	0.071	0.852
ShpGF	0.452	1.567	8.648	0.750
NSpline	0.037	0.080	0.368	0.808
OnSurf	0.015	0.037	0.123	0.908
VIPSS	0.518	4.327	9.383	0.890
Ours	**0.012**	**0.160**	**0.022**	**0.909**

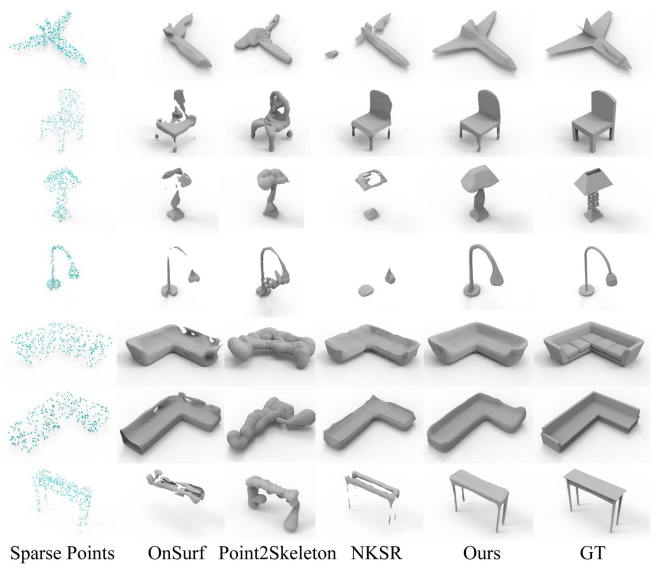

Fig. 5. Visual comparison with the state-of-the-art prior-based methods under ShapeNet dataset.

TABLE III
ACCURACY OF RECONSTRUCTION WITH 300 POINTS UNDER MGN IN TERMS OF CD_{L1}, CD_{L2} AND NC

Method	$CD_{L1} \times 10$	$CD_{L2} \times 100$	NC
PSR	0.195	0.129	0.924
NPull	0.151	0.027	0.913
CAP-UDF	0.277	0.157	0.685
Ours	**0.124**	**0.018**	**0.967**

Evaluations under MGN: We report our evaluations under MGN to highlight our advantage in reconstructing open surfaces. We compare our methods with the state-of-the-art methods, including PSR [109], NeuralPull (NPull) [9] and CAP-UDF [92]. Notably, CAP-UDF specializes in reconstructing open surfaces. We use PSR to reconstruct meshes from sparse points, and we train NeuralPull and CAP-UDF to overfit the sparse points. As shown in Table III, the result indicates that we achieve the best results which show our superior performance over other methods in handling open surfaces. Both PSR and NeuralPull struggle to reconstruct reasonable surfaces from merely 300 points. Similarly, The performance of CAP-UDF degenerates on open surfaces due to the extreme sparsity of the points. The visual comparisons in Fig. 8 highlight our advantages in reconstructing more complete and smoother surfaces.

Evaluations on SACD: We report our evaluations on the Structure-Aware Chamfer Distance (SACD) under ShapeNet with overfitting-based methods NDrop [96], ShpGF [108], VIPSS [97], NSpline [100], and prior-based methods OnSurf [45], P2Skeleton [84], NKSR [86]. As shown in Table I, the result of "Ours+SACD" indicates that we achieve the best results which show our superior performance over the latest prior-based methods. This superiority stems from SACD's ability to extract more structure from sparse points, enabling the reconstruction of more accurate and reasonable surfaces despite the points' extreme non-uniformity and the presence of missing parts. This is further evidenced by our visual comparisons in Fig. 6. We choose some typical cases with missing parts such as chairs with incomplete backrests, lamps lacking supports, and tables missing several legs. While overfitting-based methods like NDrop, ShpGF and VIPSS struggle to learn implicit functions from merely 300 points. NSpline exhibits artifacts in extremely sparse regions. The prior-based methods like Onsurf, P2Skeleton, and NKSR fail to reconstruct the missing parts effectively. Benefiting from SACD, we achieve better accuracy and smoother surfaces on sparse points with missing parts.

B. Surface Reconstruction for Scenes

Dataset and Metrics: We further evaluate our method in surface reconstruction for scenes in 3D Scene [110] and KITTI [111]. For results under 3D scene, we follow previous methods [9], [10] to randomly sample 100 points per m^2. For results under KITTI dataset, we use point clouds in single frames to conduct a comparison. Similarly, we evaluate the reconstructed surfaces using L1 Chamfer Distance (CD_{L1}), L2 Chamfer Distance (CD_{L2}), and normal consistency (NC), where we sample $1000\,k$ points on the reconstructed surfaces and ground truth surfaces respectively to measure errors.

Evaluations: We compare our methods with the state-of-the-art methods including PSR [109], NeedleDrop (NDrop) [96], NeuralPull (NPull) [9], SAP [11], and NeuralSplines (NSpline) [100]. We train each method to overfit each single point cloud. Similarly, we provide NSpline the ground truth normals as input. Our numerical comparisons in Table IV show that our method can reveal more accurate geometry in a 3D scene. Our reconstructed surfaces in Fig. 12 are smoother and more complete than others, and do not have artifacts in empty space as NSpline, which justifies our capability of handling sparsity in point clouds.

We further show our reconstructed surfaces from KITTI dataset. Since there are no ground truth meshes, we evaluate our method in visual comparisons with screened possion reconstruction (PSR) [109], NeedleDrop (NDrop) [96], NeuralPull (NPull) [9], SAP [11], and OnSurf [45]. The visual comparisons in reconstructing cars, pedestrians, and roads are shown in Figs. 9, 10, and 11. Note that the cars, pedestrians, and roads obtained from LiDAR scans in the KITTI dataset are represented as single-sided point clouds, i.e., open surfaces. Our reconstructed surfaces yield more complete surfaces with more geometry details, such as the walking poses of pedestrians, and the windows of cars. The smooth roads we construct highlight our ability to reconstruct thin structures, even when working with sparse open surfaces. Our method does not use any learnable priors, and performs much better than the methods without learning priors, such as NeedleDrop and NeuralPull, and also OnSurf which learns a prior for sparse points.

We also show our reconstructed surfaces from a large scale scan of a road from KITTI [96] in Fig. 13. We separate the point cloud into different sections, and use our method to reconstruct a mesh from point clouds in each section. Our reconstruction shows that we can handle sparse point clouds and reconstruct smooth and complete surfaces.

