Error-aware Sampling in Adaptive Shells for Neural Surface Reconstruction

Qi Wang[1] , **Yuchi Huo**[1,2,*] , **Qi Ye**[1] , **Rui Wang**[1] and **Hujun Bao**[1]
[1]State Key Lab of CAD&CG, Zhejiang University
[2]Zhejiang Lab
{wqnina1995, huo.yuchi.sc}@gmail.com, {qi.ye, ruiwang}@zju.edu.cn, bao@cad.zju.edu.cn,

Abstract

Neural implicit surfaces with signed distance functions (SDFs) achieve superior quality in 3D geometry reconstruction. However, training SDFs is time-consuming because it requires a great number of samples to calculate accurate weight distributions and a considerable amount of samples sampled from the distribution for integrating the rendering results. Some existing sampling strategies focus on this problem. During the training, they assume a spatially-consistent convergence speed of kernel size, thus still suffering from low convergence or errors. Instead, we introduce an error-aware sampling method based on thin intervals of valid weight distributions, dubbed adaptive shells, to reduce the number of samples while still maintaining the reconstruction accuracy. To this end, we first extend Laplace-based neural implicit surfaces with learned spatially-varying kernel sizes which indicates the range of valid weight distributions. Then, the adaptive shell for each ray is determined by an efficient double-clipping strategy with spatially-varying SDF values and kernel sizes, fitting larger kernel sizes to wider shells. Finally, we calculate the error-bounded cumulative distribution functions (CDFs) of shells to conduct efficient importance sampling, achieving low-variance rendering with fewer calculations. Extensive results in various scenes demonstrate the superiority of our sampling technique, including significantly reducing sample counts and training time, even improving the reconstruction quality. The code is available at https://github.com/erernan/ESampling.

1 Introduction

3D reconstruction from multi-view images is a hot research area. Recently, neural radiance field (NeRF) [Mildenhall *et al.*, 2020] and its subsequent works have demonstrated great potential for novel view synthesis. However, due to the lack of effective constraints on geometry, the reconstructed geometry of NeRFs generally suffers from discernible noise and artifacts. Neural implicit surfaces [Yariv *et al.*, 2021; Wang *et al.*, 2021] significantly improve the reconstruction results by constraining the scene to signed distance function (SDF) fields, thus achieving SOTA geometry reconstruction quality. However, the implicit SDF reconstruction requires hours to train multi-layer perceptron (MLP) networks via dense sampling algorithms, limiting its applications in practice. Some acceleration methods have been proposed to cope with this problem, which mainly focuses on two aspects: sophisticated spatial coding algorithms for reducing network parameters and accurate sampling for faster convergence speed. While the adaption of several effective spatial coding algorithms [Takikawa *et al.*, 2021; Barron *et al.*, 2021; Müller *et al.*, 2022; Fridovich-Keil *et al.*, 2022] to SDF has produced promising results, only **one** sampling algorithm is dedicated to training neural implicit surfaces.

Representatively, VolSDF [Yariv *et al.*, 2021] proposes an error-bounded sampling method dedicated to Laplace distribution-based neural implicit surfaces by mapping the SDF values to density values. They derive the maximum Riemann sum error of the weights along rays and iteratively increase the number of sampling points until the maximum error is small enough. It guarantees the accuracy of weight distribution at the cost of more than 500 sampling points for each ray to query the MLP, which significantly slows down the convergence speed. Meanwhile, a large number of importance samples also prolongs the training time.

Because of the similar volume rendering process, **two** advanced sampling methods, i.e., the occupancy grid (occupancy) and the proposal network (proposal), originally designed for NeRFs can also be applied to neural implicit surfaces. The occupancy methods discretize the continuous SDF field into a 3D grid. By calculating the validity of ray samples in the grid, invalid samples are weeded out before querying the MLP. Yet, a low grid update frequency causes inaccuracy and a high update frequency reduces efficiency. The proposal methods replace NeRF's [Mildenhall *et al.*, 2020] coarse network with a lightweight network to predict the weight distribution for reducing queries of the original MLP. However, this weight distribution is a rough approximation that leads to erroneous reconstructions. Furthermore, these two methods ignore the fact that the SDF fields converge from volume-like to surface-like, but the coarse occupancy and proposal methods designed for volume-like fields might fail in surface-like

*Corresponding author

fields, leading to reconstruction artifacts.

Different from these **three** methods, we consider the convergence process from volume to surface, proposing a novel error-aware adaptive shell. Equipped with the shell, we not only solve the above drawbacks, that is, to obtain an accurate weight distribution with less computation but also maintain the accuracy of the CDF for importance sampling for further acceleration. In particular: **1**) We extend Laplace-based neural implicit surfaces [Yariv *et al.*, 2021] with learned spatially-varying kernel sizes that represent the convergence of the surface. A larger kernel size indicates a wider weight distribution, viz., a wider range of possible surface locations. **2**) We calculate the density distribution of each ray utilizing spatially-varying kernel sizes and SDFs, then cut off ray segments corresponding to illegal densities. A similar weight clipping operation is applied to the remaining ray segments to determine the shell of each ray. **3**) We derive the upper bound of CDF's bias as a function of the shell size and the number of uniform samples. Therefore, according to the adaptive shell, we calculated the number of uniform sampling points required. By fitting the sparsely sampled density distribution, we obtain an accurate CDF. **4**) We adaptively adjust the number of importance sampling points for each ray based on the shell size to achieve low-variance rendering with less computation.

