2S-UDF: A Novel Two-stage UDF Learning Method for Robust Non-watertight Model Reconstruction from Multi-view Images

Junkai Deng[1,2] **Fei Hou**[1,2*] **Xuhui Chen**[1,2] **Wencheng Wang**[1,2] **Ying He**[3]
[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]School of Computer Science and Engineering, Nanyang Technological University
{dengjk, houfei, chenxh, whn}@ios.ac.cn yhe@ntu.edu.sg

Abstract

Recently, building on the foundation of neural radiance field, various techniques have emerged to learn unsigned distance fields (UDF) to reconstruct 3D non-watertight models from multi-view images. Yet, a central challenge in UDF-based volume rendering is formulating a proper way to convert unsigned distance values into volume density, ensuring that the resulting weight function remains unbiased and sensitive to occlusions. Falling short on these requirements often results in incorrect topology or large reconstruction errors in resulting models. This paper addresses this challenge by presenting a novel two-stage algorithm, 2S-UDF, for learning a high-quality UDF from multi-view images. Initially, the method applies an easily trainable density function that, while slightly biased and transparent, aids in coarse reconstruction. The subsequent stage then refines the geometry and appearance of the object to achieve a high-quality reconstruction by directly adjusting the weight function used in volume rendering to ensure that it is unbiased and occlusion-aware. Decoupling density and weight in two stages makes our training stable and robust, distinguishing our technique from existing UDF learning approaches. Evaluations on the DeepFashion3D, DTU, and BlendedMVS datasets validate the robustness and effectiveness of our proposed approach. In both quantitative metrics and visual quality, the results indicate our superior performance over other UDF learning techniques in reconstructing 3D non-watertight models from multi-view images. Our code is available at https://bitbucket.org/jkdeng/2sudf/.

1. Introduction

As the success of neural radiance field (NeRF) [29], numerous volume rendering based 3D modeling methods are proposed to learn signed distance fields (SDF) for 3D model reconstruction from multi-view images [7, 34, 36, 40]. These approaches map signed distance value to a density function, thereby enabling the use of volume rendering to learn an implicit SDF representation. To calculate pixel colors, they compute the weighted sum of radiances along each light ray. Achieving an accurate surface depiction requires the density function to meet three essential criteria. Firstly, the weights, which are derived from the density function, must reach their maximum value when the distance is zero, ensuring unbiasedness. Secondly, as a ray traverses through the surface, the accumulated density should tend towards infinity, rendering the surface opaque — a property referred to as occlusion-awareness. Finally, the density function should be bounded to prevent numerical issues. The popular SDF approaches, such as NeuS [34] and VolSDF [40], adopt an S-shaped density function that meets all these requirements.

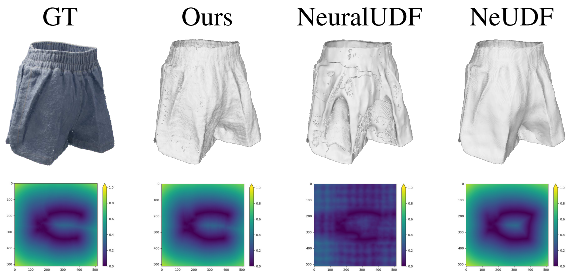

Figure 1. We learn a UDF from multiview images for non-watertight model reconstruction. As illustrated in the cross sections of learned UDFs, our learned UDF approximates to the ground truth. In contrast, the learned UDF of NeuralUDF [25] is choppy leading to significant artifacts, e.g., unexpected pit. The learned UDF of NeUDF [23] is almost closed struggling to generate open surface.

While SDF-based methods excel at reconstructing watertight models, they have limitations in representing open models. This is due to the intrinsic nature of SDF, which

[*]Corresponding author

differentiates between the interior and exterior of a model, thus failing to accommodate open boundaries. Recent advances have attempted to mitigate this constraint by employing unsigned distance fields (UDF) [23, 25, 27]. Unlike signed distance fields, UDFs have non-negative distance values, making them suitable for representing non-watertight models. However, learning a UDF from multi-view images is a challenging task since the gradients of the UDF are unstable due to directional changes near the zero level-set, making it difficult to train the neural network. Another major challenge lies in formulating a UDF-induced density function that can simultaneously meet the above-mentioned three requirements. Unlike SDFs, UDFs cannot distinguish between the front and back of a surface based on distance values, thus, directly using an S-shaped density function is off the table. Opting for a bell-shaped density function brings its own issues. It is impossible for these integrations to approach infinity, so as to be occlusion-aware, unless the density becomes boundless at zero distance values. These conflicting requirements make UDF learning a non-trivial task, forcing existing methods to sacrifice at least one of these conditions. As shown in Figure 1, the existing methods NeuralUDF [25] and NeUDF [23] result in either choppy or nearly closed UDFs.

As designing a UDF-induced density function that simultaneously fulfills the three aforementioned conditions remains an unresolved challenge, we propose a novel approach that learns a UDF from multi-view images in two separate stages. In the first stage, we apply an easily trainable but slightly biased and transparent density function for coarse reconstruction. Such a UDF, although being approximate, provides an important clue so that we can determine where to truncate the light rays. This accounts for the occlusion effect, where points behind the surface are not visible and should not contribute to the output color. With *truncated* light rays, we are able to derive the weights from UDF directly bypassing the density function, to further refine the geometry and appearance in the second stage. Our two-stage learning method, called 2S-UDF, leads to an unbiased and occlusion-aware weight function. Furthermore, by sidestepping density function learning in Stage 2, we effectively bypass the challenges associated with ensuring its boundedness. This strategy enhances the numerical stability of our method. Evaluations on benchmark datasets DeepFashion3D [43] and DTU [19] show that 2S-UDF outperforms existing UDF learning methods in terms of both reconstruction accuracy and visual quality. Additionally, we observe that the training stability of 2S-UDF is notably superior compared to other UDF learning neural networks.