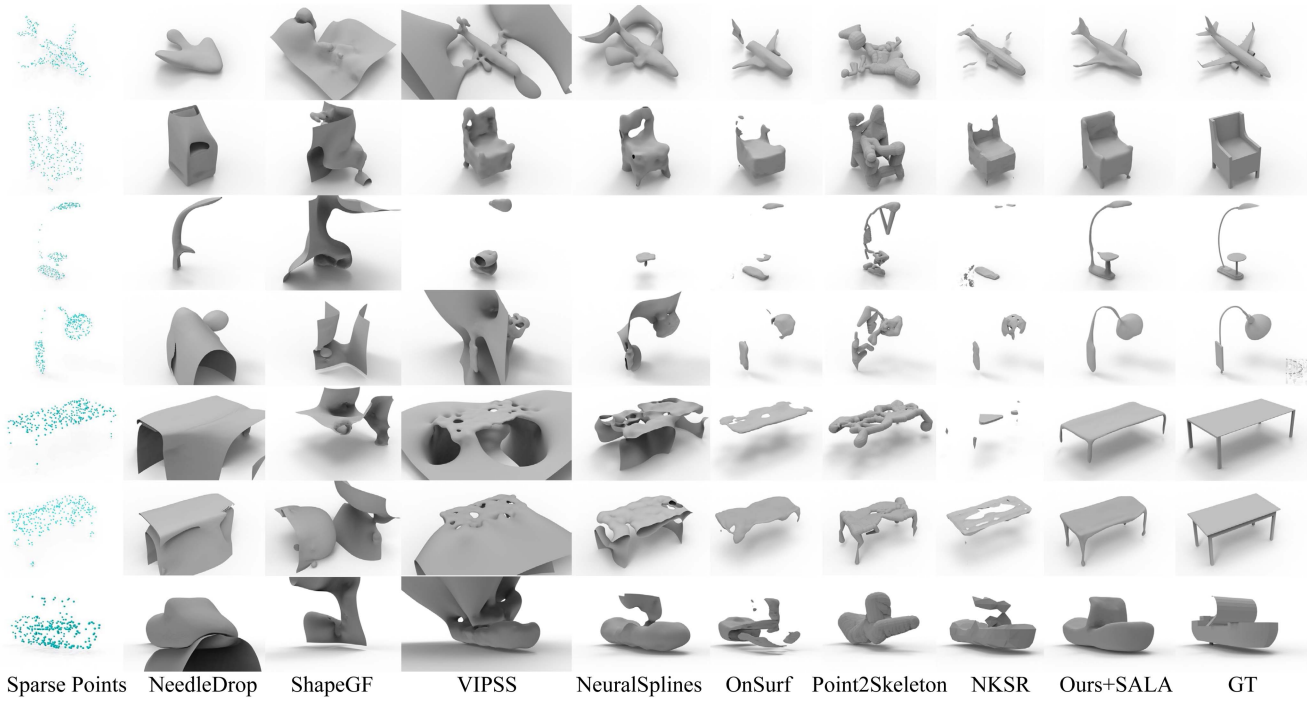

Fig. 6. Visual comparison with the state-of-the-art with large missing parts or structures under ShapeNet dataset.

TABLE IV
ACCURACY OF RECONSTRUCTION UNDER 3D SCENE IN TERMS OF L2CD, L1CD AND NC

Method	Burghers			Copyroom			Lounge			Stonewall			Totempole			Mean		
	CD_{L1}	CD_{L1}	NC	CD_{L1}	CD_{L1}	NC	CD_{L1}	CD_{L1}	NC	CD_{L1}	CD_{L1}	NC	CD_{L1}	CD_{L1}	NC	CD_{L1}	CD_{L1}	NC
PSR	0.178	0.205	0.874	0.225	0.286	0.861	0.280	0.365	0.869	0.300	0.480	0.866	0.588	1.673	0.879	0.314	0.602	0.870
NDrop	0.200	0.114	0.825	0.168	0.063	0.696	0.156	0.050	0.663	0.150	0.081	0.815	0.203	0.139	0.844	0.175	0.089	0.769
NPull	0.064	0.008	0.898	0.049	0.005	0.828	0.133	0.038	0.847	0.060	0.005	0.910	0.178	0.024	0.908	0.097	0.016	0.878
SAP	0.153	0.101	0.807	0.053	0.009	0.771	0.134	0.033	0.813	0.070	0.007	0.867	0.474	0.382	0.725	0.151	0.100	0.797
NSpline	0.135	0.123	0.891	0.056	0.023	0.855	**0.063**	0.039	0.827	0.124	0.091	0.897	0.378	0.768	0.892	0.151	0.209	0.889
Ours	**0.055**	**0.005**	**0.909**	**0.045**	**0.003**	**0.892**	0.129	**0.022**	**0.872**	**0.054**	**0.004**	**0.939**	**0.103**	**0.017**	**0.935**	**0.077**	**0.010**	**0.897**

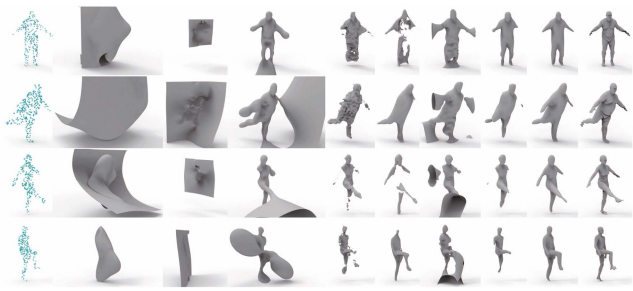

Fig. 7. Visual comparison with the state-of-the-art under D-FAUST dataset.

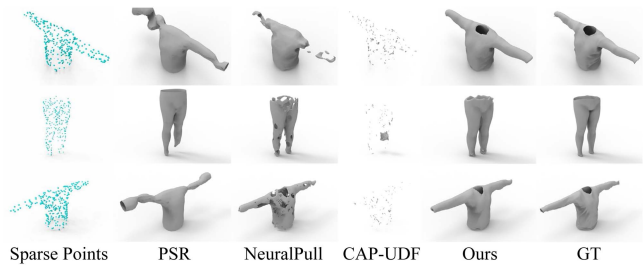

Fig. 8. Visual comparisons with the state-of-the-art under MGN dataset.

C. Analysis

Level Sets: We first visualize the signed distance field that our method learns in Fig. 14. To highlight our performance, we conduct visual comparisons with SAP [11], NeedleDrop [96], and NeuralSplines [100] on a 2D case. SAP and NeedleDrop estimate occupancy fields, while NeuralSplines and ours learn signed distance fields, and we also provide the ground truth normals to NeuralSplines to produce its results. We use each method to learn an SDF from a sparse 2D point cloud that is nonuniformly sampled on a moon-like shape, where we show these points as blue dots in Fig. 14(d). The visual comparisons of level sets learned by each method indicate that our method can employ TPS to reveal smoother level sets with the highest accuracy among the counterparts. Specifically, SAP and NeedleDrop do not deal with sparsity well. Although NeuralSplines also use splines to fit signed distances, it uses the distances along normals as the ground truth, which easily produces artifacts near sharp areas.

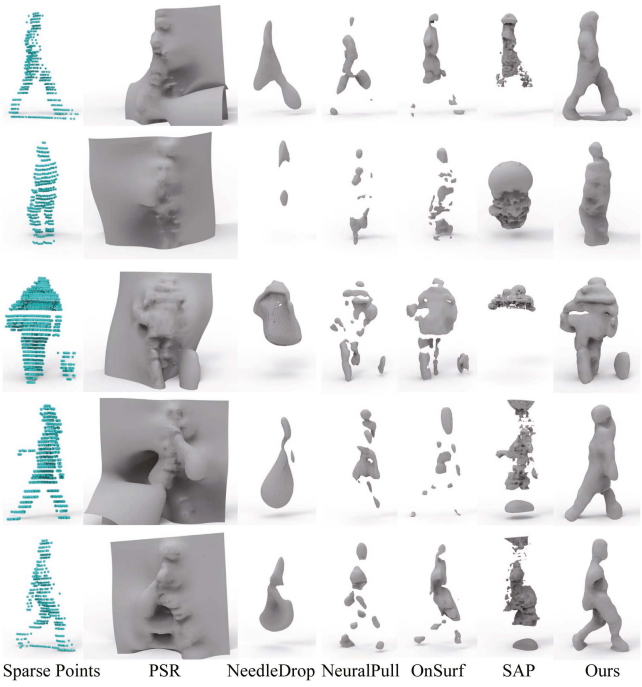

Fig. 9. Visual comparisons of pedestrians in KITTI.

