FIRe: Fast Inverse Rendering using Directional and Signed Distance Functions

Tarun Yenamandra[1] Ayush Tewari[2] Nan Yang[1] Florian Bernard[3]
Christian Theobalt[4] Daniel Cremers[1]
[1]TU Munich, MCML [2]Massachusetts Institute of Technology [3]University of Bonn [4]MPI Informatics, SIC

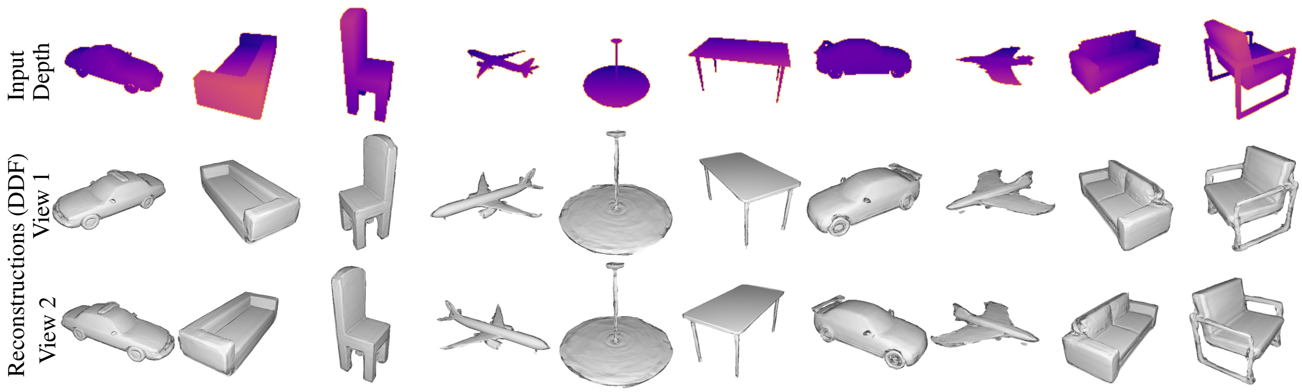

Figure 1. We propose a novel neural scene representation based on directional distance function (DDF), which enables us to replace sphere tracing for rendering images from a signed distance function (SDF) model. We learn the SDF and DDF models on a class of 3D shapes. During inference, given a depth map (top row), we reconstruct 3D shapes by means of our proposed algorithm (FIRe) which is 15 times faster (per iteration) and more accurate than competing methods. In the last two rows, we show images of reconstructions rendered using our DDF model with just a **single network evaluation per ray**.

Abstract

Neural 3D implicit representations learn priors that are useful for diverse applications, such as single- or multiple-view 3D reconstruction. A major downside of existing approaches while rendering an image is that they require evaluating the network multiple times per camera ray so that the high computational time forms a bottleneck for downstream applications. We address this problem by introducing a novel neural scene representation that we call the directional distance function (DDF). To this end, we learn a signed distance function (SDF) along with our DDF model to represent a class of shapes. Specifically, our DDF is defined on the unit sphere and predicts the distance to the surface along any given direction. Therefore, our DDF allows rendering images with just a single network evaluation per camera ray. Based on our DDF, we present a novel fast algorithm (FIRe) to reconstruct 3D shapes given a posed depth map. We evaluate our proposed method on 3D reconstruction from single-view depth images, where we empirically show that our algorithm reconstructs 3D shapes more accurately and it is more than 15 *times faster (per iteration) than competing methods.*

1. Introduction

The field of generating 3D shapes [26,32,33,54] has seen unprecedented growth in the recent past due to novel neural network architectures. Yet, there are many open challenges for generating realistic 3D shapes, such as data availability and 3D shape representations. Further, using the 3D generative models for accurately reconstructing 3D shapes given partial observations such as depth maps or point clouds is still in the early stages.