2. Related Work

3D Reconstruction from Multi-View Images. Surface reconstruction from multi-view images has been a subject of study for several decades, and can generally be classified into two categories: voxel-based and point-based methods. Voxel-based methods [3, 8, 20, 21, 33] divide the 3D space into voxels and determine which ones belong to the object. These methods can be computationally expensive and may not be suitable for reconstructing complex surfaces. Point-based methods [13, 31, 38] use structure-from-motion [16] to calibrate the images and generate a dense point cloud using multi-view stereo [12]. Finally, surface reconstruction methods (e.g., [2, 17, 22]) are used to generate a mesh. Since multi-view stereo requires dense correspondences to generate a dense point cloud, which are often difficult to compute, its results often contain various types of artifacts, such as noise, holes, and incomplete structures.

Neural Volume Rendering. Neural network-based 3D surface reconstruction has received attention in recent years with the emergence of neural rendering [29]. Several methods have been proposed for volume rendering and surface reconstruction using neural networks. VolSDF [40] uses the cumulative distribution function of Laplacian distribution to evaluate the density function from SDF for volume rendering and surface reconstruction. NeuS [34] adopts an unbiased density function to the first-order approximation of SDFs for more accurate reconstruction. SparseNeuS [24] extends NeuS to use fewer images for reconstruction. HF-NeuS [36] improves NeuS by proposing a simplified and unbiased density function and using hierarchical multi-layer perceptrons (MLPs) for detail reconstruction. Geo-NeuS [10] incorporates structure-from-motion to add more constraints. NeuralWarp [7] improves the accuracy by optimizing consistency between warped views of different images. PET-NeuS [37] further improves the accuracy by introducing tri-planes into the SDF prediction module, incorporating with MLP. All these methods learn SDFs, which can only reconstruct watertight models. Recently, Long *et al.* proposed NeuralUDF [25] for learning UDF for reconstructing open models. It adapts the S-shaped density function for learning SDF to UDFs by introducing an indicator function. However, the indicator function is complicated to learn, and also introduces biases. Liu *et al.* proposed NeUDF [23] adopting a bell-shaped density. However, to make it occlusion-aware, the density has to be unbounded resulting in an improper integral, which reduces accuracy. Meng *et al.* proposed NeAT [27] to learn SDF with validity so as to reconstruct open models from SDF. However, it needs foreground masks for data.

3D Reconstruction from Point Clouds. There has been recent interest in surface representation using signed distance fields (SDFs) and occupation fields. Several methods have been proposed for learning SDFs [4, 26, 30, 32, 35], while occupation fields have been used in methods such as [5, 28]. However, both SDFs and occupation fields can only represent watertight models. To represent non-watertight models, some methods are proposed to learn UDF from 3D point clouds [6, 41, 42]. Our proposed method also uses UDF for non-watertight models representation, but we learn it directly from multi-view images, which is a challenging problem.

3. Method

At the foundation of UDF-based learning approaches is the task of crafting a density function that converts unsigned distance values into volume density, ensuring that the resulting weight function is unbiased and responsive to occlusions. None of the existing UDF learning methods [23, 25] can simultaneously meet the three critical requirements, i.e., ensuring the density function is bounded, and that the weight function remains both unbiased and occlusion aware.

We tackle these challenges by decoupling the density function and weight function across two stages. In the initial stage (Section 3.1), we utilize an easy-to-train, bell-shaped density function (which is inherently bounded) to learn a coarse UDF. While the resulting weight function is not theoretically unbiased or occlusion-aware, we can make it practically usable by choosing a proper parameter. Moving into the second stage (Section 3.2), we sidestep the density function entirely, focusing instead on refining the UDF by directly adjusting the weight function within the neural volume rendering framework. Specifically, we truncate light rays after they hit the front side of the object and obtain a weight function that is both unbiased and sensitive to occlusions, without the overhang of density function boundedness concerns. Finally, Section 3.3 presents the training details.

3.1. Stage 1: Coarse UDF Learning via a Simple Density Function

We consider the scenario of a single planar plane $\mathcal{M}$ and a single ray-plane intersection. Inspired by HF-NeuS [36], we propose an easy-to-learn density function σ_1 that maps unsigned distance f to density

$$\sigma_1(f(t)) = \frac{ce^{-sf(t)}}{1+e^{-sf(t)}},\ s>0,\ c>0, \tag{1}$$

where $c > 0$ is a fixed, user-specified parameter and $s > 0$ is a learnable parameter controlling the width of the bell-shaped curve. Straightforward calculation shows that the weight function $w_1(f(t)) = e^{-\int_0^t \sigma_1(f(u))\mathrm{d}u}\sigma_1(f(t))$ is monotonically decreasing behind the plane $\mathcal{M}$ and the maximum value occurs at a point t^* in front of $\mathcal{M}$ with an unsigned distance value of $f(t^*) = \frac{1}{s}\ln\frac{c}{|\cos(\theta)|}, (c > |\cos(\theta)|)$ or $f(t^*) = 0, (0 < c \leq |\cos(\theta)|)$, where θ is the incident angle between the light ray and the surface normal. This means that the weight function w_1 is not unbiased. Furthermore, the line integral $\int_0^t \sigma_1(f(u))\mathrm{d}u$ does not approach infinity when a light ray passes through the front-most layer of the surface, indicating w_1 is only partially occlusion-aware.

While the density function σ_1 is not perfect in theory, by selecting an appropriate c, we can practically minimize bias and enhance opacity. Clearly, a smaller c value decreases $f(t^*)$, thereby reducing bias. To gauge the effect of c on opacity, we now consider the most extreme scenario where the incident light ray is perpendicular to the planar surface $\mathcal{M}$, and assume that the intersection point is located at $t = 1$. In such a situation, the unsigned distance function is $f(t) = 1 - t$ for points in front of $\mathcal{M}$. Since σ_1 is symmetrical on either side of $\mathcal{M}$, the surface transparency is the square of the transparency of the front side. The theoretic transparency is,

$$\begin{aligned}\left(e^{-\int_0^1 \sigma_1(f(t))\mathrm{d}t}\right)^2 &= \left[\exp\left(-\int_0^1 \frac{cse^{-s(1-t)}}{1+e^{-s(1-t)}}\mathrm{d}t\right)\right]^2 \\ &= \left(\frac{1+e^{-s}}{2}\right)^{2c}.\end{aligned}$$

Therefore, we should choose a relatively large c to reduce transparency. In our implementation, we set the constant $c = 5$ based on the typical value of the learned parameter s which usually ranges between 1000 and 2000. Calculations of bias and translucency show that this setting offers a good balance between occlusion-awareness and unbiasedness in the first stage training. Please refer to the supplementary material for a detailed analysis.