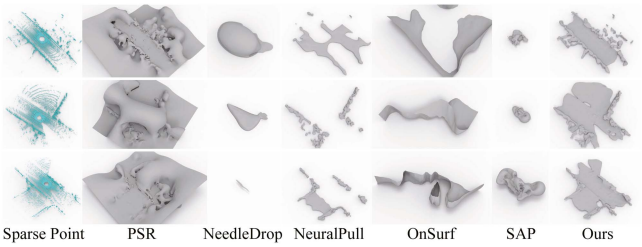

Fig. 10. Visual comparisons of roads in KITTI.

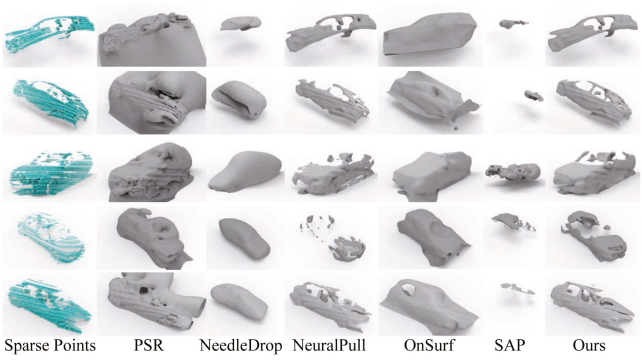

Fig. 11. Visual comparisons of cars in KITTI.

Point numbers: Although we aim to reconstruct complete surfaces from extremely sparse points, our method is insensitive to point numbers and performs well with both sparse and dense point clouds. As the points increase from 300 to 10 000, our reconstruction results become more and more accurate, as shown in Fig. 16.

TABLE V
ACCURACY OF RECONSTRUCTION UNDER AIRPLANE CLASS OF SHAPENET WITH 300 AND 2000 POINTS IN TERMS OF L2CD, L1CD AND NC

Point number		ConvOcc	P2S	OnSurf	SAP	Ours
300	$CD_{L1}\times10$	0.653	0.872	0.153	0.141	**0.095**
	$CD_{L2}\times100$	0.271	0.549	0.112	0.063	**0.030**
	NC	0.731	0.591	0.864	0.774	**0.899**
2000	$CD_{L1}\times10$	0.018	0.028	0.044	0.033	**0.012**
	$CD_{L2}\times100$	0.007	0.010	0.023	0.012	**0.005**
	NC	0.879	0.873	0.872	0.801	**0.906**

TABLE VI
ACCURACY OF RECONSTRUCTION WITH 2000 POINTS UNDER D-FAUST IN TERMS OF CD_{L2} AND NC

Method	$CD_{L2}\times100$			NC
	5%	50%	95%	
ConvOcc	0.011	0.029	0.093	0.792
P2S	0.013	0.033	0.109	0.866
OnSurf	0.008	0.019	0.044	0.931
SAP	0.006	0.012	0.026	0.904
Ours	**0.005**	**0.009**	**0.012**	**0.941**

We first evaluate our method in surface reconstruction with 300 points and 2000 points in the airplane class under ShapeNet [105]. We compare our methods with the state-of-the-art methods including ConvOcc [22], Point2surf (P2S) [23], OnSurf [45] and SAP [11]. For supervised methods including ConvOcc, P2S, and OnSurf, we produce their results using the parameters pre-trained they provide on airplane class of ShapeNet. And for unsupervised method SAP, We train it to overfit each single point cloud. We report numerical comparisons in Table V. The comparisons indicate that we achieve the best results which show our superior performance over the latest methods on both sparse and dense point clouds. The visual comparisons in Fig. 15 highlight our advantages in reconstructing more complete and smooth surfaces in 300 and 2000 points. Furthermore, we evaluate our method in surface reconstruction in unseen classes with 2000 points under D-FAUST dataset. For supervised methods including ConvOcc, P2S, and OnSurf, we produce results using their official parameters pre-trained on ShapeNet. The comparisons shown in Table VI indicate that our method can achieve better performance in unseen classes. Our reconstructed surfaces in Fig. 15 are smoother and more complete than others, which justifies our generalization ability with dense point clouds.

Double-Layer Structure: Surface reconstruction for double-layer structures is challenging because the points of the two layers are very close to each other, and sparsity in particular can make reconstruction more difficult. So we conduct an experiment to evaluate our performance in dense and sparse double-layer structure curved surfaces. As shown in Fig. 17, we perform effectively in handling both sparse and dense scenarios. The comparisons indicate that NeuralTPS can robustly reconstruct double-layer structure curved surfaces, accurately maintaining the correct inside and outside.

TSNE: We also use TSNE [112] to visualize the feature space learned for conducting TPS interpolation in Fig. 18. We can see that the features we learned to conduct TPS interpolation are compact, where features of queries (small dots) are closely surrounding features of surface points (big dots). This makes it

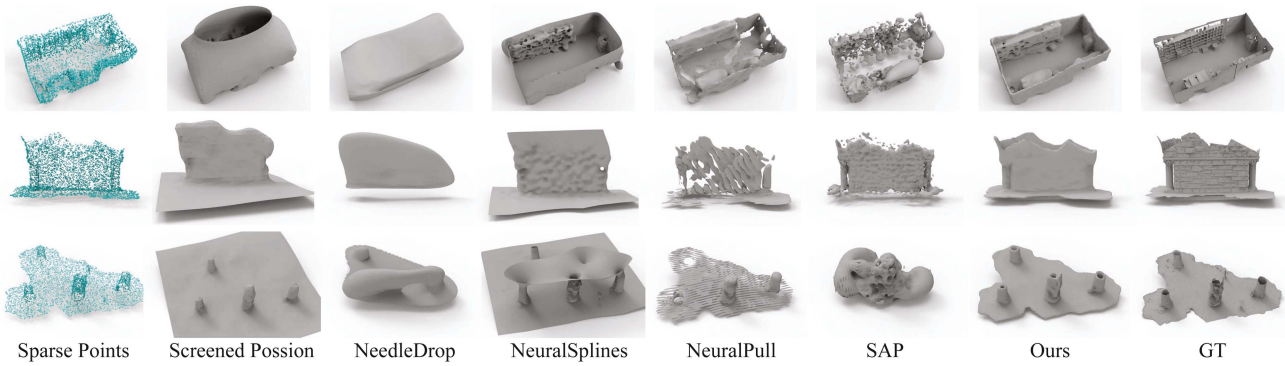

Fig. 12. Visual comparison with the state-of-the-art under 3D scene dataset.

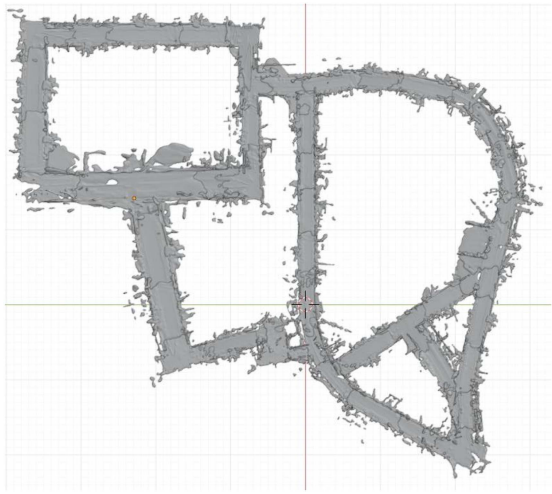

Fig. 13. Visualization of our reconstruction from a large scale scan of a road in KITTI.