Implicit scene representations have proven to be the most suitable data representations for generating 3D surfaces using deep neural networks. Among others, signed distance functions (SDFs) are commonly used. SDFs represent a 3D shape as the level-set of a function, $\{x \in \mathbb{R}^3 \mid f(x) = 0\}$. At every point in space, the SDF of a 3D shape evaluates to the minimum distance to the surface. The sign indicates if the point is inside or outside the shape. For rendering an image of the shape represented by SDFs, one must perform a line search along each camera ray to find the distance to the surface. Sphere tracing [18] accelerates this process for SDFs by exploiting the minimum distance property of SDFs. Inverse rendering is the process of optimizing for the shape and other properties from one or many images [52].

Substantial progress has been made towards single shape or scene reconstruction [27, 52] from dense multi-view images using inverse rendering. Some of the models [4, 6, 8, 28, 44] trade off memory for speed enabling real-time rendering. However, these models cannot be used as priors as they reconstruct a single scene. Further, the major focus of these methods is to generate novel views of a scene rather than reconstruct geometry. It is an open problem to train such models to represent different shapes with accurate geometry.

In contrast, a 3D generative model learns a conditional implicit function of shapes. In addition to a 3D point, a 3D generative model accepts a latent code as an input to represent different shapes. DeepSDF [32] learns a class of shapes using an autodecoding framework. Models trained on many shapes can be used as priors to reconstruct shapes from partial observations, such as images, at test time. We use inverse rendering to optimize for the latent code of a generative model during inference. However, for each optimization step, we need to render an image by sphere tracing through a neural implicit representation as done in DIST [24].

Novel Scene Representation: In this paper, we propose to accelerate inverse rendering algorithms with learned models by avoiding sphere tracing at each iteration of the algorithms. Towards that, we propose a novel scene representation called directional distance function (DDF). We propose to use DDF along with the signed distance function (SDF). We assume that the 3D shapes that our models represent are inside the unit sphere. While the SDF is defined everywhere, our DDF is defined on the surface of the unit sphere. Our DDF model learns to predict the distance to the object's surface along rays cast in all directions from the unit sphere's surface. DDF has two output components - the directional distance, and the probability of the ray hitting the surface. The learned DDF model accelerates inverse rendering algorithms by reducing the number of network evaluations required to find the object surface to 1 for each iteration.

Enabling Fast Inverse Rendering: We propose a shape optimization algorithm that utilizes our proposed neural representation to reconstruct the 3D shape given partial observations, such as single view depth. As our DDF model replaces the sphere tracing algorithm, our algorithm is $15.5\times$ faster than competing methods. Our contributions are as follows.

1. A novel neural scene representation, DDF defined on the unit sphere, for rendering images from our SDF model during inference with 1 forward pass through the model.

2. An algorithm to reconstruct 3D shapes from single view depth maps using our DDF and SDF models, which is $15.5\times$ per iteration faster than competing methods.

2. Related Work

In the following, we introduce relevant papers from different domains.

Implicit Representations: Implicit shape representations, in particular SDFs, have been studied for decades as they can represent shapes with arbitrary topology [12, 15, 21–23, 29, 35, 42, 43]. Recently, neural network-based implicits [26, 32] have proven to be a compact way to represent SDFs. DeepSDF [32] learns the SDF values using an autodecoder architecture conditioned on a learned latent code set to generate different 3D shapes. OccupancyNet [26] and IMNet [9] learn object surfaces as decision boundaries using an autoencoder architecture. Instead of encoding global priors, local priors [10, 20, 33, 51] have been explored to handle large-scale scenes and more detailed representations. However, these rely on generating large feature grids using neural networks. Further, for each new downstream task such as reconstructing from a depth map, or even handling a change in input image resolution, they need to train a new encoder. Differentiable rendering-based methods [31, 41] alleviate the need for 3D supervision by learning from images. Pixel features [37, 38, 55] have been used to condition implicit representations for novel view synthesis given a single image. However, they cannot model the geometry of the objects satisfactorily. Other novel view synthesis [40] methods suffer from similar problems. Applications of implicit representation on human [37, 38, 46, 47], face [54], and hair [36, 54] modeling are also explored and have achieved superior results to classical methods.

