

Approximating Signed Distance Fields With Sparse Ellipsoidal Radial Basis Function Networks: A Dynamic Multi-Objective Optimization Strategy

Bobo Lian, Zidong Wang, Dandan Wang, Chenjian Wu, and Minxin Chen

Abstract—In this paper, a general learning method is proposed for approximating precomputed signed distance function (SDF) fields of implicit surfaces using a relatively small number of ellipsoidal radial basis functions (ERBFs). The SDF values may be obtained from various sources, including point clouds, triangle meshes, analytical expressions, pretrained neural networks, and others. Given SDF values at spatial grid points, the proposed method approximates the SDF with a significantly reduced number of ERBFs, yielding a compact representation while preserving the geometric shape of the corresponding implicit surface. To balance sparsity and approximation precision, a dynamic multi-objective optimization strategy inspired by Pareto multi-task learning is introduced. Instead of relying on fixed coefficients, this mechanism dynamically adjusts the trade-off during training by computing an optimal convex combination based on the gradients of the accuracy and sparsity loss terms. Simultaneous optimization is performed over the weights, centers, shapes, and orientations of the ERBFs. For enhanced computational efficiency, a nearest-neighbor-based data structure is employed to confine computations to points in the vicinity of each kernel center, and CUDA-based parallelism is utilized to further accelerate the optimization process. In addition, a hierarchical refinement strategy is applied based on the SDF spatial grid points, where coarse-to-fine samples are progressively incorporated for parameter initialization and optimization. This approach improves both convergence behavior and training efficiency. Extensive experiments on multiple benchmark datasets demonstrate that the proposed method represents SDF fields with substantially fewer parameters than existing sparse implicit representation methods, while attaining superior accuracy, robustness, and computational efficiency. The corresponding executable program is publicly available at <https://github.com/lianbobo/SE-RBFNet-git>.

Index Terms—Signed distance function, implicit surface representation, sparse optimization, radial basis function, multi-objective optimization.

I. INTRODUCTION

ACCURATE and compact representation of signed distance functions (SDFs) is crucial for efficient storage,

computation, and downstream processing of 3D geometry [1]. SDFs have been widely adopted due to their implicit encoding of surfaces as the zero-level set of a continuous function [2], [3], thereby supporting tasks such as reconstruction, storage, editing, and simulation.

Most implicit surface representation methods, including radial basis function (RBF) interpolation [4]–[7] and neural implicit methods [8]–[10], have been designed to learn SDFs from raw geometric data. Although high-fidelity surfaces can be achieved by these approaches, they often involve a large number of parameters to capture fine-scale or anisotropic features. Specifically, conventional RBF networks typically employ isotropic Gaussian basis functions with spherical influence regions. While well-known for their universal approximation properties [11], [12], isotropic RBFs are less effective in capturing anisotropic structures, uneven curvature, or elongated geometries in complex 3D surfaces [13].

Inspired by the application of ellipsoidal RBFs (ERBFs) in molecular surface modeling [14] and 3D Gaussian Splatting (3DGS) [13], this paper introduces the Sparse Ellipsoidal Radial Basis Function Network (SE-RBFNet) as a fast and efficient approach for the sparse representation of precomputed SDFs. Specifically, 3DGS [13] achieves remarkable efficiency by representing complex scenes using a vast number of anisotropic Gaussian primitives, whose positions, scales, and orientations are jointly optimized, and whose influence is confined to local neighborhoods to enable real-time rendering. Motivated by the success of these anisotropic Gaussian primitives, SE-RBFNet replaces traditional spherical kernels with learnable ERBFs. In this formulation, each basis function acts analogously to a deformable 3D Gaussian primitive, explicitly parameterized by a center, anisotropic scales, and a rotation matrix. By transferring the 3DGS concept of localized anisotropic influence to SDF approximation, SE-RBFNet allows each kernel to dynamically adapt its shape and orientation to align with local geometric structures, yielding a significantly more compact representation than previous methods [1], [15] while maintaining high approximation accuracy. Ultimately, by approximating the SDF with a sparse set of these optimized ERBFs using precomputed samples drawn from diverse sources (e.g., analytical expressions, meshes, or existing implicit models), SE-RBFNet serves as a highly effective post hoc sparsification module that intrinsically complements existing SDF generation techniques.

Note that SE-RBFNet shares conceptual similarities with SparseRBF [1], but differs in two key aspects. First, the two

Manuscript received November 10, 2025; revised June 28, 2026. Minxin Chen is supported by the Jiangsu Provincial Key Research and Development Program under Grant BE2022058-4. (Corresponding authors: Chenjian Wu; Minxin Chen.)

Bobo Lian and Dandan Wang are with the School of Mathematical Sciences, Soochow University, Suzhou 215006, China (e-mail: 20244007006@stu.suda.edu.cn; 20194207032@stu.suda.edu.cn).

Zidong Wang is with the Department of Computer Science, Brunel University of London, UB8 3PH Uxbridge, U.K. (e-mail: Zidong.Wang@brunel.ac.uk).

Chenjian Wu is with the School of Electronic and Information Engineering, Soochow University, Suzhou 215006, China (e-mail: cjwu@suda.edu.cn).

Minxin Chen is with the School of Mathematical Sciences, Soochow University, Suzhou 215006, China (e-mail: chenminxin@suda.edu.cn).

methods employ fundamentally different optimization strategies. SparseRBF adopts spherical basis functions with fixed centers and reduces the problem to a linear Least Absolute Shrinkage and Selection Operator (LASSO) formulation [16]. In contrast, SE-RBFNet involves the joint optimization of ERBF centers, orientations, shapes, and coefficients, resulting in a nonlinear and nonconvex formulation. This allows for a more flexible and compact representation of complex geometries. Second, in SE-RBFNet, an inscribed-sphere initialization strategy is employed to improve adaptability to varying geometric structures.

Building upon prior work in sparse RBF networks for images [17] and 3D surfaces [18], SE-RBFNet introduces a dynamic multi-objective optimization framework [19] that adaptively balances the $\mathcal{L}_2$ fitting error and $\mathcal{L}_1$ sparsity. To further enhance computational efficiency and robustness, a nearest-neighbor search mechanism is used to restrict evaluations to points near each kernel center during both forward and backward passes, thereby reducing the computational burden. This optimization is further accelerated using CUDA-based parallelization. In addition, a hierarchical optimization strategy is proposed based on SDF grid points. Specifically, ERBF parameters are initialized and optimized using coarse grid points, while finer grid points are incrementally incorporated as optimization proceeds. This hierarchical scheme applies to both uniform grids and adaptive structures such as octrees [20]. Moreover, grid points exhibiting large $\mathcal{L}_2$ errors are identified and used to define new kernel centers, thereby improving approximation quality and convergence speed.

To tackle the sparse representation problem of SDFs using ERBFs, the following key challenges must be addressed:

- 1) *Complex optimization landscape*: Joint optimization over ERBF centers, orientations, shapes, and weights yields a high-dimensional, nonlinear, and nonconvex problem. Strong interdependencies among parameters make the optimization prone to poor local minima. To address this, initialization strategies and structured optimization procedures must be carefully designed to facilitate convergence toward meaningful solutions.
- 2) *Accuracy vs. sparsity trade-off*: It is inherently difficult to obtain high fidelity near the zero-level set while maintaining a small number of basis functions. The use of multi-objective optimization allows adaptive balancing between $\mathcal{L}_2$ approximation accuracy and $\mathcal{L}_1$ sparsity, which single-objective approaches cannot achieve. This balance is particularly important for handling geometries with varying complexity.
- 3) *Computational bottlenecks*: It is computationally intensive to evaluate ERBFs and their gradients over dense 3D grids, with costs growing quadratically with both the number of grid points and basis functions. Although nearest-neighbor filtering and hierarchical optimization help reduce computational overhead, GPU-based parallelization remains essential for scalability.
- 4) *Initialization and shape adaptation*: The effectiveness of the approach depends significantly on initialization and geometric adaptability. The inscribed-sphere initialization provides robust initial conditions, while dynamically

adding basis functions at high-error regions improves the representation of both smooth and detailed structures. This combination of smart initialization and adaptive refinement mitigates convergence issues and improves approximation accuracy across diverse geometries.

Motivated by the challenges identified above, in this paper, we aim to design the SE-RBFNet framework, which integrates dynamic multi-objective optimization, hierarchical refinement, and adaptive basis function addition to enable efficient and accurate SDF representation. Accordingly, the main contributions of this work are summarized as follows:

- 1) A novel framework of SE-RBFNet is proposed, where the sparse representation of SDFs is formulated as a nonlinear sparse optimization problem over precomputed SDF samples (see Section III-A).
- 2) A dynamic multi-objective optimization strategy is developed to balance approximation accuracy and sparsity (see Section III-B).
- 3) A coarse-to-fine hierarchical optimization scheme based on SDF grid points is introduced, with nearest-neighbor filtering incorporated to enhance computational efficiency and reduce model complexity (see Section III-C2).
- 4) An adaptive basis function refinement mechanism is designed, wherein new kernel centers are iteratively selected from high-error regions and corresponding ellipsoidal parameters are dynamically updated, improving accuracy and convergence speed (see Section III-C3).

II. RELATED WORK

Implicit surface representation has been a longstanding topic of research. Numerous methods have been developed to directly infer implicit functions (e.g. SDFs or occupancy fields) from raw geometric data. In contrast, the present work addresses a different problem: the efficient representation of a precomputed set of SDF samples in a compact and memory-efficient manner, regardless of the source from which these values were derived. By decoupling SDF estimation from representation, the proposed approach enables a focused investigation into encoding efficiency while fully utilizing high-quality SDF inputs. The remainder of this section briefly reviews both classical and learning-based methods for implicit surface representation.

A. Classical Implicit Representations

Classical methods typically define a surface as the zero-level set of a continuous scalar field, often in the form of a signed distance function (SDF). This formulation enables flexible shape representation and efficient surface extraction using algorithms such as Marching Cubes [21].

Pioneering work by Curless and Levoy [22] introduced volumetric SDF fusion, establishing a foundation for subsequent methods. Calakli and Taubin [2] further emphasized that the target implicit function should approximate the true SDF, rather than merely occupying the interior volume. To handle noisy or incomplete data, Pan and Skala [23] proposed

a continuous global optimization framework that minimizes a variational energy combining data fidelity and smoothness.

Poisson surface reconstruction (PSR) and its variants [24], [25] reformulate surface reconstruction as the solution of a spatial Poisson equation, yielding smooth and globally consistent surfaces. Building on this framework, an iterative PSR method was proposed by Hou et al. [26], in which normals are re-estimated from the reconstructed surface at each iteration to progressively refine surface quality.

RBFs represent another classical approach to implicit surface modeling. These methods typically place RBFs at data points and optimize the corresponding weights to fit the surface. Carr et al. [4] introduced multi-harmonic RBFs for smooth interpolation and developed a greedy algorithm that iteratively added centers with the largest residuals, reducing the overall number of basis functions. To further improve efficiency, Ohtake et al. [7] decomposed the global fitting problem into overlapping local subproblems, each solved via least-squares RBF fitting. This localized formulation significantly reduced the number of required basis functions.

A different center selection strategy was proposed by Samozino et al. [27], where centers were chosen from Voronoi vertices derived from the input point cloud. Although this resulted in a more uniform spatial distribution of RBF centers, equal treatment of surface and off-surface points during fitting introduced approximation errors near the true surface. Subsequent enhancements to RBF-based surface modeling explored anisotropic kernels [28], compactly supported RBFs [29], orthogonal least squares (OLS) center selection [30], and Hermite RBFs [31]–[33], which incorporate normal information directly into the fitting process. More recent advances have introduced sparse center selection strategies [18], partitioned representations [34], and novel interpolation techniques [35], broadening the applicability of RBFs in diverse surface reconstruction tasks.

Beyond RBFs, alternative formulations have also been explored for implicit surface representation. These include Implicit Moving Least Squares [36], Fourier-based methods [37], and Gaussian-based models [38], [39], all of which highlight the flexibility and extensibility of implicit approaches to 3D surface modeling.

B. Neural Implicit Representations

Deep learning has demonstrated remarkable capabilities across a wide range of applications, including load forecasting [40], multimodal learning [41], [42], reliability and security [43], [44], medical diagnosis [45], and semantic segmentation [46]. Inspired by this broader success, recent learning-based approaches have increasingly parameterized implicit functions using neural networks. Mescheder et al. [47] introduced Occupancy Networks, which represent implicit functions through neural networks combined with latent codes. Park et al. [3] proposed DeepSDF, where shapes are modeled as continuous SDFs using multilayer perceptrons (MLPs). Several extensions have been developed, including convolutional neural networks (CNNs) for learning shape priors [48], geometric regularization [9], Fourier

Feature Networks (FFN) [49], Sinusoidal Representation Networks (SIREN) [50], Wavelet Implicit Neural Representations (WIRE) [15], second-order constraints [51], and Hessian-based smoothness techniques [10]. These methods have significantly improved the representation capability of neural implicit models and achieved increasingly compact representations of complex geometric fields.

In contrast to global shape modeling, local patch-based methods such as Points2Surf [8] and PPSurf [52] adopt self-supervised learning to improve robustness against noise, sparsity, and non-uniform sampling. Ma et al. [53] further relaxed the supervision requirement by proposing an unsupervised framework that estimates surface geometry directly from point clouds without relying on ground-truth SDFs.