3.2. Stage 2: UDF Refinement through Weight Adjustment

In this stage, we refine the UDF learned in Stage 1 to improve the quality of geometry and appearance. Unlike Stage 1 and all other UDF-learning methods, inspired by [1], we truncate light rays based on the approximated UDF learned in Stage 1 and learn the weight function $w(t)$ directly instead of the density function $\sigma(t)$ to refine the UDF.

Ideally, for a single ray-plane intersection, we want a bell-shaped function $w(t)$ that attains its maximum at the points with zero distance values, and satisfies partition of unity. Therefore, we adopt the derivative of the sigmoid function as the weight function [1], defined as

$$w_2(f(t)) = \frac{se^{-sf(t)}}{(1+e^{-sf(t)})^2}\cdot|\cos(\theta)|. \tag{2}$$

with θ being the incident angle between the light ray and the surface normal.

Intuitively speaking, learning such a weight function w_2 in Stage 2 of our UDF method is similar to learning an S-shaped density function in SDF-based approaches, such as [36]. As a result, the learning process in Stage 2 is as stable as those SDF approaches. Furthermore, it can totally avoid using the visibility indicator function, which is necessary in NeuralUDF [25].

Calculation shows that the weight w_2 attains its maximum at zero distance values, therefore it is unbiased. However, if we naively predict the weight function directly, it will not be occlusion-aware, so we introduce the ray truncation. To make w_2 occlusion-aware, we can truncate the light rays after they pass through the frontmost layer of the surface, thereby preventing rendering the interior of the object. Note that we do not expect the truncation to be exactly on the frontmost layer of the surface. In fact, as long as it occurs between the frontmost layer and the second layer, we consider the truncation valid. This means that the approximate UDF learned in the first stage, which can capture the main topological features (such as boundaries) and provide a fairly good representation of the target object, is sufficient for us to determine where to cut off the light rays.

In our implementation, we adopt a simple strategy to determine the truncation point for each light ray. Specifically, the truncation point of ray **r** is the first sample point along **r** such that

- The unsigned distance value at the point is a local maxima. To avoid distance vibration interference, it should be the maximum in a window centered at the point. And
- The accumulated weight up to this point is greater than δ_{thres}.

The accumulated weight threshold δ_{thres} is intuitively set to 0.5. This choice is based on the assumption that if the Stage 1 training is performed well enough, the accumulated weights at each sample point along the ray would be either 0 (for not reaching a surface) or 1 (for having intersected with a surface). Hence, we intuitively select 0.5 for δ_{thres} because it is the midpoint between 0 and 1. With the cutoff mechanism, only the first ray-surface intersection contributes to the color of the ray, effectively achieving occlusion-awareness. Given these properties, we conclude that,

Theorem 1 *The weight w_2 with light cutting off is unbiased and occlusion-aware.*

Figure 2 is an intuitive illustration of our Stage 2 weight learning and truncation strategy. The UDF maxima point A in front of the intersection surface would not affect the cutting point selection as the accumulated weight is below δ_{thres} (0.5). The local maxima B due to UDF oscillation also would not affect it since it's not the maximum in a large enough neighborhood. The light is cut at maxima point C, and thus the weight of point D is zero without contributions to the rendering. As illustrated in Figure 2, the cutting process is robust against UDF oscillation, open boundaries, and local maxima in front of the intersection surface.

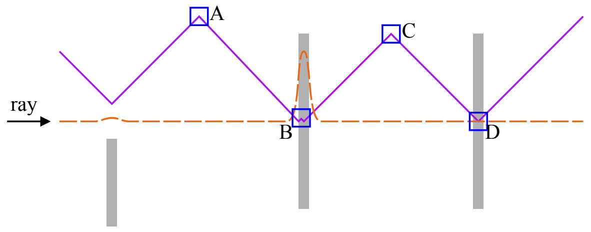

Figure 2. An intuitive illustration of our ray cutting algorithm, best viewed in color and magnified. A ray shoots from left to right, approaching the boundary of the first surface, and going through another two surfaces (gray boxes). The violet solid line represents the UDF values along the ray; the orange dashed line represents the corresponding color weight.

3.3. Training

Differentiable UDFs. NeuS uses an MLP network to learn the signed distance function f, which is a differentiable function. In contrast, UDF is not differentiable at the zero level set, making the network difficult to learn the values and gradients of the UDF close to the zero level set.

Another crucial requirement is to ensure non-negative values for the computed distances, which seems like a trivial task as one may simply apply absolute value or normalization such as ReLU [11] to the MLP output. However, applying the absolute value to the distance is not viable due to its non-differentiability at zero. Similarly, normalizing the output value using ReLU is not feasible as it is also non-differentiable at zero and its gradient vanishes for negative inputs. This can be particularly problematic for learning UDFs, since when the MLP returns a negative distance value, the ReLU gradient vanishes, hindering the update of the distance to a positive value in the subsequent iterations.

We add a softplus [9] function after the output layer of the MLP [23]. The softplus function is a smooth and differentiable approximation of the ReLU function, which is defined as softplus$(x) = \frac{1}{\beta}\ln(1+e^{\beta x})$. Softplus has the same shape as ReLU, but it is continuous and differentiable at every point and its gradients do not vanish anywhere. Using the softplus function allows us to ensure that the output of the MLP is non-negative and differentiable, making it suitable for learning the UDF. Similar to NeUDF [23], we set $\beta = 100$ in our experiments.

