APSIPA Transactions on Signal and Information Processing, 2025, 14, e105

Original

LPSR: Lightweight Point Cloud Surface Reconstruction

Qingyang Zhou[1*], Chee-An Yu[1], Xuechun Hua[1], Shan Liu[2] and C.-C. Jay Kuo[1]

[1] *University of Southern California, Los Angeles, California, USA*
[2] *Tencent Media Lab, Palo Alto, California, USA*

ABSTRACT

Surface reconstruction from point cloud scans is crucial in 3D vision and graphics. Recent approaches focus on training deep-learning (DL) models to generate representations through learned priors. These models use neural networks to map point clouds into compact representations and then decode these latent representations into signed distance functions (SDFs). Such methods rely on heavy supervision and incur high computational costs. Moreover, they lack interpretability regarding how the encoded representations influence the resulting surfaces. This work proposes a computationally efficient and mathematically transparent Green Learning (GL) solution. We name it the lightweight point-cloud surface reconstruction (LPSR) method. LPSR reconstructs surfaces in two steps. First, it progressively generates a sparse voxel representation using a feedforward approach. Second, it decodes the representation into unsigned distance functions (UDFs) based on anisotropic heat diffusion. Experimental results show that LPSR offers competitive performance against state-of-the-art surface reconstruction methods on the FAMOUS, ABC, and Thingi10K datasets at modest model complexity.

*Corresponding author: Qingyang Zhou, email: qzhou776@usc.edu.

Received 05 September 2024; revised 31 January 2025; accepted 05 February 2025
ISSN 2048-7703; DOI 10.1561/116.20240069

Keywords: Surface reconstruction, point clouds, shape modeling, green learning

1 Introduction

Surface reconstruction from point cloud scans is critical in 3D vision and graphics. It finds applications in augmented and virtual reality (AR/VR), cultural heritage preservation, building information modeling (BIM), etc. Surface reconstruction has recently gained attention in 3D AI-generated content (AIGC). One inherent challenge in surface reconstruction from point clouds stems from the ill-posed nature of reconstructing continuous surfaces from discrete points. The fact that infinitely many possible surfaces can pass through the input points makes it an open problem with no universally optimal solution.

Early research in this field [6, 7, 13, 16, 21, 28, 29, 48] employed unsupervised solutions, where prior knowledge and assumptions (e.g., smoothness, consistent normal directions) were heuristically designed to approximate an optimal solution. Their reconstruction quality is somehow limited.

More recent efforts leverage supervision by pairing point clouds and their surfaces, allowing surface reconstruction models to gain prior knowledge via supervised learning. They use deep neural networks to map point clouds into compact representations and then decode these latent representations into signed distance functions (SDFs). The supervised deep-learning (DL) methodology has become the dominant one nowadays [3, 14, 18, 35, 36, 38, 39, 41, 42, 44, 45, 47]. DL models have the "encoder" and the "decoder" modules applied to those 3D shape modeling or generative tasks. The former transforms point clouds into latent vectors while the latter converts latent vectors to output iso-surfaces [51]. Despite the superior performance of supervised models over unsupervised ones, several challenges remain unresolved.

The first one is the interpretability of supervised DL models. Point clouds are encoded into high-dimensional latent spaces and decoded into a signed distance field (SDF) using convolutional neural networks (CNNs) or transformer layers. The latent representations are difficult to explain, and the contributions of various layers are unclear due to the black-box nature of DL models. Lack of interpretability may lead to reconstruction failures and hinder effective analysis of the model's generalizability. Moreover, it complicates the conditioning process in 3D generation and reconstruction. To address this shortcoming, we propose an explainable method that yields explainable results step by step. The second one is the training and inference complexity and model sizes. Although GPUs can accelerate inference, DL models demand significant memory and computational resources. Furthermore, they need a large amount of training data. This is a concern since the sample numbers in 3D object datasets are much smaller than those in 2D image datasets (e.g., 50K samples in ShapeNet[8] vs. 1.4M images in ImageNet[12]).

To tackle interpretability and complexity, we propose a lightweight point cloud surface reconstruction (LPSR) method in this work. It extends our previous work, GPSR [59]. LPSR enhances the heat diffusion module in GPSR and employs a supervised learning paradigm to boost the performance. LPSR features a feedforward-designed encoder and a PDE-based decoder with mathematical transparency. The encoded representation has a clear physical meaning in controlling the surface shape, and it can be decoded at different grid resolutions to produce different levels of detail.

The main contributions of this work are summarized below.

- We propose a lightweight supervised learning pipeline that progressively constructs voxel representations across multi-resolutions. Feature derivation, selection, and decision-making are all implemented in a feedforward and statistical approach.
- We design a decoder that leverages geometric priors and adopts anisotropic heat diffusion to generate surfaces from representations with modest memory in an unsupervised manner. It accurately maps the representation to the 3D distance field.
- We conduct experiments on two 3D object datasets to demonstrate that LPSR achieves competitive performance against state-of-the-art (SOTA) surface reconstruction methods with lower memory consumption and at a smaller model size.

2 Related Work

Surface reconstruction methods can be classified into two categories: unsupervised and supervised. We provide a brief review of them below.

2.1 Unsupervised Surface Reconstruction Methods

Early unsupervised methods utilized combinatorial techniques, such as Delaunay triangulation or Voronoi diagrams, to directly infer point connectivity. Notable examples include ball-pivoting [4] and power crust [1]. While these methods are computationally less complex, they often struggle with accuracy and cannot guarantee watertight surfaces.

On the other hand, implicit function methods solve an underdetermined partial differential equation (PDE) system, where the solution represents the target implicit surface in the 3D space. To mitigate the ill-posedness of the system, geometric priors or constraints such as point normals and surface smoothness are typically required. A prominent example is the Screened Poisson method [28, 29]. It formulates the Poisson equation by enforcing consistency between SDF gradients and point normals. This method is attractive due to its accuracy and relatively low computational cost. However, it is sensitive to input data noise and point normals' accuracy. To address them, the iPSR method [22] enhances the Poisson surface reconstruction process by iteratively updating oriented normals and refining the surface reconstruction at each iteration. Xiao *et al.* [48] improve the orientation of normals by incorporating an iso-value constraint in the Poisson equations. The PGR method [33] eliminates the need for normals, treats surface normals and element areas as unknown parameters, and optimizes these parameters to achieve better surface representation.

Unsupervised surface reconstruction methods rely on heuristic assumptions to approximate optimal solutions. They ignore specific requirements of certain scenarios (e.g., objects, indoor scenes, outdoor environments, and human faces). Therefore, their performance is generally inferior to that of supervised methods.

2.2 Supervised Surface Reconstruction Methods

Supervised surface reconstruction methods utilize machine learning models to yield the UDF/SDF or the occupancy grid. We examine them from two angles below.

3D Representations. Since 3D UDF/SDF/occupancy grids can be redundant and less informative, alternative 3D representations have been introduced to create more compact and informative ones. One approach solves the problem in the point domain [5, 14, 37, 2]. For example, Points2Surf [14] employs a PointNet-based network to predict SDF values in the point domain. Similarly, DeepSDF [37] utilizes an auto-decoder to optimize randomly initialized latent vectors at each point. While solving the problem in the point domain is memory efficient, the positional information computation can be expensive. Sparse voxel representations, widely adopted by SOTA methods [23, 24, 39, 50], store the positional information in voxel grids. This strategy alleviates the burden on decoders but has relatively high memory consumption. Triplanes [43] and similar projective representations [34] introduce a novel representation that mitigates the memory concerns associated with voxel grids. [43] utilizes a multilayer perceptron (MLP) to decode 3D grids from projected feature planes, though its interpretability may be limited. Lastly, several methods focus on predicting the UDF [41, 57, 55, 11, 15], which then requires conversion to SDF or direct extraction of an non-watertight surface.

Architectural Issues. Early supervised methods adopted an end-to-end optimized neural network [14, 37] to generate point-wise or voxel-wise SDFs. However, these approaches lack interpretability, exhibit poor cross-dataset performance, and frequently result in failure cases. GeoUDF [41] decomposes the problem into a sequence of modules to enhance interpretability. Huang *et al.* [23, 24] replace B-spline bases with learned neural kernel bases in the screened Poisson method, which achieves SOTA performance with partial explainability. These methods rely on deep neural networks to predict various meta-information, such as normals and octree structures. Another method is based on iterative online training including Neural-Pull [3], CAP-UDF [57, 55], and Neural-IMLS [46]. Neural-Pull [3] minimizes the distance between query points and predicted SDF or UDFs. These methods do not rely on training sets from external data, thus avoiding potential biases. However, online training can be computationally demanding and carries the risk of producing non-converged results.

Despite their notable performance on specific test sets, DL-based methods demand substantial computational resources and face interpretability issues. Unexpected failures may occur when they are applied to unseen data. Consequently, there is a need for a transparent and lightweight surface reconstruction method.

2.3 Green Learning

This work adopts the Green Learning (GL) paradigm, initially proposed by Kuo *et al.* [31, 32], to mitigate computational efficiency and interpretability concerns. Green Learning has emerged as an eco-conscious approach within machine learning, emphasizing efficiency and a reduced carbon footprint during model operations. Several core features distinguish this paradigm. It promotes the development of models that are compact in size and have low computational complexity during both training and inference phases. Furthermore, GL is founded on a modularized design principle that enhances mathematical transparency and theoretical explainability.

