MixNet: Mix Different Networks for Learning 3D Implicit Representations

Bowen Lyu
School of Mathematical Sciences, UCAS
lvbowen19@mails.ucas.ac.cn

Li-Yong Shen
School of Mathematical Sciences, UCAS
lyshen@ucas.ac.cn

Chun-Ming Yuan
KLMM, AMSS
Chinese of Academic Sciences
cmyuan@mmrc.iss.ac.cn

Abstract

We introduce a neural network, MixNet, for learning implicit representations of 3D subtle models with large smooth areas and exact shape details in the form of interpolation of two different implicit functions. Our network takes a point cloud as input and uses conventional MLP networks and SIREN networks to predict different implicit fields. We use a learnable interpolation function to combine the implicit values of these two networks and achieve the respective advantages of them. The network is self-supervised with only reconstruction loss, leading to faithful 3D reconstructions with smooth planes, correct details, and plausible spatial partition without any ground-truth segmentation. We evaluate our method on ABC, the largest and most diverse CAD dataset, and some typical shapes to test in terms of geometric correctness and surface smoothness to demonstrate superiority over current alternatives suitable for shape reconstruction.

Keywords: Implicit representation, 3D reconstruction, Point cloud, Deep learning.

1. Introduction

Implicit Representations using only Multilayer Perceptions (MLPs) [21, 24, 7, 11, 9] have gained sustained interest for their simple form and effective expression in the field of 3D shapes. It has shown an excellent ability to recover a shape from the unordered point cloud compactly and efficiently. The superiority of this method is that the network takes only point clouds, i.e., 3D coordinates, as input directly without any extra operation, and outputs the corresponding signed distance fields or occupancy fields through multilayer perceptrons. Then it can be rendered easily by Marching Cubes [19] or other similar methods. In theory, as long as the network has been trained well enough, we can obtain a model that is capable of infinite subdivisions as each 3D shape is represented by a continuous field.

Since it is quite easy to find various fine features in many man-made and mechanical 3D shapes, reconstruction of shape features, particularly in the vicinity of small complex topological structures, is one of the difficulties in implicit representation learning. However, coordinate-based MLPs with ReLU activation networks are incapable of reconstructing high-frequency details of surfaces, and they fail almost every time to reconstruct correctly in the region where the values of implicit representation change subtly (the fine details or small hole structures in thin plates). Fig. 1 shows that the conventional MLP networks, such as IGR, fail to fit the small hole-like structures or fine details.

Recent studies have noted this problem and have focused on improving networks' ability to represent fine details. From the perspective of high-frequency signals, these methods [28, 30, 32] achieve the desired results in the region

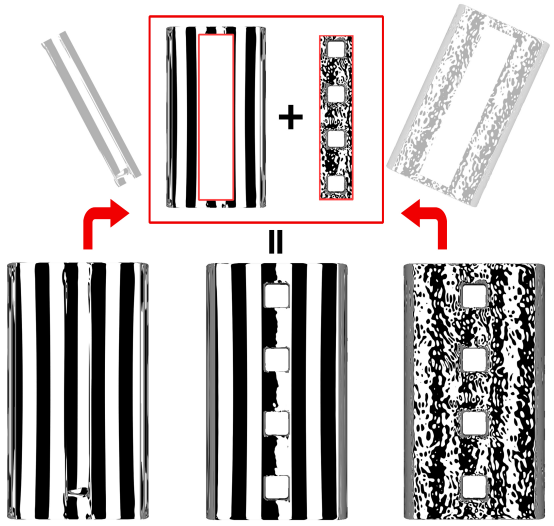

Figure 1. **Method overview**. We represent a better shape as a combination of two base shapes: IGR [11] (left) can reconstruct a smooth flat surface, but it cannot recover the *shallow* hole correctly. SIREN [28] (right) can represent the topological feature correctly, but their surface is corrugated. Our method (middle) combines both smooth planes and correct topological structures.

full of fine details, while they lose smoothness in the flat area. In addition, training these networks is more challenging because of their tendency to overfit. Another direction is to partition the space with some guidelines to train networks locally [5, 15, 10, 29]. The majority of these methods focus on spatial structures, usually hierarchical structures, for better expressiveness and generalizability. Their results are sometimes dependent on the geometric complexity, as their local methods usually have hyperparameters related to them.

Motivated by this observation, in this paper, we propose a new architecture that makes our network able to reconstruct shapes separately, i.e., using a ReLU-based MLP to represent flat areas while using a sine-based MLP to represent areas with rich details. The final implicit representation is defined as the combination of two different implicit functions. We prove the feasibility of this combination and make this process learnable. Intuitively, the new network can dynamically discriminate the points in the *complicated* regions without human experience and retain a simple representation in the *simple* regions during the training. Through experiments and ablation studies, we demonstrate the efficacy and superiority of our method over other state-of-the-art implicit reconstructions from a point cloud, especially on CAD-type shapes. Our main contribution includes the following:

- A novel mixed implicit 3D representation learning method that is able to fit fine details and flat areas at the same time.
- The theoretical feasibility of the decomposition of implicit representation and its corresponding learnable process.
- An extensible framework that can combine the expressive strengths of two types of models.

2. Related work

Neural implicit representations Neural implicit representations have recently been proven to have great promise for 3D modeling due to their global continuity that is not tied to a specific resolution and concise presentation, which makes them easily extendable for other applications. These pioneering works [21, 24, 8] use MLP networks to regress the ground truth SDFs or volume radiance values. SAL [1] and SALD [2] learn an implicit shape representation directly from raw data by introducing the sign agnostic distance. IGR [11] then proposes to train MLP networks without knowing the ground truth SDF by regarding the networks as implicit functions. BSP-Net [6] generates compact low-poly meshes via binary space partitioning. These works have a simple structure and can reconstruct aesthetically pleasing but are often lacking in detailed models.

Hierarchical neural implicit representations Many works then resort to hierarchical structures to improve the expressiveness of fine details and generalizability of more scenarios. LDIF (Local Deep Implicit Functions) [10] proposes a 3D shape representation that decomposes space into an organized set of learned implicit functions to obtain higher reconstruction accuracy. DeepMLS [18] introduces implicit moving least-squares (IMLS) surface formulation into deep neural networks for inheriting both the flexibility of point sets and the high quality of implicit surfaces. SAIL-S3 [34] learns a local implicit surface network for shared, adaptive modeling of the entire surface. ConvOccNet [25] combines convolutional encoders with implicit occupancy decoders, enabling structured reasoning in 3D space. By subdividing the whole space into some subspaces, these methods alleviate the difficulty in expressing details to some extent but might meet the problem that the model complexity and computation cost increase when the desired geometric resolution increases.