Through experiments of Section 4, we illustrate that our method can significantly accelerate the training of neural implicit fields. In the qualitative results, our reconstructed meshes acquire fewer artifacts compared to the proposal method and the occupancy method. The main contributions of our work are summarized as follows:

- We propose novel adaptive shells for Laplace-based neural implicit surfaces, which reduce the number of samples in the weight calculation stage and importance sampling stage together with the double-clipping strategy.
- We extend Laplace-based neural implicit surfaces with learned spatially-varying kernel sizes and analyze different training strategies for network convergence.
- We derive the upper bound of CDF's bias, which ensures the accuracy of our sampling method.

2 Related Work

In this section, we mainly summarize some recent neural implicit surfaces research and sampling methods related to neural implicit surfaces and NeRFs.

Neural implicit surfaces. VolSDF [Yariv *et al.*, 2021] and NeuS [Wang *et al.*, 2021] are two classic methods of neural implicit surfaces which both have a series of follow-up work. They are both designed under NeRF's volume rendering framework, but VolSDF applies Laplace distribution mapping SDF values to density values, while NeuS uses logistic distribution. Meanwhile, the density mapping method of NeuS is unbiased in planar scenes, compared with VolSDF. [Fu *et al.*, 2022; Yu *et al.*, 2022b] improve reconstruction by introducing geometric cues. [Ge *et al.*, 2023] focus on reconstructing objects with strong reflection. [Fan *et al.*, 2023] not only achieves the reconstruction of glossy objects but also estimates the illumination and material. Similarly, [Yariv *et al.*, 2023] also reconstructs the material with geometry and even achieves real-time rendering. [Jiang *et al.*, 2023; Azinović *et al.*, 2022] replace RGB inputs with RGB-D inputs which naturally results in higher reconstruction quality. To speed up training, [Rosu and Behnke, 2023] replaces the voxel hash encoding with a permutohedral lattice which optimizes faster. Recently, [Wang *et al.*, 2023a] deeply combined NeuS with Instant-NGP [Müller *et al.*, 2022] which also significantly reduces the training time of original NeuS. In the popular AIGC field, implicit SDFs can also generate 3D content [Xu *et al.*, 2023; Zheng *et al.*, 2022].

Sampling methods. In general, sampling methods for neural implicit surfaces can be classified into 4 categories: error-bounded (VolSDF) method, coarse-to-fine method, voxel-surface guided method, sampling network method, and occupancy grid method. Except for the error-bounded method, the rest are sampling algorithms derived from NeRFs. The coarse-to-fine method [Mildenhall *et al.*, 2020] guides sampling by training a coarse network. They sample uniformly on the coarse network and consider the density value of the sampling point as a PDF to guide sampling on the fine network. This sampling method is inefficient due to training an additional network. Occupancy grid methods [Li *et al.*, 2022; Müller *et al.*, 2022] discretize the scene into voxels, and query the grid to determine whether the sampling point contributes to the color of rays, thereby skipping invalid areas and achieving sampling acceleration. However, the accuracy of the query depends on the grid resolution and update frequency. A high-resolution grid takes up extra GPU memory. Also, occupancy grids require a large number of MLP queries when updating, and low update frequency greatly reduces query accuracy. Voxel-surface guided method [Sun *et al.*, 2022] is an efficient sampling algorithm that combines the occupancy grid with surface-guided sampling. However, their method requires pre-reconstructed point clouds, which limits the applicable scenarios of the algorithm. Sampling network methods [Lindell *et al.*, 2021; Piala and Clark, 2021; Barron *et al.*, 2022; Kurz *et al.*, 2022] train neural networks to directly sample or guide sampling alone rays with end-to-end or pre-training manners. For end-to-end training methods, e.g. proposal networks, gradients calculation and backpropagation consume additional time and the predicted weight distribution is not accurate. For pre-trained methods, network predictions for unknown scenes are unreliable. Recently, [Wang *et al.*, 2023b] introduces a sampling method that is also based on adaptive shells. However, their method speeds up the inference (rendering) stage, not the training stage. Their adaptive shells are explicitly extracted through the marching cube algorithm, while our adaptive shells are calculated on the fly. Although they also introduce a learnable kernel size, they have not revealed the specific training details, while we analyze the network training strategy.

3 Method

Given N calibrated multi-view images and corresponding camera intrinsic and extrinsic parameters, our method aims to reduce the number of samples to accelerate training