Specifically, GL reduces the training costs of backpropagation by adopting a purely feedforward training scheme. It employs a modular design that decomposes the whole machine-learning problem into manageable sub-problems, each solved by a transparent learning model. This approach reduces the model size and training/inference complexity and facilitates a theoretically interpretable process across various applications.

Green learning has been applied to a range of point cloud processing tasks. Notable models like PointHop [54], PointHop++ [53], R-PointHop [25], GISP [52], PCRP [27], SPA [26] have proven effective in handling various point cloud processing challenges, including 3D classification, registration, semantic segmentation, and retrieval.

3 Proposed LPSR Method

The proposed Learning-based Point Surface Reconstruction (LPSR) method, as illustrated in Figure 1, employs a sequential modular approach to transform an input point cloud into an unsigned distance field (UDF). It consists of four cascaded modules, as depicted in Figure 1. They are: 1) Point Feature Projection, 2) Sparse Green U-Shaped Learning (GUSL), 3) PDE-based UDF Solution, and 4) Post Processing. In the first module, a point feature projection operator is employed to extract point features, which are voxelized through voxel-wise aggregation. The second module progressively learns the representation in a feedforward manner. The representation output from Module 2 is used as the input to Module 3, which solves a partial differential equation (PDE) for the UDF. Finally, after the sign assignment, Module 4 uses a marching-cubes algorithm to obtain the distance field. For clarity and logical progression in explanation, the methodological description begins with the Point Feature Projection module, followed by the detailed exposition of the PDE-based UDF decoder module. The Sparse GUSL framework is discussed further in Section 3.3, and the Post Processing steps are elaborated in Section 3.4.

3.1 Point Feature Projection

In this module, the input point clouds are voxelized and yield voxel features in an unsupervised process. As discussed in Section 2.2, a voxel grid is favored among various representations because it contains positional information, and no positional decoder is needed. We employ a sparse voxel grid to avoid excessive memory consumption, considering only occupied voxels.

To convert point features to voxel features, we voxelize an input point cloud into a 3d grid of size $d \times d \times d$. Within each voxel cube, we recenter the points using the voxel geometry center. The recentered points inside the voxel can be denoted as $\{p_1, p_2, \ldots, p_n\} \subseteq \mathbb{R}^3$. To align the dimension across different voxels, we introduce a PoinNet-like structure [40]. A recentered point p_i is projected into a high-dimensional feature f_i using a shared weights W and bias b:

$$f_i = \sigma(Wp_i + b), \tag{1}$$

where $\sigma(\cdot)$ is a non-linear activation function, specifically ReLU. Then, a maximum aggregation function produces a voxel feature F by:

$$F = max(\{f_1, f_2, \ldots, f_n\}), \tag{2}$$

The major difference between our method and PointNet is that the weights W and biases b in our module are purely based on random initialization and do not require supervision for adjustment. We choose a large feature dimension and rely on a feature selection module to identify powerful features, as discussed in Section 3.2.

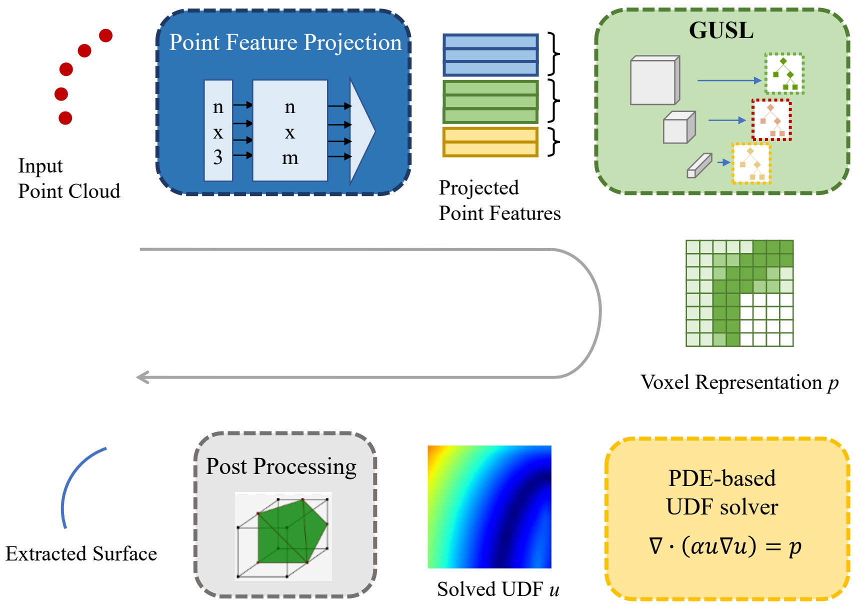


Figure 1: The four modules of the LPSR method: 1) Point Feature Extraction, 2) Sparse GUSL, 3) PDE-based UDF Solver and 4) Post Processing. Modules 1 and 2 can be viewed as the encoding process that maps a point cloud scan to a representation, denoted by p, while modules 3 and 4 decode the representation, p, to the reconstructed surface.

3.2 Heat-equation-based Surface Decoder

Motivation. The motivation behind applying a PDE-based decoder is to parameterize the UDF/occupancy field into an explainable feature space, thereby avoiding the black box decoding processes typical of MLPs or other decoding layers. This idea is shared by a group of supervised works [23, 24]. Considering the SDF/UDF is continuous in space and monotonic in empty regions, one can parameterize it to some compact representations. Differ from [24] using popular B-spline or Fourier bases to represent the SDF, we opt to use a temperature grid to mimic the UDF and control the grid by parameterizing the boundary conditions (e.g., temperature or heat flux). The advantages of using a heat-equation solution as the UDF are outlined below:

- The steady-state solution of the heat equation forms a smooth and continuous 3D grid, supporting re-scaling using the finite element method (FEM).
- Temperature values in empty voxels follow a monotonic change, with values falling between the lowest and highest temperature ranges.

- The temperature is controlled by the magnitude and spatial position of the heat source / flux, which is inherently differentiable.

This approach has proven efficient in our previous work, GPSR [59].

Implementation Details. Given a discrete 3D grid, our goal is to establish a heat flux or temperature control function $q(\mathbf{x})$ and seek the steady state solution $u(\mathbf{x})$ to the heat equation, Eq. (3), which will serve as the UDF:

$$\frac{\partial u(\mathbf{x})}{\partial t} - \nabla \cdot (D \nabla u(\mathbf{x})) = q(\mathbf{x}), \tag{3}$$

where D is the diffusion coefficient, $\frac{\partial u(\mathbf{x})}{\partial t}$ is set to 0 in the steady state when $u(\mathbf{x})$ does not change w.r.t. to time t.

Unlike our previous GPSR method, which treats the diffusion coefficient D as a constant value, we set the D as a dependent variable of u by

$$D = \alpha u, \tag{4}$$

where α is a scalar parameter. This modification leads to the establishment of a more accurate steady-state equation:

$$-\nabla \cdot (\alpha u(\mathbf{x}) \nabla u(\mathbf{x})) = q(\mathbf{x}), \tag{5}$$

which offers a more precise solution that converges to the UDF. We then discretize Eq. (5) into a matrix form:

$$\mathbf{G}_x(\alpha \mathbf{u} \odot (\mathbf{G}_x \mathbf{u})) + \mathbf{G}_y(\alpha \mathbf{u} \odot (\mathbf{G}_y \mathbf{u})) + \mathbf{G}_z(\alpha \mathbf{u} \odot (\mathbf{G}_z \mathbf{u})) = \mathbf{q}, \tag{6}$$

where $\odot$ denotes the Hadamard product. $\mathbf{G}_x, \mathbf{G}_y, \mathbf{G}_z$ are matrix form of discrete gradient operator along x, y, z directions, respectively. The solution $\mathbf{u}$ is the flattened UDF vector in the 3D grid. The $\mathbf{q}$ is the vector representing the heat source/flux controlling the shape of the UDF.

The following two constraints must be satisfied to guarantee a converged steady-state solution.

- The boundary condition is set to Neumann boundary condition with zero heat flux, indicating no heat exchange with the outside environment at the boundary.
- The vector $\mathbf{q}$ has a zero-sum, indicating that the net heat flux entering and leaving the system is zero.

Considering the non-linear parabolic PDE, Eq. (6), cannot be directly solved, we treat it as an optimization problem and use the gradient descent method to solve it iteratively. The initial guess for optimization is the q_{pred} obtained from the XGBoost regressor. We use the iterative approach to solve the PDE for u_{pred}, the output UDF.

3.3 Sparse Green U-Shaped Learning (GUSL) Model

The GUSL model is the critical module that maps the aggregated point features into a compact representation using the Green Learning paradigm. As shown in Figure 2, it follows a U-shape pipeline and progressively yields representations at different grid resolutions. The framework comprises three modules for each resolution: unsupervised Saab feature construction, semi-supervised feature selection, and supervised decision learning. We will introduce these modules in the following.