The aforementioned studies highlight the utility of SDFs as a fundamental surface representation. Most existing methods, whether classical implicit models or neural implicit models, focus on estimating or learning SDFs directly from raw data. Such methods typically require a large number of function parameters or network weights to represent detailed geometry. In contrast, the present work focuses on the sparse representation of precomputed SDF samples by approximating the implicit surface with a compact set of ERBFs. This design enables SE-RBFNet to operate independently of the SDF source, whether derived from oriented point clouds, triangle meshes, analytical functions, or outputs of neural models. As a general SDF encoding framework, SE-RBFNet provides efficient and accurate approximation with strong sparsity.

III. METHOD

In this section, the SE-RBFNet framework is presented for approximating precomputed SDFs in order to achieve a compact and sparse representation of implicit surfaces. ERBFs are employed as nodes in the hidden layers of the network, with sparsity introduced to represent the surface using a reduced number of basis functions. The overall workflow of SE-RBFNet is illustrated in Fig. 1. For illustrative purposes, SDFs derived from a given implicit surface are considered. An octree grid is used to provide spatial sampling points for training and surface extraction; however, alternative sampling strategies such as uniform grids may also be adopted.

A. Ellipsoid RBF Neural Network

A function of the form $\Phi(\mathbf{x}, \mathbf{c}) = \Phi(\|\mathbf{x} - \mathbf{c}\|)$ is referred to as an RBF, whose value depends solely on the distance between $\mathbf{x}$ and the center $\mathbf{c}$. Various forms of RBFs exist. One common example is the standard isotropic Gaussian RBF $\Phi(\mathbf{x}, \mathbf{c}) = e^{-\|\mathbf{x} - \mathbf{c}\|^2}$, which relies on the standard Euclidean distance and yields a symmetric spherical influence region, where $\mathbf{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3$, $\mathbf{c} = (c_1, c_2, c_3)^T \in \mathbb{R}^3$.

In this work, the classical RBF is extended to a more flexible ellipsoidal form, namely, ellipsoidal RBF (ERBF) [54]. The ERBF in $\mathbb{R}^3$ is defined as follows:

$$\hat{\Phi}(\mathbf{x}, \Theta) = e^{-\|\mathbf{D}\mathbf{R}(\mathbf{x} - \mathbf{c})\|^2}, \quad (1)$$

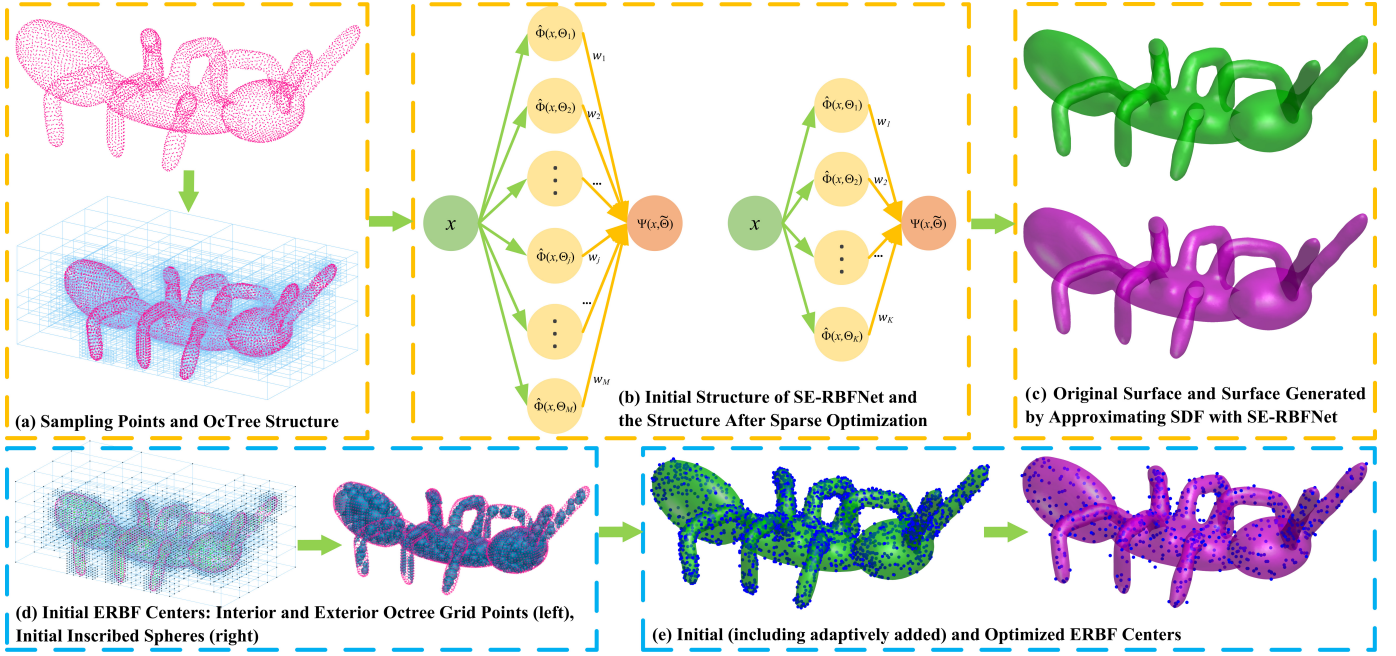


Fig. 1. The workflow and results of SE-RBFNet on SDFs of implicit surfaces. (a) shows the octree structure as an example, where SDF values at grid points are obtained using arbitrary SDF-generation methods and used as training data. (b) illustrates the initial structure of SE-RBFNet and the structure after sparse optimization, where the number of hidden-layer ERBFs is significantly reduced. (c) displays explicit surfaces extracted from the original (green) and SE-RBFNet-approximated (purple) SDF values on octree grids. (d) presents the process of extracting initial ERBF centers using maximum inscribed spheres. The left panel depicts interior (green) and exterior (black) octree grid points, while the right panel shows the computed maximum inscribed spheres (in light blue). (e) demonstrates that the number of optimized ERBF bases is significantly reduced while the surface shape is preserved.

where $\mathbf{D} = \text{diag}(d_1, d_2, d_3)$, $d_1, d_2, d_3 \in \mathbb{R}$ are the scaling factors along the ellipsoid's principal axes. $\mathbf{R}$ represents the rotation matrix, defined as follows:

$$\mathbf{R}(\theta_x, \theta_y, \theta_z) = \mathbf{R}(\theta_z) \cdot \mathbf{R}(\theta_y) \cdot \mathbf{R}(\theta_x). \quad (2)$$

The detailed form of $\mathbf{R}$ is given in Appendix A. $\mathbf{R}(\theta_x)$, $\mathbf{R}(\theta_y)$, $\mathbf{R}(\theta_z)$ are the rotation matrices of the three principal axes, θ_x , θ_y , θ_z are the rotation angles of the ellipsoid along the principal axis. $\Theta = \{c, \mathbf{D}, \theta_x, \theta_y, \theta_z\}$ represents the parameters set of a single ERBF, including its center, scaling matrix, and rotation angles. Mathematically, the standard RBF is a strict special case of the ERBF where both the scaling matrix $\mathbf{D}$ and the rotation matrix $\mathbf{R}$ degenerate to the identity matrix. It is noteworthy that the exponential term $\|\mathbf{D}\mathbf{R}(\mathbf{x} - \mathbf{c})\|^2$ in Eq. (1) is mathematically equivalent to the squared Mahalanobis distance $(\mathbf{x} - \mathbf{c})^T \Sigma^{-1}(\mathbf{x} - \mathbf{c})$, where $\Sigma^{-1} = \mathbf{R}^T \mathbf{D}^2 \mathbf{R}$. This establishes a direct correspondence between the parameters of SE-RBFNet and the primitives used in 3DGS. Mirroring the 3DGS paradigm, the center $\mathbf{c}$ anchors the primitive in space, the scaling matrix $\mathbf{D}$ controls the receptive field size along principal axes, and the rotation matrix $\mathbf{R}$ aligns the basis with the dominant curvature directions of the underlying geometry.

Fig. 1(b) shows the structure of the SE-RBFNet. When ERBF is chosen as the hidden layer activation function, given an input $\mathbf{x}_i$, the output of the ERBF network is as follows:

$$\Psi(\mathbf{x}_i, \tilde{\Theta}) = \sum_{j=1}^M w_j^2 \hat{\Phi}(\mathbf{x}_i, \Theta^{(j)}) = \sum_{j=1}^M w_j^2 e^{-\|\mathbf{D}_j \mathbf{R}_j(\mathbf{x}_i - \mathbf{c}_j)\|^2}, \quad (3)$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})^T$ is an arbitrary point in three-

dimensional space, M is the number of ERBFs in the hidden layer, and $\Theta^{(j)} = \{c_j, \mathbf{D}_j, \theta_{xj}, \theta_{yj}, \theta_{zj}\}$ specifically denotes the parameters of the j -th ERBF. $\tilde{\Theta} = \{\mathbf{C}, \mathbf{D}, \mathbf{A}, \mathbf{W}\}$ indicates all the parameters of the ERBFs in the hidden layer, which are also all the parameters to be optimized in SE-RBFNet, where $\mathbf{C} \in \mathbb{R}^{M \times 3}$ is the center of all ERBFs and $\mathbf{c}_j = (c_{j1}, c_{j2}, c_{j3})$ is the j -th row of $\mathbf{C}$, indicating the center of the j -th ERBF. Similarly, $\mathbf{D} \in \mathbb{R}^{M \times 3 \times 3}$, $\mathbf{D}_j = \text{diag}(d_{j1}, d_{j2}, d_{j3})$, where $1/|d_{ji}|$ ($i = 1, 2, 3$) are the semi-axis lengths of the j -th ellipsoid. $\mathbf{A} \in \mathbb{R}^{M \times 3}$ is the rotation angles of all ERBFs, the j -th row of $\mathbf{A}$ is $\mathbf{a}_j = (\theta_{xj}, \theta_{yj}, \theta_{zj})$. $\mathbf{R}_j = \mathbf{R}_j(\theta_{xj}, \theta_{yj}, \theta_{zj})$ is the rotation matrix of the j -th ERBF. $\mathbf{W} \in \mathbb{R}^{M \times 1}$ is the coefficient matrix from the hidden layer to the output layer, w_j^2 is the weight of the j -th ERBF from the hidden layer to the output layer. The weight is nonnegative to prevent different ERBFs from canceling each other out through opposing weighting and to make the output of Eq. (3) nonnegative and consistent with the nonnegative SDF values defined in Eq. (4).

B. Loss Function Design

Given a set of sampling points on an implicit surface $P = \{\mathbf{p}_i\}_{i=1}^{K_p}$, where K_p is the number of points, we assume that the corresponding octree grid points $G = \{\mathbf{g}_j\}_{j=1}^{K_g}$ and the signed distance value $S(\mathbf{g}_j)$ at each grid point are provided as input, where K_g is the number of grid points. The surface itself is characterized by the zero-level set of the signed distance: $\{\mathbf{g}_j \in \mathbb{R}^3, S(\mathbf{g}_j) = 0\}$. For points inside (outside) the surface, $S(\mathbf{g}_j)$ is negative (positive). As can be seen from Eq. (3), the

output of Ψ is nonnegative. Therefore, to fit the signed distance using the SE-RBFNet, we scale all signed distances S to $[0, 2]$ using the following nonlinear transformation.

$$\hat{S} = 2 \cdot e^{-h(S-m)^2}, \quad (4)$$

where $m = \min(S)$, $h = \ln(2)/m^2$. Eq. (4) normalizes S values less than 0 (inside the surface) to the interval $[1, 2]$ and values greater than 0 (outside the surface) to the interval $[0, 1]$. Notably, the zero-level set of $S = 0$ (i.e., the reconstructed surface) is mapped to $\hat{S} = 1$.