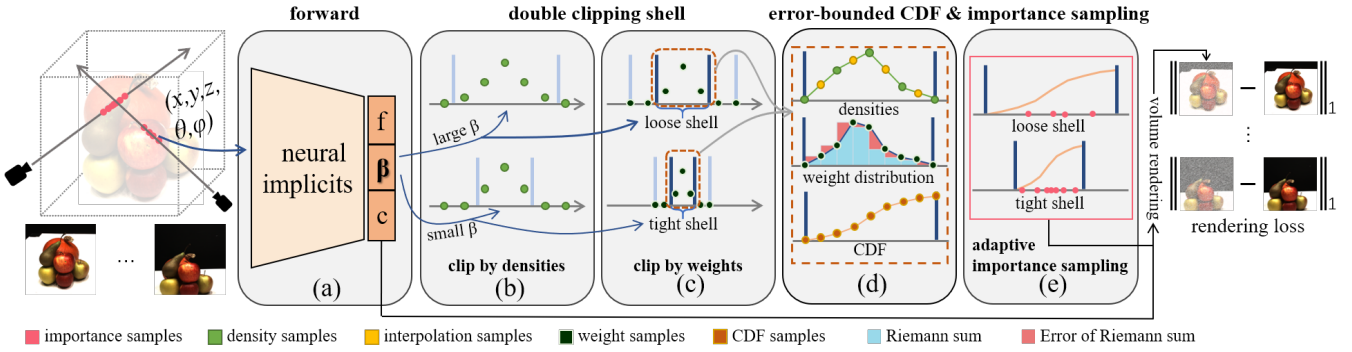


Figure 1: **Overview of our sampling method.** (a) We first extend neural implicits with a spatially-varying kernel size β. (b) Then we sample uniformly along the ray and calculate the densities of the sampling points utilizing βs and SDF values and clip ray segments with tiny densities to obtain the coares shell. (c) Next, we perform another uniform sampling within the coarse to calculate the weights. The adaptive shells are determined by clipping ray segments with tiny weights. (d) Subsequently, we conduct a coarse uniform sampling within adaptive shells and compute the samples' densities. Using function fitting, we generate more density samples. The weight samples are then calculated by all density samples. The error-bounded CDF is derived from weight samples. (e) Finally, we perform adaptive importance sampling to obtain the final importance samples according to CDF and shell size. These importance samples combined with predicted radiance are fed into the volume rendering equation to calculate the rendering loss.

while still maintaining reconstruction accuracy (see Fig. 1). Specifically, we extend Laplace-based neural implicit surfaces [Yariv *et al.*, 2021] with learned spatially-varying kernel sizes (Section 3.2). Then, a double-clipping strategy is applied to determine the shell for each ray (Section 3.3). Finally, we utilize shell sizes and function fitting to calculate the error-bounded CDFs within shells, then perform adaptive importance sampling for volume rendering (Section 3.4).

3.1 Preliminaries

Neural implicit surfaces share the same framework with NeRF [Mildenhall *et al.*, 2020], which represents a scene by a neural network (usually an MLP). For any ray parametrized as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ passing through the scene, the classic volume rendering equation takes density σ and color $\mathbf{c}$ of samples predicted by the neural network to provide a solution for ray color:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \tag{1}$$

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right), \tag{2}$$

where $C(\mathbf{r})$ denotes the predicted color of the ray, $\sigma(\mathbf{r}(t))$ denotes the volume density of point $\mathbf{r}(t)$, t_n and t_f are the near and far sampling distances, $T(t)$ denotes the accumulated transmittance along the ray from t_n to t. However, directly predicts the density of samples lacking geometric constraints. Therefore, neural implicit surfaces utilize the network to predict the SDF values of samples and map SDF values to density values through the mapping function. CDF of the Laplace distribution with zero means and β scale is one of the commonly used equations proposed by VolSDF [Yariv *et al.*, 2021]:

$$\sigma(\mathbf{r}(t)) = \alpha\Psi_\beta\left(-d_\Omega(\mathbf{r}(t))\right), \text{ where } \alpha = (1/\beta), \tag{3}$$

$$\Psi_\beta(s) = \begin{cases} \frac{1}{2}\exp\left(\frac{s}{\beta}\right) & \text{if } s \le 0 \\ 1 - \frac{1}{2}\exp\left(-\frac{s}{\beta}\right) & \text{if } s > 0 \end{cases}, \tag{4}$$

where d_Ω denotes the predicted SDF value of a sampling point. β is a global learnable parameter that approaches zero during training.

In practice, the continuous function integral in Eq. 1 is discretized by the Monte Carlo integration method. N points $\{\mathbf{p}_i = \mathbf{o} + t_i\mathbf{d} \mid i = 1, \ldots, N\}$ is sampled along the ray to approximate pixel color:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\sigma_i\delta_i\right)\right)\mathbf{c_i}, \tag{5}$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right), \tag{6}$$

where δ_i denotes the length of interval from $\mathbf{p}_{i-1}$ to $\mathbf{p}_i$, $\alpha_i = (1 - \exp(-\sigma_i\delta_i))$ is the alpha value of interval, $T_i\alpha_i$ is called the weight of a sampling point by convention, denoted by ω_i. It is obvious that the expected ray color is estimated by ω_i, and thus the optimal sampling distribution should be proportional to ω_i. Neural implicit surfaces consider PDF as a piecewise function of $\{\omega_i \mid i = 1, \ldots, N\}$.

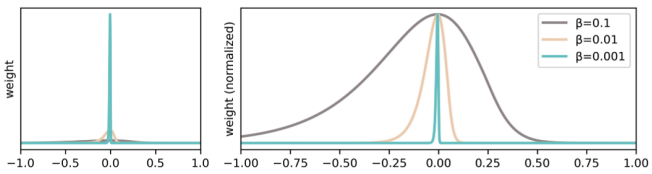
$$f_p(x) = \frac{\omega_i}{\sum_{i=1}^{N}\omega_i}, \qquad x \in [t_i, t_{i+1}). \tag{7}$$

The CDF obtained by integrating this PDF is used as the final importance sampling.

To optimize the network, the color loss $\mathcal{L}_{\text{color}}$ is defined as:

$$\mathcal{L}_{\text{color}} = \sum_{\mathbf{r}\in\mathcal{R}} \|C'(\mathbf{r}) - C(\mathbf{r})\|_1, \tag{8}$$

where $\mathcal{R}$ denotes a batch of training rays.



