PET-NeuS: Positional Encoding Tri-Planes for Neural Surfaces

Yiqun Wang
Chongqing University and KAUST

Ivan Skorokhodov
KAUST

Peter Wonka
KAUST

Abstract

A signed distance function (SDF) parametrized by an MLP is a common ingredient of neural surface reconstruction. We build on the successful recent method NeuS to extend it by three new components. The first component is to borrow the tri-plane representation from EG3D and represent signed distance fields as a mixture of tri-planes and MLPs instead of representing it with MLPs only. Using tri-planes leads to a more expressive data structure but will also introduce noise in the reconstructed surface. The second component is to use a new type of positional encoding with learnable weights to combat noise in the reconstruction process. We divide the features in the tri-plane into multiple frequency scales and modulate them with sin and cos functions of different frequencies. The third component is to use learnable convolution operations on the tri-plane features using self-attention convolution to produce features with different frequency bands. The experiments show that PET-NeuS achieves high-fidelity surface reconstruction on standard datasets. Following previous work and using the Chamfer metric as the most important way to measure surface reconstruction quality, we are able to improve upon the NeuS baseline by 57% on Nerf-synthetic (0.84 compared to 1.97) and by 15.5% on DTU (0.71 compared to 0.84). The qualitative evaluation reveals how our method can better control the interference of high-frequency noise. Code available at https://github.com/yiqun-wang/PET-NeuS.

1. Introduction

Implicit neural functions, or neural fields, have received a lot of attention in recent research. The seminal paper NeRF [25] combines neural fields with volume rendering, enabling high-quality novel view synthesis. Inspired by NeRF, NeuS [41] and VolSDF [44] introduce a signed distance function (SDF) into the volume rendering equation and regularize the SDF, so that smooth surface models can be reconstructed. However, these methods use pure MLP networks to encode SDFs. Although these two methods can reconstruct smooth surfaces, they both leave room for im-

provement when it comes to reconstructing surface details.

One research direction ([5, 6, 26, 33, 46]) explores data structures such as tri-planes or voxel grids that are suitable to improve the NeRF framework, in terms of speed or reconstruction quality. However, data structures that are successful for novel view synthesis may not bring immediate success when employed for surface reconstruction as shown in the third column of Fig. 1. While a greater expressiveness to encode local details is useful to better fit the input data, there is also less inductive bias towards a smooth surface. Therefore, noise during image acquisition, high-frequency shading, or high-frequency texture variations are more likely to result in a noisy reconstructed surface.

In our work, we explore how to increase expressiveness to encode local features while at the same time reducing the impact of noise interference. We choose to build on the tri-plane data structure since it consumes less memory and can be easier scaled to higher resolutions.

In our work, we build on EG3D and NeuS to propose a novel framework, called PET-NeuS. First, we propose a method to integrate the tri-plane data structure into a surface reconstruction framework in order to be able to model an SDF with more local details. Second, since the features between tri-plane pixels do not share learnable parameters, we use positional encoding to modulate the tri-plane features, thereby enhancing the smoothness of the learnable features. Third, the positional encoding involves functions of different frequencies. In order to better match different frequencies, we propose to use multi-scale self-attention convolution kernels with different window sizes to perform convolution in the spatial domain to generate features of different frequency bands. This further increases the fidelity of the surface reconstruction while suppressing noise.

We experiment on two datasets to verify the effectiveness of our method, the DTU dataset and the NeRF-Synthetic dataset. Since the DTU dataset contains non-Lambertian surfaces, the ability of the network to resist noise interference can be verified. The NeRF-Synthetic dataset has many sharp features, which can verify that our framework can effectively utilize its improved local expressiveness to better reconstruct local details. We show superior performance compared to state-of-the-art methods on both datasets.

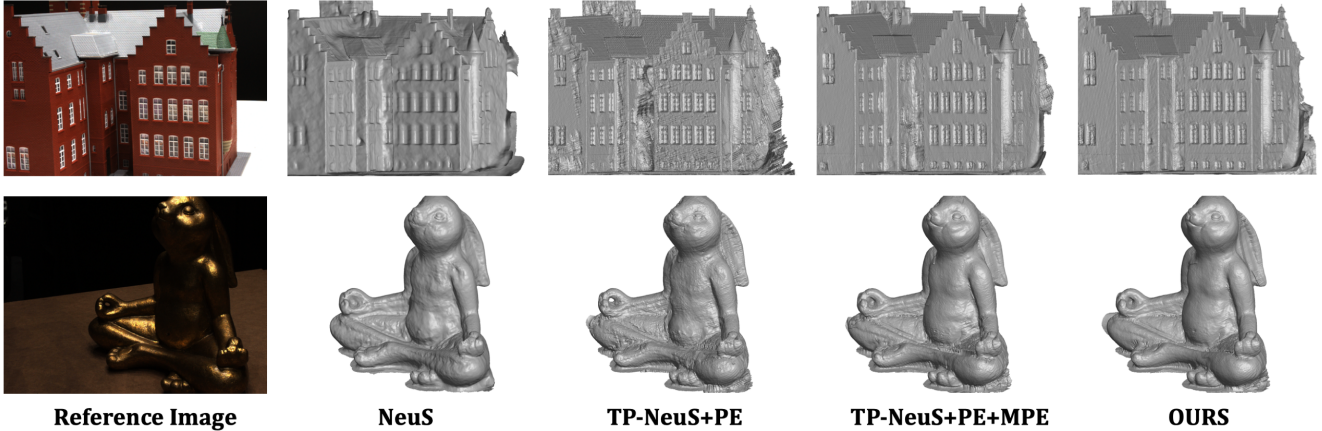| Reference Image | NeuS | TP-NeuS+PE | TP-NeuS+PE+MPE | OURS |

Figure 1. The challenge of using the tri-plane representation directly. First column: reference image. Second to the fifth column: NeuS, Learning SDF using tri-planes, OURS without self-attention convolution, and OURS.

In summary, our contributions are as follows:

- We propose to train neural implicit surfaces with a tri-plane architecture to enable the reconstructed surfaces to better preserve fine-grained local features.

- We derive a novel positional encoding strategy to be used in conjunction with tri-plane features in order to reduce noise interference.

- We utilize self-attention convolution to produce tri-plane features with different frequency bands to match the positional encoding of different frequencies, further improving the fidelity of surface reconstruction.

2. Related Work

Neural fields. Neural fields are a popular representation of 3D scenes. Two fundamental representations are to encode occupancy [7, 24] or signed distance functions [30] using MLPs. In order to improve the representational expressiveness of the models, [8, 31] use 3D convolutions on voxels to learn local shape features and construct the occupancy function and signed distance function of the shapes, respectively. Due to locality, implicit neural function representations can model fine-grained scenes. Subsequently, some works [1, 9] focus on solving the problem of learning implicit functions on shapes with boundary, while others [22, 39, 43] further exploit voxel representations to improve the quality of modeling. Then the seminal work, NeRF [25], incorporates the implicit neural function into the volume rendering formula, thus achieving high-fidelity rendering results. Due to the implicit neural function representing the scene, the method produces excellent results for novel view synthesis. Some follow-up

works [3,15,23,38,40] use multi-scale techniques or encoding strategies to learn fine-grained details. Recently, many works [5, 6, 26, 33, 46] use voxel grids or a factored representation (e.g. tri-planes) to further improve training speed or rendering quality.

Neural surface reconstruction. Surface reconstruction from multiple views is a popular topic in 3D vision. Traditional algorithms for multi-view surface reconstruction usually use discrete voxel-based representations [4, 11, 16, 18, 28, 37] or reconstruct point clouds [2, 12, 13, 35, 36]. Discrete voxel representations suffer from resolution and memory overhead, while point-based methods require additional consideration of missing point clouds and additional surface reconstruction steps. Recently, some methods based on neural implicit surfaces have emerged to reconstruct shapes using continuous neural implicit function from multi-view images. Surface rendering and volume rendering are two key techniques. DVR [27] and IDR [45] adopt surface rendering to model the occupancy functions or signed distance functions for 3D shapes, respectively. The methods based on surface rendering need to compute the precise location of the surface to render images and gradient descent is applied only on the surface. NeRF-based methods like UNISURF [29], VolSDF [44], and NeuS [41] incorporate occupancy functions or the signed distance functions into the volume rendering equation. Since the implicit function can be regularized by the Eikonal loss, the reconstructed surface can maintain smoothness. The NeuralPatch method by [10] is a post-processing step to VolSDF. It binds the colors in the volume to nearby patches with a homography transformation. Since the computation of patch warping relies on accurate surface normals, we consider the algorithm as a post-process that can be applied to any method. HF-NeuS [42] introduces an additional MLP for modeling a
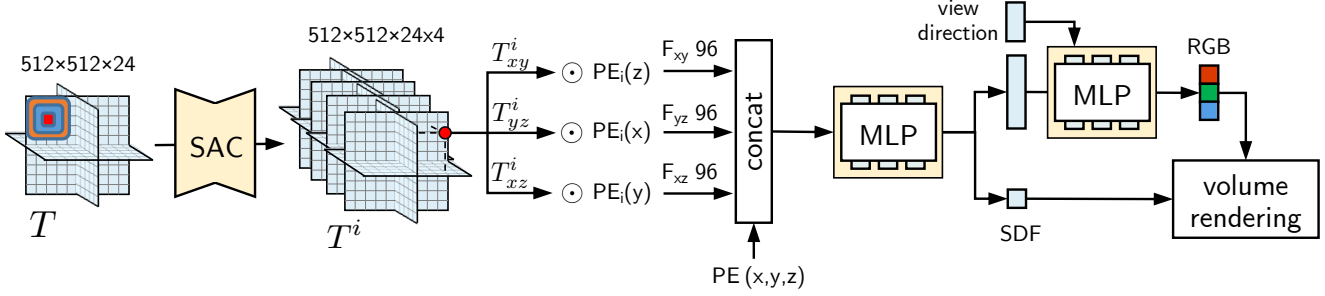
Figure 2. Our PET-NeuS framework consists of a tri-plane architecture, two types of positional encoding, self-attention convolution (SAC), and MLP mapping blocks.

displacement field to learn high-frequency details and further improve surface fidelity. We choose VolSDF, NeuS, and HF-NeuS as our state-of-the-art competitors.