We define the total training point set V as the union of the octree grid points G and the surface sampling points P . This is denoted as $V = G \cup P = \{\mathbf{v}_i\}_{i=1}^N$, where $N = K_p + K_g$ represents the total number of training points, and $\mathbf{v}_i$ denotes the i -th point in this combined set. $\hat{S}$ is merged with the all-1 vector I as the label of P , denoted as T , $T = \{\hat{S}, I\} = \{t_i\}_{i=1}^N$, t_i is the i -th SDF value. Here, the length of I is equal to the number of sampling points. The SE-RBFNet model is used to approximate the given SDF values of V as follows:

$$\begin{aligned} \mathbf{O} &= \begin{bmatrix} \hat{\Phi}(\mathbf{v}_1, \Theta^{(1)}) & \cdots & \hat{\Phi}(\mathbf{v}_1, \Theta^{(M)}) \\ \hat{\Phi}(\mathbf{v}_2, \Theta^{(1)}) & \cdots & \hat{\Phi}(\mathbf{v}_2, \Theta^{(M)}) \\ \vdots & \vdots & \vdots \\ \hat{\Phi}(\mathbf{v}_N, \Theta^{(1)}) & \cdots & \hat{\Phi}(\mathbf{v}_N, \Theta^{(M)}) \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ \vdots \\ w_M^2 \end{bmatrix} \\ &= \mathbf{F} \cdot \mathbf{W}^2 = \begin{bmatrix} \Psi(\mathbf{v}_1, \tilde{\Theta}) \\ \Psi(\mathbf{v}_2, \tilde{\Theta}) \\ \vdots \\ \Psi(\mathbf{v}_N, \tilde{\Theta}) \end{bmatrix} \approx \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}. \end{aligned} \quad (5)$$

For notational clarity, we define the feature matrix $\mathbf{F} \in \mathbb{R}^{N \times M}$ in Eq. (5), where each element $F_{ij} = \hat{\Phi}(\mathbf{v}_i, \Theta^{(j)})$ denotes the response value of the i -th training point under the influence of the j -th ERBF. Once the network parameters $\tilde{\Theta}$ are optimized, the explicit surface is extracted by applying the marching cubes algorithm [21] to the predicted SDF values on the octree grid points G . We set the loss function as:

$$\mathcal{L}(\tilde{\Theta}) = \alpha \sum_{i=1}^N \left(\Psi(\mathbf{v}_i, \tilde{\Theta}) - t_i \right)^2 + (1 - \alpha) \|\mathbf{W}\|_1. \quad (6)$$

Note that Eq. (6) can be rewritten concisely as follows:

$$\mathcal{L} = \alpha \mathcal{L}_2 + (1 - \alpha) \mathcal{L}_1,$$

where $\mathcal{L}_2$ ensures approximate accuracy by minimizing the squared error between predicted and ground-truth values on training points V . $\mathcal{L}_1$ enforces sparsity in the network parameters by penalizing the $\mathcal{L}_1$ norm of the weights, encouraging a sparse solution for $\mathbf{W}$. Weights in $\mathbf{W}$ that are close to zero correspond to basis elements with negligible contributions to the network, which can safely be removed. $\alpha \in [0, 1]$ is a dynamic weighting coefficient that balances the two loss terms. Our aim is to minimize the loss function $\mathcal{L}$ and find the corresponding optimal network parameters, $\tilde{\Theta}$.

It is worth pointing out that, in surface representation, more attention is paid to the points close to the surface, where $\{\mathbf{x} \in \mathbb{R}^3 \mid \Psi(\mathbf{x}) = 1\}$. To further reduce the computational cost, two thresholds, τ_1 (with $\tau_1 < 1$) and τ_2 (with $\tau_2 > 1$),

each close to 1, are introduced to filter the training points for loss computation at each iteration, as defined by the following formula:

$$\begin{aligned} id_0 &= \{i \mid \tau_1 < t_i < \tau_2\}, \\ id_1 &= \{i \mid t_i \leq \tau_1 \text{ and } \Psi(\mathbf{v}_i, \tilde{\Theta}) > \tau_1\}, \\ id_2 &= \{i \mid t_i \geq \tau_2 \text{ and } \Psi(\mathbf{v}_i, \tilde{\Theta}) < \tau_2\}, \\ id_{all} &= id_0 \cup id_1 \cup id_2. \end{aligned} \quad (7)$$

In Eq. (7), id_0 selects from both interior and exterior regions close to the surface, which are crucial for accurately approximating the SDF near the surface and ensuring effective surface representation through ERBFs. id_1 (id_2) selects points with erroneous predictions in the exterior (interior) region to help the network learn and optimize further, avoiding artificial bulges or dents on the surface represented by the predicted SDF. For instance, id_1 (id_2) prevents misclassification of points that should properly reside outside (inside) the surface as being erroneously close to or crossing the surface boundary. Only the points corresponding to id_{all} are subject to loss calculation, so Eq. (6) can be rewritten as:

$$\mathcal{L}(\tilde{\Theta}) = \alpha \sum_{i \in id_{all}} \left(\Psi(\mathbf{v}_i, \tilde{\Theta}) - t_i \right)^2 + (1 - \alpha) \|\mathbf{W}\|_1. \quad (8)$$

During the optimization of the loss function, the $\mathcal{L}_1$ regularization term is applied only when two conditions are simultaneously satisfied. First, starting from epoch $e_p \geq k^{l_2}$, the $\mathcal{L}_2$ loss is considered converged, meaning that its standard deviation over the last k^{l_2} epochs falls below a threshold τ^{l_2} :

$$\text{std} \left(\left\{ \mathcal{L}_2^{(e)}(\tilde{\Theta}) \right\}_{e=e_p-k^{l_2}+1}^{e_p} \right) < \tau^{l_2}, \quad (9)$$

where $\text{std}(\cdot)$ denotes the standard deviation operation. The $\mathcal{L}_2$ loss at epoch e and the current training epoch are represented by $\mathcal{L}_2^{(e)}$ and e_p , respectively. Second, the maximum absolute error between the predicted and the ground-truth SDFs must be below a threshold τ_m :

$$\max_{i \in id_{all}} \left| \Psi(\mathbf{v}_i, \tilde{\Theta}) - t_i \right| < \tau_m, \quad (10)$$

If either of these conditions is not satisfied, the optimization proceeds using only the $\mathcal{L}_2$ loss, i.e., $\alpha = 1$.

When performing sparse optimization with the $\mathcal{L}_1$ regularization term, we adopt a dynamic weighting mechanism [19] instead of using a fixed coefficient α in Eq. (8). This mechanism is inspired by the Frank-Wolfe method for Pareto multi-task learning [19] and dynamically adjusts the trade-off between accuracy and sparsity during the optimization process. Specifically, at each optimization step, the coefficient α is computed based on the gradients of the two loss terms, reflecting their current relative contributions. This dynamic approach is formulated as:

$$\alpha = \text{clip}_{[0,1]} \left(\frac{\left(\frac{\partial \mathcal{L}_1}{\partial \mathbf{W}} - \frac{\partial \mathcal{L}_2}{\partial \mathbf{W}} \right)^T \frac{\partial \mathcal{L}_1}{\partial \mathbf{W}}}{\left\| \frac{\partial \mathcal{L}_2}{\partial \mathbf{W}} - \frac{\partial \mathcal{L}_1}{\partial \mathbf{W}} \right\|_2^2} \right), \quad (11)$$

where $\text{clip}_{[0,1]}(\sigma) = \max(\min(\sigma, 1), 0)$. The detailed mathematical derivation of the dynamic weight α is provided in the Appendix B. This dynamic multi-objective optimization

strategy automatically adjusts the coefficients during training, balancing accuracy and sparsity according to the model's state, which improves generalization and robustness.

C. Optimization Algorithm

1) *Inscribed Sphere-Based Initialization for ERBF*: The hidden layer of SE-RBFNet consists of multiple ERBFs. To initialize the network parameters effectively, we use the largest inscribed sphere method [55] to calculate the initialization parameters of each ERBF. This method ensures that the initialization process can adapt to the geometry of the target surface, facilitating the learning of the SDF used for surface representation.

For interior points $G_{\text{in}} = \{g_i \mid 1 < \hat{S}_i < 2\}$ and their corresponding signed distance $\hat{S}_{\text{in}}$, we iteratively detect the maximal inscribed spheres using Algorithm 1 to obtain the initial centers $C = \{c_j\}_{j=1}^M$ and the initial weights $W = \{w_j\}_{j=1}^M$. To visually demonstrate the calculation process of the inscribed sphere, Fig. 1.(d) illustrates the interior and exterior points (left), as well as the distribution of the maximum inscribed spheres (right). As depicted, the inscribed spheres densely cover the internal space of the implicit surface, providing an effective approximation of its geometric structure.

Algorithm 1 Inscribed Sphere-Based Initialization for ERBF

Input: interior grid points G_{in} , the corresponding signed distance $\hat{S}_{\text{in}}$, sampling points P .

Compute the closest distance of each interior point to P , denoted as d^{in} . Initialize empty matrices: C and W .

while G_{in} is not empty **do**

Find the index of the maximum d^{in} value:

$$i = \arg \max(d^{\text{in}}), \quad r = d^{\text{in}}[i]$$

Append the center and weight to the matrices:

$$C \leftarrow G_{\text{in}}[i], \quad W \leftarrow \sqrt{\hat{S}_{\text{in}}[i]}$$

Compute: $d_k = \|G_{\text{in}}[k] - G_{\text{in}}[i]\|_2, \quad \forall k$

Remove points within the inscribed sphere:

$$G_{\text{in}} \leftarrow G_{\text{in}}[d_k > r]$$

Update $d^{\text{in}} \leftarrow d^{\text{in}}[d_k > r]$ and $\hat{S}_{\text{in}} \leftarrow \hat{S}_{\text{in}}[d_k > r]$

Output: Centers C , Weights W .

From Algorithm 1, the weights and centers are initialized. Then, the shape parameters $\{D_j\}_{j=1}^M$ are initialized with identical diagonal elements, i.e., $d_{j1} = d_{j2} = d_{j3}$. Assume that the Gaussian distribution is satisfied between each Gaussian kernel center and its nearest neighboring Gaussian kernel center. To minimize mutual influence between Gaussian kernels, we ensure that the value of each Gaussian kernel at its nearest neighboring kernel center is sufficiently small. Under this assumption, d_{j1} satisfies the following equation:

$$w_j^2 e^{-d_{j1}^2 (\hat{d}_j/2)^2} = \gamma. \quad (12)$$

Here, γ is a threshold value chosen to minimize the mutual influence between kernels. $\hat{d}_j = \min_{i \neq j} \|c_j - c_i\|_2, i = 1, 2, \dots, M$. This means that $\hat{d}_j$ is the shortest distance from

the current kernel center c_j to any other kernel center. The d_{j1} can be derived from Eq. (12) as:

$$d_{j1} = d_{j2} = d_{j3} = 2\sqrt{-\ln(\gamma/w_j^2)}/\hat{d}_j. \quad (13)$$

Initialize $D_j = \text{diag}(d_{j1}, d_{j2}, d_{j3})$. The rotation angle of each ellipsoid is set to zero, so that each corresponding rotation matrix R_j is initialized as the identity matrix.

2) *Hierarchical Optimization*: In this section, we describe a hierarchical optimization method for training the aforementioned SE-RBFNet on multi-level grid points. Although the method is applicable to any hierarchical grid structure, we use an octree as an example for illustration. Specifically, we perform the training process on the octree grid points layer by layer. Assuming that G^i represents the set of grid points at the i -th layer of the octree, G can be represented as $G = \{G^i\}_{i=1}^l$ and $G^i = \{G_{k_1}^i, G_{k_2}^i, \dots, G_{k_{k_i}}^i\}$, where k_i indicates the number of grid points in the i -th layer, and the total number of layers in the octree is l . Similarly, the signed distance corresponding to each grid point is also layered, that is $\hat{S} = \{\hat{S}^i\}_{i=1}^l$ and $\hat{S}^i = \{\hat{S}_{k_1}^i, \hat{S}_{k_2}^i, \dots, \hat{S}_{k_{k_i}}^i\}$. The training process begins with the grid points of the coarse layer, and data from finer layers is progressively incorporated. The specific steps are as follows:

- 1) Set the starting layer to l_s , and calculate the inscribed sphere using the points $\hat{G}^{l_s} = \{G^1, G^2, \dots, G^{l_s}\}$ to initialize the ERBF parameters (Section III-C1).
- 2) Combine $\hat{G}^{l_s}$ with the sampling points P to form $V^{l_s} = \{\hat{G}^{l_s}, P\}$ as the initial training set, send it to the model along with the corresponding label $T^{l_s} = \{\hat{S}^1, \hat{S}^2, \dots, \hat{S}^{l_s}, I\}$ for supervised optimization.
- 3) Using Eq. (8), compute the loss and optimize until convergence, then add the points of the next finer layer to the training set.
- 4) Repeat the third step until all grid points of the octree have been included in the optimization.

This hierarchical method effectively reduces the difficulty of model learning. In addition, trained parameters in the current layer are used as initialization parameters for learning in the next layer, similar to a transfer learning model, thereby improving convergence and stability.

Note that the formulations presented here, as well as those introduced below, are derived with respect to the initial layer l_s , while the same optimization is applied to subsequent layers. In practice, l_s is typically set to the third-to-last layer.

3) *Adaptive Basis Function Addition*: We introduce an adaptive basis function addition mechanism that automatically adds suitable basis functions to areas with large errors during loss optimization. Before adding new basis functions, two prerequisites must be satisfied: (1) the condition in Eq. (9) has been fulfilled, meaning the $\mathcal{L}_2$ loss has stabilized; (2) the number of effective basis functions remains stable over a predefined number of optimization steps, formulated as:

$$\max \left(\left\{ b_f^{(e)} \right\}_{e=e_p-k^{l_1}+1}^{e_p} \right) - \min \left(\left\{ b_f^{(e)} \right\}_{e=e_p-k^{l_1}+1}^{e_p} \right) < \tau^{l_1}, \quad (14)$$

where $b_f^{(e)}$ indicates the number of effective basis functions at epoch e , and is computed as: $b_f^{(e)} = \sum_{j=1}^M \mathbb{I}(\|w_j\| \geq \tau_d)$, $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if the condition is

true and 0 otherwise. The threshold τ_d is used to determine whether a basis function is effective. k^{l_1} and τ^{l_1} represent the number of iterations used to determine the stability of the number of basis functions and the corresponding threshold, respectively. Once the condition of Eq. (14) is triggered, the basis function addition process is automatically performed. After adding the new basis functions, the $\mathcal{L}_1$ regularization term is temporarily deactivated and the network switches to pure $\mathcal{L}_2$ optimization to fully train the newly added basis functions. The $\mathcal{L}_1$ regularization term is reactivated for sparse optimization when Eq. (9) and Eq. (10) are satisfied again.