(a) weigth distribution (b) weigth distribution (normalized)

Figure 2: Weight distributions under different βs.

3.2 Learnable Spatially-Varying Kernel Size

As we mentioned above, VolSDF [Yariv *et al.*, 2021] trains a learnable global kernel size β (Eq. 3, Eq. 4) that controls the magnitude and the variation rate of density distribution. A smaller β results in a larger maximum value and a larger first-order derivative. According to Eq. 5 and Eq. 6, the range of weight distribution is inversely proportional to the derivative of density distribution. Thus, as β approaches zero during training, the surface converges to an infinitesimal interval along with the weight distribution (see Fig. 2(b)). However, due to the complexity of the scene, the optimal convergence speed of different surfaces should be different. Therefore, the same convergence speed brought by global β is inappropriate.

We modify the original MLP-based geometry network in neural implicit surfaces to additionally predict a β for each input point (see Fig. 1(a)):

$$N^{geo}\left(\Psi\left(x,y,z\right)\right)=\left(f,\beta,\mathbf{f}_{geo}\right), \tag{9}$$

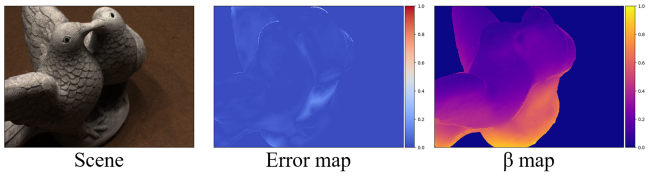
where $\Psi(x,y,z)$ is the encoding function of input point position, usually positional encoding [Mildenhall *et al.*, 2020] or hash encoding [Müller *et al.*, 2022]. f and β are the predicted SDF value and kernel size of the point respectively. These learnable betas adaptively imply variable convergence speeds at different locations based on scene characteristics (see Fig. 3). Notice that Eq. 9 is slightly different from the neural implicit network in Fig. 1, Eq. 9 only formulates the geometry network, excluding the radiance network. $\mathbf{f}_{geo}$ is a feature vector, which is input to the radiance network combined with the view direction and point position to obtain the predicted color $\mathbf{c}$. The modified geometry network can be trained without any additional inputs, and automatically control the convergence speed concerning input points.

However, naive training strategies will result in negative numbers in density distributions. Direct clamp beta to positive will cause a gradient vanishing problem. Through experiments, we choose a simple but effective function: $\exp(\beta - 3.7)$, which provides reasonable initial values and gradients of β. For other functions and initial values settings, please refer to Section 4 for more details.

3.3 Double Clipping Shell

We perform a double-clipping strategy to efficiently calculate a tight shell for each ray through two passes of sparse samplings, as shown in Figures (b) and (c) of Figure 1.

First, we sparsely sample a given ray over its near and far ends and evaluate the SDF values and βs of these sampling points by querying the MLP. Then we calculate a coarse density distribution of the ray based on these samples. We clip



Scene Error map β map

Figure 3: The spatially variable βs (*right*) indicates the convergence of different surfaces during training. Surfaces with larger β generally have higher reconstruction errors (*middle*). This reflects inherent properties of the scene, such as edges, dark regions, or highlight areas. Error map and β map are normalized to $[0,1]$ for better visualization.

the sampling point with a density smaller than the **valid density bound**, which is the maximum value of density distribution multiplied by a tiny value ϵ. According to Eq. 5, tiny densities lead to tiny weights. Thus clipping the ray segments where these samples are located means skipping meaningless areas. In particular, we first traverse all sampling points forward, find the first sampling point $\mathbf{p}_i$ whose density value is lower than **valid density bound**, and discard $\mathbf{p}_1$ to $\mathbf{p}_{i-2}$ (included). If density values of all sampling points are greater than **valid density bound**, no samples will be discarded. After that, we traverse the sampling points in reverse with a similar operation. Due to the additional overhead required to calculate weight distribution and the allowable error for the first clipping, we do not directly clip the weight distribution.

After the initial clipping, we can skip most of the intervals with zero-valued weights. However, the remaining segment of the ray is usually not small enough to closely surround sharp weight distributions accurately, especially when β is small. To address this issue, We perform another sparse sampling within the remaining ray segment and query their SDF and β values. We calculate the coarse weights using these SDF and β values and determine the **weight boundary** by calculating the maximum value of coarse weights multiplied by ϵ. Then, we apply approaches similar to the first clipping to locate the final ray segment, which is dubbed the **adaptive shell** of the ray.