3. Method

Given a set of images and their camera positions, our goal is to reconstruct the scene geometry represented by a signed distance function (SDF). In this section, we first provide the necessary details on the tri-plane representation [5, 31]. Then, we describe how they can be integrated into NeuS without losing its geometric MLP initialization and, after that, how to combine the grid-based tri-plane representation with conventional sinusoidal positional encoding. Finally, we introduce multi-frequency tri-plane features. Our overall framework is illustrated in Fig. 2.

3.1. Tri-Plane-based NeuS

A signed distance function (SDF) S is a powerful surface representation suitable for many downstream applications and easily convertible to other representation types [21, 30]. It takes a 3D coordinate $(x, y, z) \in \mathbb{R}^3$ as an input, and outputs the (signed) distance $S(x, y, z) = d_s \in \mathbb{R}$ to the nearest point on the scene surface. Neural surface (NeuS) [41] is the most common way to recover such an SDF-based representation of a scene geometry from its (posed) multi-view images. It uses a multi-layer perceptron (MLP) to model the SDF and optimizes it for scene reconstruction using volumetric rendering [25]. The conventional NeuS relies solely on neural networks to encode the scene, which is an expensive representation with limited expressivity [38]. In this work, we explore tri-planes [5, 31] as an additional learnable data structure for modeling SDFs (TP-NeuS).

A tri-plane T is a grid-based 3D data structure, which is composed of three learnable feature planes $T = (T_{xy}, T_{yz}, T_{xz})$ of resolution $R \times R$ and feature dimensionality n_f (i.e., $T_* \in \mathbb{R}^{R \times R \times n_f}$). These planes are orthogonal to each other and form a 3D cube of size L^3 centered at the origin $(0, 0, 0)$. To extract a feature $T(x, y, z) = \boldsymbol{w} \in \mathbb{R}^{3n_f}$ at a 3D coordinate $(x, y, z) \in [-\frac{L}{2}, \frac{L}{2}]^3$,

we project it onto each of the three planes, bilinearly interpolate its nearby feature vectors and concatenate them: $\boldsymbol{w} = (\boldsymbol{w}_{xy}, \boldsymbol{w}_{yz}, \boldsymbol{w}_{xz})$.

After computing the feature vector $\boldsymbol{w}$, we use it to estimate the signed distance d_s. We do this via a shallow 3-layer MLP: $(d_s, \boldsymbol{u}) = \text{MLP}_d(\boldsymbol{w}_{xy}, \boldsymbol{w}_{yz}, \boldsymbol{w}_{xz})$. In addition to the distance value d_s, it produces a feature vector $\boldsymbol{u} \in \mathbb{R}^{256}$, which is passed to the color branch. It is also represented as a 3-layer MLP and predicts view-dependent RGB color value $\text{MLP}_c(\boldsymbol{u}, \boldsymbol{v}_d) = \text{RGB} \in [0, 1]^3$ from the feature vector $\boldsymbol{u}$ and the view direction $\boldsymbol{v}_d \in \mathbb{R}^3$.

There are multiple ways to perform volume rendering with an SDF [29, 41, 42, 44]. We follow HF-NeuS [42] and compute the density value $\sigma \in \mathbb{R}_0^+$ by modeling the transparency as the transformed SDF:

$$\sigma(x, y, z) = s\left(\Psi_s\left(\mathsf{S}(x, y, z)\right) - 1\right) \nabla \mathsf{S}(x, y, z) \cdot \boldsymbol{v}_d \quad (1)$$

where Ψ_s is the sigmoid function with scale parameter $s \in \mathbb{R}$ and $\boldsymbol{v}_d$ is the viewing direction. Following NeRF [25], the volume rendering integral is approximated via α-compositing with $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$, where $\delta_i \in \mathbb{R}_+$ is the distance between adjacent ray samples. The integrated color values constitute the pixel color for the corresponding ray.

3.2. Geometric initialization for TP-NeuS

Prior works (e.g., [1, 41]) showed that proper initialization of the surface can substantially improve its final reconstruction quality. In our experiments, we confirm this observation and find that TP-NeuS converges to a worse solution when naively initialized from random noise (see the Fig. 6 in Appx B). Popular *geometric initialization* [1] initializes the parameters of an MLP-based SDF in such a way, that it (approximately) represents a sphere. Unfortunately, it is designed exclusively for MLPs and thus not directly applicable to grid-based surface representations like tri-planes. To circumvent this, we develop the following simple technique of adapting geometric initialization for a tri-plane-based SDF.

Our initialization strategy is based on converting a properly initialized MLP into a tri-plane representation. For this, we take an 8-layer MLP, initialize it with conventional geometric initialization, and then substitute its first 5 layers with tri-planes. Each $T_{xy}, T_{yz},$ and T_{xz} feature plane is set to values from the MLP by looking up its features in $R \times R$ coordinates of the form $(x, y, 0)$, $(0, y, z)$, and $(x, 0, z)$, respectively. The remaining 3 layers are used as the MLP head on top of the tri-planes to estimate the distance value (see Sec 3.1).

3.3. Positional Encoding for TP-NeuS

Traditional positional encoding [25, 41] uses sinusoidal functions to map raw 3D coordinates into multiple different frequencies to make the network capture the characteristics of different frequency scales. In contrast, tri-planes are an interpolation-based grid representation, which makes it difficult to incorporate similar inductive biases. To mitigate this, we first derive an implicit function representation as a weighted sum of sinusoidal positional embeddings and then design a method of combining them with tri-planes.

A neural implicit function learns a mapping from the coordinates into the function values. A continuous unary function with compact support can be expanded as a Fourier series with a frequency scale $M \mapsto \infty$. In practice, we have only a finite number of frequencies. An MLP can model the coefficients of the Fourier decomposition and also approximate the error caused by finite truncation:

$$f(x) = a_0 + \sum_{m=1}^{M} a_m \cos(mx) + \sum_{m=1}^{M} a_{-m} \sin(mx) \quad (2)$$

$$= \text{MLP}(\{\cos(mx), \sin(mx)\}_{m=1}^{M}), \quad (3)$$

where a_m, a_{-m} denote the amplitude coefficients. We can rewrite the $f(x)$ decomposition as:

$$f(x) = \sum_{m=-M}^{M} a_m \Theta_m^x \quad (4)$$

where Θ_t^x is an auxiliary variable introduced for brevity:

$$\Theta_t^x = \begin{cases} \cos(tx) & t > 0 \\ 1 & t = 0 \\ \sin(tx) & t < 0 \end{cases} \quad (5)$$

We can similarly re-write the Fourier series decomposition for an implicit function $f(x, y, z)$ in 3D:

$$f(x, y, z) = \sum_{k=-K}^{K} \sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{mnk} \Theta_m^x \Theta_n^y \Theta_k^z \quad (6)$$

where m, n, and k are the different frequencies for $x, y,$ and z axes with the maximum number of frequency scales

of $M, N, K \mapsto \infty$, and a_{mnk} denote the corresponding amplitude values.

In the above representation, sine and cosine waves from different dimensions x, y, z are entangled with each other through multiplications. We want to obtain a representation where each sinusoidal function can be treated individually as an independent input. To do this, we perform a series of substitutions.

First, we substitute Θ_m^x back using its definition (5), we get:

$$f = \sum_{k=-K}^{K} \sum_{n=-N}^{N} \left(\sum_{m=1}^{M} a_{mnk} \cos(mx) \right) \Theta_n^y \Theta_k^z \quad (7)$$

$$+ \sum_{k=-K}^{K} \sum_{n=-N}^{N} \left(\sum_{m=1}^{M} a_{(-m)nk} \sin(mx) \right) \Theta_n^y \Theta_k^z \quad (8)$$

$$+ \sum_{k=-K}^{K} \sum_{n=-N}^{N} (a_{0nk}) \Theta_n^y \Theta_k^z \quad (9)$$

The first and second terms can be rewritten as a combination of $\cos(mx)$ and $\sin(mx)$ by introducing two auxiliary functions $\hat{g}_m(y, z)$ and $\hat{g}'_m(y, z)$:

$$\hat{g}_m(y, z) = \sum_{k=-K}^{K} \sum_{n=-N}^{N} a_{mnk} \Theta_n^y \Theta_k^z \quad (10)$$

$$\hat{g}'_m(y, z) = \sum_{k=-K}^{K} \sum_{n=-N}^{N} a_{(-m)nk} \Theta_n^y \Theta_k^z \quad (11)$$

Then, we alternatively expand Θ_n^y and Θ_k^z in a similar manner to represent the first and second terms of $f(x, y, z)$ as a combination of $\cos(ny), \sin(ny), \cos(kz),$ and $\sin(kz)$. This yields similar functions $\hat{h}_n, \hat{h}'_n, \hat{w}_k$ and $\hat{w}'_k$. We provide the detailed derivations for the third terms and yield $g_m, g'_m, h_n, h'_n, w_k$ and w'_k for $f(x, y, z)$ in Appx A.