Directional Distance Prediction: Recent efforts towards predicting the occupancy density distribution along the rays [34], or, alternatively, a region along the ray instead of distance [30], have proven to accelerate volumetric rendering. However, they only model single objects and they still need to perform local sampling for volumetric rendering. We note CPDDF [1], PRIF [16], NeuralODF [19], and SDDF [56] as our contemporary works, which propose to use DDF as a standalone representation. However, unlike these methods, we use both SDF and DDF for high-quality geometric details while defining the DDF only on the surface of the unit sphere.

Single View 3D Reconstruction: Single-view reconstruction is generally an ill-posed problem, general solutions [17, 48, 50, 53] exploit low-level geometric or photometric properties, whereas shape-specific methods [3, 31, 37, 40, 45, 49, 51, 55] solve the problem using learned priors [2, 25, 32]. The closest to our algorithm is DIST [24], which reconstructs 3D shapes given a depth map. However, it requires multiple evaluations per ray, whereas ours needs only a single evaluation.

3. Method

We learn the two neural representations, DDF and SDF. The DDF model represents the distance to the surface of an object from a point on the unit sphere along a given direction, and the probability of the ray hitting the surface. This helps avoid the computationally intensive sphere tracing step for rendering images, especially for solving image-based inverse problems such as 3D reconstruction. In the following, we first introduce our representation, followed by our network architecture, and then our proposed algorithm.

3.1. Directional Distance Representation

Our 3D shape representation consists of two components: (i) a distance d to the surface of an object along a given direction r from a point p on the surface of the unit sphere, called the directional distance (DDF) and (ii) the signed distance s at every point inside the unit sphere enclosing the object (SDF).

The directional distance d and signed distance s are related as follows - outside the surface of the object the signed distance is positive and the value is the minimum distance between a given point $x \in \mathbb{R}^3$ and the object surface (in any given direction), i.e.,

$$\mathrm{SDF}(x) = \min_r \mathrm{DDF}(x, r),$$

where $s = \mathrm{SDF}(x)$ is the signed distance at the point $x \in \mathbb{R}^3, r \in \mathbb{S}^2$ is a given direction from x pointing towards the surface of the object.

In our proposed directional distance representation, we learn to predict the DDF on the unit sphere $p \in \mathbb{S}^2$ along with a ray-hitting probability $\sigma \in [0, 1]$. Further, the value of SDF at the distance predicted along a hitting direction r must be 0, or

$$\mathrm{SDF}(p + d_{\sigma=1} r) = 0, d_{\sigma=1} = \mathrm{DDF}(p, r). \tag{1}$$

In the following, we introduce our neural model that learns to represent this function along with SDF, and show how we exploit the relationship defined in Eq. (1).

3.2. Network

Our model consists of two shape representations, SDF and DDF. For both, we use neural networks conditioned on latent codes to represent multiple shapes. Further, we make our generative model of 3D shapes viable to represent shapes with much higher accuracy. We achieve this by conditioning them on high-dimensional features that are sampled from learned feature planes for each shape category, as shown in Fig. 2. Our network architecture is inspired by Pi-GAN's [5] implementation[1].

[1] https://github.com/marcoamonteiro/pi-GAN/blob/master/siren/siren.py#L255

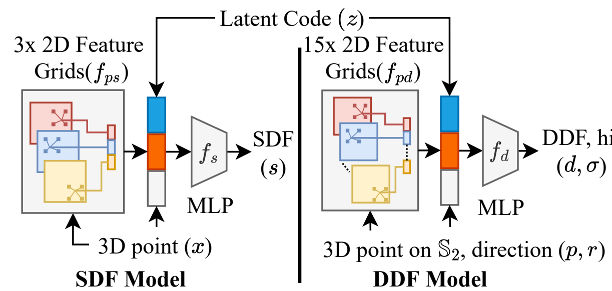

Figure 2. SDF and DDF Models: Our SDF model generalizes with high-dimensional feature inputs from 3 2D feature grids (f_{ps}) by sampling from the grid with bilinear interpolation given a 3D point ($x \in \mathbb{R}^3$) and a latent code (z). Similarly, our DDF model takes as input a point on the unit sphere and a direction ($(p, r) \in \mathbb{R}^6$) along with a latent code (z) to generalize with features from 15 2D feature grids for each shape category. The SDF and DDF models have a shared latent space for each shape category.