High-frequency representations in neural networks As many works [27, 31, 26, 3, 22, 17] have shown that deep networks tend to learn lower frequency functions, better methods are sought to resolve this issue. SIREN [28] shows remarkable progress in detail reconstruction by replacing classical ReLU-like activation with periodic activation. NeRF [22] also demonstrates that using position encoding [30] before passing low-dimension inputs directly to the network enables better fitting of data that contains high-frequency variation. Furthermore, SAPE [13] presents a spatially adaptive progressive encoding scheme, which enables MLP networks to better fit a wide range of frequencies without sacrificing training stability or requiring any domain-specific preprocessing. In addition, IDF [32] applies a coarse-to-fine frequency hierarchy to represent a complex surface as a smooth base surface plus the displacement along the base's normal directions. These approaches have their own characteristics and address the problem from the perspective of frequency.

3. Method

We propose a method for representing a shape by interpolating two distinct implicit fields, one of which is learned by conventional MLP networks and the other by SIREN networks, and the values of interpolation are learnable as well. This combination can combine the benefits of these two types of networks to reconstruct smooth flat surfaces with intricate details.

In this section, we first illustrate the poor performance when representing the shape with only a single kind of network in Section 3.1. Then we formally define the decomposition of implicit neural representations through a constructive proof in Section 3.2 . Finally, we introduce the network

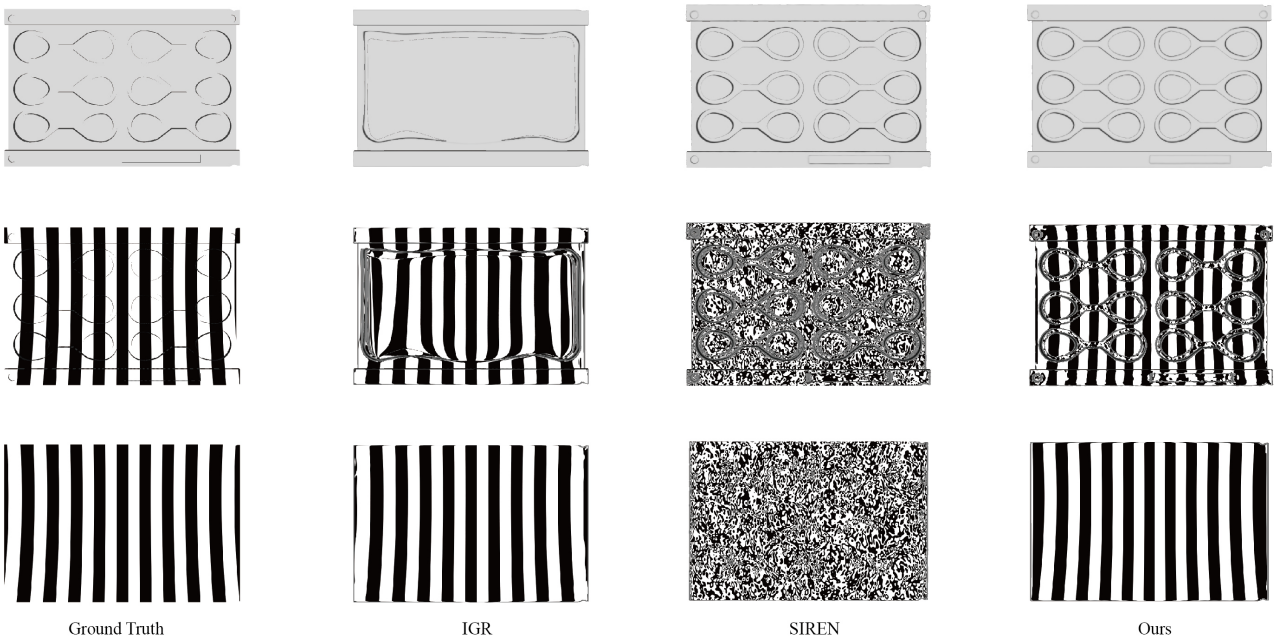

Figure 2. **Comparison of expressive power**. Different expressive powers of conventional networks and SIRENs. After adding the reflection lines, it is clear that conventional networks such as IGR can reconstruct a better smooth plane but are deficient in detail expression, while sinusoidal networks such as SIRENs and FFN have the opposite appearance. Top: Reconstruction results. Middle & Bottom: The front & back of results after adding reflection lines.

architecture that combines the advantages of both conventional MLP networks and SIREN networks, as well as the responding training strategies in Section 3.3.

3.1. Expressive Power of Single Model

The MLP structure is widely adopted in 3D reconstruction work [11, 1, 21, 24, 28, 30, 9] due to its simple structure and satisfactory results. These methods employ either conventional activations or sinusoidal activations. In some recent work [26, 30, 33], it has been proven that both have expressive limitations. We would like to explore the characteristics of each activation in this part.

Conventional MLP An increasing number of works [22, 30, 4, 13, 33] have found that having conventional MLP networks $\mathcal{F}_\theta$ directly operate on low dimensional inputs such as coordinates leads to poor performance at high-frequency variations in the areas where there are rich details or topological deformation. The spectral bias [17, 30, 14, 3, 17] of the network's output shows that the training loss does not decay evenly and independently, but instead decays more rapidly, corresponding to the larger eigenvalues of the neural kernel. As a result, the conventional MLP networks converge extremely slowly for those high-frequency details or topological changes.

On the other hand, the conventional activation's ability of plane reproduction cannot be ignored, as shown in Fig. 2. In the conventional MLP case, owing to the monotonicity of conventional activations, we could consider the conventional MLP as the combination of many linear regions [12, 23]. Meanwhile, the model's plane regions in 3D space can also be treated as a linear classification problem and be separated into many subplanes w.r.t. the linear regions of the network. From this perspective, the smoothness of the plane fitted by the conventional MLP can be guaranteed.

SIRENs Sitzmann *et al*. [28] improved the performance of the MLP with monotonic activations by using sinusoidal activations, i.e., replacing conventional activations such as ReLU-like functions with $\sin$. Similarly, FFNs [30] apply a Fourier mapping $\gamma(\boldsymbol{x}) = \sin(\boldsymbol{\omega}\boldsymbol{x} + \boldsymbol{\phi})$ on the low dimensional input before it is sent to a conventional MLP. Gizem *et al*. [33] have proved the defects of these sinusoidal networks that the expressive power of sinusoidal networks is restricted to a linear combination of certain harmonics of the feature mapping. Furthermore, the network's width and depth are finite; therefore, we can only obtain a finite frequency approximation, which results in poor plane recovery quality.

Here, we employ a straightforward example to illustrate

this. To show its intuitiveness, we reduce the dimension of this problem to 2 dimensions. Consider a 2D implicit field $f(x) : \mathbb{R}^2 \to \mathbb{R}$, where the isosurface is defined similar to that in 3D as $\mathcal{S}_\tau = \{\boldsymbol{x}|f(\boldsymbol{x}) = \tau\}$. Now we define a picture whose pixel values vary uniformly from the border to the center, and we extract the 2D isosurface $\mathcal{S}_\tau$ marked with a black line as shown in Fig. 3. From the definition above, we obtain a rectangular isosurface. To simulate the situation of finite frequencies, we now convert it to the frequency domain to eliminate high-frequency components and then convert it back to the spatial domain. This time, we can obtain another isosurface $\mathcal{S}'_\tau$. It is quite evident that $\mathcal{S}'_\tau$, which is composed of the part of frequency, looks curved relative to the original isosurface $\mathcal{S}_\tau$. This explains why SIREN does not work very well when representing the plane.