Loss functions. Following NeuralUDF [25], we adopt an iso-surface regularizer to penalize the UDF values of the non-surface points from being zero, therefore encouraging smooth and clean UDFs. The regularization loss is defined as [25]

$$\mathcal{L}_{reg} = \frac{1}{MN}\sum_{i,k}\exp\left(-\tau \cdot f(t_{i,k})\right),$$

where τ is a constant scalar that scales the learned UDF values, M is the total number of sampled rays per training iteration, and N is the number of sampled points on a single ray. τ is set to 5.0 in the first stage and 50.0 in the second stage.

The value of s, which is learnable in our method, significantly affects the quality of the reconstruction. When s is small, it introduces a larger bias and leads to a more blurred output. We observe that s typically converges to a relatively large value between 1000 and 2000, leading to visually pleasing results. However, in rare cases when s stops increasing during training, we apply a penalty to force it to increase. The penalty is defined as follows

$$\mathcal{L}_s = \frac{1}{M}\sum_{i,k}\frac{1}{s_{i,k}},$$

where M is the number of rays during a training epoch. This term $\mathcal{L}_s$ aggregates the reciprocals of all s values used for the point $t_{i,k}$ on ray r_i. Intuitively speaking, it encourages a larger s during the early stage of training. In our implementation, we make this term optional since s generally increases with a decreasing rate during training, and the penalty term is only necessary in rare cases when s stops at a relatively low value.

As in other SDF- and UDF-based methods [25, 34, 36], we adopt color loss and Eikonal loss in our approach. Specifically, the color loss $\mathcal{L}_{color}$ is the L_1 loss between the predicted color and the ground truth color of a single pixel as used in [34]. The Eikonal loss $\mathcal{L}_{eik}$ is used to regularize the learned distance field to have a unit gradient [14]. Users may also choose to adopt object masks for supervision as introduced in other SDF- and UDF-based methods [25, 34]. Putting it all together, we define the combined loss function as a weighted sum,

$$\mathcal{L} = \mathcal{L}_{color} + \lambda_1\mathcal{L}_{eik} + \lambda_2\mathcal{L}_{reg} + \lambda_3\mathcal{L}_s\,(+\lambda_m\mathcal{L}_{mask}),$$

where λ_1, λ_2, λ_3 and the optional λ_m are hyperparameters that control the weight of each loss term.

4. Experiments

Datasets. To evaluate our method, we use three datasets: DeepFashion3D [43], DTU [19] and BlendedMVS [39]. The DeepFashion3D dataset consists of clothing models, which are open models with boundaries. As only 3D points are available, we render 72 images of resolution 1024×1024 with a white background from different viewpoints for each model. In addition to DeepFashion3D images rendered by us most of which are texture-less, we also take the image data from NeuralUDF [25] most of which are texture-rich into our experiments. We call them DF3D#Ours and DF3D#NeuralUDF, respectively. The DTU dataset consists of models captured in a studio, all of which are watertight. We use this dataset to validate that our method also works well for watertight models. These datasets have been widely used in previous works such as [34, 36, 40]. In our experiments, open models such as in DeepFashion3D are trained without mask supervision; DTU is trained with mask supervision.

Baselines. To validate the effectiveness of our method, we compare it with state-of-the-art UDF learning methods: NeuralUDF [25], NeUDF [23] and NeAT [27]; and SDF learning methods: VolSDF [40] and NeuS [34].

4.1. Comparisons on Open Models

Method	#1	#2	#3	#4	#5	#6	#7	#8	#9	Mean
NeuS	6.69	13.50	10.32	15.01	8.99	12.92	12.94	9.93	9.49	11.09
VolSDF	6.36	9.44	11.87	16.03	10.78	14.91	15.06	11.34	8.96	11.64
NeAT	10.54	13.89	7.30	13.12	13.18	12.44	8.22	10.30	11.33	11.15
NeuralUDF	6.07	11.58	7.68	10.96	11.16	9.76	6.98	6.13	6.41	8.53
NeUDF	**4.39**	8.29	4.94	19.56	7.52	8.18	3.81	3.81	5.76	7.36
Ours	4.55	**5.77**	**4.27**	**7.43**	**6.59**	**4.77**	**2.88**	**3.21**	**5.73**	**5.02**

Method	LS-C0	SS-D0	LS-D0	NS-D1	LS-C1	Skirt1	SS-C0	Mean
NeuS	3.18	4.82	5.71	2.21	3.60	2.44	5.13	3.87
VolSDF	5.92	4.79	5.96	4.36	8.73	7.74	8.84	6.62
NeAT	3.06	4.33	5.92	3.52	8.84	3.91	4.30	4.84
NeuralUDF	**1.92**	2.05	4.11	1.50	2.47	2.16	2.15	2.34
NeUDF	1.95	2.93	N.A.	1.48	2.66	2.74	**1.77**	2.26
Ours	**1.92**	**1.97**	**2.46**	**1.47**	**2.14**	**1.84**	1.91	**1.96**

Table 1. Chamfer distances ($\times 10^{-3}$) on DF3D#Ours (top) and DF3D#NeuralUDF (bottom). NeAT requires mask supervision and others do not need.

We evaluate our method and compare it with baselines using the garments from DeepFashion3D [43], where the models have multiple open boundaries. VolSDF and NeuS always close the boundaries since they learn SDFs.

NeuralUDF, NeUDF and NeAT are designed to learn non-watertight models. NeAT learns SDFs for open models, and requires mask supervision to produce reasonable results, but other methods do not require mask supervision for DeepFashion3D. The released codebase of NeuralUDF indicates that it also has a two-stage training process. We evaluate the results of NeuralUDF at the end of both stages, and present whichever is better.

In contrast, NeuralUDF, NeUDF and our method learn UDFs, which can generate open models. Table 1 shows the Chamfer distances of the results on DeepFashion3D. Some of the Chamfer distances of the compared methods are large because the open holes are closed or the model is over-smoothed, resulting in significant errors.