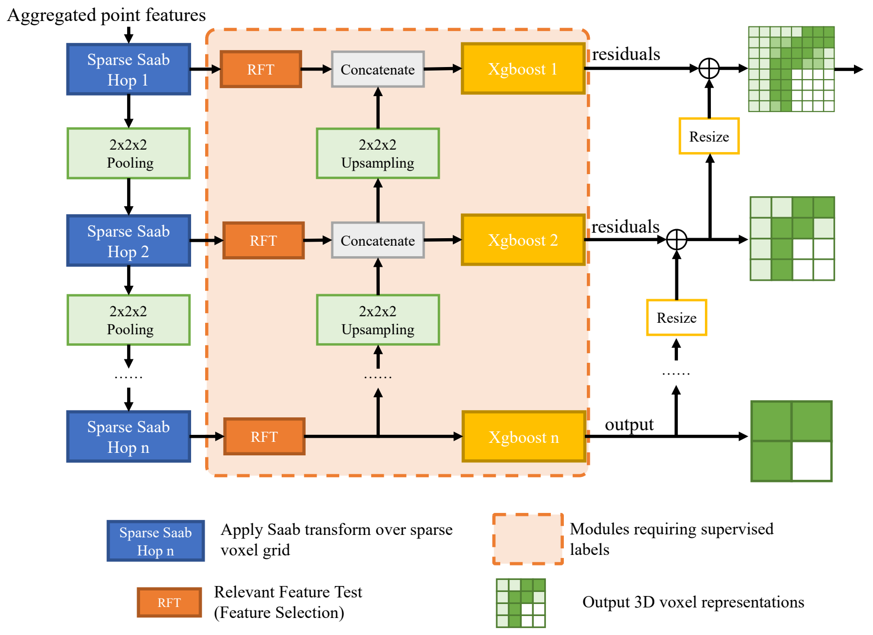


Figure 2: The pipeline of sparse GUSL, a multi-grid representation learning process. Note that the Saab Hop module and the 2x2x2 pooling/upsampling operations are based on sparse voxel operations, where empty voxels are not processed.

1) Sparse Saab Transform. To yield decorrelated representation features from the input voxel features, a channel-wise (c/w) Saab Transform [10, 32] is applied on each local voxel neighborhood to derive its new representation in terms of Saab coefficients in an unsupervised manner. Given the sparse nature of the 3D voxel representation, we only apply the c/w-Saab transform over the occupied neighborhood, following a similar scheme of sparse convolution [17]. Using the n-hop c/w-Saab transform layers increase the spectrum resolution to capture spatial correlations. The 3-layer features are sent to the multi-grid feature selection and decision learning module.

2) Feature Selection. Reducing the input feature dimension is essential to achieve a lightweight model. Therefore, we introduce a semi-supervised feature selection technique called the Relevant Feature Test (RFT) [49]. As shown in Figure 3, for a given 1D input feature, samples are ordered by their feature values, and the sample maximum and minimum bind the feature dimension. The representation is then partitioned into two sub-intervals at a set of uniformly spaced points between the maximum and minimum. We compute the mean values of the samples in the left and right sub-intervals, find the point that minimizes the weighted mean-squared error (MSE), and define this MSE value as the RFT cost for that representation. The RFT cost indicates the effectiveness of the representation in reducing regression errors. We select the top k features with the most discriminant power and discard the rest. "semi-supervised" means only a small subset of labels among the training samples are required in the feature selection process.

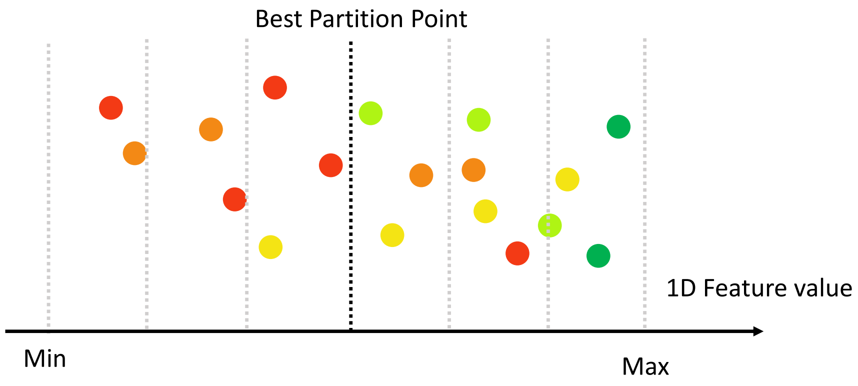


Figure 3: Visualization of the Relevant Feature Test (RFT). Dots of different colors represent samples with different labels. The partition metric is the label values' weighted mean-squared error (MSE).

One challenge with incorporating supervision in our method is the absence of backpropagation, which prevents traditional gradient-based learning. We employ an indirect labeling approach to address this by utilizing outputs from the PDE-based Module 3. Specifically, we pass the ground truth UDF u_{GT} pass through Eq. (5) introduced in Section 3.3 under different grid resolutions. In Eq. (5), α is a scalar coefficient, $u(\mathbf{x})$ represent the UDF at position $\mathbf{x}$ and $q(\mathbf{x})$ denotes the resulting representation. This process yields a set of ground truth representation labels $\{q_{GT}^{n}, q_{GT}^{n-1}, , ...q_{GT}^{2}, q_{GT}^{1}\}$ corresponding to various resolutions. These labels are then used to effectively guide the feature selection process without relying on backpropagation.

3) Decision Learning. The selected features predict each voxel's representations q_{pred}. These predictions are made using a group of progressive XGBoost regressors [9]. The XGBoost model at the coarsest grid level (Hop n) produces the coarse prediction $q^{(n)}_{pred}$ for the representation voxel, while those at finer grid levels predict the corresponding residuals $\Delta q^{(n-1)}_{pred}, ..., \Delta q^{(2)}_{pred}, \Delta q^{(1)}_{pred}$ between the ground truth and the previous predictions.

As discussed in the previous paragraph, directly inferring the u_{pred} from p_{pred} involves solving a non-linear PDE in each training iteration, which can be computationally expensive. To address this, we define an indirect loss for the representation term q:

$$\mathcal{L}_q = \left\| q_{pred}(\mathbf{x}) - q_{UDF}(\mathbf{x}) \right\|_2 . \tag{7}$$

In addition, when the q_{pred} is close to the ground truth, a correction term is required to respect the PDE. In this case, we solve the PDE and define a loss for the PDE solution u_{pred}:

$$\mathcal{L}_{UDF} = \left\| u_{pred}(\mathbf{x}) - u_{UDF}(\mathbf{x}) \right\|_2 , \tag{8}$$

where u_{pred} is the solution to the heat propagation equation, Eq. (3), given representation q. The overall learning objective function can be expressed as:

$$\mathcal{L} = \mathcal{L}_q + \lambda \mathcal{L}_{UDF}. \tag{9}$$

The parameter, λ, is initially set to 0 in the training and gradually increases to a preset value as the iteration progresses.

3.4 Post Processing

In this module, we aim to convert the unsigned distance field into a surface by assigning a sign to the UDF u_{pred} and applying a marching cube to the recovered SDF s_{pred}. To infer the sign of the predicted distance field u_{pred}, we follow the approach used in GPSR [59]. As shown in Figure 4, the process of converting UDF to SDF involves a forward heat propagation process across a 3D temperature grid T through the following steps:

- Step 1: Identify all voxels intersected by the surface in a unit 3D space with resolution $d \times d \times d$. A voxel is considered occupied (intersected by the surface) if its predicted unsigned distance is less than $0.5 * s$, where $s = 1/d$.

- Step 2: Compute the unit-length normal direction vector $\mathbf{n}$ for each occupied voxel by determining the direction of the steepest gradient in UDF.

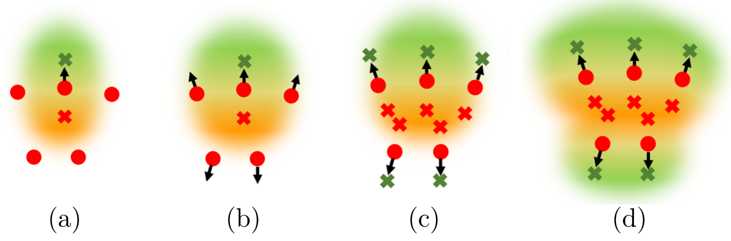


Figure 4: Illustration of the sign alignment in the post-processing step, where red dots denote the geometry center of each occupied voxel, while red and green crosses denote new heat sources and sinks in $\mathbf{P}$ and $\mathbf{N}$, respectively. The sign of occupied voxels is aligned in the heat propagation process.