Algorithm 2 Adaptive ERBF Addition Algorithm

Input: Error vector E , the training points V^{l_s} , id_{all} and τ_m .

- 1: Identify high-error points: $\mathcal{I}_{add} = \{i \mid |E_i| > \frac{\tau_m}{2}\}$.
- 2: Construct a KD-tree on $V^{l_s}(id_{all})$ and find neighbors within radius r for each selected point.
- 3: Initialize an empty list for extreme points $\mathcal{I}_{ext}$.
- 4: **for** each $i \in \mathcal{I}_{add}$ **do**
- 5: Retrieve neighbor indices $\mathcal{N}_i$ from the KD-tree.
- 6: Extract absolute errors $E_{\mathcal{N}_i}$ in the neighborhood.
- 7: **if** $|E_i| \geq \max(|E_{\mathcal{N}_i}|)$ **then** $\mathcal{I}_{ext} \leftarrow i$
- 8: **if** $|\mathcal{I}_{ext}| = 0$ **then return** $\emptyset, \emptyset, \emptyset, \emptyset$
- 9: Compute new ERBF parameters using Eq. (16):

Output: C^a, W^a, d^a, A^a .

The process alternates cyclically until the predefined number of training epochs T^e is reached. The detailed process for adding basis functions is given in Algorithm 2. First, calculate the error vector for each training point in V^{l_s} , and select the subset indexed by id_{all} , which is computed using Eq. (7):

$$E = \Psi(V^{l_s}, \tilde{\Theta})[id_{all}] - T^{l_s}[id_{all}]. \quad (15)$$

Then, we select the points with larger absolute errors based on τ_m . Among these candidates, points whose absolute error is larger than that of all other points in their local neighborhood are identified as extreme error points. The parameters of the newly added basis functions are computed as follows:

$$\begin{aligned} C^a &= V^{l_s}[id_{all}][\mathcal{I}_{ext}]; \quad d^a = \sqrt{-\ln(\varepsilon/|E[\mathcal{I}_{ext}]|)/(\bar{d})^2}; \\ W^a &= -\text{sign}(E[\mathcal{I}_{ext}]) \cdot \sqrt{|E[\mathcal{I}_{ext}]|}; \quad A^a = \mathbf{0}, \end{aligned} \quad (16)$$

where $\mathcal{I}_{ext}$ denotes the indices of the extreme error points. The length of $\mathcal{I}_{ext}$ is k^a , which indicates the number of newly added basis functions. $C^a \in \mathbb{R}^{k^a \times 3}$ represents the centers of the newly added basis functions. $W^a \in \mathbb{R}^{k^a \times 1}$ represents the corresponding weights. $d^a \in \mathbb{R}^{k^a}$ represents the shapes of the newly added basis functions. Each element in d^a is expanded into a 3×3 diagonal matrix to form $D^a \in \mathbb{R}^{k^a \times 3 \times 3}$. $A^a \in \mathbb{R}^{k^a \times 3}$ denotes the rotation angles, which are initialized to zero. $\bar{d}_i$ refers to the i -th element of $\bar{d}$, which is defined as the minimum distance from the center c_i^a to the sampling points P , $\bar{d}_i = \min \|c_i^a - P\|_2, i = 1, 2, \dots, k^a$. $\text{sign}(\cdot)$ is the sign function. Next, the SE-RBFNet parameters will be updated according to Eq. (17), and optimization will continue.

$$\begin{aligned} C &\leftarrow [C; C^a]; \quad W \leftarrow [W; W^a]; \\ \bar{D} &\leftarrow [\bar{D}; D^a]; \quad A \leftarrow [A; A^a]. \end{aligned} \quad (17)$$

In Eq. (15), E is defined as the difference between the predicted and true values. If the local error is positive, the newly added basis function must provide a negative contribution to offset this residual. However, in Eq. (3), the weights are restricted to squared values (w_j^2), which in themselves ensure non-negativity. To overcome this functional restriction while preserving optimization stability, we modify Eq. (3) as follows:

$$\Psi(v_i, \tilde{\Theta}) = \sum_{j=1}^M w_j |w_j| e^{-\|D_j R_j(v_i - c_j)\|_2^2}. \quad (18)$$

This modification introduces two critical numerical and stability effects. First, the signed formulation breaks the non-negativity constraint, enabling the network to learn both positive and negative contributions during the adaptive basis addition phases, thereby enhancing error correction capability. Second, regarding optimization stability, compared to directly using a simple linear weight w_j , the $w_j |w_j|$ formulation maintains a C^1 -continuous gradient ($2|w_j|$) with respect to w_j . This property ensures that the gradient naturally vanishes as the weight approaches zero, preventing gradient discontinuities and ensuring numerical stability when dynamically adding or pruning basis functions around the zero-weight threshold.

During SE-RBFNet optimization, some basis functions have coefficients that become so small that their contribution to the overall output can be ignored. We introduce a threshold τ_d and remove the basis functions whose w coefficients are less than τ_d at the specified iteration interval. That is:

$$\begin{aligned} \tilde{W} &= \{w_j \in W \mid |w_j| < \tau_d\}; \quad \tilde{C} = \{c_j \in C \mid |w_j| < \tau_d\}; \\ \tilde{D} &= \{D_j \in \bar{D} \mid |w_j| < \tau_d\}; \quad \tilde{A} = \{a_j \in A \mid |w_j| < \tau_d\}. \end{aligned} \quad (19)$$

Deleting these basis functions allows the network to automatically adjust its complexity, increasing sparsity and computational efficiency. When training reaches T^e epochs, the optimization focuses exclusively on the $\mathcal{L}_2$ loss term to further improve the precision of the SDF approximation, continuing until the maximum iteration count M^e is reached. At this stage, no further basis function additions or deletions occur, and $\mathcal{L}_1$ regularization is no longer applied.

4) *Additional Techniques for Training Acceleration:* In practice, the input set G can contain a very large number of points, and optimizing $\mathcal{L}$ is a high-dimensional nonlinear and nonconvex problem. To reduce computational burden and increase convergence speed, we primarily employ the following techniques. In Eq. (5), F has a high sparsity due to the fact that the ERBF value of points far from the center of the basis function approaches 0. To improve efficiency, we introduce a point screening strategy based on nearest neighbor. Specifically, for each ERBF, we only calculate those points that are relatively close to the center of the basis function, with the distance range defined as follows:

$$\|c_j - v_i\|_2 \leq \sqrt{-\ln(\varepsilon)/\lambda_j}, \quad (20)$$

where c_j is the center of the j -th ERBF, $v_i \in V$, and $\lambda_j = \min(d_{j1}^2, d_{j2}^2, d_{j3}^2)$. The entries of F can be re-expressed as:

$$F_{ij} = \begin{cases} \hat{\Phi}(v_i, \Theta^{(j)}), & \text{if } \|c_j - v_i\|_2 \leq \sqrt{-\ln(\varepsilon)/\lambda_j} \\ 0, & \text{otherwise.} \end{cases} \quad (21)$$

Algorithm 3 SE-RBFNet Sparse Optimization Algorithm

Input: Sampling points P on an implicit surface, the optimization parameters follow the default settings in Section IV-A.

- 1: Construct an octree from the sampling points P , and compute the SDF at each octree grid point G , which is then normalized according to Eq. (4) to obtain $\hat{S}$;
- 2: Using l_s , obtain the initial training points V^{l_s} and corresponding target labels T^{l_s} , and then calculate the initial ERBF parameters $\tilde{\Theta}$ using Algorithm 1.
- 3: Set $l_{1_optim}=\text{False}$, $add_levels=\text{False}$, $e_p = 1$;
- 4: **while** $e_p < M^e$ **do**
- 5: $e_p \leftarrow e_p + 1$,
- 6: **if** add_levels and $l_s < l_{max}$ **then**
- 7: The grid points from the $(l_s + 1)$ -th layer are selected and appended to the training set, followed by updating: $V^{l_s} \leftarrow V^{l_s+1}$, $T^{l_s} \leftarrow T^{l_s+1}$ and $l_s \leftarrow l_s + 1$.
- 8: Set $add_levels = \text{False}$.
- 9: Shuffle training points and corresponding labels.
- 10: **for** $i = 0$ to N with step B_s **do**
- 11: Sample batch $V_b^{l_s}$ and $T_b^{l_s}$
- 12: Use Eq. (5) to calculate: $\mathbf{O} \leftarrow \text{SE-RBFNet}(V_b^{l_s})$
- 13: Calculate id_{all} (Eq. (7)), and the losses (Eq. (6)).
- 14: Compute the gradient $\nabla \mathcal{L}(\tilde{\Theta})$ (see Appendix A).
- 15: **if** $l_{1_optim}=\text{True}$ and Eq. (10) is True **then**
- 16: Use Eq. (11) to compute α
- 17: Update loss: $\mathcal{L} \leftarrow \alpha \mathcal{L}_2 + (1 - \alpha) \mathcal{L}_1$
- 18: **else**: Update loss: $\mathcal{L} \leftarrow \mathcal{L}_2$
- 19: Update network parameters $\tilde{\Theta}$ via Adam [56].
- 20: **if** $e_p \% k^{l_2} = 0$ and $l_{1_optim}=\text{True}$ and $e_p < T^e$ **then**
- 21: Delete invalid basis functions using Eq. (19).
- 22: **if** $l_{1_optim}=\text{False}$ and Eq. (9) is True **then**
- 23: Update $l_{1_optim}=\text{True}$.
- 24: **if** $l_{1_optim}=\text{True}$ and Eq. (14) is True **then**
- 25: Add new ERBFs according to Eq. 16.
- 26: Update $l_{1_optim}=\text{False}$.
- 27: **if** $e_p = T^e$ **then**
- 28: During the iterations from T^e to M^e , steps 20 to 26 are skipped, and only $\mathcal{L}_2$ optimization is performed. $l_{1_optim}=\text{False}$, $lr = 10^{-3}$ and adjusted using a cosine annealing schedule, with a minimum value of 10^{-5} .
- 29: **if** e_p in $[400, 800]$ **then** $add_levels = \text{True}$

Output: Network parameters $\tilde{\Theta}$, predicted SDF $\mathbf{O}$, and a surface extracted by the marching cubes algorithm [21].

The matrix $\mathbf{F}$ is used not only for the forward calculation of the network but also plays a role in the calculation of the gradient of each parameter. To avoid repeated calculations, we explicitly compute the gradient of each parameter based on $\mathbf{F}$ and the loss $\mathcal{L}$ (Eq. (6)) according to the chain rule. The detailed derivations of the gradients can be found in the Appendix. When the value of $\mathbf{F}$ at a certain location is close to zero, the corresponding gradient also approaches zero. Consequently, to improve the calculation efficiency, we take advantage of the nearest-neighbor strategy described in Eq. (20) and calculate only the gradient values of the points near the centers of the basis functions.

Finally, due to the inherent independence of each ERBF in the network, we leverage CUDA to parallelize the computation of the value of the loss function and the gradient with respect

to each parameter in Θ . This independence allows each ERBF to be computed independently and concurrently, leading to a significant reduction in computational time. Note that SE-RBFNet is optimized using the Adam method [56], with the complete optimization details provided in Algorithm 3.

Remark 1: The proposed SE-RBFNet introduces a novel framework for compact and sparse representation of pre-computed SDFs, distinguishing itself from existing methods that focus primarily on direct SDF estimation from raw data. By decoupling SDF generation from representation, SE-RBFNet operates in a data-agnostic manner, capable of encoding SDFs from any source. It leverages flexible ERBFs with learnable centers, orientations, and anisotropic scales to adaptively fit complex geometries using fewer parameters. A dynamic multi-objective optimization strategy is developed to jointly minimize approximation error and promote sparsity, while a hierarchical coarse-to-fine refinement process and adaptive basis function addition further enhance efficiency and accuracy. Localized computation via nearest-neighbor filtering and CUDA-based parallelism ensures scalability, making SE-RBFNet a compact, interpretable, and efficient alternative to existing implicit surface representation techniques.

IV. EXPERIMENT AND ANALYSIS

In this section, we evaluate the accuracy, sparsity, and efficiency of SE-RBFNet. Note that SE-RBFNet is designed to approximate precomputed SDFs of implicit surfaces rather than reconstruct surfaces directly from raw point clouds. Therefore, we do not perform direct comparisons with full surface reconstruction methods. We focus on two complementary experiments: one using ground-truth SDFs from triangle meshes, and the other using predicted SDFs from neural implicit (or any other SDF-generating methods).

First, to assess accuracy and parameter efficiency, we compare SE-RBFNet with SparseRBF [1] and a recent neural implicit baseline, WIRE [15]. In the original SparseRBF, the inputs consist of surface points (with zero SDF), offset points, and center points, where the SDF values are approximated from Voronoi diagram poles [57] rather than computed exactly. To ensure a fair comparison, we instead provide all three methods with identical input data: area-weighted uniformly sampled surface points (SDF = 0) and octree grid points, whose SDF values are defined as the directional point-to-mesh distances and computed directly from the triangle mesh using the GPU-accelerated Kaolin API [58]. The octree construction follows [39]. This setup eliminates potential inaccuracies introduced by approximate SDF estimation and ensures that the comparison focuses solely on the parameter efficiency and approximation accuracy of different methods. For WIRE, we employ its official implementation and train it for 1,000 epochs. To keep its parameter budget comparable to SE-RBFNet, we reduce the original architecture from two hidden layers to one hidden layer while keeping 256 hidden neurons, resulting in approximately 33.8K trainable parameters.