3.4 Error-bounded CDF & Adaptive Sampling

In this section, we derive the error bound of the CDF mentioned in Section 3.1 to ensure the accuracy of importance sampling and thereby reduce the number of sampling points. For a set of N uniform samples $\mathcal{S}=\{t_i\}_{i=1}^{N}, t_n<t_1<t_2<\cdots<t_N<t_f$, we denote $\delta_i=t_i-t_{i-1}$, and ω_i is the weight corresponding to each sample. Due to uniform sampling, $\delta_1=\delta_2=\cdots=\delta_n=\delta_u$. Multiply the numerator and denominator of the Eq. 7 by δ_u, the PDF and CDF function is converted into (left) Riemann sum form:

$$f_p(x)=\frac{\omega_i\cdot\delta_u}{\sum_{i=1}^{N}\omega_i\cdot\delta_u}, \qquad x\in[t_i,t_{i+1}). \tag{10}$$

$$F_p(x)=\sum PDF(x) \tag{11}$$

$$=\frac{\sum_{j=1}^{i}\omega_j\cdot\delta_u}{\sum_{i=1}^{N}\omega_i\cdot\delta_u}, \qquad x\in[t_i,t_{i+1}). \tag{12}$$

Theorem 1. *The maximum error of (left) Riemann sum of ω, i.e. $\widehat{\mathbf{W}}$ satisfies*

$$\max|\mathbf{W}-\widehat{\mathbf{W}}| \leq (\omega_{max}-\omega_{min})\cdot\delta_u, \tag{13}$$

where $\mathbf{W}$ is the constant unbiased integral value of the weight distribution for a given ray. ω_{max} and ω_{max} denote the maximum and minimum value of weight distribution respectively, which are constants for a given ray.

The proof of Theorem 1 is provided in the supplementary. Now consider the error bound of CDF. Obviously, both numerator and denominator in Eq. 12 are Riemann sum. According to Theorem 1, the maximum error value of the numerator is bounded by $(\omega_{max}-\omega_{min})\cdot\delta_u$, and the minimum value of the denominator is bounded by $\mathbf{W}-(\omega_{max}-\omega_{min})\cdot\delta_u$. Thus, the error bound of CDF is:

$$\max|F_p(x)-\widehat{F_p}(x)| \leq \frac{(\omega_{max}-\omega_{min})\cdot\delta_u}{\mathbf{W}-(\omega_{max}-\omega_{min})\cdot\delta_u}, \tag{14}$$

where $\widehat{F_p}(x)$ denotes CDF function estimated with Eq. 12. Eq. 14 is a monotonically increasing function of δ_u. When δ_u is small enough, that is, the uniform sampling is dense enough, Eq. 14 converges to zero. In practice, ω_{min} and ω_{miax} are hard to get, so we approximate them by the maximum and minimum value among ω_i.

As the sampling range reduces, δ_u also decreases. Benefiting from the adaptive tight shell obtained in Section 3.3, the number of uniform sampling points N_u in the shell can be greatly reduced while still satisfying a small error bound. However, N_u is still large for efficient training. Therefore, we utilize function fitting to approximate density sampling distribution with fewer samples (see Fig. 1(d)). In practice, we uniform sample 16 points within the shell. Then we query MLP to get the SDF and β values of these samples to calculate the density distribution. After that, we obtain a dense density distribution by linearly interpolating the density values between each sample. Although linear interpolation is a naive method, it already provides a sufficiently accurate approximation with a tiny computational overhead. The dense weight distribution within the shell is then calculated by the dense interpolated density distribution. Finally, this weight distribution is used to compute the error-bounded CDF for importance sampling.

Utilizing double-clipping strategy, adaptive shells, and linear interpolation, we already reduced a large number of sampling points used to calculate weight distribution and CDF. Now we take the number of importance samples into account for further acceleration (see Fig. 1(e)). Recall that the Monte Carlo integral operator and variance are:

$$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}, g(x) = \frac{f(x)}{p(x)}, \tag{15}$$

$$Var[F_N] = \sigma^2_{F_N} = \frac{Var[g(x)]}{N}, \tag{16}$$

where F_N is the Monte Carlo integral operator, N is the number of importance samples, $f(x)$ is the integrand of x, $p(x)$ is the PDF of x, X_i is the samples and $\sigma^2_{F_N}$ is the variance of F_N. Thus, rewrite the Eq. 5 into Monte Carlo integral form, and we get the approximated variance of $\hat{C}(\mathbf{r})$:

$$\sigma^2_{\hat{C}(\mathbf{r})} \propto \frac{Var[\omega(x)]}{N_i}, \tag{17}$$

where N_i is the number of importance samples, $\omega(x)$ is the weight distribution within the shell. For a smaller β (smaller shell), the variance of weight is larger (see Fig. 2(a)). Therefore, we adaptive sample different numbers of importance samples based on shell size to restrict the rendering variance to a small value:

$$N_i = \lfloor\left(1-\frac{1}{1+\exp(-10\cdot(\max(l\cdot 2, 1.0)-0.5))}\right)\cdot 32\rfloor, \tag{18}$$

where l is the shell size. A smaller N_i reduces the sample count within the loose shell for acceleration, while a larger N_i ensures low variance volume rendering within the tight shell.

	VolSDF	NeuS	Proposal	Occupancy	Ours
weight calculating ↓	512	128	352	128 / 2.048	80
rendering ↓	64	128	48	128 / 16	32
rays /s ($\times 10^3$) ↑	8.8	13.5	21.7	13.1 / 36.7	26

Table 1: The number of samples per ray and training rays per second of different methods.

4 Experiments

	w/o hash encoding				**w/** hash encoding				
Chamfer Distance ↓	Scan	VolSDF	NeuS	Ours	VolSDF	NeuS	Proposal	Occupancy	Ours
	24	6.17	6.66	**4.45**	4.28	8.4	3.35	7.19	**3.24**
	55	3.82	/	**3.65**	2.06	7.19	2.10	6.70	**1.82**
	65	5.66	7.72	**4.00**	2.97	6.41	2.50	5.79	**2.11**
	106	7.81	/	**3.73**	5.20	7.83	3.98	7.62	**2.93**
	114	6.10	8.50	**3.54**	2.29	7.35	3.01	5.37	**1.54**
	122	7.66	/	**4.41**	5.33	/	4.02	6.57	**1.89**
	Avg	6.20	/	**3.96**	3.69	/	3.16	6.54	**2.26**

Table 2: **Chamfer distance on 6 scans of the DTU dataset.** By utilizing hash encoding or not, we divide all methods into two categories, separated by a vertical bar. Our method outperforms other methods in both categories. "/" means that the network has not converged at all and no valid value can be obtained. See the supplementary material for longer training results and more scenario results.