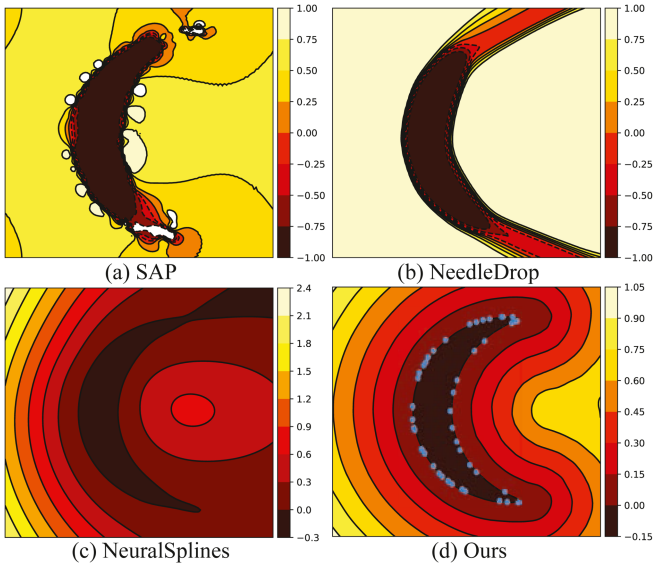

Fig. 14. Visual comparison of learned fields with SAP, NeedleDrop, NeuralSplines on a 2D case.

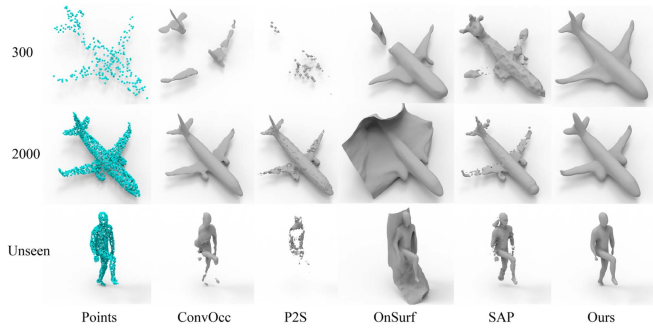

Fig. 15. Visual comparisons with 300 and 2000 points and unseen classes.

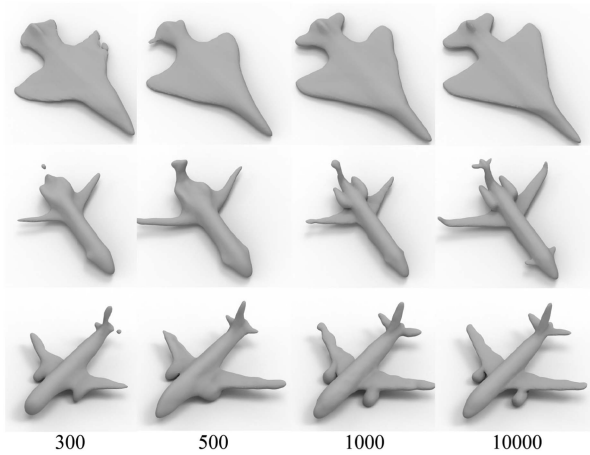

Fig. 16. Comparisons with different point numbers.

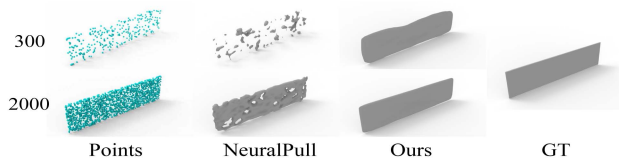

Fig. 17. Comparisons in dense and sparse double-layer structure curved surfaces.

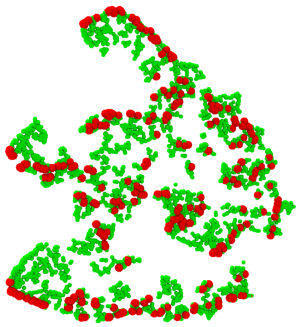

Fig. 18. We use TSNE to visualize the feature space learned for conducting TPS interpolation. Large dots indicate features of surface points, and small dots indicate features of 3D queries.

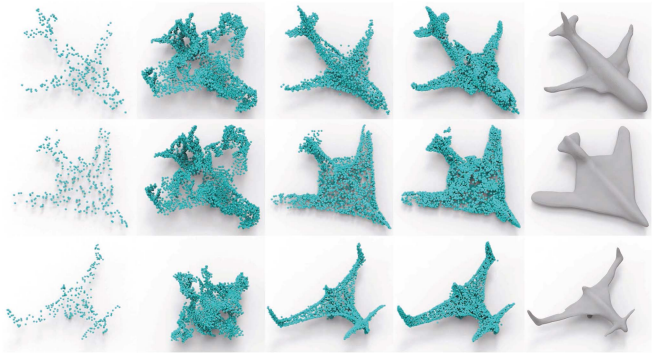

Fig. 19. Visual comparisons of surface parameterizations.

TABLE VII
EFFECT OF SURFACE PARAMETERIZATION

	Separate	Upsample	GradDiff	Ours
$CD_{L1}\times 10$	0.105	0.582	0.102	**0.095**
$CD_{L2}\times 100$	0.118	0.815	0.047	**0.030**
NC	0.891	0.724	0.897	**0.899**

TABLE VIII
EFFECT OF PATCH NUMBERS

	1	3	5
$CD_{L1}\times 10$	**0.095**	0.114	0.115
$CD_{L2}\times 100$	**0.030**	0.071	0.077
NC	**0.899**	0.898	0.896

TABLE IX
EFFECT OF THE SDF INFERENCE

	PSR	NP	Ours
$CD_{L1}\times 10$	0.325	0.213	**0.095**
$CD_{L2}\times 100$	0.218	0.187	**0.030**
NC	0.813	0.854	**0.899**

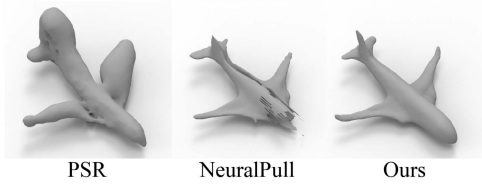

Fig. 20. Visual comparisons of the SDF Inference.

easier to use surface points in basis functions for approximating SDFs using TPS interpolation.

We visualize our optimization including learned level sets, the feature space to perform TPS interpolation and more results with real scans.

D. Ablation Studies

To justify each module of our method, we conduct our ablation studies in airplane class that we used in Section IV-A under ShapeNet. We report results in surface reconstruction with different variations of our method.

Surface Parameterizations: We first highlight the benefits of end-to-end training in Table VII. Hence, we optimize surface parameterization and SDF inference separately, and merely use the predicted 5000 points from surface parameterization to infer SDF, as shown by the result of "Separate". The degenerated results show that we can infer more accurate SDFs by observing surface estimation in different iterations. Then, we replace surface parameterization into the latest point cloud upsampling method [113] to upsample the 300 point input to 5000 points. The result of "Upsample" indicates that the upsampling method can not generalize the learned prior to upsample 300 points into a plausible shape with 5000 points. As shown in Fig. 19, the upsampling method is sensitive to the density of input points, resulting in distorted shapes after upsampling. In contrast, both of our surface parameterization and the points pulled to the surface fill the gaps of the sparse point cloud well, which leads to a smooth and complete shape. Next, we show the effect of gradient stop by turning it off. The result of "GradDiff" indicates that the error backpropagated from the SDF inference brings too much uncertainty to the surface parameterizations, which turns to degenerate the SDF inference.