2D Feature Grids: High-dimensional features stored in a high-resolution grid have proven to be effective in reducing rendering times for representing complex shapes [8, 39, 44]. For example, given a 3D point $x \in \mathbb{R}^3$, we sample a feature from a learned high-resolution grid, e.g. 256^3, by fetching 8 nearest features in the grid and trilinearly interpolating in the cube formed by the 8 neighboring features. We assume that the function we learn is linear in the high-dimensional feature space, and process the features using an MLP to obtain the value of the function at the given 3D point. Recent efforts [6, 8] to factorize the 3D grids into three 2D grids have proven to be effective. We elucidate in the following text.

2D Feature Grids for SDF:
For 3-dimensions: A feature grid $M_{xyz} \in \mathbb{R}^{N \times N \times N} \times \mathbb{R}^K$ with resolution N and K-dimensional features can be factored into three 2D feature grids $(M_{xy}, M_{yz}, M_{zx}) \in \mathbb{R}^{3 \times N \times N} \times \mathbb{R}^K$. Performing this factorization, we assume that the distribution of high-dimensional features in (x, y) is independent of z, (y, z) is independent of x, and that of (z, x) is independent of y. We expect that the MLP handles cases where this assumption is broken. For SDF, we define three feature grids, $(M^s_{xy}, M^s_{yz}, M^s_{zx}) \in \mathbb{R}^{3 \times N \times N} \times \mathbb{R}^K$. Given a 3D point $x \in \mathbb{R}^3$, we retrieve the features $m^s_{xy} \sim M^s_{xy}$ [2], $m^s_{yz} \sim M^s_{yz}$, and $m^s_{zx} \sim M^s_{zx}$, where m^s_{xy}, m^s_{yz}, and $m^s_{zx} \in \mathbb{R}^K$. Using this, we define a function $f^{ps} : \mathbb{R}^3 \to \mathbb{R}^{3 \times K}$ as

$$f^{ps}(x) = (m^s_{xy}, m^s_{yz}, m^s_{zx}). \tag{2}$$

2D Feature Grids for DDF: Our DDF representation is defined on a $6D$ grid, therefore we need to factorize a

[2] By '$\sim$' we mean to sample 4 neigbouring features of a given 2D location in the grid and bilinearly interpolating between the features.

6D grid into 2D grids. The number of 2D grids we need is $\binom{6}{2} = 15$, which is the number of 2D tuples we can make from a 6D tuple i.e., $(p_x, p_y, p_z, r_x, r_y, r_z)$ factorizes into $\{(p_x, p_y), (p_y, p_z), \dots, (r_z, r_x)\}$. For points $p = (p_x, p_y, p_z) \in \mathbb{S}^2$ on the unit sphere, and directions $r = (r_x, r_y, r_z) \in \mathbb{S}^2$, we define 15 feature grids, $(M^d_{p_xp_y}, \dots, M^d_{r_zr_x}) \in \mathbb{R}^{15\times N\times N} \times \mathbb{R}^K$. Given a 6D tuple $(p, r) \in \mathbb{R}^6$, we retrieve the features $m^d_{p_xp_y} \sim M^d_{p_xp_y}, \dots$, and $m^d_{r_zr_x} \sim M^d_{r_zr_x}$, where $m^d_{p_xp_y}, \dots, m^d_{r_zr_x} \in \mathbb{R}^K$. Using this, we define a function, $f^{pd} : \mathbb{R}^6 \to \mathbb{R}^{15\times K}$ as

$$f^{pd}(p, r) = (m_{p_xp_y}, \dots, m_{r_zr_x}). \tag{3}$$

Without this factorization, the memory required to store a 6D grid scales as $\mathcal{O}(N^6)$ for 2D featured grids, and is computationally highly inefficient. The 2D feature grids' memory requirements scale quadratically $\mathcal{O}(N^2)$ with the grid resolution. The 2D feature grids, $f^{pd}(p, r)$, and $f^{ps}(x)$, are learned per shape class and not per object.