4.1 Experiment Settings

Dataset. We use 6 scenes from the DTU dataset [Jensen *et al.*, 2014]. These scenes contain different geometry structures and materials and are commonly used to evaluate neural implicit surfaces. We train the neural implicit surface model with our sampling method to represent the full scenes and extract meshes with the provided masks to remove noisy backgrounds. Note that all scenes are mostly observed from the front view, so reconstructions of the back of the model are not meaningful.

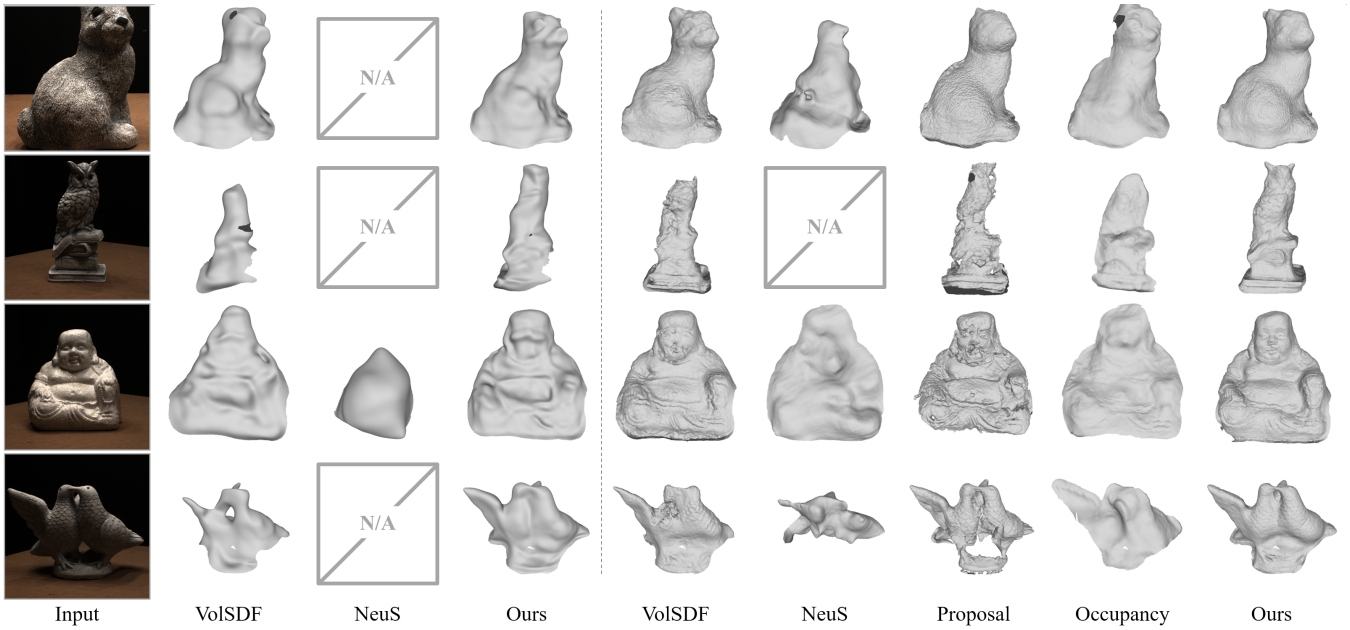


Figure 4: **Qualitative results on DTU dataset with 1.3 minutes training.** The left side of the dotted line shows results without hash encoding, while the right side shows results with hash encoding. "N/A" indicates no valid reconstruction result.

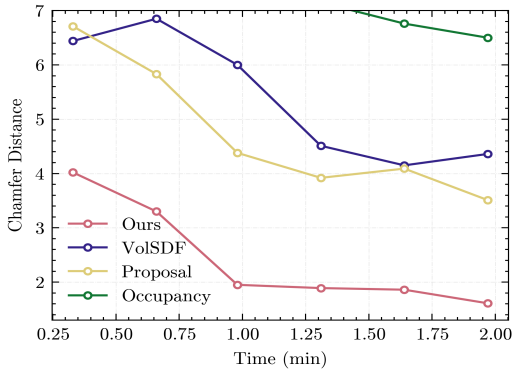


Figure 5: **Comparison of Chamfer distance between our method and proposal under the same training time.** Training time is distributed between **0.3 minutes** and **2 minutes**.

Implementation Details. We implement our method based on VolSDF [Yariv *et al.*, 2021] within the SDFStudio [Yu *et al.*, 2022a] framework. The structure of the radiance network is the same as that of VolSDF [Yariv *et al.*, 2021]. The geometry network is extended with a spatially-varying β as discussed above. The radiance network consists of 4 hidden layers with a hidden size of 256 while the geometry network consists of 8 hidden layers with the same hidden size. We also train the network with hash encoding [Müller *et al.*, 2022], in which the geometry and radiance networks are all reduced to 2 hidden layers with a hidden size of 256. We sample 64 points during the double-clipping stage (32 for a single pass), and 16 points during the linear interpolation stage. As for the final volume rendering stage, 32 samples combining uniform samples and importance samples are taken into account (See Table 1). The batch size of all experiments is 1, and the number of sampling rays in each batch is 1024. All experiments are deployed on a single Tesla-V100s GPU.