Second, to demonstrate the generality of SE-RBFNet, we evaluate its ability to sparsely approximate arbitrary SDF data regardless of the source or accuracy of the SDF computation. In this scenario, SDF values on the octree grid

points are obtained using neural implicit methods or arbitrary SDF-generation methods. SE-RBFNet further sparsifies the representation by approximating the input SDFs using significantly fewer parameters, while preserving high accuracy of the surface extracted from the approximated SDF. This highlights the compatibility of SE-RBFNet with existing SDF generation methods and its advantage in reducing storage and transmission costs.

Together, these experiments allow us to evaluate both the sparsity and parameter efficiency of SE-RBFNet, as well as its generality across different SDF sources. All experiments were performed on Ubuntu 18.04 with an Intel Core i9-7940X CPU @ 3.1GHz and an NVIDIA GeForce RTX 2080 GPU.

A. Datasets and Parameter Settings

We evaluated SE-RBFNet on four common datasets: the Famous dataset [8], MeshSegBenchmark (MeshSeg) [59], ABC [60], and Thingi10K [61]. The Famous dataset contains 22 well-known models, and MeshSeg includes 380 meshes from 19 categories. Since the meshes within each category share similar characteristics, we randomly selected 60 meshes for our experiments. The ABC dataset contains approximately 1 million CAD models, while Thingi10K includes 10,000 3D printable meshes. To balance efficiency and representativeness, we follow the experimental setup of Points2Surf [8] and select a subset of 100 meshes from each dataset for our experiments.

SE-RBFNet parameters were set as: batch size $B_s = 10000$, max epoch $M^e = 2000$, $T^e = 1600$, learning rate $lr = 0.01$, octree max depth $l_{max} = 10$, $\gamma = 10^{-3}$, $\varepsilon = 10^{-7}$. Training point filtering thresholds: $\tau_1 = 0.9$ and $\tau_2 = 1.1$. $\tau^{l_1} = 5$ and $k^{l_1} = 50$ are used to determine whether the number of basis functions has stabilized, while $\tau^{l_2} = 0.5$ and $k^{l_2} = 10$ are used to judge the stability of the $\mathcal{L}_2$ loss. Sparsity control: $\tau_m = 0.02$. $\tau_d = 0.01$ is used as the deletion threshold for ERBFs. The initial training layer l_s is set by default to the third-to-last layer. However, it must also satisfy the following condition: $|G_{in}^{l_s}| > 100$, otherwise, l_s is incremented by one, i.e., $l_s = l_s + 1$. This constraint ensures that the number of interior grid points in the l_s layer must be greater than 100 to avoid having too few initial ERBFs. Experiments have found that l_s is generally 6 or 7. Unless otherwise specified above, all remaining hyperparameters of the comparison methods follow their official default settings.

B. Evaluation Metrics

In our setting, SE-RBFNet aims to approximate a precomputed ground-truth SDF. To ensure a fair and meaningful evaluation, the ground-truth SDF is computed directly from the original triangle mesh geometry, avoiding intermediate errors of SDF values. It is important to note that our loss function assigns greater emphasis to points close to the surface (see Eq. (7)). Therefore, accurately approximating SDF values on all grid points is not the purpose of SE-RBFNet. Instead, SE-RBFNet aims to preserve the geometry of the zero-level set (i.e., the implicit surface). Accordingly, our evaluation includes not only the direct approximation error of the SDF values near the surface, but also the geometric deviation between the

zero-level set surface output by SE-RBFNet and the isosurface from the ground-truth SDF, which is measured by surface-to-surface distances and normal consistency. Specifically, the Root Mean Square Error (RMSE) is first computed using the predicted normalized SDF values and the corresponding ground-truth normalized SDF values at the near-surface grid points indexed by id_0 , which are defined in Eq. (7). Next, the network-predicted normalized SDF values are transformed back to the physical SDF range according to Eq. (4). Using the same octree structure, the iso-surface at the zero level is then extracted from the transformed ERBF-approximated SDF and compared with the ground truth surface directly obtained from the ground-truth SDF, which represents the true zero-level set. The explicit surface extraction from the SDF values on grids is performed using the method provided in [39]. We use the following metrics—RMSE, Hausdorff Distance (HD) [39], Chamfer Distance (CD) [8], and Cosine Similarity (CS) [3]—to evaluate the fidelity of the SDF representation. RMSE quantifies the preservation of signed distance values in the vicinity of the surface, whereas HD, CD, and CS assess the geometric consistency between the extracted zero-level set surface and the ground-truth isosurface. Lower RMSE, HD, and CD values and higher CS values indicate that the surfaces extracted from the approximated SDF closely match the true zero-level set, thus reflecting higher fidelity of the SDF representation.

Suppose that $\mathcal{M}_t$ and $\mathcal{M}_r$ denote the ground truth surface and the surface extracted from the approximated SDF, respectively. Let $\hat{S}_{pred}$ and $\hat{S}_{gt}$ denote the predicted and ground-truth normalized SDF values, respectively. The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{|id_0|} \sum_{i \in id_0} \left(\hat{S}_{pred}^{(i)} - \hat{S}_{gt}^{(i)} \right)^2},$$

Both HD and CD are computed using point-to-point distances, where the distance from a point x to a surface $\mathcal{M}$ is defined as the minimum Euclidean distance to any point on that surface. Formally, the distance is defined as:

$$d(x, \mathcal{M}) = \min_{y \in \mathcal{M}} \|x - y\|_2,$$

where $y \in \mathcal{M}$ denotes a surface point.

The HD measures the worst-case deviation between the surface $\mathcal{M}_t$ and $\mathcal{M}_r$, and is defined as:

$$\text{HD}(\mathcal{M}_t, \mathcal{M}_r) = \max \left(\max_{x \in \mathcal{M}_t} d(x, \mathcal{M}_r), \max_{y \in \mathcal{M}_r} d(y, \mathcal{M}_t) \right).$$

The CD measures the bidirectional average deviation between the surface $\mathcal{M}_t$ and $\mathcal{M}_r$, and is computed as:

$$\text{CD}(\mathcal{M}_t, \mathcal{M}_r) = \frac{1}{2} \sum_{k \in \{t, r\}} \frac{1}{|\mathcal{M}_k|} \sum_{x \in \mathcal{M}_k} d(x, \mathcal{M}_{\{r, t\} \setminus k}),$$

where $|\cdot|$ denotes the number of points sampled uniformly on the surface.

In addition to geometric distances, normal consistency is another critical factor in assessing surface representation quality. To evaluate the similarity between normal vectors, we compute the cosine similarity between the normals of corresponding

TABLE I
AVERAGE APPROXIMATION ACCURACY, TIME AND MEMORY USAGE, AND SPARSITY OF IMPLICIT SURFACE REPRESENTATIONS ON FOUR DATASETS.

Dataset	Method	Surface Geometric Metrics				Time and Memory Usage			Parameter Comparison		
		RMSE	HD	CD	CS	Time (s)	CPU (GB)	GPU (GB)	Final Basis	Param	Ratio
ABC	Ours	0.0026	0.0071	0.0021	0.9910	119.0	3.76	7.50	1170	11700	/
	WIRE	0.0064	0.0239	0.0037	0.9672	255.1	3.78	2.23	/	33848	0.34
	SparseRBF	0.0207	0.0529	0.0049	0.9616	923.8	12.49	N/A	6079	30395	0.38
Famous	Ours	0.0031	0.0058	0.0019	0.9846	167.8	3.98	7.23	2050	20500	/
	WIRE	0.0063	0.0265	0.0024	0.9664	246.3	3.99	2.23	/	33848	0.60
	SparseRBF	0.0127	0.0384	0.0022	0.9643	918.6	11.77	N/A	5969	29845	0.68
MeshSeg	Ours	0.0029	0.0086	0.0024	0.9957	103.7	3.79	6.44	582	5820	/
	WIRE	0.0054	0.0156	0.0031	0.9870	237.3	3.78	2.23	/	33848	0.17
	SparseRBF	0.0067	0.0195	0.0026	0.9910	797.5	9.27	N/A	4961	24805	0.23
Thing10k	Ours	0.0027	0.0066	0.0018	0.9940	140.8	3.78	7.45	1097	10970	/
	WIRE	0.0059	0.0103	0.0019	0.9797	258.1	3.81	2.23	/	33848	0.32
	SparseRBF	0.0085	0.0191	0.0021	0.9844	895.5	11.15	N/A	5086	25430	0.43

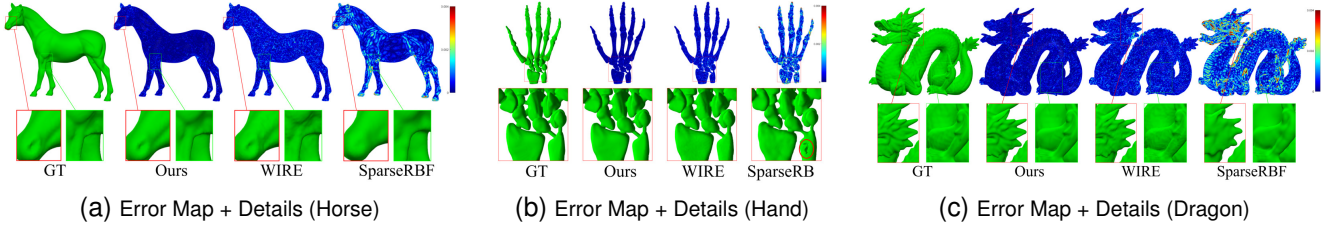


Fig. 2. Comparison of implicit surface approximation results obtained by our method, WIRE and SparseRBF. (a), (b), and (c) are all from the Famous dataset.

nearest-neighbor points:

$$CS(\mathcal{M}_t, \mathcal{M}_r) = \sum_{k \in \{t, r\}} \frac{1}{2|\mathcal{M}_k|} \sum_{x \in \mathcal{M}_k} |v(x) \cdot v(\text{closest}_{\mathcal{M}_{\{t, r\} \setminus k}}(x))|,$$

where $v(x)$ is the surface normal at point x , and $\text{closest}_{\mathcal{M}_r}(x)$ is the point in $\mathcal{M}_r$ closest to x . For HD, CD, and CS, 10^5 points are uniformly sampled from each surface for evaluation.

C. Sparsity and Parameter Efficiency Evaluation

In this experiment, we evaluate the accuracy and parameter efficiency of SE-RBFNet by comparing it with SparseRBF [1] and WIRE [15]. SparseRBF represents SDFs using a sparse set of isotropic RBFs, whereas WIRE parameterizes the SDF using a compact neural implicit representation with wavelet activations. For each mesh, 40K points are first sampled using the `sample_surface_even` function in the Trimesh [62] API, which performs area-weighted uniform sampling over the mesh surface. These sampled points are then used to construct an octree, and the ground-truth SDF values at the octree grid points (defined by the directional point-to-mesh distances) are precomputed directly from the mesh geometry using the Kaolin API [58], serving as the training samples.

We evaluated the generality and robustness of SE-RBFNet on the ABC, Famous, MeshSeg, and Thing10K datasets. Table I reports a comprehensive comparison among SE-RBFNet, SparseRBF, and WIRE in terms of SDF preservation accuracy, geometric reconstruction quality, sparsity, and computational efficiency (including time and memory usage). In terms of surface quality and approximation accuracy, our method consistently achieves significantly lower RMSE, HD, and CD

values, and higher CS scores across all datasets, indicating more accurate implicit surface representations and better preservation of fine geometric details. WIRE, as a representative compact neural implicit representation method, is able to capture the overall surface geometry. However, it still exhibits larger deviations than SE-RBFNet, resulting in slightly inferior geometric accuracy. Similarly, SparseRBF tends to produce over-smoothed surface representations, which leads to relatively lower approximation accuracy and reconstruction quality. Beyond accuracy, our method also demonstrates strong sparsity advantages. Using as few parameters as possible to represent the SDFs of implicit surfaces is crucial for reducing storage and computational overhead. As demonstrated in [1], SparseRBF achieves substantially better sparsity than prior RBF-based methods [4], [7], [27] by combining medial-axis-based center selection with a linear $\mathcal{L}_1$ sparsity optimization strategy. The right columns of Table I compare the average numbers of effective basis functions and parameters used by our method with those of the baselines. Here, Final Basis denotes the number of effective basis functions remaining after the optimization. In addition, Param denotes the number of parameters required to represent an implicit surface. Each ERBF uses 10 parameters (center, rotation angles, shapes, and weight), compared with 5 per isotropic RBF in SparseRBF, while WIRE utilizes fixed network weights ($\approx 33.8K$). Since the parameter count directly reflects the sparsity of an implicit representation, the Ratio column reports the proportion of final parameters required by SE-RBFNet relative to each baseline method. The results demonstrate that, compared with WIRE, SE-RBFNet requires on average only about 36% of the pa-

TABLE II

AVERAGE APPROXIMATION ACCURACY, TIME AND MEMORY USAGE, AND PARAMETER COMPARISON OF SE-RBFNET, WIRE, AND SPARSERBF FOR IMPLICIT SURFACE REPRESENTATIONS ON FOUR DATASETS. THE GROUND-TRUTH SDFS USED IN THIS EVALUATION ARE OBTAINED FROM NSH [10].