We conduct experiments to explore the effect of using more patches for surface parameterizations. As shown in Fig. 2, we use more branches like AtlasNet [102] to cover the surface by generating more patches, such as $\{1, 3, 5\}$. The comparison in Table VIII shows that more patches degenerate reconstruction accuracy. Since more patches result in larger gaps between patches as we pointed out in Fig. 2, which does not resolve the sparsity on the surface.

SDF Inference: We further validate the effectiveness of our SDF inference as shown in Table IX and Fig. 20. We first replace the SDF inference with the classic method Poisson Surface Reconstruction (PSR) [109]. We reconstruct the surface of shape $\mathcal{G}$ from surface parameterization directly with PSR, as shown by the result of "PSR". The degenerated results and the swollen surface indicate that PSR is unable to perform accurate surface reconstruction for the coarse shape $\mathcal{G}$. Then we removed the NeuralTPS module and replaced it with MLPs, which are equivalent to NeuralPull [9]. The result of "NP" indicates that NeuralPull is unable to predict the accurate SDF from the coarse shape $\mathcal{G}$. Even though the shape $\mathcal{G}$ is denser compared to the ground truth $\mathcal{P}$ (only 300 points), there are still large gaps between some

TABLE X
EFFECT OF LOSSES

	No L_{CD}	No L_{Surf}	Ours
$CD_{L1}\times10$	0.146	0.108	**0.095**
$CD_{L2}\times100$	0.109	0.041	**0.030**
NC	0.844	0.898	**0.899**

TABLE XI
EFFECT OF BALANCE WEIGHTS α AND β

α,β	0.02,0.02	0.1,0.1	0.1,0.5	0.5,0.1	0.5,0.5
$CD_{L1}\times10$	0.152	**0.095**	0.133	0.129	0.144
$CD_{L2}\times100$	0.093	**0.030**	0.078	0.071	0.080
NC	0.866	**0.899**	0.841	0.852	0.847

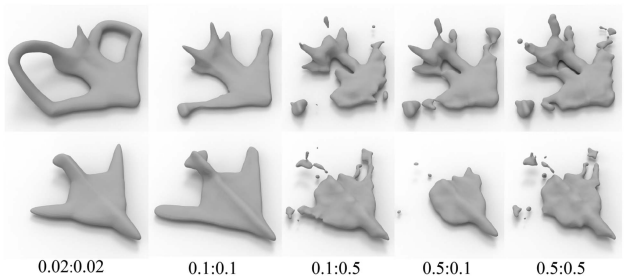

Fig. 21. Comparisons with different weights α and β.

TABLE XII
EFFECT OF THE DECAY PARAMETER δ

δ	10	30	50	70
$CD_{L1}\times10$	0.119	0.111	**0.095**	0.122
$CD_{L2}\times100$	0.060	0.063	**0.030**	0.066
NC	0.863	0.869	**0.899**	0.866

TABLE XIII
EFFECT OF THIN PLATE SPLINES

	$\lvert r\rvert^3$	No Feature	No Disp	Ours
$CD_{L1}\times10$	0.110	0.791	0.159	**0.095**
$CD_{L2}\times100$	0.043	1.699	0.193	**0.030**
NC	0.895	0.691	0.898	**0.899**

TABLE XIV
EFFECT OF STRUCTURE-AWARE CHAMFER DISTANCE

	Ours	Ours+L_{SACD}	Ours+L_{SACD}+Strategy	Ours+All
$CD_{L1}\times10$	0.095	0.091	0.089	**0.088**
$CD_{L2}\times100$	0.030	0.028	0.027	**0.026**
NC	0.899	0.903	0.909	**0.912**

TABLE XV
EFFECT OF BALANCE WEIGHTS γ AND δ IN SACD

γ,δ	0.02,0.02	0.1,0.1	0.1,0.5	0.5,0.1	0.5,0.5
$CD_{L1}\times10$	0.132	**0.088**	0.122	0.131	0.147
$CD_{L2}\times100$	0.082	**0.026**	0.075	0.079	0.092
NC	0.874	**0.912**	0.853	0.865	0.849

points. As shown in Fig. 20, NeuralPull is sensitive to the density of input points, resulting in incomplete surfaces. In contrast, our NeuralTPS module performs TPS interpolation in the feature space to produce a smooth field, which somewhat fills the gaps on the shape $\mathcal{G}$ and thus yields smooth and complete surfaces.

Loss: We conduct experiments to explore the importance of each term in our loss function in (8). We remove each of them respectively, and report results in Table X. Since we learn SDF, we keep L_{Pull} in all experiments. We first remove L_{CD} and pull queries directly on the sparse points rather than the output of surface parameterization. The degenerated results of "No L_{CD}" show that the surface parameterization provides an important surface estimation to infer SDFs. Then, we remove L_{Surf}, and get slightly worse results of "No L_{Surf}". These results show the effectiveness of each term, and surface parameterizations supervised by L_{CD} are the most important.

Balance Weights: We set α and β to balance the amplitude of the three terms when the network converges and report the effect of balance weights α and β in our loss function in Table XI. It shows that $\alpha = 0.1$ and $\beta = 0.1$ are the best. We reduce the values of α and β, which results in the SDF inference module being insufficiently trained to infer the accurate surface. When we increase the values of α and β, this causes the SDF inference module to optimize too fast, while the surface parameterization module at the same time does not obtain an optimal shape $\mathcal{S}$, making the additional coarse surface estimation $\mathcal{G}$ to be sparse and incomplete, thus producing an incomplete surface. As shown in Fig. 21, larger α and β highlight the effect of sparse input, and produce incomplete surface, while smaller ones over smooth the surface, even connect nearby surfaces for some shapes, which is caused by the inaccurate SDF inference.

We further report the effect of the decay parameter δ in (7). Table XII shows that both smaller and larger δ can degenerate results.

Thin Plate Splines: We report ablation studies related to TPS in Table XIII. We first replace TPS $\psi(r) = r^2 log(r)$ into other splines like $\psi(r) = |r|^3$. The result of "$|r|^3$" shows that the basis function we use performs better. Then, we highlight the feature space where we do TPS interpolation by removing the MLP1 in Fig. 3. Instead, we do TPS directly in the spatial space. The result of "No Feature" drastically degenerates, which indicates it is more effective to perform TPS interpolation in an optimized feature space than in spatial space. Next, we explore the effect of the displacement d_{FEA} by removing the MLP3 in Fig. 3. The result of "No Disp" shows that predicting SDFs directly from the feature is a good remedy to the TPS prediction.

Structure-Aware Chamfer Distance: We report the effect of the Structure-Aware Chamfer Distance (SACD) in Table XIV. We first replace L_{CD} into L_{SACD}. The result of "Ours+L_{SACD}" shows that L_{SACD} can capture more structural information from the sparse points than L_{CD}. Then, we further apply the training strategy introduced in SACD. The result of "Ours+L_{SACD}+Strategy" indicates that we can infer SDFs more accurately with the help of training strategy. Next, we explore the effect of the L2 parameter regularization. The result of "Ours+All" has a slightly improvement.