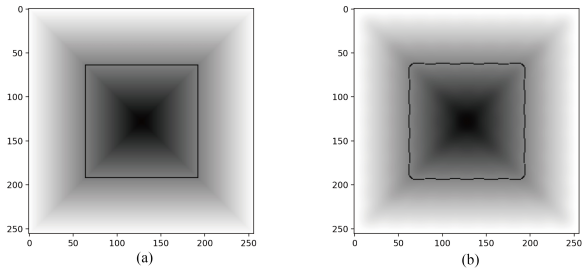

Figure 3. **Expression Flaws of SIRENs**. Left: The original 2D implicit field. Right: Implicit field generated after removing high frequencies. The black line in the figure is the extracted contour, and the contour on the right is approximated by a threshold.

3.2. Mix Two Models

Before we introduce our network architecture, we first briefly review the reconstruction problem. The problem of implicit surface reconstruction can be formulated as the task of finding an implicit representation $\boldsymbol{F} : \mathbb{R}^d \to \mathbb{R}$ whose zero level set $\mathcal{S} = \{\boldsymbol{x}|\boldsymbol{F}(\boldsymbol{x}) = 0\}$ is the estimated surface. Our approximation in various ways can be defined as $\boldsymbol{f}(\boldsymbol{x})$ and satisfies $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \varepsilon(\boldsymbol{x})$, where $\varepsilon(\boldsymbol{x})$ is the error function. According to Section 3.1, this error function is difficult to eliminate due to the limitations of the number of network layers and training times. Therefore, we assume that different types of network activations correspond to different distributions of error functions. It is obvious that using only one kind of network would prevent us from getting smooth planes and shape details at the same time.

We now introduce the interpolation function $\boldsymbol{p} : \mathbb{R}^d \to [0, 1]$, which is nontrivial when $\boldsymbol{p}(\boldsymbol{x})$ is not always equal to 1 or 0. And we have

Theorem 1. *Differentiable implicit representation $\boldsymbol{F}(\boldsymbol{x})$ can be decomposed of two different differentiable functions $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ within an nontrivial interpolation function $\boldsymbol{p}(\boldsymbol{x})$, i.e. $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{x})\boldsymbol{f}_1(\boldsymbol{x}) + (1 - \boldsymbol{p}(\boldsymbol{x}))\boldsymbol{f}_2(\boldsymbol{x})$, when $\boldsymbol{x} \in \mathcal{S}$.*

Proof. Assume that $\varepsilon_1(\boldsymbol{x})$, $\varepsilon_2(\boldsymbol{x})$ are two different differentiable functions, and $\varepsilon_1(\boldsymbol{x}) \cdot \varepsilon_2(\boldsymbol{x}) < 0$ when $\boldsymbol{x} \in \mathcal{S}$. Let $\boldsymbol{f}_1(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \varepsilon_1(\boldsymbol{x})$, $\boldsymbol{f}_2(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \varepsilon_2(\boldsymbol{x})$, due to the differentiability of $\boldsymbol{F}(\boldsymbol{x})$ and $\varepsilon_{1,2}(\boldsymbol{x})$, $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ are still differentiable functions. Let $\boldsymbol{p}(\boldsymbol{x}) = \frac{\varepsilon_2(\boldsymbol{x})}{\varepsilon_2(\boldsymbol{x}) - \varepsilon_1(\boldsymbol{x})}$, we can easily obtain that $\boldsymbol{p}(\boldsymbol{x}) \in [0, 1]$ and is differentiable. Therefore, we have

$$\begin{aligned} &\boldsymbol{p}(\boldsymbol{x})\boldsymbol{f}_1(\boldsymbol{x}) + (1 - \boldsymbol{p}(\boldsymbol{x}))\boldsymbol{f}_2(\boldsymbol{x}) \\ &= \frac{\varepsilon_2(\boldsymbol{x})}{\varepsilon_2(\boldsymbol{x}) - \varepsilon_1(\boldsymbol{x})}(\boldsymbol{F}(\boldsymbol{x}) + \varepsilon_1(\boldsymbol{x})) + \\ &(1 - \frac{\varepsilon_2(\boldsymbol{x})}{\varepsilon_2(\boldsymbol{x}) - \varepsilon_1(\boldsymbol{x})})(\boldsymbol{F}(\boldsymbol{x}) + \varepsilon_2(\boldsymbol{x})) \\ &= \boldsymbol{F}(\boldsymbol{x}) \end{aligned} \tag{1}$$

□

Remark 1. *This is a constructive proof, and we have only demonstrated one form of existence. However, it intuitively shows that the precise representation $\boldsymbol{F}(\boldsymbol{x})$ can be obtained by interpolating two crude approximations $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$. In addition, the closer $\boldsymbol{f}_1(\boldsymbol{x})$ is to the exact solution $\boldsymbol{F}(\boldsymbol{x})$, the closer the $\boldsymbol{p}(\boldsymbol{x})$ is to 1.*

However, although Theorem 1 proves that $\boldsymbol{F}(\boldsymbol{x})$ can be decomposed, the assumption $\varepsilon_1(\boldsymbol{x}) \cdot \varepsilon_2(\boldsymbol{x}) < 0$ used in the proof is too restrictive. This requires that our two approximations $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ must have opposite signs near the ground truth surface, which is very difficult to guarantee in practical estimation. Therefore, we relax the conditions and define a new approximation

$$\hat{\boldsymbol{F}}(\boldsymbol{x}) = \hat{\boldsymbol{p}}(\boldsymbol{x})\boldsymbol{f}_1(\boldsymbol{x}) + (1 - \hat{\boldsymbol{p}}(\boldsymbol{x}))\boldsymbol{f}_2(\boldsymbol{x}) \tag{2}$$

where

$$\hat{\boldsymbol{p}}(\boldsymbol{x}) = \max\{0, \min\{\frac{\varepsilon_2(\boldsymbol{x})}{\varepsilon_2(\boldsymbol{x}) - \varepsilon_1(\boldsymbol{x})}, 1\}\} \tag{3}$$

$$\boldsymbol{f}_i(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \varepsilon_i(\boldsymbol{x}), i = 1, 2 \tag{4}$$

Theorem 2. *If $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are different approximations of $\boldsymbol{F}(\boldsymbol{x})$ on $\mathcal{S}$. $\hat{\boldsymbol{F}}(\boldsymbol{x})$ defined by Eq. (2) is a better approximation than $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$.*