Dataset	Method	Surface Geometric Metrics				Time and Memory Usage			Parameter Comparison		
		RMSE	HD	CD	CS	Time (s)	CPU (GB)	GPU (GB)	Final Basis	Param	Ratio
ABC	Ours	0.0059	0.0214	0.0029	0.9846	110.6	3.81	7.19	1865	18650	/
	WIRE	0.0106	0.0283	0.0030	0.9801	259.7	3.76	2.23	/	33848	0.55
	SparseRBF	0.0266	0.2128	0.0084	0.9767	955.7	11.55	N/A	10163	50815	0.36
Famous	Ours	0.0044	0.0247	0.0027	0.9823	134.6	3.96	7.26	2209	22090	/
	WIRE	0.0092	0.0269	0.0028	0.9747	252.9	3.87	2.23	/	33848	0.65
	SparseRBF	0.0221	0.0521	0.0031	0.9727	991.0	11.71	N/A	10287	51435	0.43
MeshSeg	Ours	0.0038	0.0258	0.0034	0.9912	94.9	4.00	6.70	751	7510	/
	WIRE	0.0055	0.0267	0.0033	0.9896	236.2	3.77	2.23	/	33848	0.22
	SparseRBF	0.0191	0.0409	0.0046	0.9868	791.1	9.28	N/A	6607	33035	0.22
Thing10k	Ours	0.0036	0.0285	0.0026	0.9920	112.5	3.95	5.67	1458	14580	/
	WIRE	0.0063	0.0376	0.0027	0.9892	256.4	3.79	2.23	/	33848	0.43
	SparseRBF	0.0299	0.0821	0.0044	0.9867	865.7	11.15	N/A	8821	44105	0.33

rameters while achieving consistently better RMSE, HD, CD, and CS values. Relative to SparseRBF, our method reduces the number of basis functions by about 78% on average, corresponding to only about 44% of the parameters required by SparseRBF, while still maintaining superior accuracy. This reduction in parameter complexity translates to significant advantages in computational speed. The middle columns of Table I report the training time and peak memory usage of each method, our method achieves approximately $2\times$ speedups over WIRE and $6\text{--}7\times$ speedups over SparseRBF. Regarding CPU memory usage, SE-RBFNet requires only about 3.76–3.98 GB, which is comparable to WIRE and substantially lower than the 9.27–12.49 GB required by SparseRBF. Although SE-RBFNet allocates more GPU memory during the CUDA-based ERBF optimization, it achieves substantially faster training and better approximation accuracy. Note that the time values reported in Table I refer only to the optimization/training process and do not include the time required for data preparation, extracting the explicit surface from the SDF, etc. Taken together, for these four datasets, SE-RBFNet achieves a favorable balance between approximation accuracy, sparsity, and efficiency.

Fig. 2 shows the extracted surfaces for several representative cases. As shown in the figure, the surfaces obtained from SE-RBFNet’s approximated SDF more closely match the ground-truth surface, effectively preserving geometric details while avoiding artifacts. WIRE also achieves competitive implicit surface representation quality. However, the corresponding error maps reveal that WIRE still exhibits larger deviations than SE-RBFNet, particularly in regions containing rich geometric details. In contrast, SparseRBF often oversmooths surfaces or produces holes (e.g., Fig. 2b). These qualitative observations are consistent with the quantitative results in Table I and further demonstrate the effectiveness of SE-RBFNet in preserving both near-surface SDF fidelity and the geometric accuracy of the extracted zero-level set surfaces.

D. General SDF Sparse Representation

SE-RBFNet is a general sparse implicit surface representation method that can be trained on arbitrary SDF samples,

independent of their source (e.g., point clouds, meshes, analytical SDFs, or neural SDFs). To further validate its universality, we apply SE-RBFNet to SDF data generated by the neural implicit method Neural-Singular-Hessian (NSH) [10]. Specifically, NSH outputs SDF values at octree nodes, upon which SE-RBFNet constructs its sparse representation. We further compare SE-RBFNet against WIRE and SparseRBF. For fairness, all methods are evaluated using the same input data to assess approximation quality. Quantitative comparisons are reported in Table II. For consistency, explicit surfaces are extracted under identical octree grids using the program provided in [39], and comparisons are made against the target surfaces generated from NSH under identical conditions. Across the four benchmark datasets, SE-RBFNet achieves better approximation quality than WIRE in most cases, while consistently outperforming SparseRBF. More importantly, compared with WIRE, SE-RBFNet requires only 46% of the parameters on average while reducing the optimization time by more than a factor of two. Several representative cases are shown in Fig. 3. While both SE-RBFNet and WIRE faithfully reproduce the target surfaces generated by NSH, the error maps reveal that SE-RBFNet generally yields smaller local deviations, particularly around complex geometric structures. SparseRBF, in contrast, produces noticeably larger errors and struggles to preserve fine details in challenging regions. Notably, NSH itself requires average 264K parameters to represent the SDF on grids of the test cases in Table II, whereas SE-RBFNet achieves high-fidelity sparse representation using fewer than 2K basis functions ($< 20\text{K}$ parameters), yielding over $10\times$ parameter compression. These results demonstrate that SE-RBFNet not only outperforms previous sparse representation methods, but also serves as an effective post hoc compression module for neural implicit representations, providing a lightweight and efficient sparse representation well-suited for scenarios with storage and computation constraints.

E. Parameter Analysis

We analyze the impact of key parameters on SE-RBFNet, focusing on the threshold τ_m and the batch size B_s . Other

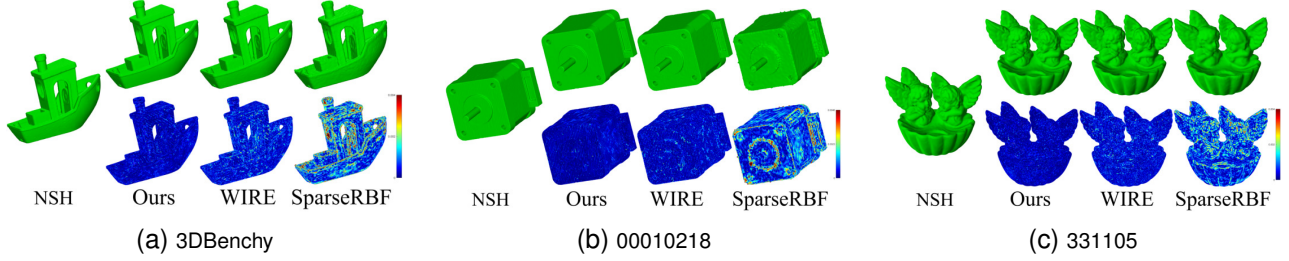
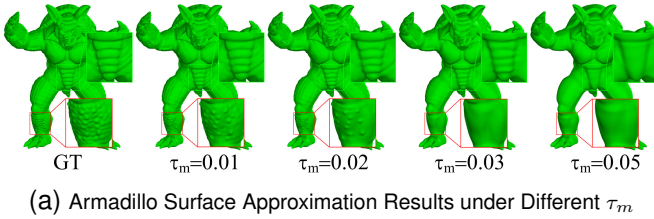


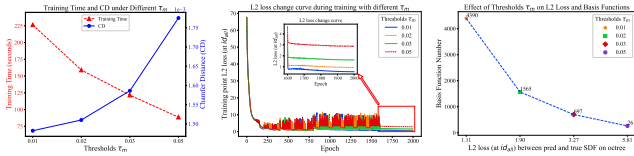
Fig. 3. Comparison of the implicit surfaces extracted from the approximated SDFs by our method, WIRE, and SparseRBF. (a) is from the Famous dataset, (b) is from the ABC dataset, while (c) is from the Thing10k dataset.

parameters follow the settings in Section IV-A.

1) *Impact of τ_m on Approximation Results:* The threshold τ_m , introduced in Algorithm 2 and Eq. (10), controls both the detection of local error extrema and the activation of sparse optimization. As shown in Fig. 4, larger τ_m triggers earlier pruning, reducing the number of basis functions but also discarding surface details (e.g., stripes and bumps). Fig. 4b confirms that smaller τ_m yields lower CD and $\mathcal{L}_2$ loss but requires more basis functions and longer training. Empirically, $\tau_m = 0.02$ offers a good balance between accuracy and efficiency: it achieves a convergence loss close to that of $\tau_m = 0.01$ while maintaining a similar CD, but requires only one-third the number of basis functions and significantly reduces computation time.



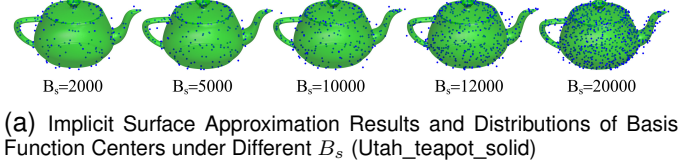
(a) Armadillo Surface Approximation Results under Different τ_m



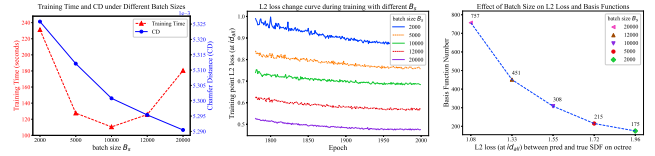
(b) CD, Time, $\mathcal{L}_2$ Loss, and Basis Function Count vs. τ_m

Fig. 4. Effect of τ_m on implicit surface approximation results, (a) Armadillo implicit surface approximation results under different τ_m . (b) shows the corresponding metric variation curves. From left to right, the curves show the training time (left y-axis) and CD (right y-axis), loss convergence, and the number of effective basis functions for different τ_m .

2) *Impact of B_s on Approximation Results:* Batch size B_s also affects both stability and efficiency. As illustrated in Fig. 5a, increasing B_s leads to more effective basis functions: as shown in the rightmost plot of Fig. 5b, the number of basis functions increases from 175 at $B_s = 2000$ to 757 at $B_s = 20000$. This is attributed to training stability: small B_s induces large oscillations in the gradient and loss, leading to excessive pruning, whereas large B_s stabilizes training and increases the number of effective basis functions. Fig. 5b shows that larger B_s reduces CD and converged loss, but overly large values increase training time due to the growth in basis functions.



(a) Implicit Surface Approximation Results and Distributions of Basis Function Centers under Different B_s (Utah_teapot_solid)



(b) CD, Time, $\mathcal{L}_2$ Loss, and Basis Function Count vs. B_s

Fig. 5. Effect of batch size B_s on implicit surface approximation results. (a) the implicit surface approximation results and the distribution of the basis function centers for the Utah_teapot_solid surface under different batch sizes. (b) shows the corresponding metric curves, from left to right: training time (left y-axis) and CD (right y-axis), loss convergence, and the number of effective basis functions.

F. Ablation Studies

To comprehensively validate the effectiveness of the core structural designs in SE-RBFNet, ablation studies were conducted focusing on three key aspects: (1) the joint optimization of ERBF parameters, (2) the hierarchical optimization strategy, and (3) the dynamic multi-objective optimization mechanism. For all ablation experiments, except for the parameter being studied, all other parameters and experimental settings were kept unchanged, including the training epochs, stopping criteria, and optimization parameters. The quantitative results of these ablation experiments, averaged across the four datasets, are summarized in Table III.

TABLE III
QUANTITATIVE ABLATION STUDY OF SE-RBFNET. ALL VALUES ARE AVERAGED OVER THE FOUR DATASETS.

Variant	Final Basis	Time (s)	RMSE
(A) W Only	3992	191.3	0.01741
(B) $C + W$	2998	158.1	0.00622
(C) $C + D + W$	1489	125.1	0.00433
(D) Without Hierarchical Opt.	725	310.4	0.00396
(E) Fixed Loss Weight $\lambda = 0.1$	74	87.9	0.01720
(F) Fixed Loss Weight $\lambda = 0.01$	205	88.3	0.00817
(G) Fixed Loss Weight $\lambda = 0.001$	533	103.1	0.00395
(H) Fixed Loss Weight $\lambda = 0.0$	2473	175.9	0.00219
(I) Full SE-RBFNet	1101	127.9	0.00274

(1) Joint optimization of ERBF parameters. Regarding the joint optimization of ERBF parameters, we considered three restricted variants in Table III. Variant (A) optimizes only the weights $\mathbf{W}$. Similarly, Variant (B) optimizes only the centers $\mathbf{C}$ and weights $\mathbf{W}$. Variant (C) optimizes the centers $\mathbf{C}$, scales $\mathbf{D}$, and weights $\mathbf{W}$. As shown in Table III, the full SE-RBFNet, i.e., Variant (I), achieves the lowest RMSE of 0.00274. When the rotation angles are fixed (Variant C), the RMSE increases from 0.00274 to 0.00433. Further removing the optimization of scales (Variant B) results in a substantial degradation in approximation quality (RMSE = 0.00622). Finally, optimizing only the weights (Variant A) leads to the worst performance (RMSE = 0.01741). These results demonstrate that jointly optimizing the ERBF centers, rotations, scales, and weights is crucial for achieving both high approximation accuracy and compact sparse representations.