As a result, we can approximate the implicit function $f(x, y, z)$ with an MLP with sine/cosine inputs of the following form:

$$f(x, y, z) \approx \text{MLP}\left(\left\{ \begin{array}{c} \cos(mx) \\ \sin(mx) \\ g_m(y, z)\cos(mx) \\ g'_m(y, z)\sin(mx) \\ \cos(ny) \\ \sin(ny) \\ h_n(x, z)\cos(ny) \\ h'_n(x, z)\sin(ny) \\ \cos(kz) \\ \sin(kz) \\ w_k(x, y)\cos(kz) \\ w'_k(x, y)\sin(kz) \end{array} \right\}_{mnk}^{\textit{flatten}} \right) \quad (12)$$

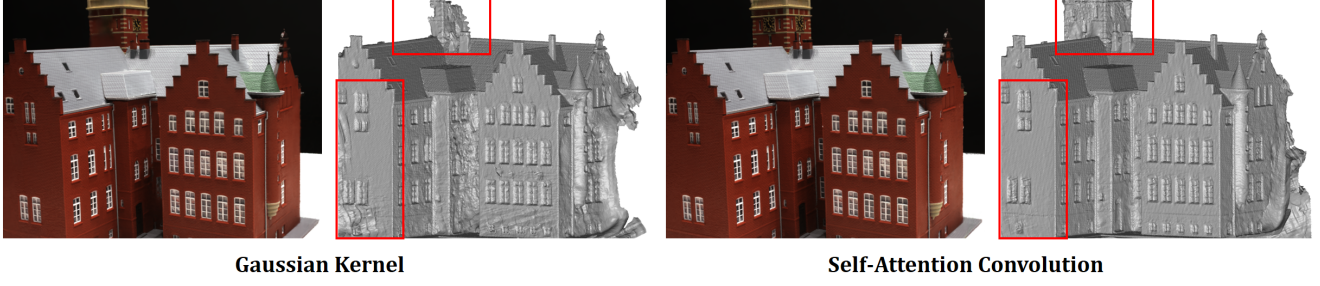**Gaussian Kernel** **Self-Attention Convolution**

Figure 3. Comparing Gaussian kernels with our self-attention kernels. For each method, the left shows the reconstructed image and the right the reconstructed surface.

Such an approximation is directly applicable to our SDF function S. Conventional neural surface reconstruction methods represent the surface entirely through a neural network: $S(x, y, z) = \text{MLP}(\text{PE}(x, y, z))$. Since the functions g, h, and w in (12) are all highly nonlinear, the underlying MLP should be of high capacity to have a low reconstruction error. Tri-planes can take advantage of the high-capacity features to replace the large MLP backbone network, but this direct replacement poses the following issue. Tri-planes do not carry any inductive biases about positional encoding, and due to the discrete discontinuities of their grid-based representation and the absence of frequency constraints, the tri-plane features will introduce high-frequency noise. Analyzing Eq. (12) suggests the following way to integrate positional information into tri-planes.

From (12), we observe that the coefficients g, h, and w of positional encoding are consistent with the tri-plane features since these coefficients are all binary functions of the two-dimensional coordinates. We propose to regard g, h, and w functions as the tri-plane features and modulate/multiply them with sin and cos functions of different frequencies. In this way, the output features of tri-planes contain a frequency bound, and the base function should be easier to fit. This analysis leads to the following parametrization:

$$S(x, y, z) = \text{MLP} \left(\begin{bmatrix} \text{PE}(x, y, z) \\ T_{xy}(x, y) \odot \text{PE}(z) \\ T_{yz}(y, z) \odot \text{PE}(x) \\ T_{xz}(x, z) \odot \text{PE}(y) \end{bmatrix} \right) \quad (13)$$

where $\odot$ denotes element-wise multiplication. Such modulation of tri-plane features via conventional positional embeddings endows the grid-based representation with frequency information, which suppresses high-frequency noise.

3.4. Tri-Planes with Self-Attention

In practice, one can simply set the feature dimensionality of the tri-planes n_f to the dimension of the positional embedding. However, in order to better learn the features of different frequencies, we propose to generate tri-plane features with different frequency bands, where each band contains multiple frequency scales.

Since the product in the frequency domain is the convolution in the spatial domain, one can generate multi-frequency tri-plane features by smoothing them with fixed Gaussian kernels. But when experimenting with this simple technique, we noticed that it smooths features across depth discontinuities, e.g. foreground to background, as depicted in Fig. 3. This happens because each feature plane of the tri-plane representation is an orthogonal projection of the 3D space, and foreground features are getting affected by the background features due to the direct convolution on the plane. This results in the wrong structure of the generated surface, although the rendered image is reasonable. Therefore, we looked for alternative dynamic convolution operations that could be performed.

Inspired by window self-attention convolution [32] and the Swin Transformer architecture [19], we propose to use self-attention convolution with different window sizes to generate features in different frequency bands. We found that using either a sliding window or a shifted window exponentially increases the computational cost of the optimization procedure. To reduce it, we use a single-layer self-attention convolution and directly divide the tri-plane into regular non-overlapping patches with different window sizes (we use the window sizes of 4, 8, and 16 everywhere unless stated otherwise). Since the subsequent MLP combines features of different scales, the features across windows also interact with each other. We take the output features $T^i \in \mathbb{R}^{R \times R \times n_f}$ produced by the i-th SAC convolution, and concatenate them all together with the original features T^0 to form the final tri-plane representation for four frequency bands as follows:

$$T = \text{concat} \left[T^i \right]_{i=0}^{3}. \quad T^i = \left\{ T^i_{xy}, T^i_{yz}, T^i_{xz} \right\}, \quad (14)$$

where $\text{concat}[\cdot]$ is the channel-wise concatenation operation. We multiply the features of the four frequency bands

Table 1. Quantitative results on the NeRF-synthetic dataset. Chamfer distance is on the left, and PSNR is on the right.

Method	Chair	Ficus	Lego	Materials	Mic	Ship	**Mean**	Chair	Ficus	Lego	Materials	Mic	Ship	**Mean**
NeRF	2.12	5.17	3.05	1.51	4.77	3.54	3.36	**33.00**	**30.15**	**32.54**	29.62	32.91	**28.34**	**31.09**
VOLSDF	1.26	1.54	2.83	1.35	3.62	2.92	2.37	25.91	24.41	26.99	28.83	29.46	25.65	26.86
NeuS	0.74	1.21	2.35	1.30	3.89	2.33	1.97	27.95	25.79	29.85	29.36	29.89	25.46	28.05
HF-NeuS	0.69	1.12	0.94	1.08	0.72	2.18	1.12	28.69	26.46	30.72	29.87	30.35	25.87	28.66
PET-NeuS (ours)	**0.65**	**0.71**	**0.58**	**1.05**	**0.49**	**1.57**	**0.84**	29.57	27.39	32.40	**29.97**	**33.08**	26.83	29.87

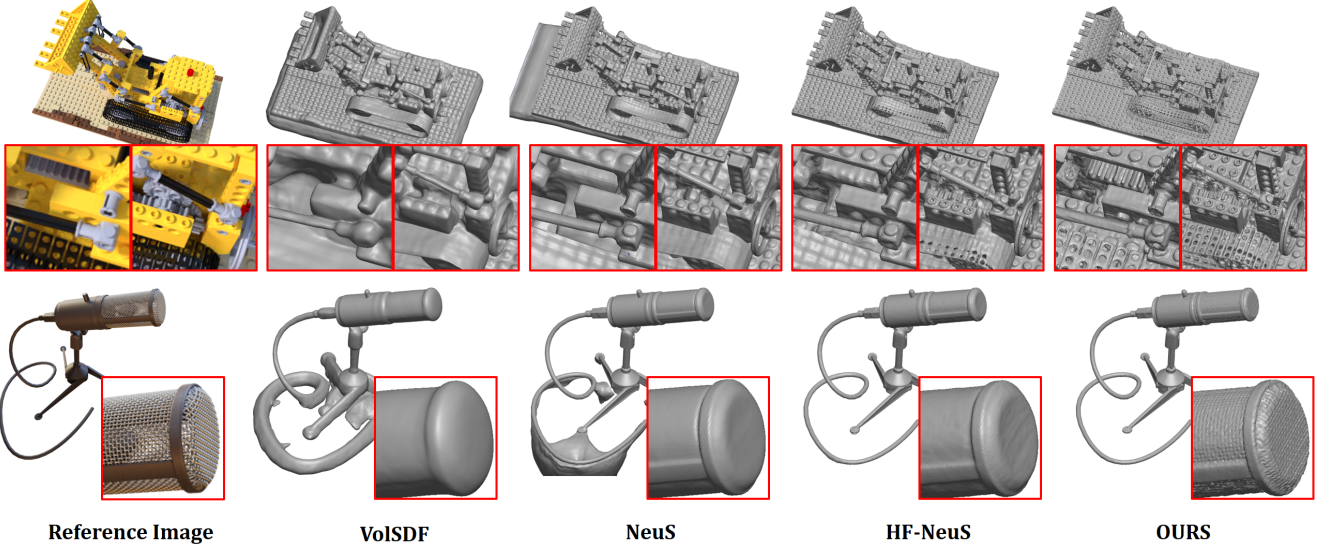**Reference Image** **VolSDF** **NeuS** **HF-NeuS** **OURS**

Figure 4. Qualitative evaluation on the Lego and Mic models. First column: reference images. Second to the fifth column: VolSDF, NeuS, HF-NeuS, and OURS.

with the corresponding low-frequency to high-frequency positional encoding. In this way, we obtain the tri-plane representation with adaptive features for different frequencies, which is well suited for surface reconstruction tasks.