SDF Model: The SDF model takes as input a latent code per shape $z \in \mathbb{R}^L$, a high-dimensional feature vector $f_{ps}(x) \in \mathbb{R}^{3\times K}$ from Eq. (2), and a point inside the unit sphere $x \in \mathbb{R}^3$. With that, it outputs an SDF value $s \in \mathbb{R}$. We learn the SDF model, $f_s : \{\mathbb{R}^3, \mathbb{R}^L, \mathbb{R}^{3\times K}\} \to \mathbb{R}$, using an MLP with parameters Θ_s as

$$f_s(x, z, f_{ps}(x); \Theta_s) = s\,. \tag{4}$$

DDF Model: The DDF model takes as input a latent code (per shape) $z \in \mathbb{R}^L$, a high-dimensional feature vector $f_{pd}(x) \in \mathbb{R}^{15\times K}$ from Eq. (3), a tuple with a point on the unit sphere, and a direction $(p, r) \in \mathbb{R}^6$. The model outputs a DDF value $d \in \mathbb{R}_+$, and a ray hit probability $\sigma \in [0, 1]$. We learn the DDF model $f_d : \{\mathbb{R}^6, \mathbb{R}^L, \mathbb{R}^{15\times K}\} \to \{\mathbb{R}_+, [0, 1]\}$ using an MLP with parameters Θ_d as

$$f_d((p, r), z, f_{pd}(x); \Theta_d) = (d, \sigma)\,. \tag{5}$$

We encode the inputs to our models, x and (p, r), with positional encoding from NeRF [27].

3.3. Training

We train a model for each class of the ShapeNet dataset [7].

Data Preprocessing: For training the network, we use the ground truth signed distance and the ground truth directional distance supervision. We use the preprocessing pipeline from DeepSDF [32] to sample about 1 million points for SDF supervision. We randomly sample 1 million points on the unit sphere and random directions that point to the surface of an object using the object's point cloud for DDF supervision. We also sample $500k$ points and random missing directions. We render the 1.5mil rays using Trimesh [13] to obtain ground truth distances and ray hit supervision.

Losses: We train the network with the following losses:
SDF loss $\mathcal{L}_s$. We supervise the SDF network to predict signed distances s (Eq. (4)), with ground truth SDFs s_{GT} using

$$\mathcal{L}_s(s) = \|s - s_{GT}\|_1\,. \tag{6}$$

DDF loss $\mathcal{L}_d$. We learn the DDF model by supervising the model to predict directional distances d (from Eq. (5)) which are close to their corresponding ground truth distances d_{GT} using

$$\mathcal{L}_d(d) = \|d - d_{GT}\|_1\,. \tag{7}$$

Ray hit loss $\mathcal{L}_\sigma$. We supervise the ray hit predictions σ from the DDF model with the ray hit ground truths σ_{GT} using the binary cross entropy loss as

$$\mathcal{L}_\sigma(\sigma) = -(1 - \sigma_{GT}) \log(1 - \sigma) - \sigma_{GT} \log(\sigma)\,. \tag{8}$$

TV regularizer $\mathcal{L}_{tv}$. We enforce that the gradient of each of the 2D feature grids is small so that the features learned in the grid result in shapes that are not noisy, for both the feature grids of DDF and SDF models. The loss is given by

$$\mathcal{L}_{tv}(M) = \sum_i \|\nabla M^s_i\|_2 + \sum_i \|\nabla M^d_i\|_2\,, \tag{9}$$

where the gradients, ∇M^s_i and ∇M^d_i, are computed using finite differences similar to how it is done for 3D feature grids [14], $i = xy, yz, zx$ for SDF feature grids and $i = p_xp_y, p_yp_z, \dots, r_zr_x$ for DDF feature grids.