(2) Hierarchical optimization strategy. To evaluate the effectiveness of the hierarchical optimization strategy, Variant (D) was designed to skip the coarse-to-fine optimization process and directly optimizes the network on the finest octree level from scratch. Without hierarchical guidance, network optimization becomes highly inefficient. As shown in Table III, the optimization time for Variant (D) increased from 127.9 s to 310.4 s. This increase mainly stems from the inscribed-sphere initialization on the finest octree level, which introduces a significantly larger number of initial basis functions (7981 versus 467). Moreover, directly fitting fine-scale details without first establishing the low-frequency structure makes the optimization more prone to local minima, leading to inferior approximation accuracy (RMSE = 0.00396 vs. 0.00274). This confirms that the proposed hierarchical strategy effectively controls optimization complexity, accelerates convergence, and stabilizes the numerical approximation.

(3) Dynamic multi-objective optimization. To evaluate the effectiveness of dynamic multi-objective optimization, the Pareto-based dynamic mechanism was replaced with a conventional fixed-weight objective, $\mathcal{L} = \mathcal{L}_2 + \lambda \mathcal{L}_1$, with different values of λ tested in Variants (E)–(H). As shown in Table III, the fixed-weight strategy is highly sensitive to the choice of λ . When the regularization weight is too large, e.g., $\lambda = 0.1$, the resulting representations become overly sparse and suffer from poor approximation quality (RMSE = 0.01720). Reducing λ improves the approximation accuracy, but it also increases the number of retained basis functions. In particular, when $\lambda = 0$, the $\mathcal{L}_1$ regularization term is removed entirely, yielding the lowest RMSE of 0.00219 but requiring 2473 basis functions. In contrast, the full SE-RBFNet achieves a more favorable balance between approximation accuracy and sparsity. Although its RMSE slightly increases to 0.00274, it requires only 1101 basis functions, reducing the basis count by more than 55% compared with the $\lambda = 0$ variant. These results demonstrate that fixed loss weighting requires careful trial-and-error to balance accuracy and sparsity, whereas the proposed Pareto-based dynamic optimization adaptively adjusts the loss weights during training and avoids manual tuning. Consequently, it achieves a robust trade-off across diverse geometries while maintaining high approximation precision and compact representation.

V. CONCLUSION

This paper has introduced SE-RBFNet, a sparse ellipsoidal radial basis function network designed for efficient and compact approximation of SDFs of implicit surfaces. Unlike existing methods that aim to generate SDFs directly, the proposed approach has focused on the sparse approximation of precomputed SDF samples obtained from diverse sources, including point clouds, triangle meshes, analytical SDFs, and neural models. Specifically, the implicit surface representation problem has been reformulated as a sparse nonlinear optimization task, where SE-RBFNet approximates spatially sampled SDF values using a compact set of ellipsoidal RBFs, thereby preserving the zero-level set while substantially reducing the number of parameters. To balance approximation accuracy and model complexity, a dynamic multi-objective optimization framework has been proposed by incorporating a coarse-to-fine hierarchical strategy and nearest-neighbor filtering to improve efficiency and reduce redundancy. In addition, an adaptive basis function adjustment mechanism has been developed, which iteratively selects new centers from high-error regions and updates ellipsoidal parameters to better capture complex surface geometry. Extensive experiments have demonstrated that SE-RBFNet achieves high-fidelity approximation of SDFs with significantly fewer parameters, thereby reducing both storage and computational overhead.

APPENDIX A

GRADIENT COMPUTATION FOR SE-RBFNET

SE-RBFNet addresses the sparse approximation of SDFs for implicit surfaces by optimizing the following loss function:

$$\mathcal{L}(\tilde{\Theta}) = \alpha \sum_{i \in id_{all}} \left(\Psi(\mathbf{v}_i, \tilde{\Theta}) - t_i \right)^2 + (1 - \alpha) \|\mathbf{W}\|_1,$$

where $\Psi(\mathbf{v}_i, \tilde{\Theta}) = \sum_{j=1}^M w_j |w_j| e^{-\|\mathbf{D}_j \mathbf{R}_j(\mathbf{v}_i - \mathbf{c}_j)\|_2^2}$. The relevant parameters used in this appendix are defined as in the main text (Section III). $\mathbf{R}(\theta_{xj}, \theta_{yj}, \theta_{zj})$ denotes the rotation matrix of the j -th ERBF, whose detailed form is:

$$\mathbf{R}(\theta_{xj}, \theta_{yj}, \theta_{zj}) = \begin{bmatrix} r_1^j & r_2^j & r_3^j \\ r_4^j & r_5^j & r_6^j \\ r_7^j & r_8^j & r_9^j \end{bmatrix},$$

where

$$\begin{aligned} r_1^j &= \cos \theta_{xj} \cos \theta_{yj}; & r_4^j &= \cos \theta_{yj} \sin \theta_{xj}; \\ r_2^j &= -\cos \theta_{zj} \sin \theta_{xj} - \sin \theta_{yj} \sin \theta_{zj} \cos \theta_{xj}; \\ r_3^j &= \sin \theta_{xj} \sin \theta_{zj} - \cos \theta_{xj} \cos \theta_{zj} \sin \theta_{yj}; \\ r_5^j &= \cos \theta_{xj} \cos \theta_{zj} - \sin \theta_{xj} \sin \theta_{yj} \sin \theta_{zj}; \\ r_6^j &= -\sin \theta_{zj} \cos \theta_{xj} - \cos \theta_{zj} \sin \theta_{xj} \sin \theta_{yj}; \\ r_7^j &= \sin \theta_{yj}; & r_8^j &= \cos \theta_{yj} \sin \theta_{zj}; & r_9^j &= \cos \theta_{yj} \cos \theta_{zj}; \end{aligned}$$

The gradient of each parameter is denoted as:

$$\nabla \mathcal{L}(\tilde{\Theta}) = \left\{ \alpha \frac{\partial \mathcal{L}_2}{\partial \mathbf{C}}, \alpha \frac{\partial \mathcal{L}_2}{\partial \mathbf{D}}, \alpha \frac{\partial \mathcal{L}_2}{\partial \mathbf{A}}, \alpha \frac{\partial \mathcal{L}_2}{\partial \mathbf{W}} + (1 - \alpha) \frac{\partial \mathcal{L}_1}{\partial \mathbf{W}} \right\}.$$

A. Weight Gradient

$$\frac{\partial \mathcal{L}_2}{\partial w_j} = \sum_{i \in id_{all}} 4w_j \text{sign}(w_j) \mathbf{F}_{ij} E_i; \quad \frac{\partial \mathcal{L}_1}{\partial w_j} = \text{sign}(w_j).$$

B. Axis Length Gradient

$$\frac{\partial \mathcal{L}_2}{\partial d_{jk}} = \sum_{i \in id_{all}} -4 \text{sign}(w_j) w_j^2 d_{jk} \mathbf{F}_{ij} \tilde{D}_{x_{ik}}^2 E_i, \quad k = 1, 2, 3.$$

where $\Delta_{x_i} = \mathbf{x}_i - \mathbf{c}_j$ and $\tilde{D}_{x_i} = \mathbf{R}(\theta_{xj}, \theta_{yj}, \theta_{zj}) \cdot \Delta_{x_i}$.

C. Center Gradient

$$\frac{\partial \mathcal{L}_2}{\partial c_{jk}} = \sum_{i \in id_{all}} 4 \text{sign}(w_j) \tilde{F}_{ik} E_i, \quad k = 1, 2, 3,$$

$$\tilde{\mathbf{E}}_i = \begin{pmatrix} d_{j1}^2 w_j^2 \tilde{D}_{x_{i1}} \mathbf{F}_{ij} \\ d_{j2}^2 w_j^2 \tilde{D}_{x_{i2}} \mathbf{F}_{ij} \\ d_{j3}^2 w_j^2 \tilde{D}_{x_{i3}} \mathbf{F}_{ij} \end{pmatrix}, \quad \tilde{\mathbf{F}}_i = \mathbf{R}(\theta_{xj}, \theta_{yj}, \theta_{zj})^T \cdot \tilde{\mathbf{E}}_i.$$

D. Rotation Angle Gradient

1). Gradient w.r.t. θ_{xj}

$$\frac{\partial \mathcal{L}_2}{\partial \theta_{xj}} = \sum_{i \in id_{all}} -4 \text{sign}(w_j) E_i \left(\tilde{E}_{i1}(-\tilde{D}_{x_{i2}}) + \tilde{E}_{i2}(-\tilde{D}_{x_{i1}}) \right).$$

2). Gradient w.r.t. θ_{yj}

$$\frac{\partial \mathcal{L}_2}{\partial \theta_{yj}} = \sum_{i \in id_{all}} -4 \text{sign}(w_j) E_i \left(\tilde{E}_{i1} \tilde{D}_X + \tilde{E}_{i2} \tilde{D}_Y + \tilde{E}_{i3} \tilde{D}_Z \right),$$

$$\tilde{D}_X = -\cos \theta_{xj} \sin \theta_{yj} \Delta_{x_{i1}} - r_1^j \sin \theta_{zj} \Delta_{x_{i2}} - r_1^j \cos \theta_{zj} \Delta_{x_{i3}},$$

$$\tilde{D}_Y = -\sin \theta_{xj} \sin \theta_{yj} \Delta_{x_{i1}} - r_4^j \sin \theta_{zj} \Delta_{x_{i2}} - r_4^j \cos \theta_{zj} \Delta_{x_{i3}},$$

$$\tilde{D}_Z = \cos \theta_{yj} \Delta_{x_{i1}} - \sin \theta_{yj} \sin \theta_{zj} \Delta_{x_{i2}} - \sin \theta_{yj} \cos \theta_{zj} \Delta_{x_{i3}}.$$

3). Gradient w.r.t. θ_{zj}

$$\frac{\partial \mathcal{L}_2}{\partial \theta_{zj}} = \sum_{i \in id_{all}} -4 \text{sign}(w_j) E_i \left(\tilde{E}_{i1} \hat{D}_X + \tilde{E}_{i2} \hat{D}_Y + \tilde{E}_{i3} \hat{D}_Z \right),$$

$$\hat{D}_X = r_3^j \Delta_{x_{i2}} - r_2^j \Delta_{x_{i3}}, \quad \hat{D}_Y = r_6^j \Delta_{x_{i2}} - r_5^j \Delta_{x_{i3}},$$

$$\hat{D}_Z = r_9^j \Delta_{x_{i2}} - r_8^j \Delta_{x_{i3}}.$$

APPENDIX B

DERIVATION OF THE DYNAMIC WEIGHT α

The dynamic weight α introduced in Eq. (11) is derived from the perspective of multi-objective optimization. Following the Frank-Wolfe optimization approach for Pareto multi-task learning [19], the objective is to find a Pareto optimal solution by identifying the minimum-norm point within the convex hull of the gradients of the two loss functions: the approximation loss $\mathcal{L}_2$ and the sparsity penalty $\mathcal{L}_1$.

Let $\nabla_{\mathbf{W}} \mathcal{L}_2$ and $\nabla_{\mathbf{W}} \mathcal{L}_1$ denote the gradients of $\mathcal{L}_2$ and $\mathcal{L}_1$ with respect to the weights $\mathbf{W}$, respectively. The goal is to find a coefficient $\alpha \in [0, 1]$ that minimizes the squared norm of the convex combination of these two gradients. This is formulated as the following one-dimensional quadratic optimization problem:

$$\min_{\alpha \in [0, 1]} J(\alpha) = \|\alpha \nabla_{\mathbf{W}} \mathcal{L}_2 + (1 - \alpha) \nabla_{\mathbf{W}} \mathcal{L}_1\|_2^2. \quad (22)$$

Expanding Eq. (22) gives a quadratic function of α :

$$J(\alpha) = \alpha^2 \|\nabla_{\mathbf{W}} \mathcal{L}_2\|_2^2 + (1 - \alpha)^2 \|\nabla_{\mathbf{W}} \mathcal{L}_1\|_2^2 + 2\alpha(1 - \alpha)(\nabla_{\mathbf{W}} \mathcal{L}_1)^T \nabla_{\mathbf{W}} \mathcal{L}_2. \quad (23)$$

To find the minimum, the derivative of $J(\alpha)$ with respect to α is taken and set to zero:

$$\frac{\partial J(\alpha)}{\partial \alpha} = 2\alpha \|\nabla_{\mathbf{W}} \mathcal{L}_2\|_2^2 - 2(1 - \alpha) \|\nabla_{\mathbf{W}} \mathcal{L}_1\|_2^2 + 2(1 - 2\alpha)(\nabla_{\mathbf{W}} \mathcal{L}_1)^T \nabla_{\mathbf{W}} \mathcal{L}_2 = 0. \quad (24)$$

Rearranging the terms and isolating α gives:

$$\alpha \|\nabla_{\mathbf{W}} \mathcal{L}_2 - \nabla_{\mathbf{W}} \mathcal{L}_1\|_2^2 = (\nabla_{\mathbf{W}} \mathcal{L}_1 - \nabla_{\mathbf{W}} \mathcal{L}_2)^T \nabla_{\mathbf{W}} \mathcal{L}_1. \quad (25)$$

Solving for α provides the analytical solution:

$$\alpha = \frac{(\nabla_{\mathbf{W}} \mathcal{L}_1 - \nabla_{\mathbf{W}} \mathcal{L}_2)^T \nabla_{\mathbf{W}} \mathcal{L}_1}{\|\nabla_{\mathbf{W}} \mathcal{L}_2 - \nabla_{\mathbf{W}} \mathcal{L}_1\|_2^2}. \quad (26)$$

Finally, to ensure that the combination strictly remains within the valid convex hull (i.e., $\alpha \in [0, 1]$), the clipping operation $\text{clip}_{[0, 1]}(\cdot)$ is applied to the analytical solution α .