4. Results

4.1. Optimization

We use two different losses in the training (identical to what has been used in previous work NeuS and HF-NeuS). The first one is the color reconstruction loss. The second is the Eikonal loss [14]. We found that total variation regularization [20] (TVloss) can also regularize the SDF like Eikonal loss, but Eikonal loss is especially suitable for learning SDF. Color reconstruction loss is the L1 distance between ground truth colors and the volume rendered colors of sampled pixel set S.

$$\mathcal{L}_{\text{color}} = \frac{1}{|S|} \sum_{s \in S} \left\| \hat{C}_s - C_s \right\|_1 \quad (15)$$

Eikonal loss is a regularization loss on sampled point set I that constrains the implicit function and makes the SDF

smooth.

$$\mathcal{L}_{\text{reg}} = \frac{1}{|I|} \sum_{i \in I} \left[(\|\nabla \mathsf{S}(x_i, y_i, z_i)\|_2 - 1)^2 \right] \quad (16)$$

We employ both loss functions to train our network with a hyperparameter λ. Note that in all settings we do not provide masks and ignore mask loss in the training.

$$\mathcal{L} = \mathcal{L}_{\text{color}} + \lambda \mathcal{L}_{\text{reg}} \quad (17)$$

Datasets. The NeRF synthetic dataset [25] contains posed multi-view images of 800×800 resolution with detailed and sharp features. The DTU dataset [17] is a real dataset that contains posed multi-view images of 1600×1200 resolution. We select the same 15 models as shown in other works for a fair comparison. The DTU dataset contains non-Lambertian surfaces which are testing for methods sensitive to noise. Besides the DTU dataset, 6 challenging scenes are selected from the NeRF-synthetic dataset. Ground truth surfaces and camera poses are provided in both datasets. We also conduct an experiment on other real-world scenes in the Appx D.

Table 2. Quantitative results on DTU (the header's numbers denote scene IDs). Chamfer distance is on top and PSNR is on the bottom.

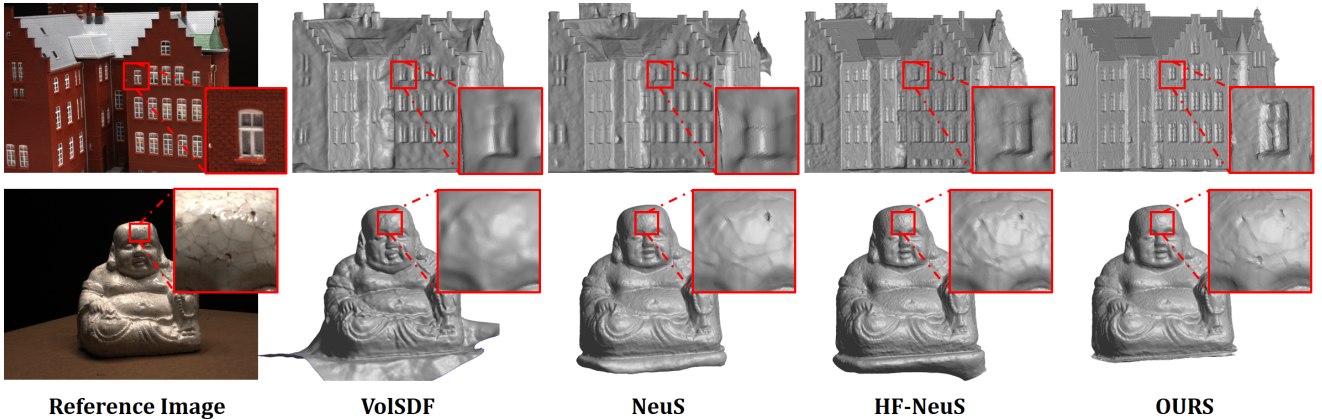| Method	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	**Mean**
NeRF	1.90	1.60	1.85	0.58	2.28	1.27	1.47	1.67	2.05	1.07	0.88	2.53	1.06	1.15	0.96	1.49
VOLSDF	1.14	1.26	0.81	0.49	1.25	0.70	0.72	1.29	1.18	0.70	0.66	1.08	0.42	0.61	0.55	0.86
NeuS	1.00	1.37	0.93	0.43	1.10	0.65	**0.57**	1.48	1.09	0.83	0.52	1.20	0.35	**0.49**	0.54	0.84
HF-NeuS	0.76	1.32	0.70	0.39	1.06	**0.63**	0.63	**1.15**	1.12	0.80	0.52	1.22	**0.33**	**0.49**	0.50	0.77
PET-NeuS (ours)	**0.56**	**0.75**	**0.68**	**0.36**	**0.87**	0.76	0.69	1.33	**1.08**	**0.66**	**0.51**	**1.04**	0.34	0.51	**0.48**	**0.71**
NeRF	26.24	25.74	26.79	27.57	31.96	31.50	29.58	32.78	28.35	32.08	33.49	31.54	31.0	35.59	35.51	30.65
VOLSDF	26.28	25.61	26.55	26.76	31.57	31.50	29.38	33.23	28.03	32.13	33.16	31.49	30.33	34.90	34.75	30.38
NeuS	28.20	27.10	28.13	28.80	32.05	33.75	30.96	34.47	29.57	32.98	35.07	32.74	31.69	36.97	37.07	31.97
HF-NeuS	29.15	27.33	28.37	28.88	32.89	**33.84**	**31.17**	34.83	**30.06**	33.37	35.44	33.09	32.12	37.13	37.32	32.33
PET-NeuS (ours)	**30.15**	**27.69**	**29.17**	**29.55**	**33.78**	33.65	30.96	**35.21**	29.53	**33.43**	**36.58**	**33.54**	**32.34**	**38.50**	**37.61**	**32.78**

Figure 5. Qualitative evaluation on DTU house and Buddha models. First column: reference images. Second to the fifth column: VolSDF, NeuS, HF-NeuS, and OURS.

Baselines. Four state-of-the-art baselines are considered: VolSDF [44] embeds an SDF into the density function and employs an error bound by using a sampling strategy. The training time is 12 hours on the DTU dataset. NeuS [41] incorporates an SDF into the weighting function and uses sigmoid functions to control the slope of the function. The training time is 16 hours on the DTU dataset. HF-NeuS [42] builds on NeuS using offset functions. The training time is 20 hours on the DTU dataset. NeRF [25] does not focus on high-fidelity surface reconstruction but high-quality image synthesis, hence producing low-quality surfaces. We include this method for completeness and the training time is 10 hours on the DTU dataset. Since NeuS and VolSDF compared to older methods and demonstrated better results for surface reconstruction, we do not compare with methods such as IDR [45], or UNISURF [29].

Evaluation metrics. For the DTU dataset, we follow the official evaluation protocol to evaluate the Chamfer distance. For the NeRF synthetic dataset, we compute the Chamfer distance between the ground truth shape and the reconstructed surface. For completeness, PSNR metric is used to

measure the quality of reconstructed images. However, we would like to emphasize that the Chamfer distance is the most important metric for surface reconstruction methods.

Implementation details. We use two MLPs to model the SDF and color function on top of tri-plane features. Each MLP consists of only 3 layers. The hyperparameter for the Eikonal regularization is $\lambda = 0.1$. We normalize scenes to fit $L = 3.0$. The resolution of each tri-plane is 512×512. The number of tri-plane feature channels is $n_f = 24$. Our three window sizes are set to 4, 8, and 16. We use positional encoding with 8 scales, which means $M = N = K = 8$ and each band contains 2 scales. The Adam optimizer with a learning rate $5e-4$ is utilized for network training using a single NVIDIA TITAN V100 graphics card. The training time is 9 hours on the DTU dataset, which is faster than all competitors. After training, the time for extracting a mesh with 512 grid resolution is 20 seconds and for rendering a 1600x1200 resolution image is around 100 seconds.

Table 3. Ablation study results (Chamfer distance) on DTU (the header's numbers denote scene IDs). "PE" stands for positional encoding, "MSA" — for multi-scale architecture, "SAC" — self-attention convolutions, "MPE" — modulating positional encoding. We consider tri-planes with PE, MPE and SAC as our full model.

Method	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	**Mean**
TP-NeuS	1.51	1.32	1.77	0.66	1.53	1.34	1.11	1.59	1.58	0.90	0.83	1.84	0.92	0.77	0.72	1.23
TP-NeuS + PE	1.21	1.13	1.26	0.46	1.07	0.90	0.82	1.41	1.21	0.83	0.58	1.69	0.43	0.58	0.55	0.94
+ MSA	1.00	0.90	1.09	0.40	0.95	0.86	0.79	1.39	1.17	0.80	0.62	1.69	0.40	0.57	0.53	0.88
+ SAC	0.75	0.92	0.99	0.41	0.93	0.88	0.74	1.37	1.16	0.80	0.58	1.38	0.38	0.54	0.53	0.82
+ MPE	0.68	0.94	0.92	0.40	0.91	0.83	0.72	1.36	1.18	0.77	0.55	1.37	0.37	0.53	0.51	0.80
+ MPE + SAC	0.56	0.75	0.68	0.36	0.87	0.76	0.69	1.33	1.08	0.66	0.51	1.04	0.34	0.51	0.48	0.71

4.2. Comparison

We first report quantitative comparisons on the NeRF-synthetic dataset [25]. In Table 1, we show Chamfer distance on the left and the PSNR values on the right. The results show that our proposed framework PET-NeuS has the best surface reconstruction quality compared to all other methods. This means that our network has the ability to better preserve local features. NeRF performs very well in PSNR. The reason could be that NeRF only focuses on reproducing the colors while the SDF-based reconstruction methods loose some of their color-fitting abilities due to the geometric regularization. Besides outperforming other baselines in terms of quantitative error, we also show the visual effect of the improved reconstruction (Fig. 4). We find the reconstruction of the bumps and the wheel holes of the Lego model and the grid of the Mic model to be particularly impressive. The reconstructed fine-grained structures are a lot better than what can be achieved with previous work.

The quantitative results on the DTU dataset [17] are shown in Table 2. We show Chamfer distance on the top and the PSNR values on the bottom. For the Chamfer distance, PET-NeuS surpasses NeuS and VolSDF. Compared with HF-NeuS, PET-NeuS is even better. Our PSNR outperforms all other competitors on the DTU dataset. The qualitative results compared with other methods are shown in Fig. 5. The reconstructed surfaces by PET-NeuS preserve fine-grained details. For instance, the holes between the eyes of the Buddha and the windows are more obvious.