As demonstrated in Figure 3, we test various types of garments, some of which have rich textures, while others are nearly a single color. Learning UDFs for texture-less models is more challenging since various regions of a model are ambiguous without clear color differences. However, our 2S-UDF generates satisfactory results even without masks. Though with mask supervision, the results of

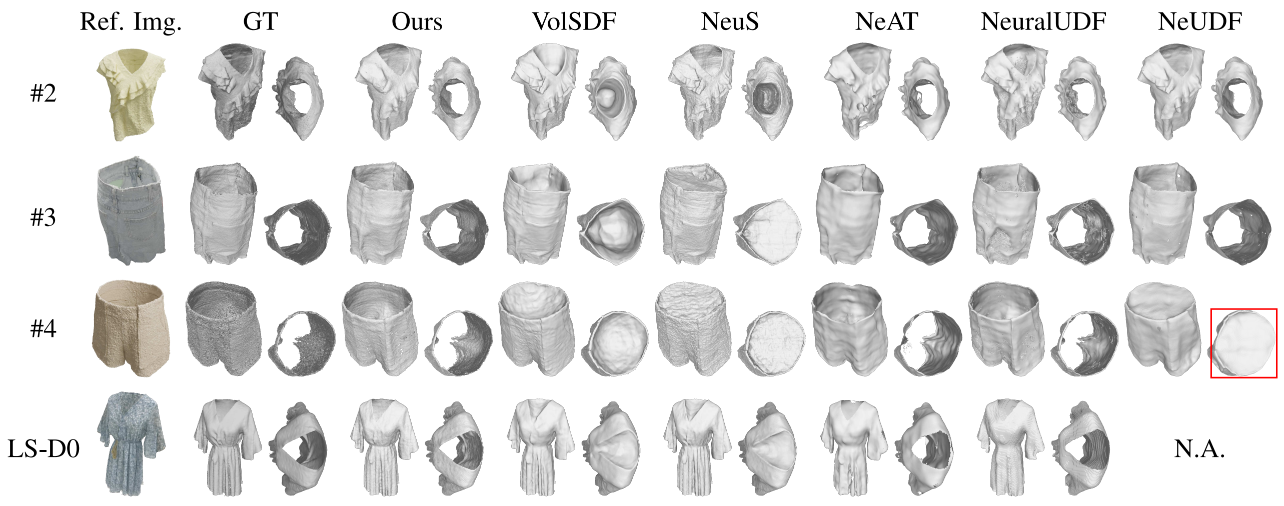

Figure 3. Visual comparisons on selected models of the DeepFashion3D [43] dataset. The surfaces produced by NeuS and VolSDF are closed watertight models, thereby post-processing is required to remove the unnecessary parts. NeAT can produce open models by learning an SDF and predicting which surfaces in the extracted meshes should be removed, but it needs mask for supervision. NeuralUDF can generate open surfaces, but struggles with textureless inputs, leading to double-layered regions and large reconstruction errors. NeUDF generally performs well, but its training is unstable and may stumble on less distinguished, darker models like LS-D0. In contrast, our 2S-UDF consistently delivers effective reconstructions of non-watertight models. See the supplementary material for additional results.

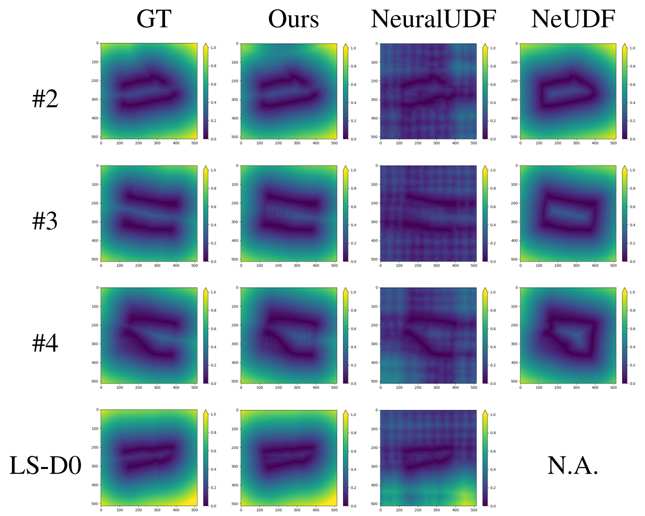

Figure 4. Visualization of the learned UDFs on cross sections. Compared with the ground truth, our method can learn a UDFs that most closely resemble the ground truth, among our method, NeuralUDF, and NeUDF. NeAT is omitted in this visualization, because it learns SDFs in lieu of UDFs. Note that for LS-D0, NeUDF completely collapses without a reasonable UDF learned.

NeAT [27] are over-smoothed, missing details, resulting in large Chamfer distance errors. NeuralUDF [25] is unable to properly reconstruct textureless models on most models, possibly due to their complex density function which is difficult to converge. Some of the NeUDF [23] models become watertight. To analyze the reasons, we illustrate these UDFs cross sections in Figure 4. To compute the ground truth UDFs, we sample 30,000 points from every input point model and compute the distances to the nearest sample point for every point in a 3D grid of resolution $512 \times 512 \times 512$. All other UDFs are extracted by querying the distance neural network in a 3D grid of the same resolution. Our learned UDFs resemble the ground truth with little difference. While, the UDFs of NeuralUDF deviate from the ground truth significantly explaining its difficulty to converge. The UDFs of NeUDF are better, but the distances approach to zero around open holes. As a result, it is challenging and tricky to generate non-watertight models and some of them are even closed. NeAT learns SDF, so we do not show their distance fields.

As illustrated in Figure 5, perhaps due to the absolute of an MLP for UDF representation, NeuralUDF possibly generates two layers of zero level-sets on both sides of the surface resulting in double-layered regions after Stage 1 learning. However, in its Stage 2 refinement, the surface is crushed into pieces and the Chamfer distance errors surge suddenly.