Track-SDF Regularizer. The predicted directional distance and the signed distance for an object need not agree, therefore, we additionally constrain that the DDF prediction results in a point close to the surface predicted by the SDF using the Track-SDF regularizer. Towards that, we compute the points using the predictions of the DDF model as $p + dr$ for point and direction pairs that point to the object surface. We enforce that these points are close to 0 using

$$\mathcal{L}_{ts}(d) = \|f_s(p + dr)\|_1\,, \tag{10}$$

where (p, r) are point-direction tuples that point to a surface, and d is the predicted directional distance for the point-direction tuples as in Eq. (5). Note that we only train the DDF model, and not the SDF model, with this loss.
Latent code regularizer $\mathcal{L}_l$. As we use an autodecoder framework [32], we enforce that the latent codes for different shapes are close to each other. This can be achieved by

penalizing latent codes with large magnitudes so that latent codes are close to zero, i.e.,

$$\mathcal{L}_l(z) = \|z\|_2 \,, \tag{11}$$

where z is the latent code for a given shape. Note that SDF and DDF have the same latent code for a given shape.

Training loss. The complete training loss is given as

$$\begin{aligned}\mathcal{L} = w_s\mathcal{L}_s + w_d\mathcal{L}_d + w_\sigma\mathcal{L}_\sigma \\ + w_{tv}\mathcal{L}_{tv} + w_{ts}\mathcal{L}_{ts} + w_l\mathcal{L}_l \,,\end{aligned} \tag{12}$$

where w_s, w_d, w_σ, w_{tv}, w_{ts}, and w_l are the weights for the SDF loss, DDF loss, Ray hit loss, TV regularizer, Track-SDF regularizer, and latent code regularizer respectively.

Optimization: We optimize the loss in Eq. (12) for the neural network weights, feature on the grid, and shape latent codes, Θ_s, Θ_d, M, and Z, where $Z = \{z_i | i = 1 \dots J\}$ is the set of latent codes representing all the J training shapes, Θ_d are the learnable network parameters of the DDF model f_d, Θ_s are the learnable network parameters of the SDF model f_s, and $M = (M^s, M^d)$ are the SDF and DDF feature grids where $M^s = \{M_i^s \mid i = xy, yz, zx\}$ are the SDF feature grids and $M^d = \{M_i^d \mid i = p_x p_y, \dots, r_z r_x\}$ are the DDF feature grids.

3.4. Reconstruction from Single-view Depth Maps

Our autodecoder framework allows us to work with any type of data without having to learn a new encoder for each type of data. Hence, during test time we merely need to optimize for the latent code z, while keeping the network and feature grids fixed. The highlight of our reconstruction algorithm is that it obviates the need for sphere tracing at every iteration of the optimization.

For 3D reconstruction, we assume a depth map with an object mask and a given camera pose as input. We obtain the points of intersection of the rays r from the camera with the unit sphere as p. At every iteration, we do the following:

1. with the latent code z corresponding to the current iteration, evaluate the DDF model for the directional distance, $d, \sigma = f_d((p, r), f_{pd}(p, r), z)$ from p along r

2. compute the 3D point inside the sphere predicted by the DDF model as $x = p + dr$

3. evaluate the SDF model at x as $s = f_s(x, f_{ps}(x), z)$

4. optimize for the latent code z of the object from the given depth map using the loss function,

$$\mathcal{L}_{rec} = w_S\mathcal{L}_S + w_D\mathcal{L}_D + w_l\mathcal{L}_l \,, \tag{13}$$

where $\mathcal{L}_S$ is the silhouette loss, $\mathcal{L}_D$ is the depth loss, and $\mathcal{L}_l$ is the regularizer (Eq. (11)) for learning the latent code with w_S, w_D, and w_l as their respective weights. Depth loss and silhouette loss are explained in the following.

Depth Loss $\mathcal{L}_D$. The depth loss is the error between the given depth λ_{GT} and the predicted depth λ, i.e.

$$\mathcal{L}_D(\lambda) = \|\lambda - \lambda_{GT}\|_1 \,. \tag{14}$$

We obtain the predicted depth using $\lambda u = Px$, where u are the image coordinates, P is the given projection matrix of the camera and x is the 3D point obtained in step 2 above.