Baseline. We compared our sampling method with the following methods: (1) Error-bounded (VolSDF) [Yariv *et al.*, 2021], (2) Hierarchical (NeuS) [Wang *et al.*, 2021], (3) Proposal Network (proposal) [Barron *et al.*, 2022] (4) Occupancy Grid (occupancy) [Müller *et al.*, 2022]. Both VolSDF and NeuS are trained with and without hash encoding, while others are only trained with hash encoding because the non-hash networks converge too slowly to be meaningful for comparison. The proposal and occupancy sampling method are built based on NeuS. All methods are implemented by SDFStudio. We only modified the number of rendering samples of VolSDF to 64 for faster convergence. Note that SDFStudio uses hash coding to train proposal sampling, which greatly enhances its sampling accuracy, and the occupancy method calls NerfAcc [Li *et al.*, 2023] API, which is a cuda-accelerated version.

Evaluation metrics. We use Chamfer distance (CD) to measure the accuracy of geometry reconstruction quantitatively.

4.2 Comparisons

Sample count and training speed. The main time cost of sampling is querying the MLP. To assess the efficiency of sampling methods, we first counted the number of samples taken by different sampling methods on each ray. Fewer sampling points mean fewer MLP queries. As shown in Table 1, our method only samples 80 sampling points in the weight distribution calculation stage, which is smaller than VolSDF, NeuS, and proposal. The occupancy method is supposed to

use the NeuS sampling method for warmup in the first 2000 iterations to obtain the initial SDF field. Therefore, the sample count of our method is still less than occupancy at this stage. In the following iterations, the occupancy method updates the SDF value of the 128^3-resolution grid every 1000 iterations, averaging 2.048 samples per ray per iteration. It should be noted that although the proposal uses 352 sample points per ray, these sampling points query a network with a smaller number of parameters, rather than the MLP of the geometric network. So the actual calculation overhead is not much higher than our method. As for the volume rendering stage, our method achieves the lowest number of samples except for the occupancy method after 2000 iterations. Reducing the number of these sampling points further improves training speed because of evading additional radiance network queries and gradient calculations. Training rays per second in Table 1 shows that our method achieves the highest training efficiency compared to all other methods within 2000 iterations. Although the occupancy method trains faster after 2000 iterations, our method already achieves much better results at 2000_{th} iteration, which will be discussed later.

Quantitative results. We evaluate the Chamfer distance metric of our method with SOTA methods on 6 scans of the DTU [Jensen *et al.*, 2014] dataset. All scans are trained for **1.3 minutes** with different sampling methods. Table 2 shows that our method outperforms other methods in both settings —w/o hash encoding and w/ hash encoding. Similar to VolSDF, NeuS also introduces a learnable global kernel size for SDF-density mapping and weight calculation. However, NeuS fixes the kernel size during sampling, causing the sampled PDF to be far different from the real PDF. The large number of samples further slows down the convergence speed, leading to the worst reconstruction results. The occupancy method uses the NeuS sampling method before 2000 iterations, so the reconstruction result is also unacceptable. Although the VolSDF method calculates an accurate weight distribution, the excessive number of sampling points hinders training. The proposal method takes advantage of a reasonable simple count and a shallow sampling network with hash encoding which predicts a relatively accurate weight distribution. However, compared with our method, their CDF is still not accurate enough. Meanwhile, our adaptive sampling further reduces the number of importance sampling points, thus achieving better reconstruction results. In Fig. 5, we additionally compare our method with other methods with hash encoding on scan 122. As we can see, our method converges significantly faster than the remaining methods and achieves better results all the time. NeuS has no valid results within the two-minute training.

Qualitative results. As can be seen in Fig. 4, our method yields promising improvements in geometry reconstruction. NeuS and occupancy can only reconstruct the rough outline of the model with **1.3 minutes** of training. Although VolSDF performs some details, each model has incorrect reconstructions. The proposal causes collapsed artifacts and excessive detail in the reconstruction. Thanks to the efficient and accurate sampling algorithm, our method achieves impressive reconstruction results regardless of model structure or details.

4.3 Ablation Studies

In Fig. 6, we conduct an ablation study on the number of linear interpolation points that are used for calculating weight distributions and CDF. When the interpolation points are reduced, the reconstruction results suffer from loss of details and holes, which proves the effectiveness of our error-bounded CDF.

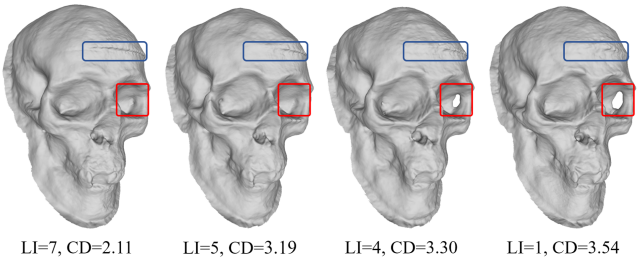


Figure 6: **Ablation study on the number of linear interpolation points.** LI represents the number of linear interpolation points between every two samples.