4.3. Ablation study

In Table 3, we conduct an ablation study to analyze the effect of each component. "TP-NeuS" refers to just using tri-planes and MLPs to model the SDF: $\mathsf{S}(x, y, z) = \mathrm{MLP}([T(x, y, z)])$. Learning SDFs with positional encoding ("PE") is denoted as "TP-NeuS + PE" with $\mathsf{S}(x, y, z) = \mathrm{MLP}([\mathrm{PE}(x, y, z), T(x, y, z)])$. We consider this as our baseline. "MPE" means we modulate tri-plane features with positional encoding as in (13). "SAC" refers to generating features with different frequencies using self-attention convolution. To show the effectiveness of SAC, we compare

it against the architecture with multi-scale tri-planes representation, denoted as "MSA". It uses tri-planes for multiple resolutions and is described in the Appx C. We regard "TP-NeuS + PE + MPE + SAC" as our proposed full architecture PET-NeuS. We conduct experiments on the DTU dataset quantitatively. From the results, we can observe that the result of using only "TP-NeuS + PE" still results in a large geometric error. We believe that this is due to the discretization discontinuities. Modulating tri-plane features using positional encoding can suppress noise interference. Using self-attention convolution will match the positional encoding on different frequencies and smooth the noise, which is better than the multi-scale setting. We also provide quantitative results in Fig. 1, in which an improvement in reconstruction quality can be observed.

5. Conclusion and Limitations

We propose PET-NeuS, a novel tri-plane based method for multi-view surface reconstruction. By modulating tri-plane features using positional encoding and producing tri-plane features with different frequencies using self-attention convolution, our surface reconstruction can reduce noise interference while maintaining high fidelity. PET-NeuS produces fine-grained surface reconstruction and outperforms other state-of-the-art competitors in qualitative and quantitative comparisons. One limitation is that we still require a long computation time. It would be an exciting avenue of future work to improve computation time by one or two orders of magnitude without drastically sacrificing quality. Another limitation we observed is a trade-off between reconstructing fine details and adding high-frequency noise to otherwise flat surface areas. As we experimented with many versions of our framework, we observed that network architectures that are more expressive to model surface detail tend to be more prone to overfitting and hallucinating details, e.g. in areas of high-frequency changes in light transport. It would be interesting to investigate this trade-off from a theoretical perspective. Finally, we would like to state that we do not expect a noteworthy negative societal impact due to research on surface reconstruction.

6. Acknowledgements

References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 2, 3, 13

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[4] Adrian Broadhurst, Tom W Drummond, and Roberto Cipolla. A probabilistic framework for space carving. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 388–393. IEEE, 2001. 2

[5] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 1, 2, 3

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 1, 2

[7] Zhiqin Chen and Hao 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

[8] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 2

[9] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020. 2

[10] François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces geometry with patch warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6260–6269, 2022. 2

[11] Jeremy S De Bonet and Paul Viola. Poxels: Probabilistic voxelized volume reconstruction. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 418–425, 1999. 2

[12] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009. 2

[13] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Gipuma: Massively parallel multi-view stereo reconstruction. *Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e. V*, 25(361-369):2, 2016. 2

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 6

[15] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Sape: Spatially-adaptive progressive encoding for neural optimization. In *Advances in Neural Information Processing Systems*, 2021. 2

[16] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 2

[17] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 6, 8

[18] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International journal of computer vision*, 38(3):199–218, 2000. 2

[19] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 5

[20] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 6

[21] Steve Marschner and Peter Shirley. *Fundamentals of computer graphics*. CRC Press, 2018. 3

[22] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788*, 2021. 2

[23] Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14214–14223, 2021. 2

[24] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 2

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 3, 4, 6, 7, 8

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 1, 2

[27] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 2

[28] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013. 2

[29] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021. 2, 3, 7

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2, 3

[31] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 2, 3

[32] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019. 5, 15

[33] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 1, 2

[34] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision*, 2021. 15

[35] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 2, 15

[36] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016. 2

[37] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. 2

[38] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. 2, 3

[39] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 2

[40] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. *arXiv preprint arXiv:2112.03907*, 2021. 2

[41] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 2021. 1, 2, 3, 4, 7, 15

[42] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Improved surface reconstruction using high-frequency details. *Advances in Neural Information Processing Systems*, 2022. 2, 3, 7, 15

[43] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9949–9958, 2021. 2

[44] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 2, 3, 7

[45] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020. 2, 7

[46] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021. 1, 2

In these supplementary materials, we first provide a derivation for incorporating positional encoding into tri-planes. We then provide ablation studies for geometric initialization, frequency bands, other architectures, and EMD evaluation. We also show the details of self-attention convolution and the multi-scale architecture we compared to. Furthermore, we conduct experiments and show some examples of other real-world scenes. We finally show more qualitative comparisons to supplement the main text.

A. Derivations for Positional Encoding Tri-Planes

In this section, we derive Eq. 13 in the main text to incorporate positional encodings into tri-planes. Recall the 3D function $f(x, y, z)$ can be expanded as follows.

$$f(x, y, z) = \sum_{k=-K}^{K} c_k(x, y) \, \Theta_k^z \tag{18}$$

$$= \sum_{k=-K}^{K} \sum_{n=-N}^{N} b_{nk}(x) \, \Theta_n^y \Theta_k^z \tag{19}$$

$$= \sum_{k=-K}^{K} \sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{mnk} \Theta_m^x \Theta_n^y \Theta_k^z \tag{20}$$

where m, n, and k are the different frequencies for x, y, z with maximum number of scales $M, N, K \mapsto \infty$ and

$$\Theta_t^v = \begin{cases} \cos(tv) & t > 0 \\ 1 & t = 0 \\ \sin(tv) & t < 0 \end{cases} \tag{21}$$

The idea is to use $\cos(tv)$ and $\sin(tv)$ to represent the function $f(x, y, z)$. We first expand Θ_m^x to represent $f(x, y, z)$ into the form of $\cos(mx)$ and $\sin(mx)$ as follows.

$$f = \sum_{k=-K}^{K} \sum_{n=-N}^{N} \left(\sum_{m=1}^{M} a_{mnk} \cos(mx) \right) \Theta_n^y \Theta_k^z \tag{22}$$

$$+ \sum_{k=-K}^{K} \sum_{n=-N}^{N} \left(\sum_{m=1}^{M} a_{(-m)nk} \sin(mx) \right) \Theta_n^y \Theta_k^z \tag{23}$$

$$+ \sum_{k=-K}^{K} \sum_{n=-N}^{N} (a_{0nk}) \, \Theta_n^y \Theta_k^z \tag{24}$$

There are three terms in the equation. The first and second terms can be rewritten as a combination of $\cos(mx)$

and $\sin(mx)$ as follows.

$$\sum_{k=-K}^{K} \sum_{n=-N}^{N} \left(\sum_{m=1}^{M} a_{mnk} \cos(mx) \right) \Theta_n^y \Theta_k^z \tag{25}$$

$$= \sum_{m=1}^{M} \left(\sum_{k=-K}^{K} \sum_{n=-N}^{N} a_{mnk} \Theta_n^y \Theta_k^z \right) \cos(mx) \tag{26}$$

$$\sum_{k=-K}^{K} \sum_{n=-N}^{N} \left(\sum_{m=1}^{M} a_{(-m)nk} \sin(mx) \right) \Theta_n^y \Theta_k^z \tag{27}$$

$$= \sum_{m=1}^{M} \left(\sum_{k=-K}^{K} \sum_{n=-N}^{N} a_{(-m)nk} \Theta_n^y \Theta_k^z \right) \sin(mx) \tag{28}$$

We define $\hat{g}_m(y, z) = \sum_{k=-K}^{K} \sum_{n=-N}^{N} a_{mnk} \Theta_n^y \Theta_k^z$ and $\hat{g}'_m(y, z) = \sum_{k=-K}^{K} \sum_{n=-N}^{N} a_{(-m)nk} \Theta_n^y \Theta_k^z$. In this way, the function can be expanded in the following form.

$$f = \left(\sum_{m=1}^{M} \hat{g}_m(y, z) \cos(mx) \right) \tag{29}$$

$$+ \left(\sum_{m=1}^{M} \hat{g}'_m(y, z) \sin(mx) \right) \tag{30}$$

$$+ \sum_{k=-K}^{K} \sum_{n=-N}^{N} (a_{0nk}) \, \Theta_n^y \Theta_k^z \tag{31}$$

However, $\cos(mx)$ and $\sin(mx)$ do not appear in the third term. We therefore would like to find some other way to express the third term using trigonometric functions. We observe that we can alternately expand the other two terms Θ_n^y and Θ_k^z in $f(x, y, z)$ of Eq. 3 in the same way and add

them together as follows.

$$3f = \sum_{m=1}^{M} \hat{g}_m(y, z) \cos(mx) \tag{32}$$

$$+ \sum_{m=1}^{M} \hat{g}'_m(y, z) \sin(mx) \tag{33}$$

$$+ \sum_{k=-K}^{K} \sum_{n=-N}^{N} (a_{0nk}) \Theta_n^y \Theta_k^z \tag{34}$$

$$+ \sum_{n=1}^{N} \hat{h}_n(x, z) \cos(ny) \tag{35}$$

$$+ \sum_{n=1}^{N} \hat{h}'_n(x, z) \sin(ny) \tag{36}$$

$$+ \sum_{k=-K}^{K} \sum_{m=-M}^{M} (a_{m0k}) \Theta_m^x \Theta_k^z \tag{37}$$

$$+ \sum_{k=1}^{K} \hat{w}_k(x, y) \cos(kz) \tag{38}$$

$$+ \sum_{k=1}^{K} \hat{w}'_k(x, y) \sin(kz) \tag{39}$$

$$+ \sum_{k=-N}^{N} \sum_{n=-M}^{M} (a_{mn0}) \Theta_m^x \Theta_n^y \tag{40}$$

where

$$\hat{h}_n(x, z) = \sum_{k=-K}^{K} \sum_{m=-M}^{M} a_{mnk} \Theta_m^x \Theta_k^z \tag{41}$$

$$\hat{h}'_n(x, z) = \sum_{k=-K}^{K} \sum_{m=-M}^{M} a_{m(-n)k} \Theta_m^x \Theta_k^z \tag{42}$$

$$\hat{w}_k(x, y) = \sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{mnk} \Theta_m^x \Theta_n^y \tag{43}$$

$$\hat{w}'_k(x, y) = \sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{mn(-k)} \Theta_m^x \Theta_n^y \tag{44}$$