- Step 3: Arbitrarily select an occupied voxel $\mathbf{p}_i = (x_i, y_i, z_i)$ with normal direction $\mathbf{n}_i$ as the starting point. The sign of $\mathbf{n}_i$ can be arbitrarily assigned.
- Step 4: Compute 2 candidate locations $\mathbf{l1}_i$ and $\mathbf{l2}_i$ for each point $\mathbf{p}_i$ as follows:
$$\mathbf{l1}_i = round(\mathbf{p}_i + \mathbf{n}_i) \tag{10}$$
$$\mathbf{l2}_i = round(\mathbf{p}_i - \mathbf{n}_i) \tag{11}$$
Append $\mathbf{l1}_i$, $\mathbf{l2}_i$ to two sets $\mathbf{P}$ and $\mathbf{N}$, respectively.
- Step 5: Designate all locations in $\mathbf{P}$ and $\mathbf{N}$ as constant heat sources with temperature +1 and sinks with temperature -1, respectively. Perform a forward heat diffusion iteration over T, as demonstrated in Figure 4 (a).
- Step 6: For each adjacent occupied voxel $\mathbf{p}_j$, append $\mathbf{l1}_j$ to $\mathbf{P}$ if $T(\mathbf{l1}_j) > 0$, and to $\mathbf{N}$ if $T(\mathbf{l1}_j) < 0$; apply the same criteria to $\mathbf{l2}_j$. This operation is demonstrated in Figure 4 (b).
- Step 7: Repeat Step 5, cycling through each voxel until all occupied voxels are processed, as shown in Figures 4 (c) and (d).

Following the propagation of all occupied voxels, the signs from the temperature field T are extracted. The SDF s_{pred} is calculated as follows:

$$s_{pred} = Sign(T) \odot u_{pred}, \tag{12}$$

where $\odot$ denotes the element-wise product. Finally, the marching cubes algorithm is applied to s_{pred} to extract the surface.

Our surface extraction method shares similarities with MeshUDF [20] in determining the sign of the distance field. However, there are distinct differences between the two approaches. MeshUDF utilizes the projection of neighboring gradient vectors to compute the vote values for sign determination. While our method is a more straightforward approach by assigning preset temperatures to specific locations and utilizing a heat transmission process to infer values throughout the grid.

4 Experiments

4.1 Experimental Settings

We compare the performance of various surface reconstruction methods on three datasets: FAMOUS [14], ABC [30], and Thingi10K [58]. They have various types of degradation, including missing parts and different noise levels. The input point cloud sets are categorized into three types: noise-free, original, and extra-noise, with the variance of noise set to $0.00L$, $0.01L$, and $0.05L$, respectively. The L denotes the largest side of the mesh bounding box. The model is trained over the ABC [30] dataset following a similar setting to [14].

We compare our LPSR method with several other surface reconstruction methods, including Screened Poisson Reconstruction (SPR) [29], PGR [33], Neural-Pull [3], Points2surf [14], PCPNet [19], POCO [5], NKSR [24], GeoUDF [41], CAP-UDF [55], and LevelSetUDF [56]. SPR and PGR are unsupervised, non-data-driven methods that do not incorporate learning modules, while the remaining seven are powered by learning-based approaches. It is worth noting that PCPNet generates oriented point normals as outputs, and SPR is used to yield its surface for fair comparison. SPR+PCPNet denotes the resulting method. Experimental results for different point set categories are reported by following the configuration settings specified by each method's papers for consistency and comparability.

The reconstruction quality is measured by the similarity between the ground truth mesh, $Mesh_{GT}$, and the reconstructed mesh surface, $Mesh_{rec}$. We normalize both meshes into unit size and uniformly sample 10,000 points to yield point cloud sets, PC_{GT} and PC_{rec}. Following existing works [33, 14], we use the symmetric Chamfer distance (CD) and the Hausdorff distance (HD) between PC_{GT} and PC_{rec}, and the symmetric mesh cosine similarity (CS) between $Mesh_{GT}$ and $Mesh_{rec}$ as three quality metrics. For a comprehensive assessment of the reconstruction quality, F-score (F-S.) and normal consistency (N.C.) are used. Following the existing work of [24], the F-score adopt a threshold of 1% for evaluation.

We normalize all input point clouds to a unit scale within the range [0, 1]. The point clouds are then voxelized with a resolution of 256x256x256. The point feature dimension is set to 48. A three-hop sparse GUSL module is applied, with each Saab transform conducted over a window size of 3x3x3. Pooling and upsampling operations are set with a stride of 2. The feature selection module ensures that the input features to each XGBoost model are reduced to $k = 256$.

4.2 Quantitative Comparison of Reconstructed Surface Quality

The experimental results for the FAMOUS [14], ABC [30], and Thingi10K [58] datasets are presented in Tables 1, 2, and 3, respectively. The best and second-best results are highlighted in bold and underlined, respectively. Additionally, to provide a comparative analysis of several leading methods, the F-score and normal consistency across these three datasets are detailed in Tables 4, 5, and 6, respectively.

Table 1: Comparison of Reconstructed Surface Quality Metrics for the FAMOUS Dataset.

Method	Noisefree			Original			ExtraNoisy		
	CD↓	HD↓	CS↑	CD↓	HD↓	CS↑	CD↓	HD↓	CS↑
PGR [33]	1.470	0.212	0.850	2.711	0.349	0.386	5.626	0.358	0.662
GPSR [59]	1.530	0.211	0.867	1.922	0.244	0.791	4.047	0.319	0.688
Points2surf [14]	1.633	0.218	0.822	1.750	**0.235**	0.793	**3.103**	0.332	0.585
SR+PCPNet [19]	1.746	0.258	0.830	1.947	0.263	0.782	4.005	0.327	0.540
Neural-Pull [3]	2.091	0.264	0.860	2.200	0.250	0.764	5.275	0.376	0.245
POCO [5]	**1.403**	0.223	0.869	**1.734**	0.252	**0.818**	3.213	0.323	0.638
NKSR [24]	1.684	**0.206**	0.871	2.151	0.236	0.633	3.490	**0.301**	**0.694**
Geo-UDF [41]	**1.403**	0.207	0.023	2.370	0.239	0.026	6.274	0.407	0.042
CAP-UDF [55]	1.456	0.219	0.027	2.367	0.240	0.028	5.738	0.373	0.036
LevelSetUDF [56]	1.433	0.214	0.023	2.318	0.242	0.029	6.253	0.393	0.036
LPSR (Ours)	1.511	0.209	**0.872**	1.832	0.237	0.809	3.546	0.311	**0.694**

Table 2: Comparison of Reconstructed Surface Quality Metrics for the ABC dataset.

Method	Noisefree			Original			ExtraNoisy		
	CD↓	HD↓	CS↑	CD↓	HD↓	CS↑	CD↓	HD↓	CS↑
PGR [33]	2.014	0.242	0.816	3.549	0.319	0.331	5.434	0.357	0.609
GPSR [59]	1.932	0.267	0.849	3.225	0.299	0.693	5.579	0.367	0.600
Points2surf [14]	2.391	0.255	0.842	2.267	0.271	0.713	5.178	0.351	0.450
SR+PCPNet [19]	3.530	0.331	0.708	3.703	0.341	0.597	5.791	0.367	0.381
Neural-Pull [3]	3.431	0.296	0.788	4.891	0.342	0.582	8.898	0.422	0.262
POCO[5]	1.854	0.238	**0.860**	2.119	**0.257**	**0.805**	**3.511**	**0.303**	0.656
NKSR[24]	2.197	0.257	0.615	5.557	0.371	0.313	16.187	0.591	0.077
Geo-UDF [41]	**1.836**	0.237	0.022	3.721	0.293	0.039	11.13	0.461	0.092
CAP-UDF [55]	2.116	0.263	0.041	3.614	0.294	0.033	9.850	0.423	0.067
LevelSetUDF [56]	1.991	0.261	0.036	3.535	0.293	0.035	9.628	0.418	0.073
LPSR (Ours)	1.881	**0.232**	0.855	**2.111**	0.260	0.802	4.047	0.319	**0.688**

Table 3: Comparison of Reconstructed Surface Quality Metrics for the Thingi10K dataset.

Method	Noisefree			Original			ExtraNoisy		
	CD↓	HD↓	CS↑	CD↓	HD↓	CS↑	CD↓	HD↓	CS↑
PGR [33]	1.457	**0.192**	**0.918**	3.055	0.352	0.402	5.672	0.355	**0.695**
GPSR[59]	1.543	0.218	0.893	2.521	0.244	0.819	4.477	0.361	0.581
Points2surf [14]	1.552	0.200	0.884	**1.689**	**0.214**	0.860	3.601	0.342	0.603
SR+PCPNet [19]	2.462	0.270	0.828	2.602	0.271	0.780	4.937	0.336	0.533
Neural-Pull [3]	2.115	0.222	0.878	2.486	0.245	0.832	6.737	0.395	0.253
POCO[5]	1.579	0.224	0.909	1.839	0.232	**0.880**	**3.611**	**0.324**	0.679
NKSR [24]	1.911	0.221	0.679	2.454	0.230	0.603	12.991	0.564	0.103
Geo-UDF [41]	**1.394**	0.200	0.027	2.709	0.239	0.056	7.454	0.429	0.068
CAP-UDF [55]	1.507	0.208	0.039	3.021	0.241	0.046	7.483	0.402	0.055
LevelSetUDF [56]	1.517	0.211	0.032	2.874	0.231	0.043	8.200	0.415	0.052
LPSR (Ours)	1.503	0.211	0.902	1.869	0.240	0.825	4.047	0.349	0.604

Table 4: Comparison of F-score (F-S.) and Normal Consistency (N.C.) for the FAMOUS Dataset.