We also explore the effect of the balance weights used in SACD. We first report the effect of balance weights γ and δ under the condition $\lambda = 1e-4$. As shown in Table XV, similar to α and β, optimal results occur when both γ and δ are set to 0.1. Deviations from this value, either increasing or decreasing, lead to worse results. Subsequently, we report the effect of balance

TABLE XVI
EFFECT OF BALANCE WEIGHTS λ IN SACD

λ	1e-6	1e-5	1e-4	1e-3	1e-2
$CD_{L1}\times10$	0.089	**0.088**	**0.088**	0.095	0.122
$CD_{L2}\times100$	0.027	0.027	**0.026**	0.034	0.070
NC	0.909	0.910	**0.912**	0.887	0.860

TABLE XVII
EFFECT OF DECAY PARAMETERS ϵ, η IN SACD

	Param	5	10	15	20
$CD_{L1}\times10$	ϵ	0.090	**0.088**	0.091	0.093
$CD_{L2}\times100$		0.024	**0.026**	0.029	0.031
NC		0.910	**0.912**	0.909	0.900
$CD_{L1}\times10$	η	0.093	**0.088**	0.091	0.101
$CD_{L2}\times100$		0.032	**0.026**	0.030	0.039
NC		0.899	**0.912**	0.904	0.884

TABLE XVIII
EFFECT OF PERCEPTUAL PARAMETERS r, k IN SACD

	Param	2	3	4	5
$CD_{L1}\times10$	r	0.090	**0.088**	0.094	0.099
$CD_{L2}\times100$		0.027	**0.026**	0.030	0.035
NC		0.906	**0.912**	0.901	0.891
$CD_{L1}\times10$	k	0.089	**0.088**	0.089	0.092
$CD_{L2}\times100$		0.027	**0.026**	0.029	0.030
NC		0.908	**0.912**	0.902	0.897

TABLE XIX
EFFECT OF DISTANCE FUNCTIONS FOR OPEN SURFACES

	Ours(UDFs)	Ours(SDFs)
$CD_{L1}\times10$	0.130	**0.124**
$CD_{L2}\times100$	0.023	**0.018**
NC	0.965	**0.967**

weight λ, with γ and δ stayed to 0.1, as detailed in Table XVI. Increasing the value of λ enhances the L2 regularization term, making the features too sparse to effectively model the sparse point cloud. Conversely, decreasing λ diminishes the influence of the L2 regularization term, making it negligible.

We further explore the effect of decay parameters ϵ, η and perceptual parameters r, k. As shown in Tables XVII and XVIII, both excessively large and small values of these parameters lead to a decline in performance.

UDFs: We further explore the effect of Unsigned Distance Functions (UDFs) on open surfaces under the MGN datasets. To do this, we extend NeuralTPS from Signed Distance Functions (SDFs) to UDFs, renaming our signed distance function inference to unsigned distance function inference. Specifically, we replace our pulling loss in UDF inference with the pulling loss introduced for unsigned distances in CAP-UDF [92]. Additionally, we employ the surface extraction algorithm from CAP-UDF to extract surfaces from UDFs. An experimental comparison of UDF-based NeuralTPS against SDF-based NeuralTPS reveals that using UDFs to represent open surfaces does not yield significant improvements, as shown in Table XIX.

Noise Levels: We report the effect of noises in our method in Table XX. We add Gaussian noises with standard deviations including $\{1\%, 2\%, 3\%\}$. Our results slightly degenerate with 1% and 2% noises and get much worse with 3% noises. Compared to SAP, our method is more robust to noises, as shown in Fig. 22.

TABLE XX
EFFECT OF NOISE LEVELS

	Method	0%	1%	2%	3%
$CD_{L1}\times10$	SAP	0.141	0.254	0.332	0.561
$CD_{L2}\times100$		0.063	0.152	0.212	0.348
NC		0.774	0.645	0.622	0.601
$CD_{L1}\times10$	Ours	**0.095**	0.103	0.177	0.272
$CD_{L2}\times100$		**0.030**	0.030	0.041	0.114
NC		**0.899**	0.878	0.835	0.803

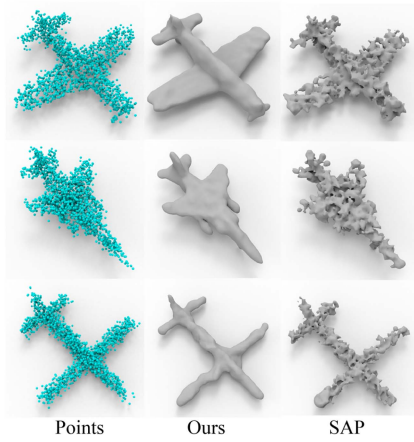

Fig. 22. Comparisons with noises.

TABLE XXI
COMPARISON OF TRAINING TIME AND NUMBER OF PARAMETERS

Method	Parameter	Time/min
NeedleDrop	5968897	26.83
ShapeGF	2668707	34.22
NeuralPull	**2169601**	**17.74**
OnSurf	7247723	31.60
Ours	2501553	20.24

Complexity: We report the comparison with other methods in terms of training time and number of parameters in Table XXI. All methods are trained on an NVIDIA GTX 1080Ti GPU. Our method has fewer parameters and a shorter training time than other methods for learning SDFs from sparse points. While we have one additional branch for surface parameterization, which makes us have more parameters and a little longer training time than NeuralPull [9].

V. CONCLUSION, LIMITATION AND FUTURE WORK

We present a method to infer SDFs from single sparse point clouds without using signed distance supervision, learned priors or even normals. We achieve this by learning surface parameterizations and SDF inference in an end-to-end manner. We parameterize the surface as a single chart, which significantly reduces the impact of the sparsity. By evaluating surface parameterization in different iterations, we provide a novel perspective to mine supervision from multiple coarse surface estimations for SDF inference. We also successfully leverage TPS interpolation in feature space to impose smooth constraints on inferring SDF from multiple coarse surface estimations in a statistical way. We justify the effectiveness of key modules and report results that

outperform the state-of-the-art methods under the widely used benchmarks.

Our current method has some failure cases in some extremely sparse and uneven cases. For example, some roads in KITTI datasets have only a very few points on some segments, resulting in poor reconstructions. To tackle this problem, we consider two potential future works of our method. First, by introducing the attention mechanism, points with the sparser neighbors have higher weights, thus guiding the network to focus more on the parts with fewer points and reconstructing a more complete and smoother surface. Second, our current reconstruction results lack details. On the one hand, the point cloud itself is extremely sparse and lacks details, on the other hand, NeuralTPS learns a global implicit field and lacks the ability to learn fine local features. In order to improve the reconstruction accuracy on large-scale scene datasets, learning a learning strategy on local regions is more promising, and we will refer to some local reconstruction algorithms [10], [16], [44] to introduce LocalNeuralTPS. It is interesting to extend our method for reconstructing high-quality surfaces on large-scale and extremely sparse point clouds with complex geometry.

Acknowledgment

Our code is available at https://github.com/chenchao15/NeuralTPS.