We further expend Eq. 17 as follows.

$$\sum_{k=-K}^{K} \sum_{n=-N}^{N} (a_{0nk}) \Theta_n^y \Theta_k^z \tag{45}$$

$$= \sum_{k=-K}^{K} \left(\sum_{n=1}^{N} a_{0nk} \cos(ny) \right) \Theta_k^z \tag{46}$$

$$+ \sum_{k=-K}^{K} \left(\sum_{n=1}^{N} a_{0(-n)k} \sin(ny) \right) \Theta_k^z \tag{47}$$

$$+ \sum_{k=-K}^{K} (a_{00k}) \Theta_k^z \tag{48}$$

$$= \sum_{n=1}^{N} \left(\sum_{k=-K}^{K} a_{0nk} \Theta_k^z \right) \cos(ny) \tag{49}$$

$$+ \sum_{n=1}^{N} \left(\sum_{k=-K}^{K} a_{0(-n)k} \Theta_k^z \right) \sin(ny) \tag{50}$$

$$+ \sum_{k=-K}^{K} (a_{00k}) \Theta_k^z \tag{51}$$

Similarly, we can expand Eq. 20, and Eq. 23.

$$\sum_{k=-K}^{K} \sum_{m=-M}^{M} (a_{m0k}) \Theta_m^x \Theta_k^z \tag{52}$$

$$= \sum_{k=1}^{K} \left(\sum_{m=-M}^{M} a_{m0k} \Theta_m^x \right) \cos(kz) \tag{53}$$

$$+ \sum_{k=1}^{K} \left(\sum_{m=-M}^{M} a_{m0(-k)} \Theta_m^x \right) \sin(kz) \tag{54}$$

$$+ \sum_{m=-M}^{M} (a_{m00}) \Theta_m^x \tag{55}$$

and

$$\sum_{k=-N}^{N} \sum_{n=-M}^{M} (a_{mn0}) \Theta_m^x \Theta_n^y \tag{56}$$

$$= \sum_{m=1}^{M} \left(\sum_{n=-N}^{N} a_{mn0} \Theta_n^y \right) \cos(mx) \tag{57}$$

$$+ \sum_{m=1}^{M} \left(\sum_{n=-N}^{N} a_{(-m)n0} \Theta_n^y \right) \sin(mx) \tag{58}$$

$$+ \sum_{n=-N}^{N} (a_{0n0}) \Theta_n^y \tag{59}$$

Therefore the function can be rewritten as follows.

$$3f = \sum_{m=1}^{M} \left(\hat{g}_m(y,z) + \sum_{n=-N}^{N} a_{mn0}\Theta_n^y \right) \cos(mx) \quad (60)$$

$$+ \sum_{m=1}^{M} \left(\hat{g}'_m(y,z) + \sum_{n=-N}^{N} a_{(-m)n0}\Theta_n^y \right) \sin(mx)$$
$$(61)$$

$$+ \sum_{n=1}^{N} \left(\hat{h}_n(x,z) + \sum_{k=-K}^{K} a_{0nk}\Theta_k^z \right) \cos(ny) \quad (62)$$

$$+ \sum_{n=1}^{N} \left(\hat{h}'_n(x,z) + \sum_{k=-K}^{K} a_{0(-n)k}\Theta_k^z \right) \sin(ny) \quad (63)$$

$$+ \sum_{k=1}^{K} \left(\hat{w}_k(x,y) + \sum_{m=-M}^{M} a_{m0k}\Theta_m^x \right) \cos(kz) \quad (64)$$

$$+ \sum_{k=1}^{K} \left(\hat{w}'_k(x,y) + \sum_{m=-M}^{M} a_{m0(-k)}\Theta_m^x \right) \sin(kz)$$
$$(65)$$

$$+ \sum_{m=-M}^{M} (a_{m00})\,\Theta_m^x \quad (66)$$

$$+ \sum_{n=-N}^{N} (a_{0n0})\,\Theta_n^y \quad (67)$$

$$+ \sum_{k=-K}^{K} (a_{00k})\,\Theta_k^z \quad (68)$$

Now we ignore the constant coefficient 3 and the function f can be rewritten by the combination of positional encoding, which can be learned by an MLP network as follows.

$$f(x,y,z) = MLP \left(\left\{ \begin{array}{c} \cos(mx) \\ \sin(mx) \\ g_m(y,z)\cos(mx) \\ g'_m(y,z)\sin(mx) \\ \cos(ny) \\ \sin(ny) \\ h_n(x,z)\cos(ny) \\ h'_n(x,z)\sin(ny) \\ \cos(kz) \\ \sin(kz) \\ w_k(x,y)\cos(kz) \\ w'_k(x,y)\sin(kz) \end{array} \right\}_{mnk}^{\text{flatten}} \right)$$
$$(69)$$

where

$$g_m(y,z) = \hat{g}_m(y,z) + \sum_{n=-N}^{N} a_{mn0}\Theta_n^y \quad (70)$$

$$g'_m(y,z) = \hat{g}'_m(y,z) + \sum_{n=-N}^{N} a_{(-m)n0}\Theta_n^y \quad (71)$$

$$h_n(x,z) = \hat{h}_n(x,z) + \sum_{k=-K}^{K} a_{0nk}\Theta_k^z \quad (72)$$

$$h'_n(x,z) = \hat{h}'_n(x,z) + \sum_{k=-K}^{K} a_{0(-n)k}\Theta_k^z \quad (73)$$

$$w_k(x,y) = \hat{w}_k(x,y) + \sum_{m=-M}^{M} a_{m0k}\Theta_m^x \quad (74)$$

$$w'_k(x,y) = \hat{w}'_k(x,y) + \sum_{m=-M}^{M} a_{m0(-k)}\Theta_m^x \quad (75)$$

B. Additional Ablations

B.1. Geometric Initialization

In this section, we provide an ablation study for geometric initialization. We compare our geometric initialization method (Our Geo Init) with two different settings. The first one is to initialize tri-planes from random noise and the 3-layer MLP with standard geometric initialization [1] (MLP Geo Init). The random noise is from a normal distribution $\mathcal{N}(0,1)$. In the second setting, the standard geometric initialization is not used during the initialization stage (No Geo Init). We show a representative example in Fig. 6 to compare the difference. We can observe that the method without geometry initialization causes the foreground and background to stick together and inconsistent reconstruction results on the belly. Using random noise to initialize tri-planes and standard geometric initialization to initialize MLP results in high-frequency details that are artifacts on the generated surface model. We also show a comparison of chamfer distance in Tab. 4. We run three times for each setting and take the average as a metric to evaluate the different initialization methods. Our geometric initialization leads to more consistent surface reconstruction results.

Table 4. Ablations for geometric initialization.

Method	1	2	3	**Mean**
No Geo Init	1.86	1.79	1.92	1.86
MLP Geo Init	1.39	1.29	1.33	1.34
Our Geo Init	1.04	1.03	1.09	1.05

Reference Image   **No Geo Init**   **MLP Geo Init**   **Our Geo Init**
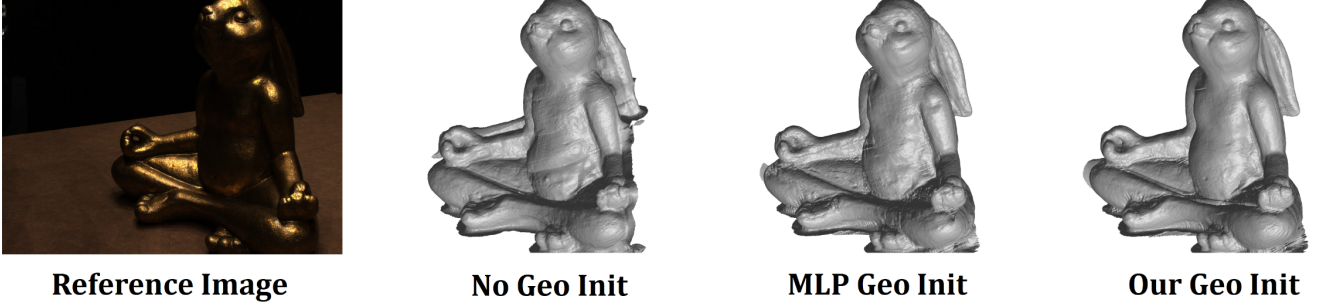
Figure 6. Qualitative evaluation for geometric initialization. First column: reference images. Second to the fourth column: no geometric initialization, geometric initialization for MLP, and our geometric initialization.

Table 5. Additional ablations and EMD evaluation on DTU.