REFERENCES

- [1] M. Li, F. Chen, W. Wang, and C. Tu, "Sparse rbf surface representations," *Computer Aided Geometric Design*, vol. 48, pp. 49–59, 2016.
- [2] F. Calakli and G. Taubin, "Ssd: Smooth signed distance surface reconstruction," *Computer Graphics Forum*, vol. 30, no. 7, pp. 1993–2002, 2011.
- [3] 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, pp. 165–174.
- [4] 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*, 2001, pp. 67–76.
- [5] H. Q. Dinh, G. Turk, and G. Slabaugh, "Reconstructing surfaces by volumetric regularization using radial basis functions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 10, pp. 1358–1371, 2002.
- [6] G. Turk and J. F. O'Brien, "Modelling with implicit surfaces that interpolate," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 4, pp. 855–873, 2002.
- [7] Y. Ohtake, A. Belyaev, and H.-P. Seidel, "Sparse surface reconstruction with adaptive partition of unity and radial basis functions," *Graphical Models*, vol. 68, no. 1, pp. 15–24, 2006.
- [8] 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, pp. 108–124.
- [9] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *arXiv preprint arXiv:2002.10099*, 2020.
- [10] Z. Wang, Y. Zhang, R. Xu, F. Zhang, P.-S. Wang, S. Chen, S. Xin, W. Wang, and C. Tu, "Neural-singular-hessian: Implicit neural representation of unoriented point clouds by enforcing singular hessian," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 6, pp. 1–14, 2023.
- [11] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural Computation*, vol. 5, no. 2, pp. 305–316, 1993.
- [12] A. Ismayilova and M. Ismayilov, "On the universal approximation property of radial basis function neural networks," *Annals of Mathematics and Artificial Intelligence*, vol. 92, no. 3, pp. 691–701, 2024.
- [13] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 139–1, 2023.
- [14] S. Gui, Z. Chen, B. Lu, and M. Chen, "Molecular sparse representation by a 3d ellipsoid radial basis function neural network via l1 regularization," *Journal of Chemical Information and Modeling*, vol. 60, no. 12, pp. 6054–6064, 2020.
- [15] V. Saragadam, D. LeJeune, J. Tan, G. Balakrishnan, A. Veeraraghavan, and R. G. Baraniuk, "Wire: Wavelet implicit neural representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 18 507–18 516.
- [16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

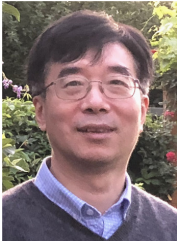
- [17] Z. Chen, S. Gui, H. Chen, C. Wu, and M. Chen, "sparse representation of images based on rbf neural network," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1. IEEE, 2020, pp. 830–835.
- [18] D. Wang, H. Chen, C. Wu, and M. Chen, "Point cloud surface reconstruction using sparse ellipsoid radial basis function neural network," in *2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 5. IEEE, 2021, pp. 1613–1618.
- [19] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *Advances in neural information processing systems*, vol. 31, 2018.
- [20] J. Wilhelms and A. Van Gelder, "Octrees for faster isosurface generation," *ACM Transactions on Graphics (TOG)*, vol. 11, no. 3, pp. 201–227, 1992.
- [21] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 347–353.
- [22] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.
- [23] R. Pan and V. Skala, "Continuous global optimization in surface reconstruction from an oriented point cloud," *Computer-Aided Design*, vol. 43, no. 8, pp. 896–901, 2011.
- [24] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7. Eurographics Association, 2006, p. 61–70.
- [25] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013.
- [26] F. Hou, C. Wang, W. Wang, H. Qin, C. Qian, and Y. He, "Iterative poisson surface reconstruction (ipsr) for unoriented points," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–13, 2022.
- [27] M. Samozino, M. Alexa, P. Alliez, and M. Yvinec, "Reconstruction with voronoi centered radial basis functions," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. Eurographics Association, 2006, p. 51–60.
- [28] G. Casciola, D. Lazzaro, L. B. Montefusco, and S. Morigi, "Shape preserving surface reconstruction using locally anisotropic radial basis function interpolants," *Computers & Mathematics with Applications*, vol. 51, no. 8, pp. 1185–1198, 2006.
- [29] C. Walder, O. Chapelle, and B. Schölkopf, "Implicit surfaces with globally regularised and compactly supported basis functions," *Advances in Neural Information Processing Systems*, vol. 19, 2006.
- [30] Q. Xia, M. Y. Wang, and X. Wu, "Orthogonal least squares in partition of unity surface reconstruction with radial basis function," in *Geometric Modeling and Imaging—New Trends (GMAI'06)*. IEEE, 2006, pp. 28–33.
- [31] E. V. Brazil, I. Macedo, M. C. Sousa, L. H. de Figueiredo, and L. Velho, "Sketching variational hermite-rbf implicits," in *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, 2010, pp. 1–8.
- [32] I. Macêdo, J. P. Gois, and L. Velho, "Hermite radial basis functions implicits," *Computer graphics forum*, vol. 30, no. 1, pp. 27–42, 2011.
- [33] S. Liu, C. C. Wang, G. Brunnett, and J. Wang, "A closed-form formulation of hrbf-based surface reconstruction by approximate solution," *Computer-Aided Design*, vol. 78, pp. 147–157, 2016.
- [34] K. P. Drake, E. J. Fuselier, and G. B. Wright, "Implicit surface reconstruction with a curl-free radial basis function partition of unity method," *SIAM Journal on Scientific Computing*, vol. 44, no. 5, pp. A3018–A3040, 2022.
- [35] Y. Zeng and Y. Zhu, "Implicit surface reconstruction based on a new interpolation/approximation radial basis function," *Computer Aided Geometric Design*, vol. 92, p. 102062, 2022.
- [36] C. Shen, J. F. O'Brien, and J. R. Shewchuk, "Interpolating and approximating implicit surfaces from polygon soup," in *ACM SIGGRAPH 2004 Papers*, 2004, p. 896–904.
- [37] M. Kazhdan, "Reconstruction of solid models from oriented point sets," in *Proceedings of the third Eurographics symposium on Geometry processing*, 2005, pp. 73–es.
- [38] W. Lu, Z. Shi, J. Sun, and B. Wang, "Surface reconstruction based on the modified gauss formula," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 1, pp. 1–18, 2018.
- [39] 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*, vol. 42, no. 2, pp. 1–19, 2022.
- [40] Q. Ma, D. Wu, and X. Luo, "A review of deep learning-based power load forecasting methods," *International Journal of Network Dynamics and Intelligence*, vol. 4, no. 4, p. 100027, Dec. 2025.
- [41] X. Yan, C. Wang, and Y. Jin, "Federated bimodal graph neural networks for text-image retrieval," *International Journal of Network Dynamics and Intelligence*, vol. 4, no. 2, p. 100009, Jun. 2025.
- [42] Y. Zhan, R. Yang, J. You, M. Huang, W. Liu, and X. Liu, "A systematic literature review on incomplete multimodal learning: techniques and challenges," *Systems Science & Control Engineering*, vol. 13, no. 1, p. 2467083, 2025.
- [43] Q. Lan, A. Kaul, and S. Jones, "Prompt injection detection in llm integrated applications," *International Journal of Network Dynamics and Intelligence*, vol. 4, no. 2, p. 100013, Jun. 2025.
- [44] B. Song, S. Zhao, L. Dang, H. Wang, and L. Xu, "A survey on learning from data with label noise via deep neural networks," *Systems Science & Control Engineering*, vol. 13, no. 1, p. 2488120, 2025.
- [45] R. Deepika and T. S. Pradeep Kumar, "An interpretable dual-level lightweight framework for malaria diagnosis for enhanced remote diagnosis in resource-constrained settings," *Systems Science & Control Engineering*, vol. 13, no. 1, p. 2579988, 2025.
- [46] Y. Zhang, X. Zhang, D. Miao, and H. Yu, "Real-time semantic segmentation of road scenes via hybrid dilated grouping network," *International Journal of Network Dynamics and Intelligence*, vol. 4, no. 1, p. 100006, Mar. 2025.
- [47] 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, pp. 4460–4470.
- [48] 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, pp. 523–540.
- [49] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghuvaran, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in neural information processing systems*, vol. 33, pp. 7537–7547, 2020.
- [50] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems*, vol. 33, pp. 7462–7473, 2020.
- [51] Y. Ben-Shabat, C. H. Koneputugodage, and S. Gould, "Digs: Divergence guided shape implicit neural representation for unoriented point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 323–19 332.
- [52] P. Erler, L. Fuentes-Perez, P. Hermosilla, P. Guerrero, R. Pajarola, and M. Wimmer, "Ppsurf: Combining patches and point convolutions for detailed surface reconstruction," *Computer Graphics Forum*, vol. 43, no. 1, p. e15000, 2024.
- [53] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces," *arXiv preprint arXiv:2011.13495*, 2020.
- [54] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "Ewa volume splatting," in *Proceedings Visualization, 2001. VIS'01*. IEEE, 2001, pp. 29–538.
- [55] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Transactions on visualization and computer graphics*, vol. 13, no. 3, pp. 530–548, 2007.
- [56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [57] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," in *Proceedings of the sixth ACM symposium on Solid modeling and applications*, 2001, pp. 249–266.
- [58] K. M. Jatavallabhula, E. Smith, J.-F. Lafleche, C. F. Tsang, A. Rozantsev, W. Chen, T. Xiang, R. Lebariedian, and S. Fidler, "Kaolin: A pytorch library for accelerating 3d deep learning research," *arXiv preprint arXiv:1911.05063*, 2019.
- [59] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3d mesh segmentation," *ACM Transactions on Graphic (TOG)*, vol. 28, no. 3, pp. 1–12, 2009.
- [60] 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, pp. 9601–9611.
- [61] Q. Zhou and A. Jacobson, "Thing10k: A dataset of 10,000 3d-printing models," *arXiv preprint arXiv:1605.04797*, 2016.
- [62] Dawson-Haggerty et al., "trimesh," 2019, <https://trimesh.org/>.



Bobo Lian received the B.S. degree in Mathematics and Applied Mathematics from Hangzhou Dianzi University, Hangzhou, China, and the M.S. degree in Mathematics from Soochow University, Suzhou, China. He is currently pursuing the Ph.D. degree with the School of Mathematical Sciences, Soochow University. His research interests include machine learning, artificial intelligence, and their applications in scientific computing.



Minxin Chen is an Associate Professor of Computational Mathematics at Soochow University, Suzhou, Jiangsu, China. He received the Ph.D. degree in Computational Mathematics from the Chinese Academy of Sciences, Beijing, China. His research interests include machine learning, numerical algorithms, and computational mathematics.



Zidong Wang was born in Jiangsu, China, in 1966. He received the B.Sc. degree in mathematics in 1986 from Suzhou University, Suzhou, China, and the M.Sc. degree in applied mathematics in 1990 and the Ph.D. degree in electrical engineering in 1994, both from Nanjing University of Science and Technology, Nanjing, China.

He is currently Professor of Dynamical Systems and Computing in the Department of Computer Science, Brunel University London, U.K. From 1990 to 2002, he held teaching and research appointments in universities in China, Germany and the U.K. Prof. Wang's research interests include dynamical systems, signal processing, bioinformatics, control theory and applications. He has published more than 600 papers in international journals. He is a holder of the Alexander von Humboldt Research Fellowship of Germany, the JSPS Research Fellowship of Japan, William Mong Visiting Research Fellowship of Hong Kong.

Prof. Wang serves (or has served) as the Editor-in-Chief for *International Journal of Systems Science*, the Editor-in-Chief for *Neurocomputing*, the Editor-in-Chief for *Systems Science & Control Engineering*, and an Associate Editor for 12 international journals including *IEEE Transactions on Automatic Control*, *IEEE Transactions on Control Systems Technology*, *IEEE Transactions on Neural Networks*, *IEEE Transactions on Signal Processing*, and *IEEE Transactions on Systems, Man, and Cybernetics-Part C*. He is a Member of the Academia Europaea, a Member of the European Academy of Sciences and Arts, an Academician of the International Academy for Systems and Cybernetic Sciences, a Fellow of the IEEE, a Fellow of the Royal Statistical Society, and a member of program committees for many international conferences.



Dandan Wang received the M.S. degree in Mathematics from the School of Mathematical Sciences, Soochow University, Suzhou, China, in 2022. She is currently engaged in research and development in the field of artificial intelligence. Her research interests include machine learning, image processing, and computer vision.



Chenjian Wu is an Associate Professor in the School of Electronic and Information Engineering at Soochow University, Suzhou, Jiangsu, China. He received his Ph.D. degree in Electronic Circuit and System from Southeast University in 2013. His research mainly focuses on machine learning and artificial intelligence chip design.