In Figure 6, we conduct additional experiments on some open model dataset provided by NeUDF [23]. For the rack model, the thin structures reconstructed by NeuralUDF [25] and NeUDF [23] seem eroded, but ours don't. The thin structures reconstructed by NeAT [27] is the closest to the reference image, but the surface is dented inward with visible artifacts due to imperfect SDF validity learning.

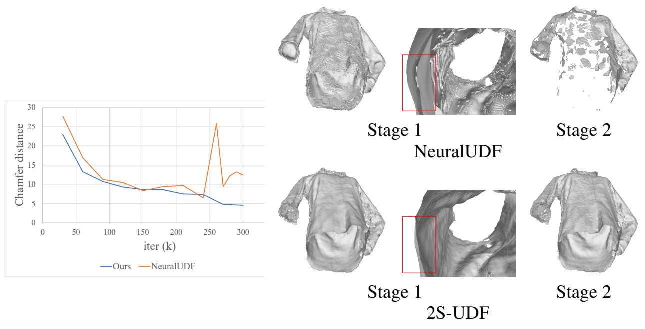

Figure 5. Plots of the Chamfer distance throughout the training process. Our method consistently reduces CD across both stages. In contrast, NeuralUDF, which also adopts a two-stage learning strategy, exhibits instability and yields a fragmented output following the second stage. The first-stage output of NeuralUDF, however, contains double-layered regions as marked above. In this figure, both methods start their stage 2 training at 250k iterations.

Method	37	55	65	69	97	105	106	114	118	122	Mean
NeuralUDF	1.18	**0.44**	**0.66**	**0.67**	**0.94**	0.95	**0.57**	**0.37**	**0.56**	0.55	**0.69**
NeAT	1.18	0.47	0.82	0.84	1.09	0.75	0.76	0.38	**0.56**	0.55	0.74
NeUDF	0.90	0.65	0.73	0.97	1.07	**0.63**	0.94	0.59	0.72	0.62	0.78
Ours	**0.89**	0.55	0.68	0.88	1.15	0.70	0.74	0.41	0.61	**0.51**	0.71

Table 2. Chamfer distances on DTU dataset.

The plant model does not have an object mask, making NeAT [27] impractical for training. NeuralUDF [25] completely fails to reconstruct a reasonable surface. Between our method and NeUDF [23] which can reconstruct a sensible model, the flower pot region marked in red is missing in NeUDF but not in ours. These show our method's ability to reconstruct non-watertight models more robustly compared to other methods.

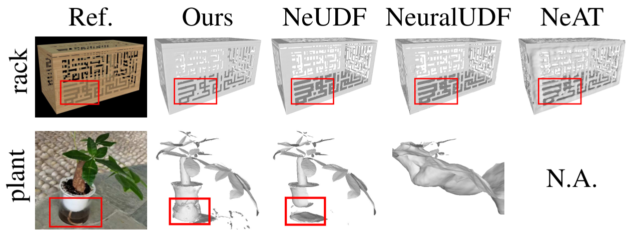

Figure 6. Qualitative comparisons with NeAT [27], NeuralUDF [25] and NeUDF [23] on some example data released by NeUDF [23]. Note that NeAT cannot reconstruct "plant" dataset because the ground truth mask for "plant" is unavailable.

4.2. Comparisons on Watertight Models

Other methods can also be used as the first stage of our 2S-UDF. We use NeUDF for the first stage training on the DTU dataset [19]. As detailed in Table 2, we compare the Chamfer distances of the reconstruction results with NeuralUDF, NeAT and NeUDF without our second-stage training. SDFs generally excel at learning watertight models, and it is worth pointing out that NeuralUDF takes the absolute value of the output of MLP as the UDF value of a given point. Therefore for closed models, they can easily learn an SDF and take its absolute value to produce a UDF. NeAT, on the other hand, explicitly learns an SDF. NeUDF and our method truly learn UDFs. While UDF learning is much more complicated than SDF learning because the UDF gradient nearby 0 is blurry and the gradient is not available at 0, our method still improves the reconstruction quality of NeUDF by around 10% as shown in Figure 7. We further provide a close-up view of specific parts of the models for detailed comparisons in Figure 7. These local callouts exhibit the ground truth points located on both sides of our surfaces, whereas most of the points are only on one side of the surfaces of NeUDF. These illustrate our reconstructed surfaces are closer to the ground truth points and thus improving the resulting quality over NeUDF, on a par with NeuralUDF and NeAT.

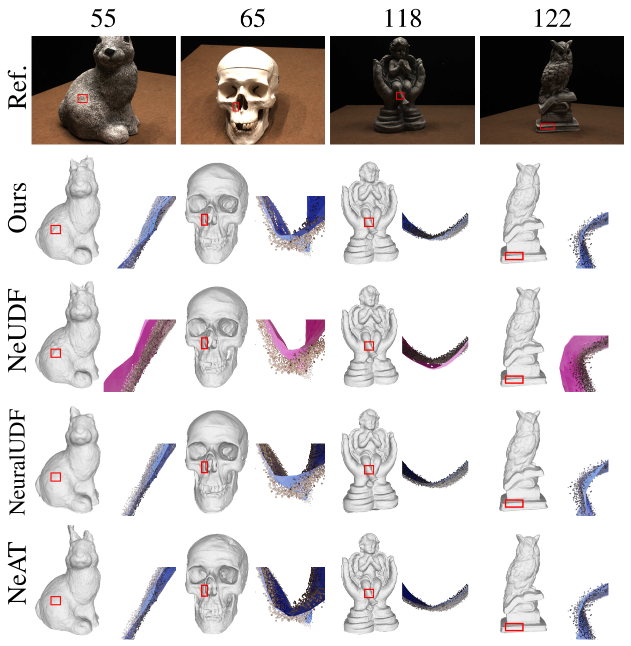

Figure 7. Qualitative comparisons with NeAT, NeuralUDF and NeUDF on the DTU [19] dataset and close-up comparisons against NeUDF. Our method can reconstruct surfaces closer to the ground truth point clouds in various places such as the marked region, generally improving the reconstruction accuracy of NeUDF by around 10%, on a par with NeuralUDF and NeAT at the bottom two rows.