Method	Noisefree		Original		ExtraNoisy	
	F-S.↑	N.C.↑	F-S.↑	N.C.↑	F-S.↑	N.C.↑
PGR [33]	0.964	0.904	0.644	0.702	0.277	0.746
Neural-Pull [3]	0.759	0.753	0.763	0.749	0.280	0.568
POCO [5]	0.963	0.913	**0.871**	**0.871**	**0.505**	**0.766**
NKSR [24]	0.861	0.891	0.613	0.690	0.249	0.546
Geo-UDF [41]	0.981	0.918	0.682	0.720	0.265	0.575
CAP-UDF [55]	**0.974**	0.924	0.674	0.681	0.281	0.548
LevelSetUDF [56]	**0.974**	**0.931**	0.690	0.699	0.252	0.553
LPSR (Ours)	0.963	0.910	0.814	0.835	0.487	0.751

Table 5: Comparison of F-score (F-S.) and Normal Consistency (N.C.) for the ABC Dataset.

Method	Noisefree		Original		ExtraNoisy	
	F-S.↑	N.C.↑	F-S.↑	N.C.↑	F-S.↑	N.C.↑
PGR [33]	0.836	0.916	0.543	0.677	0.270	0.793
Neural-Pull [3]	0.740	0.793	0.531	0.716	0.192	0.611
POCO [5]	0.882	0.953	0.757	0.883	**0.391**	**0.821**
NKSR [24]	0.766	0.897	0.520	0.662	0.159	0.547
Geo-UDF [41]	**0.956**	**0.970**	0.537	0.670	0.169	0.577
CAP-UDF [55]	0.916	0.949	0.533	0.662	0.167	0.566
LevelSetUDF [56]	0.927	0.944	0.550	0.704	0.158	0.590
LPSR (Ours)	0.897	0.933	**0.763**	**0.891**	0.352	0.801

For the FAMOUS dataset, LPSR significantly improved the performance of our previous work, GPSR. We also see a reduction in the Chamfer and the Hausdorff distances in all three datasets. This is attributed to the in-

Table 6: Comparison of F-score (F-S.) and Normal Consistency (N.C.) for the Thingi10K Dataset.

Method	Noisefree		Original		ExtraNoisy	
	F-S.↑	N.C.↑	F-S.↑	N.C.↑	F-S.↑	N.C.↑
PGR [33]	0.928	0.952	0.527	0.742	0.244	<u>0.805</u>
Neural-Pull [3]	0.806	0.919	0.591	0.788	0.221	0.589
POCO [5]	0.916	0.941	**0.817**	**0.920**	**0.394**	**0.823**
NKSR [24]	0.768	0.925	0.671	0.873	0.194	0.559
Geo-UDF [41]	**0.975**	<u>0.959</u>	0.546	0.754	0.209	0.586
CAP-UDF [55]	0.966	0.958	0.495	0.704	0.211	0.561
LevelSetUDF [56]	<u>0.972</u>	**0.963**	0.525	0.744	0.185	0.572
LPSR (Ours)	0.910	0.942	<u>0.794</u>	<u>0.889</u>	<u>0.363</u>	<u>0.805</u>

troduced supervision. LPSR employs supervision to control the heat equation, leading to finer distance fields and better handling of various distortions. POCO [5], NKSR [24], Geo-UDF[41], and CAP-UDF[55] exhibit competitive performance in terms of Chamfer distance and Hausdorff distance under noise-free conditions. Among them, POCO excels in robustness against noisy and degraded inputs. On the other hand, it demands a high memory cost since it uses query points to describe the occupancy map. NKSR [24] offers impressive performance on the FAMOUS dataset. However, its accuracy relies on the orientation of input point normals, which can be problematic with noisy inputs in the ABC and Thingi10k datasets.

Regarding UDF-based methods [41, 55, 56], they display excellent performance under noise-free conditions. However, their mesh cosine similarity scores are typically lower than those achieved by other methods. This lower performance can be attributed to the fact that UDF-reconstructed surfaces are generally non-watertight and their surface normals are poorly aligned. These factors significantly impact the mesh cosine similarity of their outputs.

4.3 Complexity Comparison

The complexity is measured over the FAMOUS dataset with noise-free inputs. The model sizes, the computational complexity (in terms of floating operation counts, or FLOPs), and the peak memory requirement are measured by taking the average for the input point clouds. For a fair comparison, the output resolution for most methods are typically set to 256, if possible.

Table 7 compares model sizes, the computational complexity (in terms of floating operation counts, or FLOPs), the peak memory requirement of LPSR, and two unsupervised and seven supervised methods. LPSR exhibits the lowest computational complexity and maintains a notably small model size among all the supervised methods evaluated. Specifically, LPSR's model

Table 7: Comparison of model sizes, computational complexities (in FLOPs), and peak memory requirements.

Method		Model Size	Total FLOPs	Peak Memory	Runtime
Unsupervised	PGR [33]	**<100 B**	-	11815 MB (x88)	7.41 s
	GPSR[59]	**<100 B**	**10 G (x1)**	**134 MB (x1)**	13.20 s
Supervised	Points2surf [14]	22 MB (x3.93)	5301 G (x62)	7805 MB (x8.71)	46.49 s
	SR+PCPNet [19]	87 MB (x15.53)	2688 G (x31)	4967 MB (x5.54)	8.97 s
	Neural-Pull [3]	15 MB (x2.67)	85486 G (x994)	*	362 s
	POCO [5]	133 MB (x23.75)	2370 G (x28)	33276 MB (x37.13)	39.53 s
	NKSR [24]	57 MB (x10.18)	340 G (x4)	3409 MB (x3.80)	**1.70 s**
	Geo-UDF [41]	3.2 MB (x0.57)	690 G (x8.02)	5742 MB (x6.41)	65.63 s
	CAP-UDF [55]	2.0 MB (x0.35)	256491 G (x2982)	*	947 s
	LevelSetUDF [56]	**1.8 MB (x0.32)**	330702 G (x3845)	*	789 s
	LPSR (Ours)	5.6 MB (x1)	**86 G (x1)**	**896 MB (x1)**	21.17 s

*Neural-Pull, CAP-UDF and LevelSetUDF requires online training for each query point cloud. Its peak memory is GPU specific.

size is 23 times smaller and its total FLOPs are 28 times lower than those of the POCO method.

The low FLOP consumption can be attributed to the efficient feature selection module in our sparse GUSL model. The Relevant Feature Test (RFT) technique constrains the feature dimension to a specified size and discards less relevant features, maintaining high performance while reducing the computational load. The small model size results from two key factors. First, RFT effectively constrains the feature space dimension, thereby reducing the size of the decision-learning process. Second, the surface decoder of LPSR is purely PDE-based without any learnable parameters. While GeoUDF and CAP-USF also have smaller model sizes, their total FLOP consumption during inference significantly exceeds that of LPSR. This makes LPSR particularly efficient in terms of both model size and computational requirements.

Regarding memory consumption, LPSR exhibits a lower cost than other supervised models, although it is slightly higher than our previous unsupervised work, GPSR. The low memory cost is due to sparse voxel representation and a multi-grid scheme.

We also compared the runtime of these methods. The self-supervised methods, such as those detailed in [3, 55, 56], exhibit significantly longer runtimes than others due to their reliance on online training for each point cloud scan. This reliance makes them less suitable for real-time tasks. NKSR, on the other hand, has the quickest response time compared to our LPSR and other supervised methods. This difference in speed is primarily because most modules of LPSR are implemented on the CPU, whereas other supervised methods benefit from GPU acceleration on Neural Networks. Consequently, the reported runtimes should be considered as informative rather than definitive indicators of true computational complexity.

4.4 Visual Quality Comparison

In Figures 5, 6, and 7, we show input point cloud scans, ground truth surfaces, and reconstructed surfaces using six methods - SPR+PCPNet, Points2Surf, Neural-Pull, POCO, NKSR, Geo-UDF, CAP-UDF, LevelsetUDF and LPSR (ours) for noise-free, original, and extra noisy categories, respectively. They are used for qualitative comparison of reconstructed surfaces under different settings. Besides, a visual illustration of different resolution outputs of LPSR are shown in Figure 8.

Generally speaking, LPSR produces accurate surfaces with no apparent failures. Although some details may be lost due to noisy inputs, no strange textures or artifacts exist in its reconstructed surfaces. In contrast, DL-based methods such as Neural-Pull and NKSR have partial failures, while Points2Surf exhibits unusual textures. Geo-UDF and CAP-UDF produce high-quality detailed geometry, though they struggle to maintain continuous surfaces in some local regions, resulting in broken pieces and non-manifold surfaces.

The success of LPSR is attributed to the PDE-based UDF decoder, which produces a smooth and monotonic UDF across most regions. In contrast, the DL-based models may decode surfaces from an ambiguous latent space, leading to unexpected local optima and, thus, unpredictable failures.

4.5 Ablation Study

In this section, we assess the effectiveness of each component using a subset of the ABC dataset.

1) Point Feature Projection. The number of projection directions in the point feature projection module significantly impacts model performance. We explored different configurations by varying the number of projection directions with $m = 8, 16, 24, 32, 64, 128$ and report the results in Table 8. We observed that the performance tends to drop with a smaller value m. This indicates that fewer projection directions limit the representation capability. When the number of projections is above 64, the performance saturates. Considering the complexity caused by larger feature dimensions, we selected $m = 48$.

Table 8: Effect of projection dimensions of Point Feature Projection Module.