References

[1] J. Wang et al., "NeuRIS: Neural reconstruction of indoor scenes using normal priors," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 139–155.
[2] A. Dai and M. Nießner, "Neural poisson: Indicator functions for neural fields," 2022, *arXiv:2211.14249*.
[3] X. Long et al., "NeuralUDF: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 20834–20843.
[4] Q. Xu et al., "Point-NeRF: Point-based neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5438–5448.
[5] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3789–3799.
[6] W. Bing et al., "RangeUDF: Semantic surface reconstruction from 3 D point clouds," 2022, *arXiv:2204.09138*.
[7] M. Atzmon and Y. Lipman, "SAL:Sign agnostic learning of shapes from raw data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2565–2574.
[8] A. Boulch and R. Marlet, "POCO: Point convolution for surface reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6302–6314.
[9] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-Pull: Learning signed distance function from point clouds by learning to pull space onto surface," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 7246–7257.
[10] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser, "Local implicit grid representations for 3D scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6001–6010.
[11] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable poisson solver," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 13032–13044.
[12] W. Zhao, J. Lei, Y. Wen, J. Zhang, and K. Jia, "Sign-agnostic implicit learning of surface self-similarities for shape modeling and reconstruction from raw point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10256–10265.
[13] M. Atzmon and Y. Lipman, "SALD: Sign agnostic learning with derivatives," in *Proc. Int. Conf. Learn. Representations*, 2021.
[14] Y. Ben-Shabat, C. Hewa Koneputugodage, and S. Gould, "DiGS: Divergence guided shape implicit neural representation for unoriented point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19323–19332.
[15] W. Yifan, S. Wu, C. Oztireli, and O. Sorkine-Hornung, "ISO-Points: Optimizing neural implicit surfaces with hybrid representations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 374–383.
[16] C. Chen, Y.-S. Liu, and Z. Han, "Latent partition implicit with surface codes for 3D representation," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 322–343.
[17] Z. Mi, Y. Luo, and W. Tao, "SSRNet: Scalable 3D surface reconstruction network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 970–979.
[18] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser, "Learning shape templates with structured implicit functions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 7154–7164.
[19] M. Jia and M. Kyan, "Learning occupancy function from point clouds for surface reconstruction," 2010, *arXiv: 2010.11378*.
[20] S.-L. Liu, H.-X. Guo, H. Pan, P. Wang, X. Tong, and Y. Liu, "Deep implicit moving least-squares functions for 3D reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1788–1797.
[21] J. Tang, J. Lei, D. Xu, F. Ma, K. Jia, and L. Zhang, "SA-ConvONet: Sign-agnostic optimization of convolutional occupancy networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6504–6513.
[22] L. M. M. P. A. G. Songyou and M. P. Niemeyer, "Convolutional occupancy networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 523–540.
[23] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer, "Points2Surf: Learning implicit surfaces from point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 108–124.
[24] S. Li, J. Zhou, B. Ma, Y.-S. Liu, and Z. Han, "NeAF: Learning neural angle fields for point normal estimation," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 1396–1404.
[25] B. Ma, J. Zhou, Y.-S. Liu, and Z. Han, "Towards better gradient consistency for neural signed distance functions via level set alignment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17724–17734.
[26] Q. Li et al., "SHS-Net: Learning signed hyper surfaces for oriented normal estimation of point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 13591–13600.
[27] M. Wang et al., "LP-DIF: Learning local pattern-specific deep implicit function for 3D objects and scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 21856–21865.
[28] S. Li, J. Zhou, B. Ma, Y.-S. Liu, and Z. Han, "Learning continuous implicit field with local distance indicator for arbitrary-scale point cloud upsampling," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 3181–3189.
[29] Q. Li et al., "NeuralGF: Unsupervised point normal estimation by learning neural gradient function," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 66006–66019.
[30] Q. Li, H. Feng, K. Shi, Y. Fang, Y.-S. Liu, and Z. Han, "Neural gradient learning and optimization for oriented point normal estimation," in *Proc. SIGGRAPH Asia 2023 Conf. Papers*, 2023, pp. 1–9.
[31] H. Huang, Y. Wu, J. Zhou, G. Gao, M. Gu, and Y.-S. Liu, "NeuSurf: On-surface priors for neural surface reconstruction from sparse input views," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 2312–2320.
[32] S. Li, G. Gao, Y. Liu, Y.-S. Liu, and M. Gu, "GridFormer: Point-grid transformer for surface reconstruction," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 3163–3171.
[33] A. Ouasfi and A. Boukhayma, "Unsupervised occupancy learning from sparse point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 21729–21739.
[34] C. Chen, Y.-S. Liu, and Z. Han, "GridPull: Towards scalability in learning implicit representations from 3D point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 18322–18334.
[35] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. P. Eriksson, "Deep level sets: Implicit surface representations for 3D shape inference," 1901, *arXiv: 1901.06802*.
[36] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.
[37] J. Huang, H.-X. Chen, and S.-M. Hu, "A neural Galerkin solver for accurate surface reconstruction," *ACM Trans. Graph.*, vol. 41, no. 6, pp. 1–16, 2022.