4.3. Ablation Studies

In this section, we present main ablation studies. We refer interested readers to the supplementary material for additional ablation studies.

Effect of the two-stage training. We conduct an ablation

Method	#1	#7	#8	LS-D0
S1 & S2	**4.55**	<u>2.88</u>	**3.21**	**2.46**
S1	7.22	**2.46**	<u>3.38</u>	6.04
S2	<u>5.75</u>	4.00	5.96	<u>3.65</u>
Method	NS-D1	LS-C1	DTU 114	DTU 122
S1 & S2	<u>1.47</u>	**2.14**	**0.41**	**0.51**
S1	**1.46**	6.23	<u>0.59</u>	0.62
S2	1.64	<u>2.98</u>	0.63	<u>0.60</u>

Table 3. Chamfer distances of models learned by both Stage 1 and 2 (S1 & S2), only Stage 1 (S1) and only Stage 2 (S2) on selected datasets. Models learned by two stages yield similar Chamfer distances, but when trained with only Stage 1 or Stage 2, the Chamfer distances generally become significantly higher.

study on the effect of the two-stage learning. We compare the Chamfer distances among both two stages, only Stage 1 and only Stage 2 training, shown in Table 3. Our results show that two-stage training improves the Chamfer distance (lower is better) compared to training with only Stage 1 or 2, under most circumstances.

It should be noted that training by the second stage from scratch is also capable of generating a generally reasonable result. However, the Chamfer distances, as shown in Table 3, indicate that its learning ability is limited. Therefore, the second refinement learning stage should cooperate with the first coarse learning stage to generate the best results.

Choice of accumulated weight threshold δ_{thres}. In Stage 2, being a ray truncate point requires the accumulated weight up until that point to be greater than δ_{thres}, where we intuitively select $\delta_{thres} = 0.5$. Figure 8 shows the reconstruction results for other choices of δ_{thres}, namely 0.3 and 0.7, respectively. We observe that all threshold choices successfully reconstruct the model. Setting the threshold δ_{thres} up to 0.7 produces visually similar results. Setting the threshold δ_{thres} down to 0.3 also works fine generally despite that it may introduce more holes to the reconstructed meshes. We deduce that setting a lower threshold increases the possibility that a ray may be truncated prematurely, leading to less desirable results. Nevertheless, we still have a considerable range of δ_{thres} from 0.3 to 0.7 without major result regression, indicating that our Stage 2 training exhibits robustness against δ_{thres}.

4.4. Limitations

Since the light is cut off after going through a layer of surface, our method relinquishes the ability to model planes with transparency. Occasionally, due to learning uncertainty, the Chamfer distance may increase slightly in the second stage, but the difference is quite small without visual impact. Overall, the two-stage learning improves the quality significantly. For watertight models, SDF learning is more suitable than UDF learning, since UDF learning is

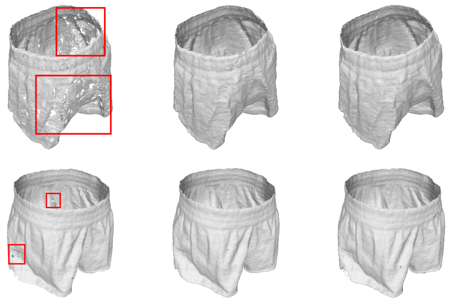

Figure 8. Qualitative comparisons on different choices of accumulated weight δ_{thres}. Setting a higher threshold works well few little visual differences; Setting a lower threshold generally works fine, but may introduce more holes in reconstructed meshes.

more complicated than SDF learning. We still advise using SDF learning, e.g., NeuS [34], HF-NeuS [36] or PET-NeuS [37], for watertight model reconstruction. Also, the mesh extraction of MeshUDF [15] tends to generate holes and "staircase" artifacts affecting the mesh reconstruction quality. Adopting a more robust extraction method, e.g., DoubleCoverUDF [18], could alleviate the problem, but we use MeshUDF here for all methods for a fair comparison.

5. Conclusions

Overall, 2S-UDF offers a promising approach to the problem of reconstructing both open and watertight models from multi-view images. Its advantages over existing methods lie in the use of a simple and more accurate density function, and a smooth differentiable UDF representation, so that the learned UDF approximates the ground truth as much as possible. A two-stage learning strategy further eliminates bias and improves UDF accuracy. Results from our experiments on the DeepFashion3D, DTU and BlendedMVS datasets demonstrate the effectiveness of our method, particularly in learning smooth and stably open UDFs revealing the robustness of 2S-UDF. Moreover, our method does not rely on object masks for open model reconstruction, making it more practical in real-world applications.

Acknowledgments

This project was supported in part by the National Natural Science Foundation of China under Grants (61872347, 62072446), in part by the National Key R&D Program of China under Grant 2023YFB3002901, in part by the Basic Research Project of ISCAS under Grant ISCAS-JCMS-202303 and in part by the Ministry of Education, Singapore, under its Academic Research Fund Grants (MOE-T2EP20220-0005, RG20/20 & RT19/22).