# of projection directions	8	16	24	32	48	64	128
L2 CD	1.952	1.664	1.573	1.548	1.540	1.551	1.539

2) PDE-based decoder. We assess the effect of our PDE-based decoder by comparing our performance against a Sparse GUSL framework without a PDE-based decoder. The latter directly uses the unsigned distance value of each voxel as the prediction label.

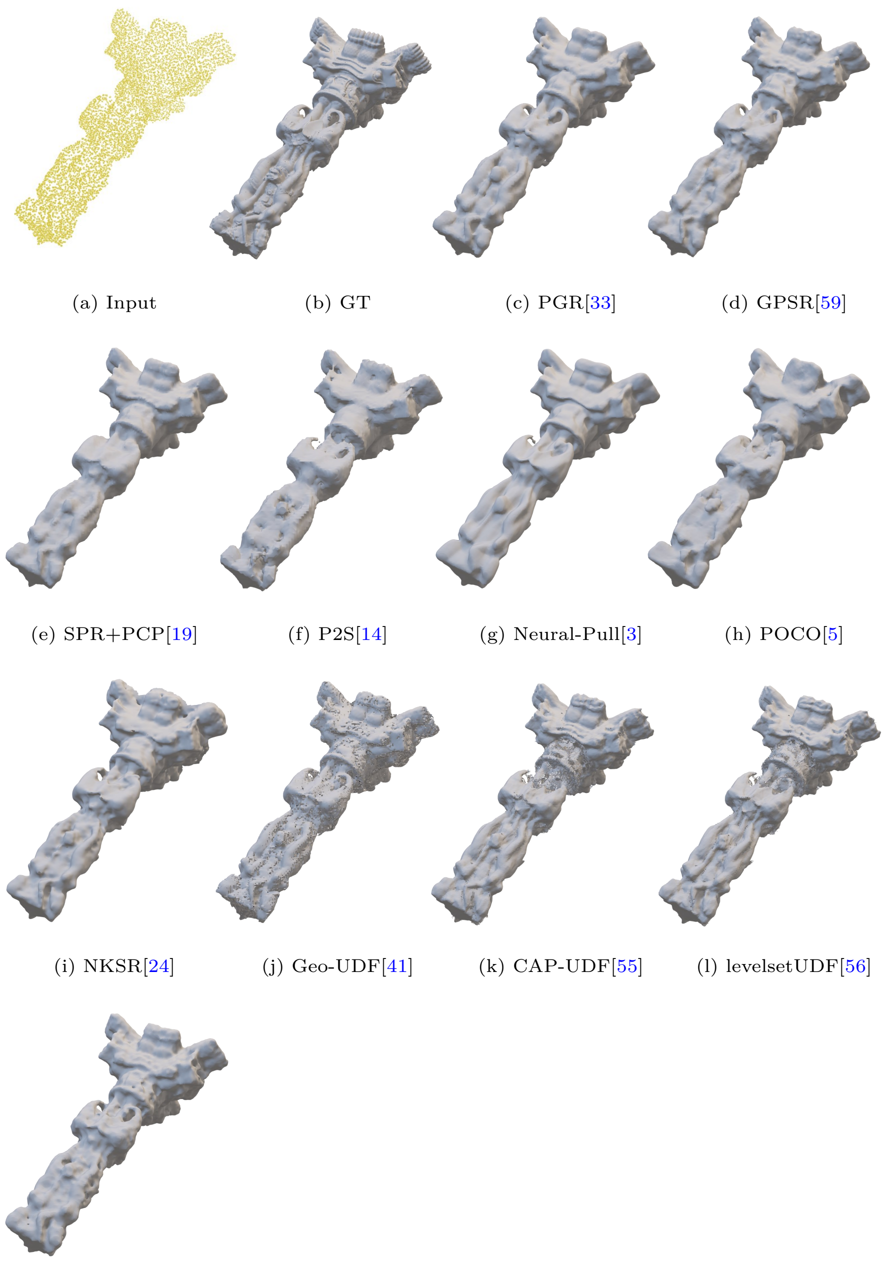
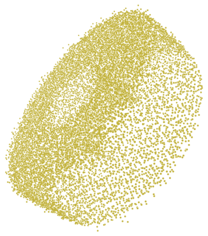
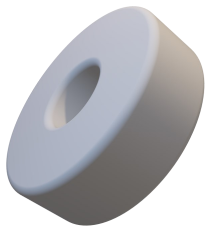
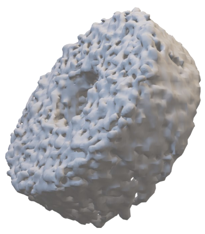
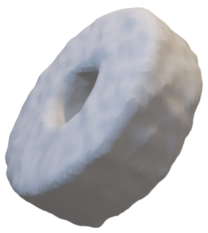


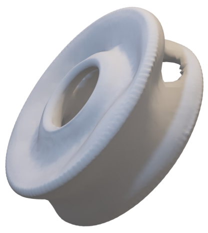
Figure 5: Visualization of reconstructed surfaces obtained by LPSR and ten benching methods, where the input point cloud belongs to the category of noise-free.

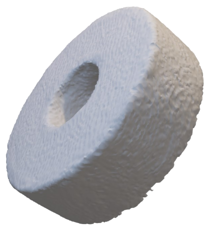

(a) Input

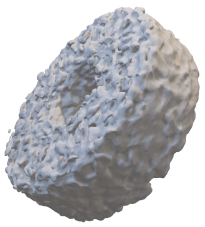

(b) GT

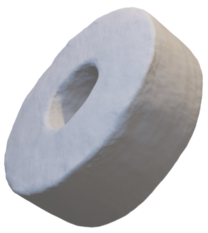

(c) PGR[33]

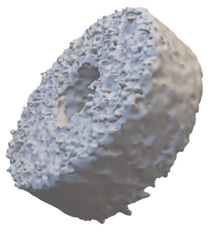

(d) GPSR[59]

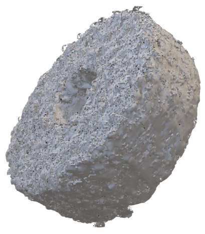

(e) SPR+PCP[19]

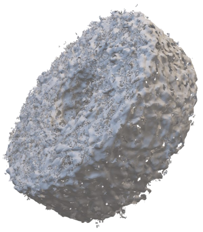

(f) P2S[14]

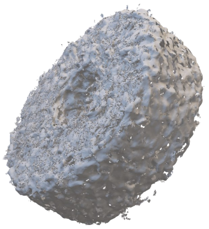

(g) Neural-Pull[3]


(h) POCO[5]


(i) NKSR[24]


(j) Geo-UDF[41]


(k) CAP-UDF[55]


(l) levelsetUDF[56]

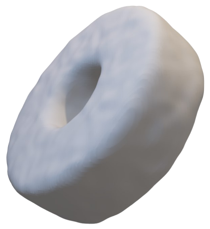
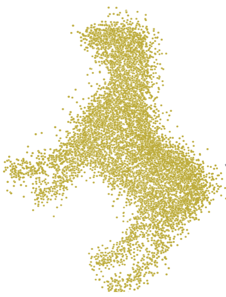
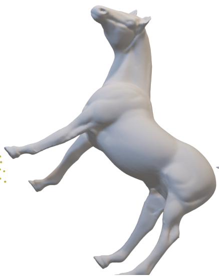
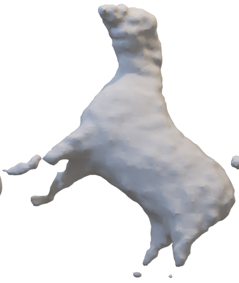
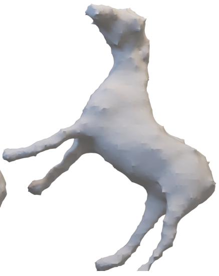

(m) Our LPSR

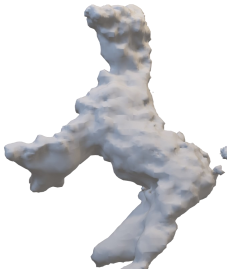
Figure 6: Visualization of reconstructed surfaces obtained by LPSR and ten benching methods, where the input point cloud belongs to the category of original.