[38] A. Pumarola, A. Sanakoyeu, L. Yariv, A. Thabet, and Y. Lipman, "VisCo grids: Surface reconstruction with viscosity and coarea grids," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 18060–18071.
[39] A. Ouasfi and A. Boukhayma, "Few 'zero level set'-shot learning of shape signed distance functions in feature space," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 561–578.
[40] T. Li, X. Wen, Y. Liu, H. Su, and Z. Han, "Learning deep implicit functions for 3D shapes with dynamic code clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12830–12840.
[41] Q. Li et al., "Learning signed hyper surfaces for oriented point cloud normal estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2024, doi: 10.1109/TPAMI.2024.3431221.
[42] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, "Deep geometric prior for surface reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10130–10139.
[43] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "PatchNets: Patch-based generalizable deep implicit 3D shape representations," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 293–309.
[44] R. Chabra et al., "Deep local shapes: Learning local SDF priors for detailed 3D reconstruction," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 608–625.
[45] B. Ma, Y. Liu, and Z. Han, "Reconstructing surfaces for sparse point clouds with on-surface priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6305–6315.
[46] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7462–7473.
[47] J. Chibane, A. Mir, and G. Pons-Moll, "Neural unsigned distance fields for implicit function learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 21638–21652.
[48] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 567–585, Jun. 1989.
[49] C. Chen, Z. Han, and Y.-S. Liu, "Unsupervised inference of signed distance functions from single sparse point clouds without learning priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17712–17723.
[50] B. Mildenhall, P.P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 405–421.
[51] M. Oechsle, S. Peng, and A. Geiger, "UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 5589–5599.
[52] Z. Han, C. Chen, Y.-S. Liu, and M. Zwicker, "DRWR: A differentiable renderer without rendering for unsupervised 3D structure learning from silhouette images," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3994–4005.
[53] Z. Han, G. Qiao, Y.-S. Liu, and M. Zwicker, "SeqXY2SeqZ: Structure learning for 3D shapes by sequentially predicting 1D occupancy segments from 2D coordinates," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 4321–4331.
[54] C. Chen, Z. Han, Y.-S. Liu, and M. Zwicker, "Unsupervised learning of fine structure generation for 3D point clouds by 2D projections matching," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 12466–12477.
[55] P. Xiang et al., "SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 5499–5509.
[56] X. Wen et al., "PMP-Net: Point cloud completion by transformer-enhanced multi-step point moving paths," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 852–867, Jan. 2023.
[57] T. Takikawa et al., "Neural geometric level of detail: Real-time rendering with implicit 3D shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11358–11367.
[58] J. N. P. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein, "ACORN: Adaptive coordinate networks for neural scene representation," *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–13, 2021.
[59] K. Rematas, R. Martin-Brualla, and V. Ferrari, "ShaRF: Shape-conditioned radiance fields from a single view," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8948–8958.
[60] Z. Han, C. Chen, Y.-S. Liu, and M. Zwicker, "ShapeCaptioner: Generative caption network for 3D shapes by learning a mapping from parts detected in multiple views to sentences," in *Proc. ACM Int. Conf. Multimedia*, 2020, pp. 1018–1027.
[61] Z. Han, X. Wang, Y. Liu, and M. Zwicker, "Hierarchical view predictor: Unsupervised 3D global feature learning through hierarchical prediction among unordered views," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 3862–3871.
[62] P. Xiang et al., "Snowflake point deconvolution for point cloud completion and generation with skip-transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 6320–6338, May 2023.
[63] W. Zhang, K. Shi, Y.-S. Liu, and Z. Han, "Learning unsigned distance functions from multi-view images with volume rendering priors," in *Proc. Eur. Conf. Comput. Vis.*, 2024.
[64] C. Chen, Y.-S. Liu, and Z. Han, "Learning local pattern modularization for point cloud reconstruction from unseen classes," in *Proc. Eur. Conf. Comput. Vis.*, 2024.
[65] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.
[66] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5939–5948.
[67] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3D-structure-aware neural scene representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1121–1132.
[68] S. Liu, Y. Zhang, S. Peng, B. Shi, M. Pollefeys, and Z. Cui, "DIST: Rendering deep implicit signed distance function with differentiable sphere tracing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2019–2028.
[69] Y. Jiang, D. Ji, Z. Han, and M. Zwicker, "SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1251–1261.
[70] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon, "Autolabeling 3D objects with differentiable rendering of SDF shape priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12224–12233.
[71] S. Liu, S. Saito, W. Chen, and H. Li, "Learning to infer implicit surfaces without 3D supervision," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8295–8306.
[72] Y. Wu and Z. Sun, "DFR: Differentiable function rendering for learning 3D generation from images," *Comput. Graph. Forum*, vol. 39, no. 5, pp. 241–252, 2020.
[73] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3504–3515.
[74] C.-H. Lin, C. Wang, and S. Lucey, "SDF-SRN: Learning signed distance 3D object reconstruction from static images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 11453–11464.
[75] L. Yariv et al., "Multiview neural surface reconstruction by disentangling geometry and appearance," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 2492–2502.
[76] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 4805–4815.
[77] Q. Fu, Q. Xu, Y. Ong, and W. Tao, "Geo-Neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 3403–3416.
[78] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 27171–27183.
[79] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 25018–25032.
[80] Y. Wang, I. Skorokhodov, and P. Wonka, "HF-NeuS: Improved surface reconstruction using high-frequency details," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 1966–1978.
[81] D. Vicini, S. Speierer, and W. Jakob, "Differentiable signed distance function rendering," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 125:1–125:18, 2022.
[82] B. Ma, Y.-S. Liu, M. Zwicker, and Z. Han, "Inferring 3D occupancy fields through implicit reasoning on silhouette images," in *Proc. ACM Int. Conf. Multimedia*, 2024.
[83] M. Liu, X. Zhang, and H. Su, "Meshing point clouds with predicted intrinsic-extrinsic ratio guidance," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 68–84.

[84] C. Lin, C. Li, Y. Liu, N. Chen, Y.-K. Choi, and W. Wang, "Point2Skeleton: Learning skeletal representations from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4277–4286.
[85] F. Williams et al., "Neural fields as learnable kernels for 3D reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18500–18510.
[86] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams, "Neural kernel surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 4369–4379.
[87] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.
[88] B. Ma, Y. Liu, M. Zwicker, and Z. Han, "Surface reconstruction from point clouds by learning predictive context priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6316–6327.
[89] B. Ma, Y.-S. Liu, and Z. Han, "Learning signed distance functions from noisy 3D point clouds via noise to noise mapping," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 23338–23357.
[90] B. Ma, J. Zhou, Y.-S. Liu, and Z. Han, "Towards better gradient consistency for neural signed distance functions via level set alignment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17724–17734.
[91] J. Zhou, B. Ma, Y.-S. Liu, Y. Fang, and Z. Han, "Learning consistency-aware unsigned distance functions progressively from raw point clouds," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 16481–16494.
[92] J. Zhou et al., "CAP-UDF: Learning unsigned distance functions progressively from raw point clouds with consistency-aware field optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–18, 2024, doi: 10.1109/TPAMI.2024.3392364.
[93] J. Zhou, B. Ma, Y.-S. Liu, and Z. Han, "Fast learning of signed distance functions from noisy point clouds via noise to noise mapping," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2024, doi: 10.1109/TPAMI.2024.3416068.
[94] J. Zhou, W. Zhang, B. Ma, K. Shi, Y. Liu, and Z. Han, "UDiFF: Generating conditional unsigned distance fields with optimal wavelet diffusion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 21496–21506.
[95] J. Zhou, B. Ma, S. Li, Y.-S. Liu, and Z. Han, "Learning a more continuous zero level set in unsigned distance fields through level set projection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 3181–3192.
[96] A. Boulch, P.-A. Langlois, G. Puy, and R. Marlet, "NeeDrop: Self-supervised shape representation from sparse point clouds using needle dropping," in *Proc. Int. Conf. 3D Vis.*, 2021, pp. 940–950.
[97] Z. Huang, N. Carr, and T. Ju, "Variational implicit point set surfaces," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–13, 2019.
[98] J. Zhao and H. Zhang, "Thin-plate spline motion model for image animation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 3647–3656.
[99] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7509–7520.
[100] F. Williams, M. Trager, J. Bruna, and D. Zorin, "Neural splines: Fitting 3D surfaces with infinitely-wide neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9949–9958.
[101] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 206–215.
[102] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A Papier-Mâché approach to learning 3D surface generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 216–224.
[103] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 728–737.
[104] G. Turk and J. F. O'Brien, "Modelling with implicit surfaces that interpolate," *ACM Trans. Graph.*, vol. 21, no. 4, pp. 855–873, 2002.
[105] A. X. Chang et al., "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
[106] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic FAUST: Registering human bodies in motion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6233–6242.
[107] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll, "Multi-garment net: Learning to dress 3D people from images," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 5420–5430.
[108] R. Cai et al., "Learning gradient fields for shape generation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 364–381.
[109] M. M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 29:1–29:13, 2013.
[110] Q. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 112:1–112:8, 2013.
[111] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[112] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.
[113] W. Feng, J. Li, H. Cai, X. Luo, and J. Zhang, "Neural points: Point cloud representation with neural fields for arbitrary upsampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18633–18642.

Chao Chen received the BS degree in software engineering from Tsinghua University, Beijing, China, in 2019, and the PhD degree from the School of Software, Tsinghua University, 2024. His research interests include deep learning and 3D computer vision.

Yu-Shen Liu (Member, IEEE) received the BS degree in mathematics from Jilin University, China, in 2000, and the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2006. From 2006 to 2009, he was a post-doctoral researcher with Purdue University. He is currently an associate professor with the School of Software, Tsinghua University. His research interests include shape analysis, pattern recognition, machine learning, and semantic search.

Zhizhong Han received the PhD degree from Northwestern Polytechnical University, China, 2017. He was a post-doctoral researcher with the Department of Computer Science, University of Maryland, College Park, USA. Currently, he is an assistant professor of computer science with Wayne State University, USA. His research interests include 3D computer vision, digital geometry processing, and artificial intelligence.