Proof. The conclusion is equivalent to proving the error of $\hat{\boldsymbol{F}}(\boldsymbol{x})$ is smaller, i.e. $\left|\boldsymbol{F}(\boldsymbol{x}) - \hat{\boldsymbol{F}}(\boldsymbol{x})\right| \leq |\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{f}_1(\boldsymbol{x})|$ and $\left|\boldsymbol{F}(\boldsymbol{x}) - \hat{\boldsymbol{F}}(\boldsymbol{x})\right| \leq |\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{f}_2(\boldsymbol{x})|$.
When $\boldsymbol{x} \in \mathcal{S}$, $\boldsymbol{F}(\boldsymbol{x}) \equiv 0$, and $\boldsymbol{f}_i = \varepsilon_i(\boldsymbol{x})$.
When $\varepsilon_1(\boldsymbol{x}) \cdot \varepsilon_2(\boldsymbol{x}) < 0$, similar to Eq. (1), we have

$$\left|\boldsymbol{F}(\boldsymbol{x}) - \hat{\boldsymbol{F}}(\boldsymbol{x})\right| = 0 \leq \min\{|\boldsymbol{f}_1(\boldsymbol{x})|, |\boldsymbol{f}_2(\boldsymbol{x})|\}. \tag{5}$$

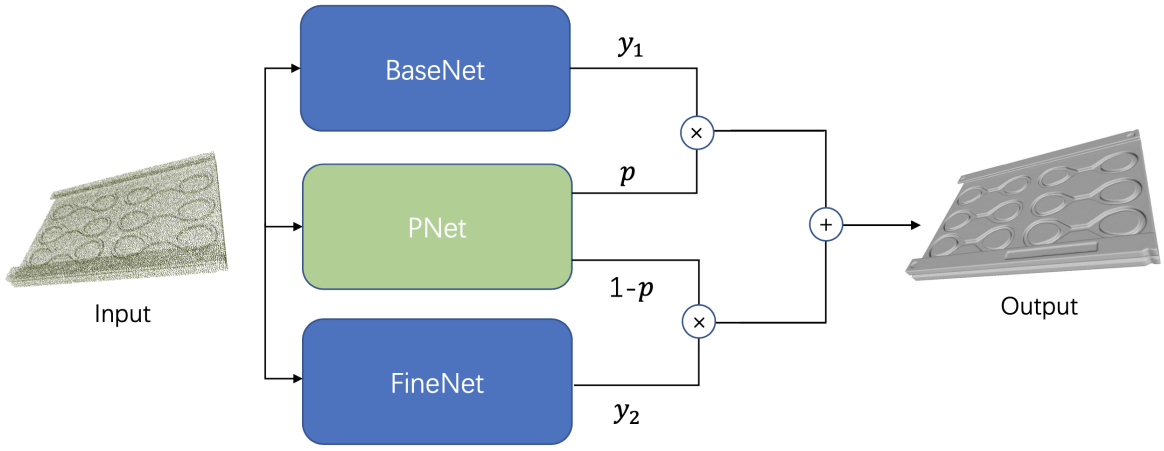

Figure 4. **Overview of our network**. Given a point cloud as input, we first send it to three different networks. The base network and fine network are responsible for fitting the shape from different features, BaseNet for smooth plat regions and FineNet for complex diverse regions. PNet is responsible for the proportion of both BaseNet and FineNet at the same point. Finally, we use the weight obtained by PNet to sum the two network outputs and obtain the implicit function of full space.

When $\varepsilon_1(\boldsymbol{x}) \cdot \varepsilon_2(\boldsymbol{x}) \geq 0$, if $\varepsilon_2(\boldsymbol{x}) > \varepsilon_1(\boldsymbol{x}) \geq 0$, then $\frac{\varepsilon_2(\boldsymbol{x})}{\varepsilon_2(\boldsymbol{x})-\varepsilon_1(\boldsymbol{x})} > 1$, so $\hat{\boldsymbol{p}}(\boldsymbol{x}) \equiv 1$, we have

$$\begin{aligned} &\left|\boldsymbol{F}(\boldsymbol{x}) - \hat{\boldsymbol{F}}(\boldsymbol{x})\right| = |\boldsymbol{f}_1(\boldsymbol{x})| \\ &\leq \min\{|\boldsymbol{f}_1(\boldsymbol{x})|, |\boldsymbol{f}_2(\boldsymbol{x})|\} = |\boldsymbol{f}_1(\boldsymbol{x})| \end{aligned} \tag{6}$$

Similarly, we can prove the other cases. □

Therefore, we are able to obtain a better approximation of the target representation by combining two different approximations with $\hat{\boldsymbol{p}}(\boldsymbol{x})$. When $\boldsymbol{x} \in \mathcal{S}$, $\boldsymbol{F}(\boldsymbol{x})$ is always equal to 0, and we can solve this equation to obtain $\hat{\boldsymbol{p}}(\boldsymbol{x})$ explicitly. However, note that the reconstruction from point cloud is an ill-posed problem, we can only cover the points sampled from the real isosurface $\mathcal{S}$ and for those points that exclude the sample point clouds, it is almost impossible to know the true values, much more the values of $\hat{\boldsymbol{p}}(\boldsymbol{x})$. Otherwise, although the expression of $\hat{\boldsymbol{p}}(\boldsymbol{x})$ is given by Theorem 2, the fractional form composed of $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ is unfavorable for backpropagation. Accordingly, we introduce another approximation denoted as $\tilde{\boldsymbol{p}}(\boldsymbol{x}) : \mathbb{R}^3 \to [0, 1]$ to fit this interpolation function so that the combination of $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ is closer to $\boldsymbol{F}(\boldsymbol{x})$, which is defined as follows:

$$\tilde{\boldsymbol{p}}(\boldsymbol{x}) = \underset{\tilde{\boldsymbol{p}}(\boldsymbol{x})}{\arg\min} \underset{x \in \mathcal{S}}{\left|\hat{\boldsymbol{F}}(\boldsymbol{x})\right|}. \tag{7}$$

This definition is quite similar to Theorem 2. Mathematically, if we solve the values of $\tilde{\boldsymbol{p}}(\boldsymbol{x})$, we also obtain the optimal $\hat{\boldsymbol{F}}(\boldsymbol{x})$ and vice versa. Therefore, we transform the problem of solving $\hat{\boldsymbol{p}}(\boldsymbol{x})$ into finding the best approximation $\hat{\boldsymbol{F}}(\boldsymbol{x})$ There are two benefits of this transformation: a) it aligns calculations with the ultimate goal ; b) the learnable $\tilde{\boldsymbol{p}}(\boldsymbol{x})$ further alleviates the problem when querying the points out of the ground truth point clouds.

3.3. Network Design and Training

We propose to model $\boldsymbol{f}_1(\boldsymbol{x})$, $\boldsymbol{f}_2(\boldsymbol{x})$ and $\tilde{\boldsymbol{p}}(\boldsymbol{x})$ with three different networks denoted as $\mathcal{F}_{base}$, $\mathcal{F}_{fine}$ and $\mathcal{P}$. $\mathcal{F}_{base}$ is responsible for a smooth base surface, $\mathcal{F}_{fine}$ is responsible for the complex geometric details and $\mathcal{P}$ is responsible for the value of interpolation, as shown in Fig. 4. We will select different types of networks according to various needs. Generally, we choose conventional networks for $\mathcal{F}_{base}$ and SIRENs for $\mathcal{F}_{fine}$, which is based on the view of Section 3.1. For more details about the selections of these types of networks, we show them in Section 4.3.