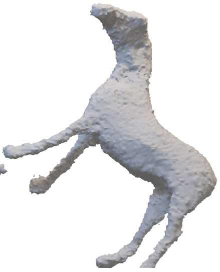

(a) Input

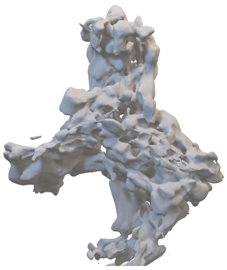

(b) GT

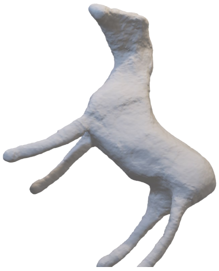

(c) PGR[33]

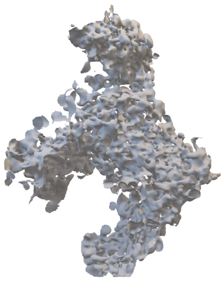

(d) GPSR[59]

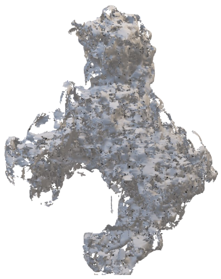

(e) SPR+PCP[19]

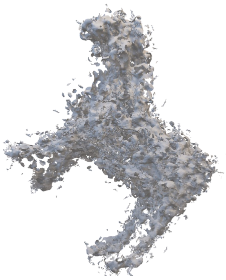

(f) P2S[14]

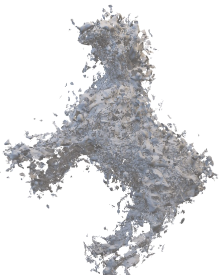

(g) Neural-Pull[3]


(h) POCO[5]


(i) NKSR[24]


(j) Geo-UDF[41]


(k) CAP-UDF[55]


(l) levelsetUDF[56]

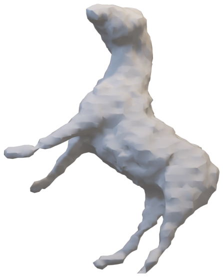

(m) Our LPSR

Figure 7: Visualization of reconstructed surfaces obtained by LPSR and ten benching methods, where the input point cloud belongs to the category of extra noisy.

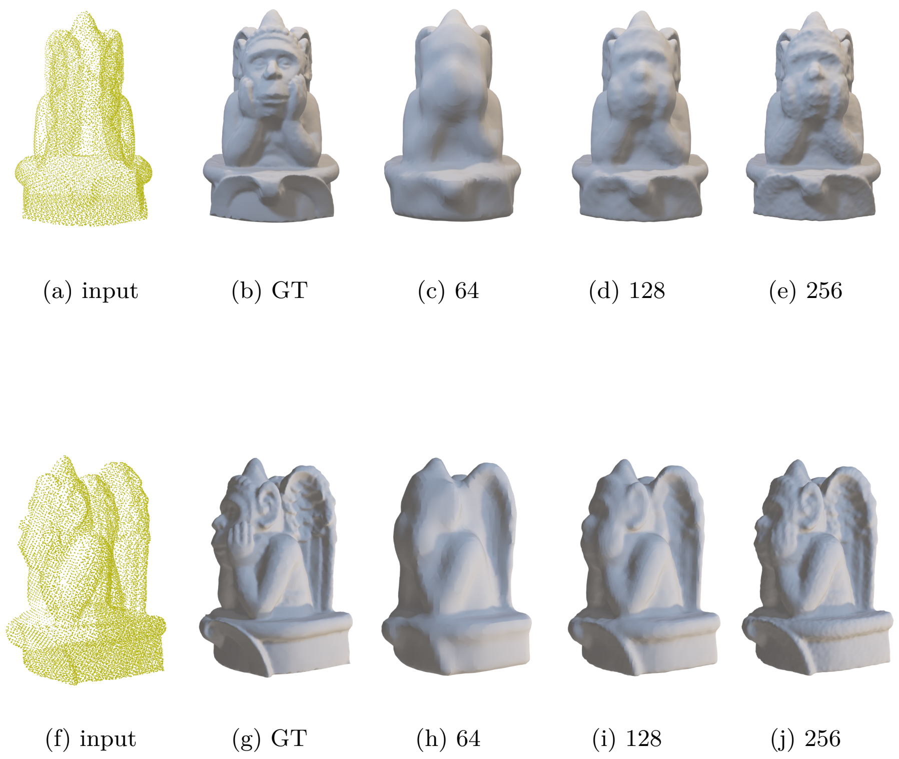


Figure 8: Visualization of reconstructed noisefree surfaces at grid resolutions of 64, 128, and 256. In our method, details are progressively added to the surface in a coarse-to-fine manner.

Performance results are detailed in Table 9, showing a significant margin that validates the effectiveness of the PDE-based decoder. Additionally, we present the validation loss curves for the XGBoost model at the coarsest layer in Figure 9. The model without the PDE-based decoder exhibits early saturation, resulting in a higher loss. This performance gap stems primarily from the absence of the decoder, where XGBoost processes each voxel independently, disregarding the spatial correlations among neighboring voxels. In contrast, the PDE-based decoder captures the smoothness across the field, preventing the generation of discontinuous UDFs by the model.

Table 9: Performance w/wo PDE-based decoder.

Method	w/o PDE decoder	w PDE decoder
L2 CD	2.23	1.54

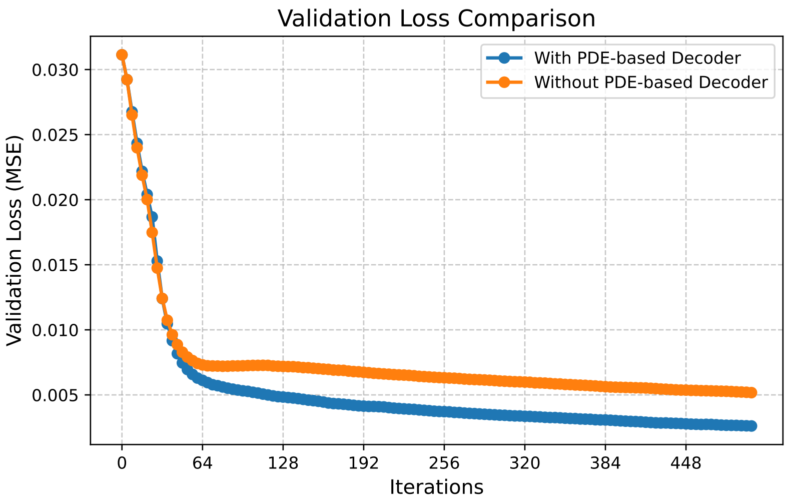


Figure 9: Comparison of validation loss of the xgboost model w/wo PDE-based decoder at the coarsest layer. The validation loss is the MSE between the predicted UDF and the ground-truth UDF.

3) The Effect of Feature Selection. To achieve an optimal balance between low complexity and high performance, we investigated the impact of varying the number of features selected by our RFT module. We experimented with feature subsets of different sizes, selecting a subset of 32, 64, 128, 256, 512, 1024, and all generated features. The performance metrics for these configurations are detailed in Table 10. Our findings indicate that the RFT effectively maintains robust performance with as few as 256 selected features. Adding more features beyond this number results in performance saturation. This trend is also verified in Figure 10, where we plot the discriminative power of each feature at the coarsest and the second coarsest layers. Both figures illustrate that only a small percentage of the features have relatively strong discriminative power and majorly contribute to the model's performance.

Table 10: Effect of selected feature dimensions of Sparse GUSL.

# of selected features	32	64	128	256	512	1024	all
L2 CD	1.725	1.632	1.578	1.540	1.532	1.561	1.533

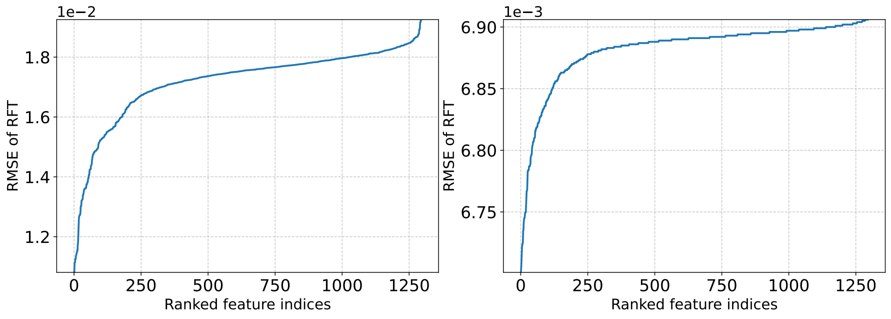


Figure 10: Ranked feature importance score of the Sparse Saab generated features for the coarsest layer (left) and the second coarsest layer (right). The smaller RMSE value of RFT indicates the stronger discriminative power of a given feature.

5 Conclusion and Future Work

A lightweight and interpretable surface reconstruction method, named LPSR, was proposed in this work. It offers high-quality reconstructed surfaces for various point cloud categories at lower computational and memory complexity. LPSR is mathematically transparent. It uses a feedforward-designed encoder and a PDE-based decoder, contributing to a deeper understanding of the surface reconstruction mechanism and model behaviors. The encoded representation has a clear physical meaning in controlling the surface shape, and it can be decoded at different grid resolutions to produce different levels of detail. The green learning paradigm further enhances the efficiency of LPSR. LPSR delivers high-quality reconstructed surfaces with a small feature dimension and minimal supervision.

We will extend LPSR to large-scale point cloud datasets, such as indoor and outdoor scenes. We aim to understand the 3D shape generation behavior in these diverse contexts while maintaining mathematical transparency and computational efficiency.

6 Acknowledgments

The authors acknowledge the gift support from the Tencent Media Lab and the Center for Advanced Research Computing (CARC) at the University of

Southern California for providing computing resources that have contributed to the research results reported in this publication. URL: https://carc.usc.edu.

References

[1] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust", in *Proceedings of the sixth ACM symposium on Solid modeling and applications*, 2001, 249–66.

[2] M. Baorui, L. Yu-Shen, and H. Zhizhong, "Reconstructing Surfaces for Sparse Point Clouds with On-Surface Priors", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] M. Baorui, H. Zhizhong, L. Yu-Shen, and Z. Matthias, "Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces", in *International Conference on Machine Learning (ICML)*, 2021.