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D Surface Reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6280–6291, 2022. 3

[2] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Vis. Comput. Graph.*, 5(4):349–359, 1999. 2

[3] A. Broadhurst, T.W. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *Int. Conf. Comput. Vis.*, pages 388–393 vol.1, 2001. 2

[4] Rohan Chabra, Jan E. Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction. In *Eur. Conf. Comput. Vis.*, pages 608–625, Cham, 2020. Springer International Publishing. 2

[5] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6968–6979, 2020. 2

[6] Julian Chibane, Mohamad Aymen mir, and Gerard Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. In *Adv. Neural Inform. Process. Syst.*, pages 21638–21652. Curran Associates, Inc., 2020. 3

[7] François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces geometry with patch warping. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6250–6259, 2022. 1, 2

[8] J. De Bonet and P. Viola. Roxels: responsibility weighted 3D volume reconstruction. In *Int. Conf. Comput. Vis.*, pages 418–425 vol.1, 1999. 2

[9] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating Second-Order Functional Knowledge for Better Option Pricing. In *Adv. Neural Inform. Process. Syst.* MIT Press, 2000. 4

[10] Qiancheng Fu, Qingshan Xu, Yew Soon Ong, and Wenbing Tao. Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-view Reconstruction. In *Adv. Neural Inform. Process. Syst.*, pages 3403–3416. Curran Associates, Inc., 2022. 2

[11] Kunihiko Fukushima. Cognitron: a self-organizing multilayered neural network. *Biological Cybernetics*, 20(3-4):121–136, 1975. 4

[12] Yasutaka Furukawa and Carlos Hernández. Multi-View Stereo: A Tutorial. *Found. Trends. Comput. Graph. Vis.*, 9 (1-2):1–148, 2015. 2

[13] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively Parallel Multiview Stereopsis by Surface Normal Diffusion. In *Int. Conf. Comput. Vis.*, pages 873–881, 2015. 2

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020. 5

[15] Benoît Guillard, Federico Stella, and Pascal Fua. MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In *Eur. Conf. Comput. Vis.*, pages 576–592, Cham, 2022. Springer Nature Switzerland. 8

[16] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. 2

[17] Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. Iterative poisson surface reconstruction (iPSR) for unoriented points. *ACM Trans. Graph.*, 41(4), 2022. 2

[18] Fei Hou, Xuhui Chen, Wencheng Wang, Hong Qin, and Ying He. Robust Zero Level-Set Extraction from Unsigned Distance Fields Based on Double Covering. *ACM Trans. Graph.*, 42(6), 2023. 8

[19] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large Scale Multi-view Stereopsis Evaluation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 406–413, 2014. 2, 5, 7

[20] Mengqi Ji, Jinzhi Zhang, Qionghai Dai, and Lu Fang. SurfaceNet+: An End-to-end 3D Neural Network for Very Sparse Multi-View Stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(11):4078–4093, 2021. 2

[21] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a Multi-View Stereo Machine. In *Adv. Neural Inform. Process. Syst.* Curran Associates, Inc., 2017. 2

[22] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), 2013. 2

[23] Yu-Tao Liu, Li Wang, Jie Yang, Weikai Chen, Xiaoxu Meng, Bo Yang, and Lin Gao. NeUDF: Leaning Neural Unsigned Distance Fields with Volume Rendering. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 237–247, 2023. 1, 2, 3, 4, 5, 6, 7

[24] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. SparseNeuS: Fast Generalizable Neural Surface Reconstruction from Sparse Views. In *Eur. Conf. Comput. Vis.*, pages 210–227, Cham, 2022. Springer Nature Switzerland. 2

[25] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. NeuralUDF: Learning Unsigned Distance Fields for Multi-View Reconstruction of Surfaces with Arbitrary Topologies. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 20834–20843, 2023. 1, 2, 3, 4, 5, 6, 7

[26] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-Pull: Learning Signed Distance Function from Point clouds by Learning to Pull Space onto Surface. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 7246–7257. PMLR, 2021. 2

[27] Xiaoxu Meng, Weikai Chen, and Bo Yang. NeAT: Learning Neural Implicit Surfaces with Arbitrary Topologies from Multi-View Images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–258, 2023. 2, 5, 6, 7

[28] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In

IEEE Conf. Comput. Vis. Pattern Recog., pages 4455–4465, 2019. 2

[29] 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 *Eur. Conf. Comput. Vis.*, pages 405–421, Cham, 2020. Springer International Publishing. 1, 2

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 165–174, 2019. 2

[31] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Eur. Conf. Comput. Vis.*, pages 501–518, Cham, 2016. Springer International Publishing. 2

[32] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Adv. Neural Inform. Process. Syst.*, pages 7462–7473. Curran Associates, Inc., 2020. 2

[33] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-Time Coherent 3D Reconstruction from Monocular Video. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 15593–15602, 2021. 2

[34] 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. In *Adv. Neural Inform. Process. Syst.*, pages 27171–27183. Curran Associates, Inc., 2021. 1, 2, 5, 8

[35] Yifan Wang, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields. In *Int. Conf. Learn. Represent.* OpenReview.net, 2022. 2

[36] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. HF-NeuS: Improved Surface Reconstruction Using High-Frequency Details. In *Adv. Neural Inform. Process. Syst.*, pages 1966–1978. Curran Associates, Inc., 2022. 1, 2, 3, 5, 8

[37] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. PET-NeuS: Positional Encoding Tri-Planes for Neural Surfaces. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12598–12607, 2023. 2, 8

[38] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent MVSNet for High-Resolution Multi-View Stereo Depth Inference. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5520–5529, 2019. 2

[39] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A Large-Scale Dataset for Generalized Multi-View Stereo Networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1787–1796, 2020. 5

[40] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume Rendering of Neural Implicit Surfaces. In *Adv. Neural Inform. Process. Syst.*, pages 4805–4815. Curran Associates, Inc., 2021. 1, 2, 5

[41] Fang Zhao, Wenhao Wang, Shengcai Liao, and Ling Shao. Learning Anchored Unsigned Distance Functions with Gradient Direction Alignment for Single-view Garment Reconstruction. In *Int. Conf. Comput. Vis.*, pages 12654–12663, 2021. 3

[42] Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning Consistency-Aware Unsigned Distance Functions Progressively from Raw Point Clouds. In *Adv. Neural Inform. Process. Syst.*, pages 16481–16494. Curran Associates, Inc., 2022. 3

[43] Heming Zhu, Yu Cao, Hang Jin, Weikai Chen, Dong Du, Zhangye Wang, Shuguang Cui, and Xiaoguang Han. Deep Fashion3D: A Dataset and Benchmark for 3D Garment Reconstruction from Single Images. In *Eur. Conf. Comput. Vis.*, pages 512–530, Cham, 2020. Springer International Publishing. 2, 5, 6