We adopt the loss from SIREN, which is designed to directly learn SDFs from oriented point clouds by solving the eikonal equation with boundary constraints on the on-surface points. The specific form of loss is

$$\begin{aligned} \mathcal{L}_{recon}(\boldsymbol{F}) =& \lambda_1 \sum_{\boldsymbol{x} \in \mathcal{P}} |\boldsymbol{F}(\boldsymbol{x})| \\ &+ \lambda_2 \sum_{\boldsymbol{x} \in \mathcal{P}, \boldsymbol{n} \in \mathcal{N}} (1 - < \nabla \boldsymbol{F}(\boldsymbol{x}), \boldsymbol{n} >) \\ &+ \lambda_3 \sum_{\boldsymbol{x} \in \Omega} (|\|\nabla \boldsymbol{F}(\boldsymbol{x})\| - 1|) \\ &+ \lambda_4 \sum_{\boldsymbol{x} \in \Omega \setminus \mathcal{P}} \exp(-100 \cdot \boldsymbol{F}(\boldsymbol{x})), \end{aligned} \tag{8}$$

where $\mathcal{P}$ is the input point cloud, $\mathcal{N}$ is the normal w.r.t. $\mathcal{P}$, Ω is the input domain (usually set to $[-1, 1]^3$), and $\lambda_{1,2,3,4}$ are the weights of these losses respectively. Here, the first

and second term of $\mathcal{L}_{recon}$ are the surface fitting loss and the normal fitting loss, the third term is *Eikonal* loss proposed by IGR, and the last term is penalization loss of off-surface points.

When $\boldsymbol{x} \in \mathcal{S}$, the target value of $\boldsymbol{F}(\boldsymbol{x})$ is determined to be 0, which implies the constraint of $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$. As mentioned above, $\boldsymbol{p}(\boldsymbol{x}) \in [0, 1]$, if we want a stable improvement, when at the same point $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ better satisfy:

$$\boldsymbol{f}_1(\boldsymbol{x}) \cdot \boldsymbol{f}_2(\boldsymbol{x}) \leq 0 \tag{9}$$

In other words, Eq. (9) guarantees a stable improvement while the values of $\boldsymbol{p}(\boldsymbol{x})$ determine the magnitude of improvement. Therefore, we add a new loss term based on Eq. (9) to slightly increase the anisotropy between the two subnetworks:

$$\mathcal{L}_{sign} = \lambda_5 \sum_{\boldsymbol{x} \in \mathcal{P}} max(\mathcal{F}_{base}(\boldsymbol{x}) \cdot \mathcal{F}_{fine}(\boldsymbol{x}), 0) \tag{10}$$

We implement a two-stage progressive training via symmetrically diminishing/increasing learning rates and loss weights for the base/entire networks. Specifically, in the first stage, we calculate the loss:

$$\begin{aligned} \mathcal{L}_{stage1} =& \kappa \cdot \mathcal{L}_{recon}(\mathcal{F}_{base}) \\ &+ (1 - \kappa)(\mathcal{L}_{recon}(\boldsymbol{F}) + \mathcal{L}_{sign}) \end{aligned} \tag{11}$$

$$\kappa = \frac{1}{2}(1 + \cos(\pi \frac{t - T}{1 - T})) \tag{12}$$

where $\mathcal{L}_{recon}(\mathcal{F}_{base})$ is the loss of merely the base network, $\mathcal{L}_{recon}(\boldsymbol{F})$ is the loss of the entire network, $T \in [0, 1]$ is the assigned training percentile, $t \in [T, 1]$ is the current training progress, and κ is called cosine annealing [20]. In the second stage, we only use the loss

$$\mathcal{L}_{stage2} = \mathcal{L}_{recon}(\boldsymbol{F}) + \mathcal{L}_{sign} \tag{13}$$

to optimize the networks.

4. Experiment

In this section, we present the results of our method. We first evaluate the effectiveness of our network on the task of single object reconstruction using the ABC dataset [16] and some famous shapes and then compare them with other state-of-the-art methods to demonstrate our advancement. Then, we evaluate various design components in an ablation study. Finally, we visualize the output of each part of our network to validate our design.

Our implementation is based on PyTorch, and all experiments were done on a desktop PC with an Intel Core i7 CPU (3.6 GHz) and a GeForce 3080 Ti GPU (16 GB memory).

4.1. Implementation Details

Network Details Three subnetworks of our method have the same MLP structure, and they have 4 hidden layers with 256 neural units each. What different is that the BaseNet $\mathcal{F}_{base}$ uses Softplus as activation while the FineNet $\mathcal{F}_{fine}$ and PNet $\mathcal{P}$ utilize $\sin$ as activation. We also adopt geometric initialization from IGR and the initialization scheme from SIREN. For different types of shapes, we choose a different ω_0 in the first layer to obtain a better reconstruction quality. We train our models for 4000 epochs using the ADAM optimizer with an initial learning rate of 10^{-3} and decay to 10^{-4} for conventional MLP, and an initial rate of 10^{-4} and decay to 10^{-5} for SIREN. All models utilize cosine annealing [20] after the 20% training process of the first stage and during the whole training process of the second stage. The weight descent of our loss function has a similar process to the learning rate.

Dataset We examine our method mainly on CAD shapes due to their large smooth area and complex topological structure. For this reason, most models in this paper come from the ABC dataset and are randomly sampled 250000 points with normals as the input point cloud. We also test our model on some famous shapes that include detailed geometric textures such as Bunny, Dragon, and Bimba.

Evaluation Metrics The quantitative metrics used in this paper for shape reconstruction are symmetric Chamfer Distance (CD) and Normal Consistency (NC). Specifically, we randomly sample a set of 25000 points from the extracted surface $\mathcal{X}$ and the ground-truth surface $\mathcal{X}_{GT}$ respectively for the calculation. In all tables that appear in this paper, the value of CD (Chamfer Distance) is scaled by 10^5, and the value of NC (Normal Consistency) is scaled by 10^2.

Visual Metrics In this paper, we used not only numerical evaluation criteria but also visual criteria. We add *reflection lines* to judge the quality of a surface. Reflection lines, which are repeated infinite, non-dispersive light sources parallel to some line, can reveal surface flaws, particularly discontinuities in normals, indicating that the surface is not $\mathcal{C}^2$, i.e., the changes of normals. A simple visual phenomenon is that the flatter the plane (the normals change more regularly), the more orderly the reflection lines we can see. Therefore, the neater reflection lines indicate the higher quality of the reconstructed surface.