[4] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction", *IEEE transactions on visualization and computer graphics*, 5(4), 1999, 349–59.

[5] A. Boulch and R. Marlet, "POCO: Point Convolution for Surface Reconstruction", 2022, https://arxiv.org/abs/2201.01831.

[6] F. Calakli and G. Taubin, "SSD: Smooth signed distance surface reconstruction", in *Computer Graphics Forum*, Vol. 30, No. 7, Wiley Online Library, 2011, 1993–2002.

[7] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions", in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01*, Association for Computing Machinery, 2001, 67–76.

[8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository", 2015, https://arxiv.org/abs/1512.03012.

[9] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *KDD 16*, ACM, August 2016, DOI: 10.1145/2939672.2939785.

[10] Y. Chen, M. Rouhsedaghat, S. You, R. Rao, and C.-C. J. Kuo, "Pixelhop++: A small successive-subspace-learning-based (ssl-based) model for image classification", in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, 3294–8.

[11] J. Chibane, A. Mir, and G. Pons-Moll, "Neural Unsigned Distance Fields for Implicit Function Learning", in *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database", in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, 248–55, DOI: 10.1109/CVPR.2009.5206848.

[13] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane", in *IEEE Transactions on Information Theory*, 1983, https://api.semanticscholar.org/CorpusID:6983029.

[14] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer, "Points2surf learning implicit surfaces from point clouds", in *European Conference on Computer Vision*, Springer, 2020, 108–24.

[15] M. Fainstein, V. Siless, and E. Iarussi, "DUDF: Differentiable Unsigned Distance Fields with Hyperbolic Scaling", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, 4484–93.

[16] J. Giesen, S. Spalinger, and B. Schölkopf, "Kernel Methods for Implicit Surface Modeling", in *Advances in Neural Information Processing Systems*, ed. L. Saul, Y. Weiss, and L. Bottou, Vol. 17, MIT Press, 2004, https://proceedings.neurips.cc/paper_files/paper/2004/file/64a08e5f1e6c39faeb90108c430eb120-Paper.pdf.

[17] B. Graham, M. Engelcke, and L. van der Maaten, "3D Semantic Segmentation with Submanifold Sparse Convolutional Networks", *CVPR*, 2018.

[18] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit Geometric Regularization for Learning Shapes", 2020, https://arxiv.org/abs/2002.10099.

[19] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet: Learning Local Shape Properties from Raw Point Clouds", *Computer Graphics Forum*, 37(2), 2018, 75–85, DOI: 10.1111/cgf.13343.

[20] B. Guillard, F. Stella, and P. Fua, "MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks", in *European Conference on Computer Vision*, 2022.

[21] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points", in *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, 1992, 71–8.

[22] F. Hou, C. Wang, W. Wang, H. Qin, C. Qian, and Y. He, "Iterative poisson surface reconstruction (ipsr) for unoriented points", *arXiv preprint arXiv:2209.09510*, 2022.

[23] J. Huang, H.-X. Chen, and S.-M. Hu, "A Neural Galerkin Solver for Accurate Surface Reconstruction", *ACM Trans. Graph.*, 41(6), 2022, DOI: 10.1145/3550454.3555457.

[24] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams, "Neural Kernel Surface Reconstruction", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, 4369–79.

[25] P. Kadam, M. Zhang, S. Liu, and C.-C. J. Kuo, "R-PointHop: A Green, Accurate and Unsupervised Point Cloud Registration Method", *arXiv preprint arXiv:2103.08129*, 2021.

[26] P. Kadam, M. Zhang, S. Liu, and C. .-. J. Kuo, "Unsupervised Point Cloud Registration via Salient Points Analysis (SPA)", 2020, https://arxiv.org/abs/2009.01293.

[27] P. Kadam, Q. Zhou, S. Liu, and C.-C. J. Kuo, "PCRP: Unsupervised Point Cloud Object Retrieval and Pose Estimation", in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, 1596–600, DOI: 10.1109/ICIP46576.2022.9897720.

[28] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction", in *Proceedings of the fourth Eurographics symposium on Geometry processing*, Vol. 7, 2006, 0.

[29] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction", *ACM Transactions on Graphics (ToG)*, 32(3), 2013, 1–13.

[30] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, 9601–11.

[31] C.-C. J. Kuo and A. M. Madni, "Green learning: Introduction, examples and outlook", *Journal of Visual Communication and Image Representation*, 90, 2023, 103685.

[32] C.-C. J. Kuo, M. Zhang, S. Li, J. Duan, and Y. Chen, "Interpretable convolutional neural networks via feedforward design", *Journal of Visual Communication and Image Representation*, 60, 2019, 346–59.

[33] S. Lin, D. Xiao, Z. Shi, and B. Wang, "Surface reconstruction from point clouds without normals by parametrizing the gauss formula", *ACM Transactions on Graphics*, 42(2), 2022, 1–19.

[34] Y. Lu, L. Wan, N. Ding, Y. Wang, S. Shen, S. Cai, and L. Gao, "Unsigned Orthogonal Distance Fields: An Accurate Neural Implicit Representation for Diverse 3D Shapes", in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[35] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space", 2019, https://arxiv.org/abs/1812.03828.

[36] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, “Occupancy networks: Learning 3d reconstruction in function space”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, 4460–70.

[37] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”, 2019, https://arxiv.org/abs/1901.05103.

[38] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “Deepsdf: Learning continuous signed distance functions for shape representation”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, 165–74.

[39] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, “Convolutional occupancy networks”, in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, Springer, 2020, 523–40.

[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”, 2017, https://arxiv.org/abs/1612.00593.

[41] S. Ren, J. Hou, X. Chen, Y. He, and W. Wang, “Geoudf: Surface reconstruction from 3D point clouds via geometry-guided distance representation”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, 14214–24.

[42] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler, “Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis”, 2021, https://arxiv.org/abs/2111.04276.

[43] J. R. Shue, E. R. Chan, R. Po, Z. Ankner, J. Wu, and G. Wetzstein, “3D Neural Field Generation using Triplane Diffusion”, 2022, https://arxiv.org/abs/2211.16677.

[44] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, “Implicit Neural Representations with Periodic Activation Functions”, 2020, https://arxiv.org/abs/2006.09661.

[45] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”, 2020, https://arxiv.org/abs/2006.10739.

[46] Z. Wang, P. Wang, P. Wang, Q. Dong, J. Gao, S. Chen, S. Xin, C. Tu, and W. Wang, “Neural-IMLS: Self-supervised Implicit Moving Least-Squares Network for Surface Reconstruction”, *IEEE Transactions on Visualization and Computer Graphics*, 2023, 1–16, DOI: 10.1109/TVCG.2023.3284233.

[47] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, “Deep Geometric Prior for Surface Reconstruction”, 2019, https://arxiv.org/abs/1811.10943.

[48] D. Xiao, Z. Shi, S. Li, B. Deng, and B. Wang, “Point normal orientation and surface reconstruction by incorporating isovalue constraints to Poisson equation”, *Computer Aided Geometric Design*, 103, 2023, 102195.

[49] Y. Yang, W. Wang, H. Fu, C.-C. J. Kuo, *et al.*, “On supervised feature selection from high dimensional feature spaces”, *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[50] J. Ye, Y. Chen, N. Wang, and X. Wang, “GIFS: Neural Implicit Function for General Shape Representation”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[51] L. Zhang, Z. Wang, Q. Zhang, Q. Qiu, A. Pang, H. Jiang, W. Yang, L. Xu, and J. Yu, “CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets”, *arXiv preprint arXiv:2406.13897*, 2024.

[52] M. Zhang, P. Kadam, S. Liu, and C.-C. J. Kuo, “GSIP: Green Semantic Segmentation of Large-Scale Indoor Point Clouds”, *Pattern Recognition Letters*, 164 (December), December 2022, 9–15, DOI: 10.1016/j.patrec.2022.10.014.

[53] M. Zhang, Y. Wang, P. Kadam, S. Liu, and C.-C. J. Kuo, “PointHop++: A Lightweight Learning Model on Point Sets for 3D Classification”, *arXiv preprint arXiv:2002.03281*, 2020.

[54] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, “PointHop: An Explainable Machine Learning Method for Point Cloud Classification”, *IEEE Transactions on Multimedia*, 2020.

[55] J. Zhou, B. Ma, S. Li, Y.-S. Liu, Y. Fang, and Z. Han, “CAP-UDF: Learning Unsigned Distance Functions Progressively from Raw Point Clouds with Consistency-Aware Field Optimization”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[56] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

[57] J. Zhou, B. Ma, Y.-S. Liu, Y. Fang, and Z. Han, “Learning Consistency-Aware Unsigned Distance Functions Progressively from Raw Point Clouds”, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[58] Q. Zhou and A. Jacobson, “Thingi10k: A dataset of 10,000 3d-printing models”, *arXiv preprint arXiv:1605.04797*, 2016.

[59] Q. Zhou, J. Yu, S. Liu, and C.-C. J. Kuo, “GPSR: A Green Point Cloud Surface Reconstruction Method”, in *2024 IEEE International Conference on Multimedia Information (MIPR)*, IEEE, 2024.