Fig. 7 shows the ablation study of importance sampling strategy. The total sample count of the three strategies is 32. The all-uniform samples method reconstructs the wrong geometry with basically no details. As for fixing the importance sample count to 8 and 16, the results show collapsed artifacts due to the high variance of the volume rendering.

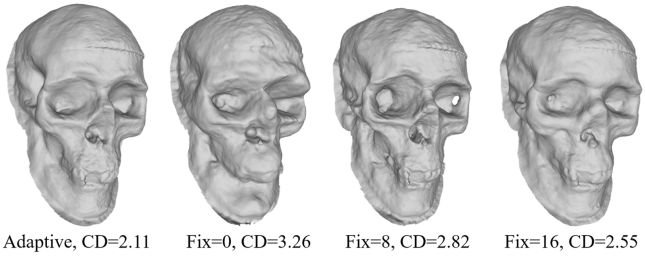


Figure 7: **Ablation study of importance sampling strategy.** From left to right, they are our adaptive sampling method, full uniform samples method, and method fixed at 8 and 16 importance samples.

Please refer to the supplementary material for additional ablation studies and results.

5 Conclusion

In this paper, we introduce an efficient error-aware sampling method for neural implicit surfaces based on spatially-varying kernel size. First, we extend the Laplace-based neural implicit surfaces with a learnable spatially-varying kernel size. Secondly, based on the kernel size and efficient double-clipping strategy, we determine the adaptive shell for each ray. Then, we derive the error bound of the sampling CDF within the shell and perform function fitting to reduce CDF bias. Finally, we conduct efficient importance sampling according to the variance of volume rendering. The experiments demonstrate the superiority of our sampling technique, including significantly reducing sample count and improving reconstruction quality within the same training time.

Acknowledgments

This work was supported by National Key R&D Program of China (No. 2024YDLN0011, No. 2023YFF0905102), NSFC (No. 62441205, No.U22B2034), Key R&D Program of Zhejiang Province (No. 2023C01039).

References

[Azinović *et al.*, 2022] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6290–6301, 2022.

[Barron *et al.*, 2021] 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, 2021.

[Barron *et al.*, 2022] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[Fan *et al.*, 2023] Yue Fan, Ivan Skorokhodov, Oleg Voynov, Savva Ignatyev, Evgeny Burnaev, Peter Wonka, and Yiqun Wang. Factored-neus: Reconstructing surfaces, illumination, and materials of possibly glossy objects. *arXiv preprint arXiv:2305.17929*, 2023.

[Fridovich-Keil *et al.*, 2022] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[Fu *et al.*, 2022] Qiancheng Fu, Qingshan Xu, Yew Soon Ong, and Wenbing Tao. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 35:3403–3416, 2022.

[Ge *et al.*, 2023] Wenhang Ge, Tao Hu, Haoyu Zhao, Shu Liu, and Ying-Cong Chen. Ref-neus: Ambiguity-reduced neural implicit surface learning for multi-view reconstruction with reflection. *arXiv preprint arXiv:2303.10840*, 2023.

[Jensen *et al.*, 2014] 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.

[Jiang *et al.*, 2023] Hanqi Jiang, Cheng Zeng, Runnan Chen, Shuai Liang, Yinhe Han, Yichao Gao, and Conglin Wang. Depth-neus: Neural implicit surfaces learning for multi-view reconstruction based on depth information optimization. *arXiv preprint arXiv:2303.17088*, 2023.

[Kurz *et al.*, 2022] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In *European Conference on Computer Vision*, pages 254–270. Springer, 2022.

[Li *et al.*, 2022] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf acceleration toolbox. *arXiv preprint arXiv:2210.04847*, 2022.

[Li *et al.*, 2023] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. *arXiv preprint arXiv:2305.04966*, 2023.

[Lindell *et al.*, 2021] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565, 2021.

[Mildenhall *et al.*, 2020] 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 *ECCV*, 2020.

[Müller *et al.*, 2022] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[Piala and Clark, 2021] Martin Piala and Ronald Clark. Terminerf: Ray termination prediction for efficient neural rendering. In *2021 International Conference on 3D Vision (3DV)*, pages 1106–1114. IEEE, 2021.

[Rosu and Behnke, 2023] Radu Alexandru Rosu and Sven Behnke. Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8466–8475, 2023.

[Sun *et al.*, 2022] Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural 3d reconstruction in the wild. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[Takikawa *et al.*, 2021] 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. 2021.

[Wang *et al.*, 2021] 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. *NeurIPS*, 2021.

[Wang *et al.*, 2023a] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[Wang *et al.*, 2023b] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic.

Adaptive shells for efficient neural radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(6):1–15, 2023.

[Xu *et al.*, 2023] Jiale Xu, Xintao Wang, Weihao Cheng, Yan-Pei Cao, Ying Shan, Xiaohu Qie, and Shenghua Gao. Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20908–20918, June 2023.

[Yariv *et al.*, 2021] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[Yariv *et al.*, 2023] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv*, 2023.

[Yu *et al.*, 2022a] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022.

[Yu *et al.*, 2022b] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in neural information processing systems*, 35:25018–25032, 2022.

[Zheng *et al.*, 2022] Xin-Yang Zheng, Yang Liu, Peng-Shuai Wang, and Xin Tong. Sdf-stylegan: Implicit sdf-based stylegan for 3d shape generation. In *Comput. Graph. Forum (SGP)*, 2022.