4.2. Comparison of Surface Reconstruction

We compare our approaches with several baseline methods: 1) IGR [11] uses 8 hidden layers with 512 neural units and has a single skip connection from the input to the middle layer. 2) SIREN [28] uses 4 hidden layers with 256 neu-

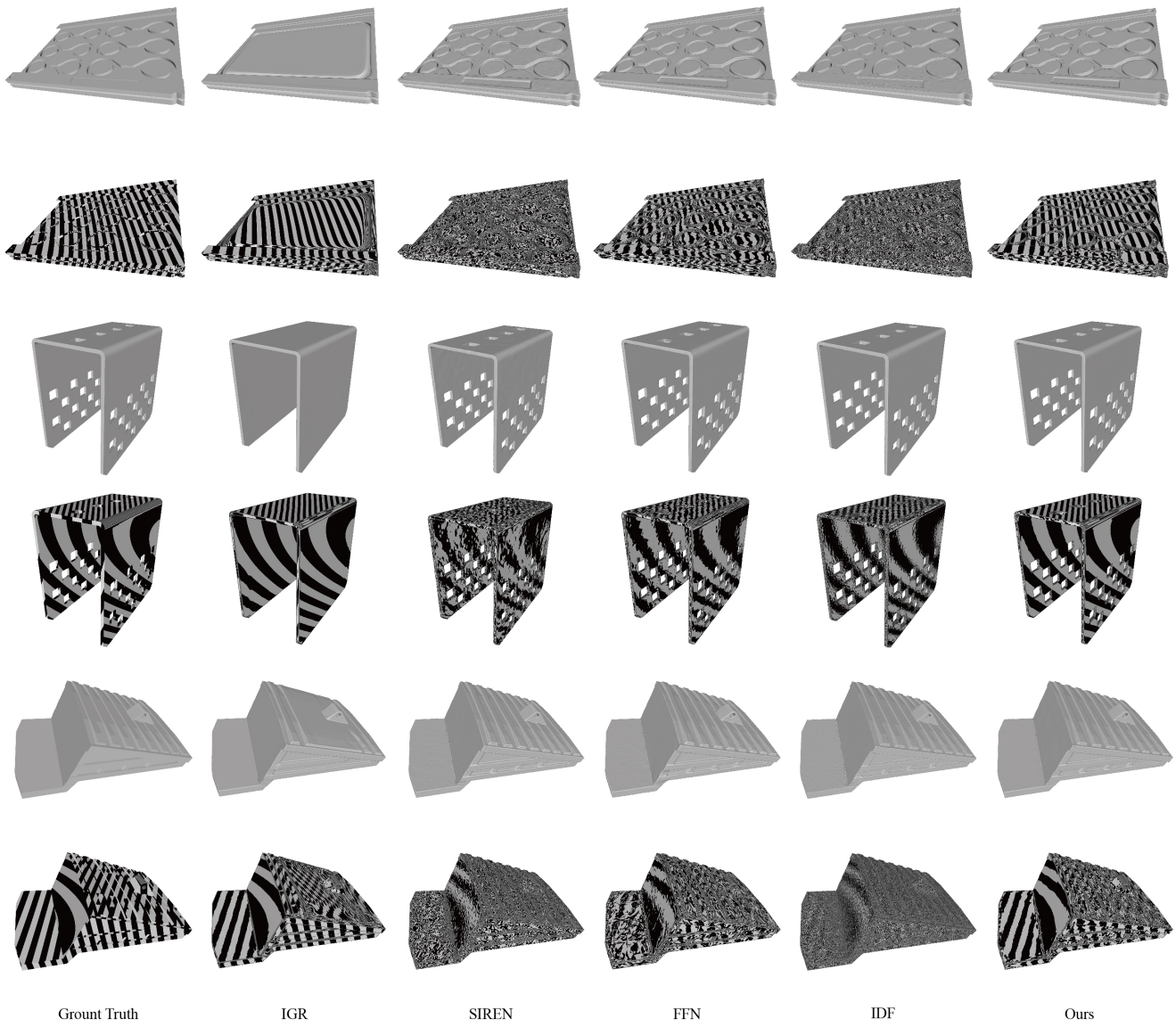

Figure 5. **Visual results of CAD-type models**. Every reconstruction result has two types of visualization. The above is the direct rendering, and the below is the rendering results after adding reflection lines, which makes it clear that our MixNet has both structural details and smooth planes. The names of these models, from top to bottom, are *mould*, *hole* and *part*.

ral units. 3)FFN [30] uses 8 hidden layers with 256 neural units. Additionally, we apply a skip connection in the middle layer as in IGR. 4) IDF [32] both the base and the displacement nets have 4 hidden layers with 256 neural units each. All these methods above use the code and training configuration provided by their papers.

The CAD-type shape reconstruction results are visualized in Fig. 5. As we can see, compared with IGR, our approach is able to reconstruct the shape details and topological characteristics correctly. As for SIREN, FFN, and IDF, their extracted surfaces may contain spurious components and reconstruct a bumpy surface, which can be seen clearly after we add the reflection lines. Note that it is non-trivial to resolve this issue for their results: the incorrect shape reconstruction is hard to repair, and the quality of the bumpy surface may be improved by some smoothing method, but it is harder to smooth precisely the region that we want to smooth. It is more likely to clean up both bumpy surfaces and shape details. Table 1 shows that our MixNet also achieves the best numerical performance among all the compared methods in the CAD-type shapes.