Silhouette Loss $\mathcal{L}_S$. The silhouette loss is enforced as

$$\mathcal{L}_S(s) = \mathcal{L}_{S_{s_+}} + \mathcal{L}_{S_{s_-}} + \mathcal{L}_\sigma \,, \tag{15}$$

$$\mathcal{L}_{S_{s_+}}(m) = \|\langle s, m\rangle\|_1 \,, \tag{16}$$

$$\mathcal{L}_{S_{s_-}}(m) = \||\langle s, 1-m\rangle| - \tau\|_1 \,, \tag{17}$$

where $m \in \{0, 1\}$ is the given image mask, s is the predicted signed distance from step 3 above, τ is the truncation distance for the SDF model, σ is the predicted ray hit probability from the DDF model in step 1, and $\mathcal{L}_\sigma$ is the DDF silhouette loss from Eq. (8). The idea behind the loss is that where the rays hit the surface, the SDF must be as low as possible and where the rays don't, SDF must be high.

4. Experiments

In this section, we evaluate our method in different settings, reconstruction from single-view depth maps, and RGB videos. We evaluate the design choices of our method and the reconstruction algorithm in the ablation study. We show reconstruction from silhouettes and provide implementation details in supp. mat.

4.1. Reconstruction from Single-view Depth Maps

Our DDF model predicts distance to the surface of a shape given the latent code representing the shape, the ray origin, and the ray direction. Therefore, it can be used to replace the expensive sphere tracing algorithm during inverse rendering with learned SDF models.

We evaluate this advantage of our method by reconstructing the 3D shape given a depth map with a camera pose. We render a depth image with the given camera pose from our network and optimize for the latent code as discussed in Sec 3.4. We test our trained models on the first 200 test instances of different classes of ShapeNet shapes – airplanes, cars, chairs, lamps, sofas, and tables. For the images, we obtain the camera parameters of the first image of the rendered ShapeNet dataset from 3D-R2N2 [11] and render a depth map with the same resolution, 137 × 137. For comparisons, we run the official implementations of IF-Net [10] and DIST [24]. With IF-Nets, we complete partial point clouds obtained by un-projecting the depth maps.

Qualitative Results: We show qualitative results in Fig. 3. It can be seen that our method can reconstruct 3D shapes accurately given a single view depth image.