Method	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	**Mean**
+ MSA + SAC (2+2 FB)	0.72	0.89	1.05	0.38	0.92	0.86	0.81	1.40	1.17	0.78	0.60	1.55	0.41	0.55	0.50	0.84
+ MSA + MPE (4FB)	0.98	0.86	0.82	0.38	0.90	0.82	0.74	1.36	1.16	0.75	0.55	1.61	0.35	0.53	0.50	0.82
+ SAC + MPE (2FB)	0.59	0.78	0.70	0.36	0.89	0.80	0.75	1.35	1.12	0.68	0.53	1.10	0.34	0.51	0.50	0.73
+ SAC + MPE (4FB)	0.56	0.75	0.68	0.36	0.87	0.76	0.69	1.33	1.08	0.66	0.51	1.04	0.34	0.51	0.48	**0.71**
+ SAC + MPE (6FB)	0.64	0.76	0.71	0.36	0.87	0.83	0.76	1.36	1.13	0.67	0.54	1.07	0.34	0.52	0.51	0.74
Earth Mover Distance (EMD) evaluation on DTU																
NeuS	1.03	1.08	0.85	0.96	1.22	0.80	0.84	0.98	1.01	0.88	0.71	0.87	0.75	0.80	0.82	0.91
Ours	0.87	0.92	0.71	0.93	1.01	0.85	0.85	0.95	1.02	0.84	0.70	0.82	0.74	0.80	0.81	0.85

B.2. Frequency Bands

We conduct an ablation study to investigate the impact of different frequency bands. Tab. 5 provides Chamfer distance results for 2, 4, and 6 frequency bands with the window sizes of {8}, {4,8,16}, and {1,2,4,8,16}, respectively. Four frequency bands performed the best in our experiments.

B.3. Other Combination of Architectures

We run the experiment for MSA+SAC and MSA+MPE and report the results in Tab. 5. To construct four frequency bands (4FB) for MSA+SAC, we use two-resolution triplanes for MSA and apply two-frequency-band SAC on each tri-plane. Compared with SAC+MPE (4FB) in Tab. 5, SAC+MPE method is superior to MSA+SAC and MSA+MPE.

B.4. EMD evaluation

We follow the very recent SotA methods (e.g., NeuS, VolSDF, IDR), which use *only* Chamfer distance to evaluate the surface quality. For additional EMD evaluation, we compare our method with NeuS and provide the results in Tab. 5. For each method, we evenly sampled 10K points on the reconstructed surface to obtain the point cloud. The results are consistent with Chamfer distance, and our method outperforms NeuS on DTU by 7% on average.

C. Method Details

C.1. Multi-Scale Architecture

Here, we discuss the details of the multi-scale architecture we compared to in the main text. One general strategy to achieve multi-scale behavior is to use a multi-resolution data structure. Therefore, it was our idea to use tri-planes with different resolutions. We use Tri-planes with four different resolutions in the multi-scale architecture to mimic four frequency bands in the SAC (self-attention convolution) structure for a fair comparison. For the highest resolution tri-plane representing high frequency, we used the same resolution as SAC, i.e. 512 dimensions. However, for the low-frequency tri-planes, we use three different tri-planes with different resolutions, namely 256×256, 128×128, and 64×64. We then concatenate features generated by these tri-planes to obtain multi-scale tri-plane features. We show the results in Table 3 in the main text and find that our results using the self-attention convolution outperform the results using the multi-scale architecture. An intuitive explanation
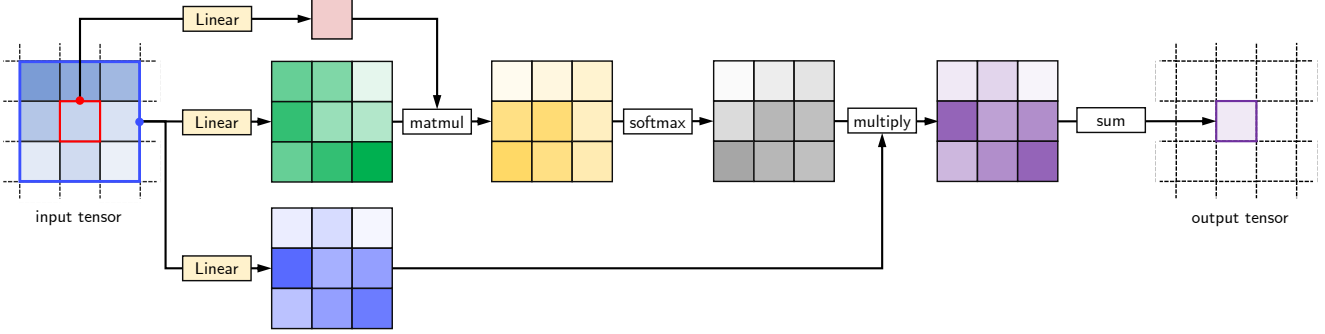
Figure 7. Self-Attention Convolution for a window size $k = 3$.

is that the smoothing kernel can deal with the influence of noise more effectively.
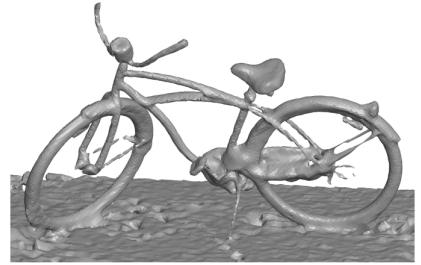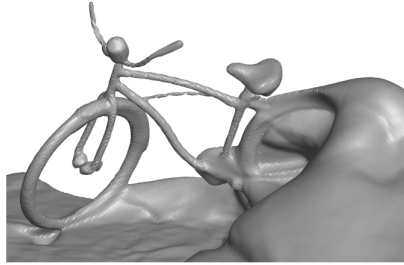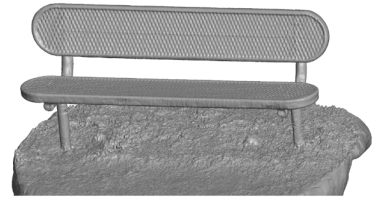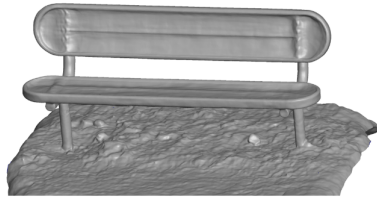
C.2. Self-Attention Convolution

We use the vanilla SAC mechanism from [32] and its detailed diagram is provided in Fig. 7. Larger window sizes represent lower frequency and smaller window sizes represent higher frequency. These window sizes are used as different kernels in the self-attention convolution and the corresponding convolved features are produced. Adding the original tri-plane features, we can get four triplane features with frequencies as our output feature maps.

D. Other Real-world Examples

We conduct an experiment on another real-world dataset namely CO3D [34]. This dataset provides data for real-world scenes. Many scenes are captured with portable devices and camera poses are extracted using COLMAP [35]. 3D reconstruction methods from datasets with this acquisition method are more general, but also introduce greater noise and challenges. We select some representative examples (bench, bicycle, and hydrant) and show the comparison with NeuS [41] on these examples in Fig. 8. Experimental results show that our method can reconstruct detailed features, such as the net structure of a bench, the rear wheel of a bicycle, and the rivets of a fire hydrant.

E. Additional Qualitative Comparisons

In this section, we provide more qualitative comparisons with NeuS [41] and HF-NeuS [42] for surfaces and images on the NeRF synthetic dataset (Fig. 9 and Fig. 10) and the DTU dataset (Fig. 11 and Fig. 12).

Reference Image **NeuS** **OURS**

Figure 8. Qualitative evaluation on CO3D dataset. First column: reference images. Second to the third column: NeuS and PET-NeuS.

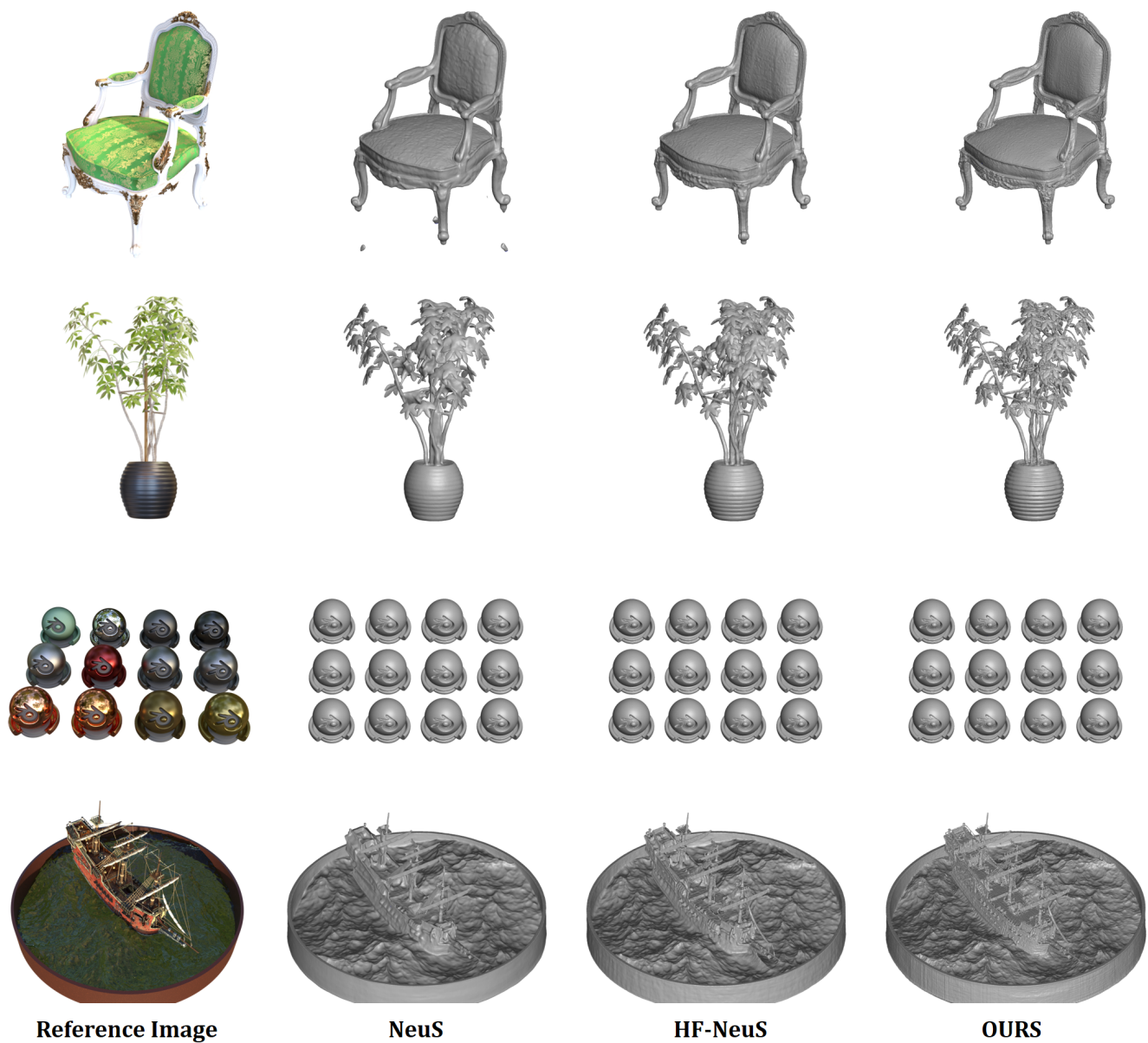| **Reference Image** | **NeuS** | **HF-NeuS** | **OURS** |