We also test our model on other types of shapes, as

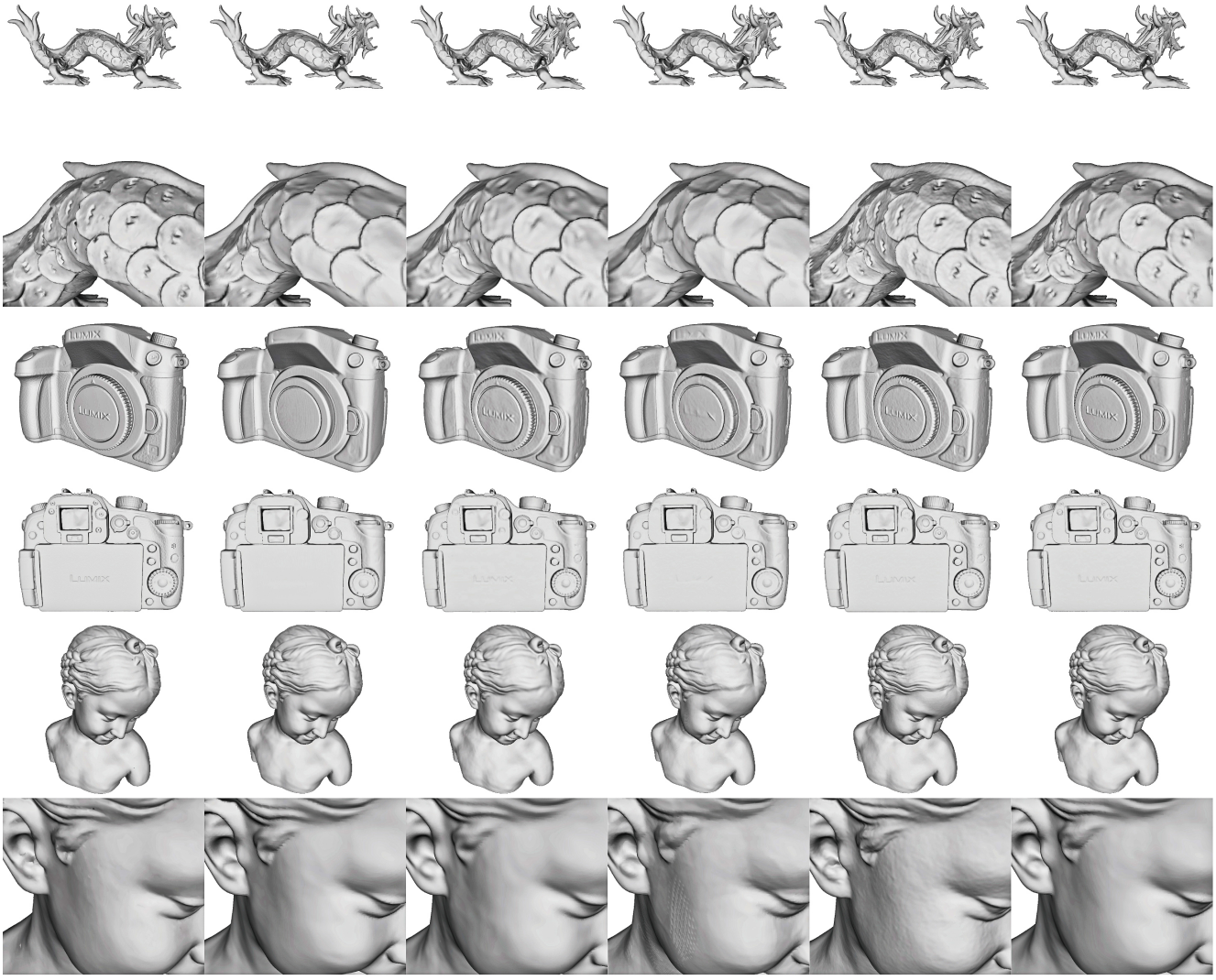

Figure 6. **Visual results of general models**. Every reconstruction result has two types of visualization. The above is the overall view, and the below is the zoom-in view or view from another perspective. The *Dragon* and *Camera* illustrate the detail recovery ability of our method. The *Bimba* illustrate the consistent reconstruction quality of our method whenever the input shapes have more details or have less details, where IDF and FFN produce undesired textures on the right cheek.

shown in Fig. 6. All these methods have good visual quality. But the zoom-in figures show that our MixNet recovers more details than IGR, SIREN, and FFN. Although IDF has the same level of accurate detail expression, its overfitting always results in substantial bumps in the flat area, which diminishes the elegance of the shapes. Compared with IDF, our method is not as good for texture representation, such as the leather texture of the *camera*, but IDF would generate undesired textures in some originally smooth areas. For example, the area outside the folds of *dragon*'s scales, the lens cap of *camera*, and the face of *bimba*, the original model is smooth, but the model reconstructed by IDF is textured. In other words, our method does not generate redundant information due to simple inputs. Table 2 demonstrates our steady improvement among these general shapes.

4.3. Ablation Study

Types of subnetworks We have done an ablation study on the selection of types of networks for each subnetwork, and we have tested the performances of different network combinations on different types of shapes, as shown in Table 3 and Fig. 7. The selection that $\mathcal{F}_{base}$ selects conventional MLP, $\mathcal{F}_{fine}$ and $\mathcal{P}$ select SIREN seems the best combination in numerical results and visualization of both CAD-

Method $\binom{\text{CD}}{\text{NC}}$	Hole	Mould	Part
IGR	29.94	14.61	19.42
	90.14	88.50	90.11
SIREN	25.55	10.84	14.68
	94.69	92.58	92.27
FFN	24.47	10.19	13.88
	95.08	92.38	92.32
IDF	21.41	10.41	13.63
	95.25	91.77	91.71
Ours	**21.18**	**9.99**	**13.52**
	95.55	**92.74**	**92.70**

Table 1. Numerical results on implicit reconstruction from CAD-type models. Our MixNet has much lower Chamfer distance than IGR.

Method $\binom{\text{CD}}{\text{NC}}$	Bunny	Dragon	Bimba	Fandisk
IGR	21.80	5.11	11.40	10.52
	97.56	91.32	97.90	97.70
SIREN	19.15	4.79	11.94	10.70
	97.66	**91.89**	97.93	97.73
FFN	20.15	4.83	11.87	10.70
	97.75	91.85	97.92	97.65
IDF	18.62	4.72	10.89	10.53
	97.73	91.31	97.90	97.59
Ours	**18.18**	**4.63**	**10.67**	**10.37**
	97.93	91.77	**97.96**	**97.82**

Table 2. Numerical results on implicit reconstruction from general models. Our MixNet also has a relatively stable improvement.

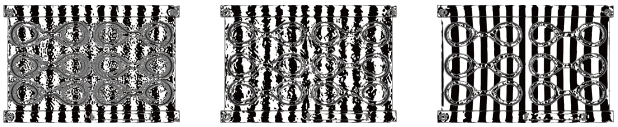

Figure 7. **Comparison of other type combinations**. The three letters represent the types of $\mathcal{F}_{base}$,$\mathcal{F}_{fine}$ and $\mathcal{P}$, respectively. Here C denotes conventional MLP, and S denotes SIREN.

type shapes and general shapes.

The addition of $\mathcal{L}_{sign}$ We also tested the impact of the new-designed loss term $\mathcal{L}_{sign}$ as shown in Table 4. The experiments show that the $\mathcal{L}_{sign}$ is able to lead to a steady improvement in the numerical standard in most cases. Although the absolute value of the improvement is relatively small, we think it is mainly because the new addition can only optimize some combination problems of subnetworks, but it cannot solve the expressive ability of the subnetwork (such as MLPs and SIRENs) itself.

4.4. The choice of ω_0

We investigated the effect of the size of ω_0 on the results. Since there are two possible places where SIREN might be used, we tested it separately, as shown in Table 5 and Table 6. It can be seen that the larger value of ω_0 does not necessarily indicate better expressive ability. In fact, we found that the larger ω_0 necessitates a more careful adjustment of the learning rate; otherwise, the model tends to collapse, which means that the outputs of subnetworks like $\mathcal{F}_{base}$ become messy and are no longer interpretable. Finally, we found $\omega_0 = 30$ to work well for all these shapes tested in this work.

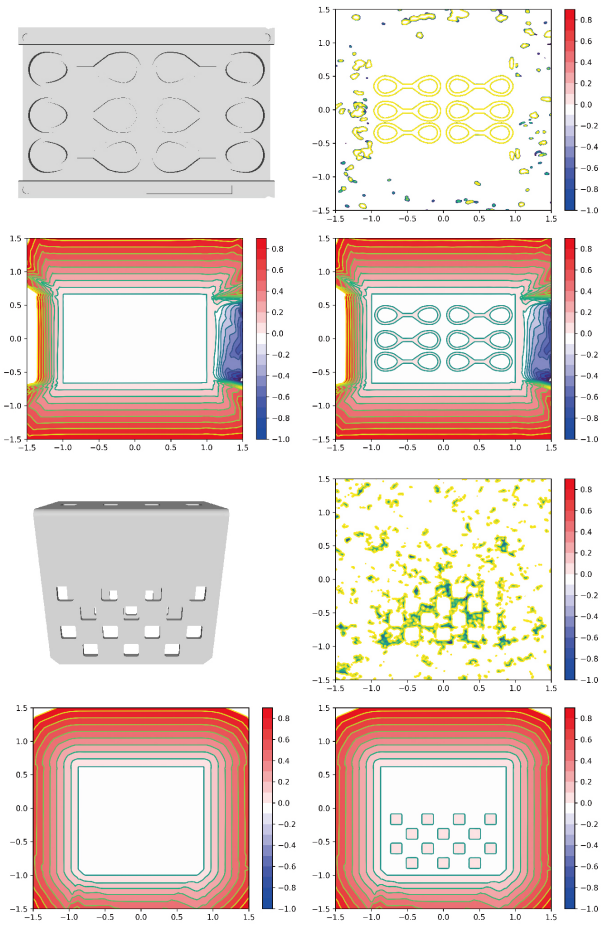

Figure 8. **Visualization of SDF slices**. Top left: The reconstruction result. Top right: SDF slice of FineNet $\mathcal{F}_{fine}$. Bottom left: SDF slice of BaseNet $\mathcal{F}_{base}$. Bottom right: SDF slice of MixNet $\hat{\boldsymbol{F}}$.

4.5. Visualization of Network

We visualize the SDF value of two subnetworks $\mathcal{F}_{base}$ and $\mathcal{F}_{fine}$ which aim to learn coarse shapes and fine features respectively, and the final network $\hat{\boldsymbol{F}}$ which aims to learn the entire precise shape.

Input Type	BaseNet	FineNet	PNet	CD	NC
CAD models	C	S	S	9.99	92.74
	C	S	C	10.31	92.34
	C	C	S	10.38	92.30
General models	C	S	S	18.18	97.93
	S	S	S	20.42	97.63
	S	S	C	19.11	97.79

Table 3. Type selection of the subnetworks. Here C denotes conventional MLP, and S denotes SIREN.

	Mould	Part	Bunny	Bimba
With $\mathcal{L}_{sign}$	9.99	13.52	18.18	10.67
Without $\mathcal{L}_{sign}$	10.04	13.83	18.55	10.84

Table 4. Abalation study of $\mathcal{L}_{sign}$

ω_0 of $\mathcal{F}_{fine}$	15	30	45	60
Bunny(CD)	20.45	20.04	18.18	19.26
Bunny(NC)	97.68	97.71	97.93	97.78
Bimba(CD)	10.96	10.67	11.25	12.42
Bimba(NC)	97.98	97.96	97.90	92.42

Table 5. Choice of ω_0 for $\mathcal{F}_{fine}$.

ω_0 of $\mathcal{P}$	15	30	45	60
Bunny(CD)	18.18	19.45	19.43	19.23
Bunny(NC)	97.93	97.71	97.74	97.80
Mould(CD)	10.40	10.30	10.27	9.99
Mould(NC)	92.24	92.31	92.43	92.74

Table 6. Choice of ω_0 for $\mathcal{P}$.

It is clear, as shown in Fig. 8, that subnetwork $\mathcal{F}_{base}$ contributes to a base smooth surface that almost has a rough shape, and the subnetwork $\mathcal{F}_{fine}$ contributes to a detailed SDF around the details or topological structure that is not captured by $\mathcal{F}_{base}$ to revise the SDF locally. Outside these regions, $\mathcal{F}_{fine}$ tends to be 0 almost everywhere. This reveals why our model can recover a large smooth plane after reconstructing the details correctly.

5. Conclusion and Future Work

In this paper, we propose a novel and effective method for learning 3D implicit signed distance fields from raw point clouds. The combination of two different networks enhances the representation power of conventional MLPs and SIRENs and even outperforms the existing positional encoding schemes, such as Fourier positional encoding, in recovering SDFs. Besides, our PNet seems capable of segmenting the shape by the frequency contained in the shape itself. And we believe this automatic frequency partition enables different networks to concentrate on different regions that are more suitable for their characteristics, significantly boosting their representational power.

In the future, we would like to explore whether it is possible to segment the shape by using two different networks that have different expressiveness and convergence such that we can segment by frequency rather than by human experience and subjective impressions.

Acknowledgements

This work is partially supported by Beijing Natural Science Foundation under Grant Z190004, the National Key Research and Development Program of China under Grant 2020YFA0713703, NSFC (Nos. 11688101, 61872332) and Fundamental Research Funds for the Central Universities.

References

[1] M. Atzmon and Y. Lipman. Sal: Sign agnostic learning of shapes from raw data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3

[2] M. Atzmon and Y. Lipman. SALD: sign agnostic learning with derivatives. In *9th International Conference on Learning Representations, ICLR 2021*, 2021. 2

[3] R. Basri, M. Galun, A. Geifman, D. Jacobs, Y. Kasten, and S. Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International Conference on Machine Learning*, pages 685–694. PMLR, 2020. 2, 3

[4] N. Benbarka, T. Höfer, A. Zell, et al. Seeing implicit neural representations as fourier series. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2041–2050, 2022. 3

[5] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020. 2

[6] Z. Chen, A. Tagliasacchi, and H. Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *Proceedings of IEEE Conference on Com-*

puter Vision and Pattern Recognition (CVPR), 2020. 2

[7] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[8] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 2

[9] Y.-F. Feng, L.-Y. Shen, C.-M. Yuan, and X. Li. Deep shape representation with sharp feature preservation. *Computer-Aided Design*, 157:103468, 2023. 1, 3

[10] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020. 2

[11] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*, pages 3569–3579, 2020. 1, 2, 3, 6

[12] B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019. 3

[13] A. Hertz, O. Perel, R. Giryes, O. Sorkine-Hornung, and D. Cohen-Or. Sape: Spatially-adaptive progressive encoding for neural optimization. *arXiv preprint arXiv:2104.09125*, 2021. 2, 3

[14] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018. 3

[15] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. 2

[16] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 6

[17] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019. 2, 3

[18] S.-L. Liu, H.-X. Guo, H. Pan, P.-S. Wang, X. Tong, and Y. Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2021. 2

[19] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 1

[20] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. 6

[21] 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 (CVPR)*, June 2019. 1, 2, 3

[22] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3

[23] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014. 3

[24] 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 (CVPR)*, June 2019. 1, 2, 3

[25] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 2

[26] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019. 2, 3

[27] B. Ronen, D. Jacobs, Y. Kasten, and S. Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[28] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 1, 2, 3, 6

[29] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 2

[30] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 1, 2, 3, 7

[31] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019. 2

[32] W. Yifan, L. Rahmann, and O. Sorkine-hornung. Geometry-consistent neural shape representation with implicit displacement fields. In *International Conference on Learning Representations*, 2021. 1, 2, 7

[33] G. Yüce, G. Ortiz-Jiménez, B. Besbinar, and P. Frossard. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19228–19238, 2022. 3

[34] W. Zhao, J. Lei, Y. Wen, J. Zhang, and K. Jia. Sign-agnostic implicit learning of surface self-similarities for shape modeling and reconstruction from raw point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10256–10265, 2021. 2