Figure 9. Qualitative evaluation on the NeRF synthetic dataset. First column: reference images. Second to the fourth column: NeuS, HF-NeuS, and PET-NeuS.

Reference Image   **NeuS**   **HF-NeuS**   **OURS**

Figure 10. Qualitative evaluation on NeRF synthetic dataset. First column: reference images. Second to the fourth column: the generated images from NeuS, HF-NeuS, and PET-NeuS.
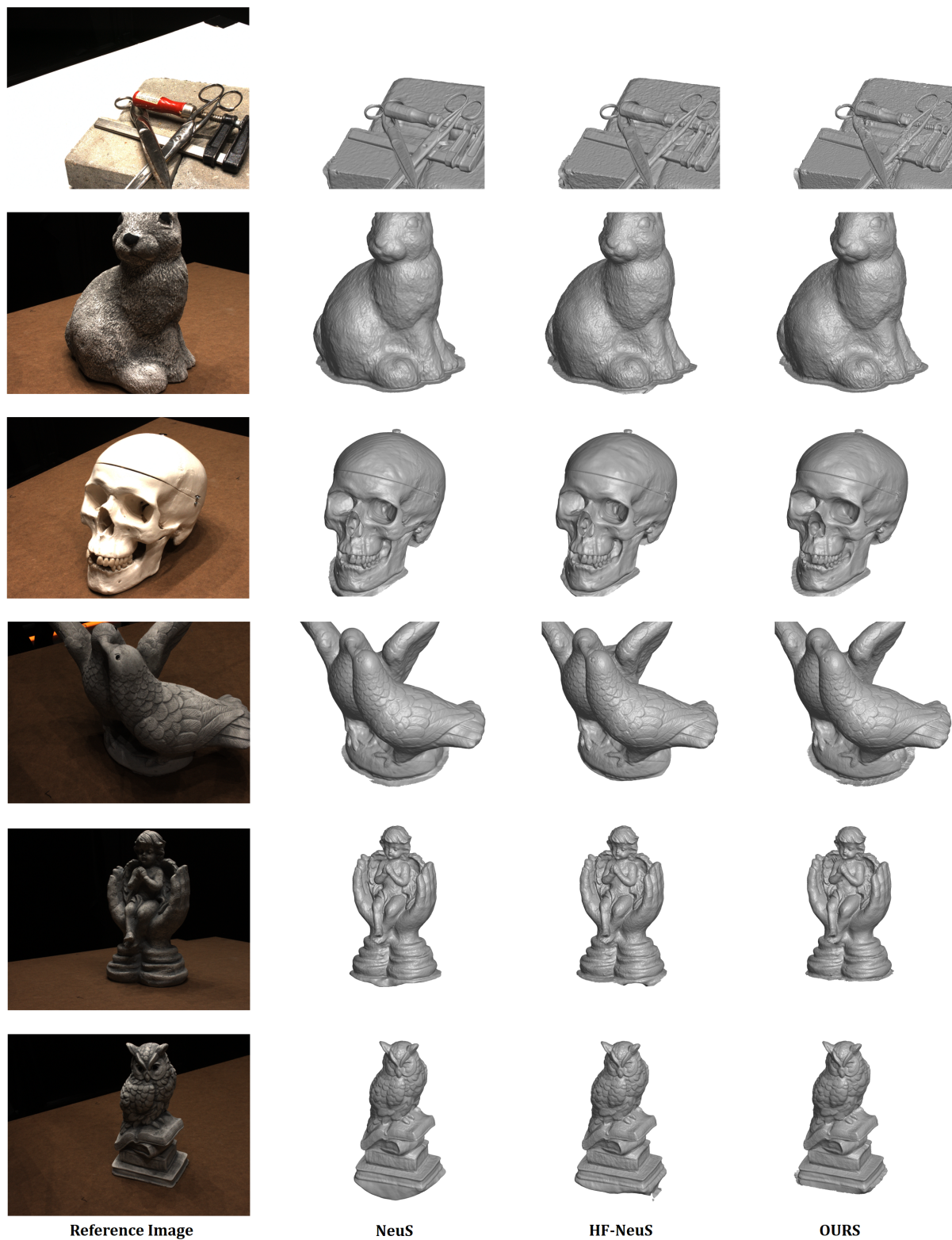
Figure 11. Qualitative evaluation on DTU dataset. First column: reference images. Second to the fourth column: NeuS, HF-NeuS, and PET-NeuS.

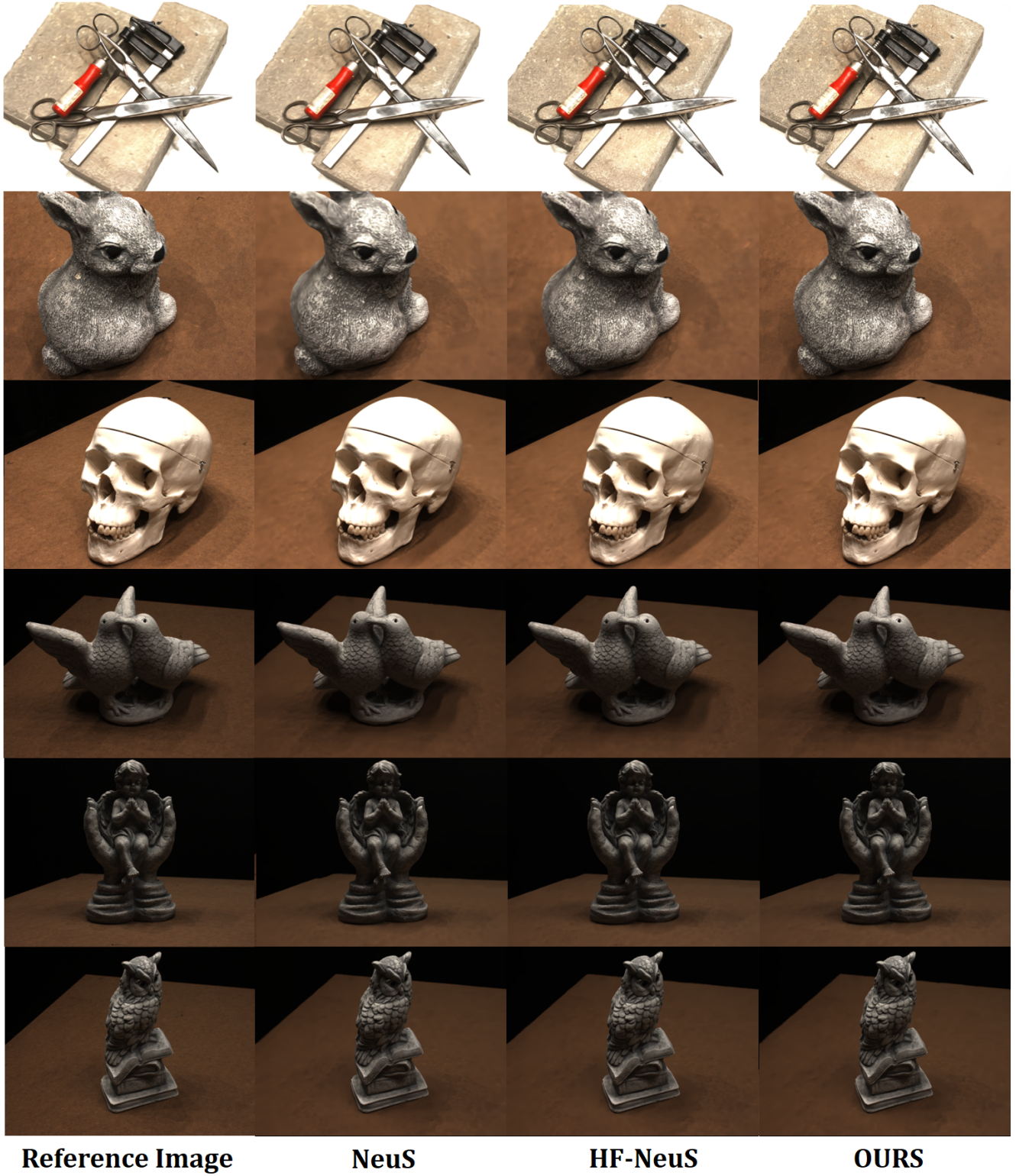**Reference Image**   **NeuS**   **HF-NeuS**   **OURS**

Figure 12. Qualitative evaluation on DTU dataset. First column: reference images. Second to the fourth column: the generated images from NeuS, HF-NeuS, and PET-NeuS.