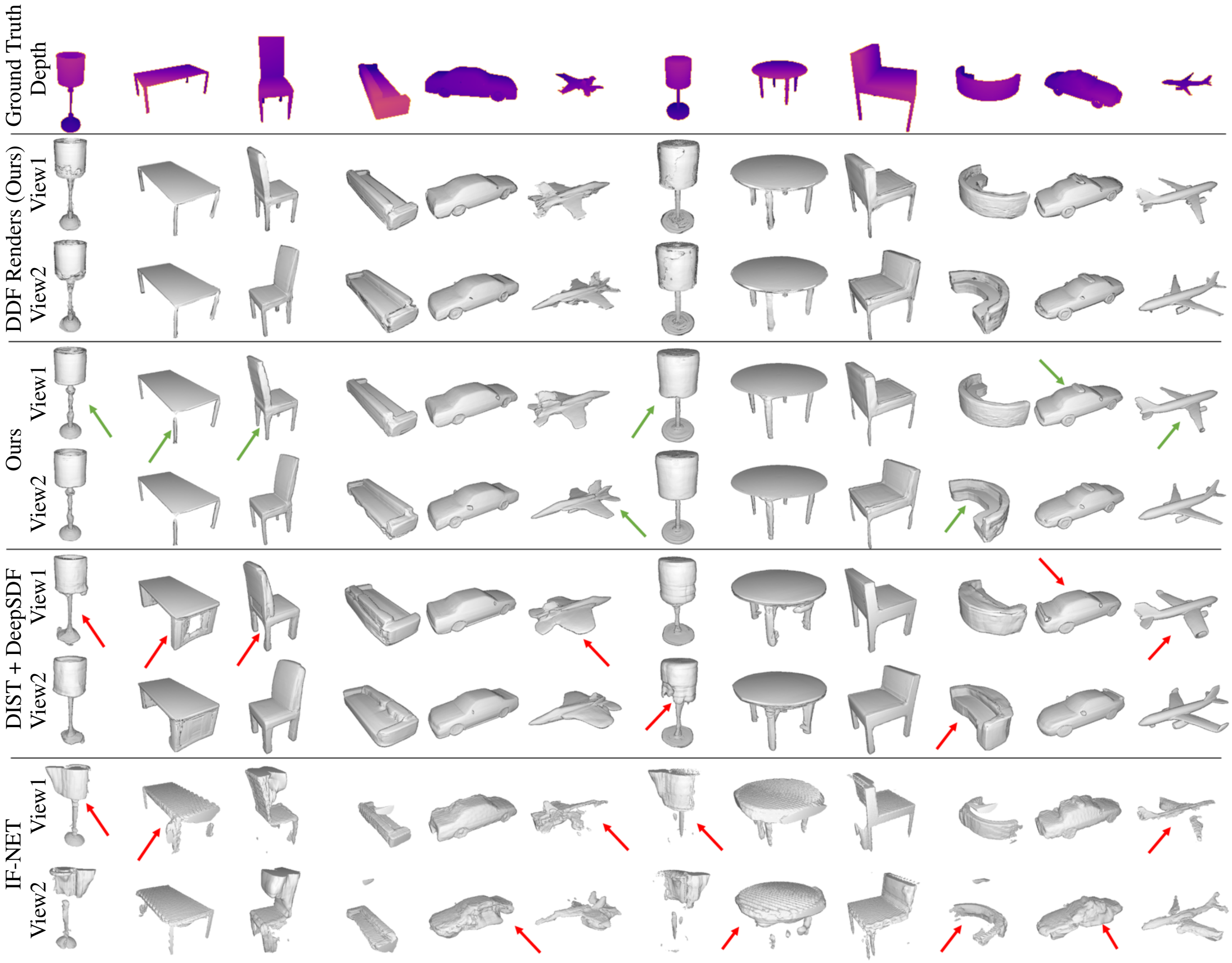

Figure 3. 3D shapes reconstructed from a given depth map. Each column shows reconstruction results for different shapes. Top row: Given depth map. Upper-middle rows: views rendered with 1 forward pass from our DDF model. Middle rows: views of 3D shapes reconstructed by our SDF model. Lower-middle rows: views of 3D shape reconstructed using DIST [24]. Last rows: views of 3D shapes reconstructed using IF-NET [10]. Our method outperforms existing methods, as it can, for example, better model fine-scale details (see e.g. the legs of the tables or chairs, or the geometry of the airplanes)

Given our feature-based network architecture, and our algorithm, our reconstructions are more detailed compared with DIST [24]. Further, our method is about $15.5\times$ faster per iteration on average (see Tab. 1). We compare our reconstruction with those of IF-NET [10], a state-of-the-art encoder-based neural implicit representation. While IF-NET leads to plausible reconstructions in the observed locations, where there are valid depth maps, it does not complete unobserved shapes, as shown in the last two rows of Fig. 3.

Quantitative Results: We show the quantitative results in Tab. 1. We use the chamfer distance defined in DeepSDF [32] to compute the accuracy. Please see supp. mat. for more details. The results are consistent with qualitative ones, as our method can fit well to the given depth maps and obtain more plausible reconstructions compared to DIST [24]. Moreover, we outperform DIST in all the classes while being $15.5\times$ faster. Further, as IF-NET [10] does not complete the shape in unobserved areas, we significantly outperform IF-NET quantitatively and qualitatively.

Model Evaluation: We compare our model with the state-of-the-art directional distance representation methods, PRIF [16], Depth-LFN [40], and NeuralODF [19] on reconstruction from depth maps. PRIF predicts the directional distance from the perpendicular foot of a camera ray. LFN predicts RGB given Plücker coordinates and ray direction. NeuralODF predicts distance to the surface, and ray hit prediction, given a point and direction in 3D space. As predicting DDF everywhere in space is a harder task, for a fair comparison, we restrict the input to NeuralODF to the unit sphere. Further, we found that predicting ray hit from the final layer results in higher accuracy for NeuralODF; hence, we use this model. For a fair comparison, we train

Method	Ours SDF	Ours DDF	DIST Our SDF	DIST DeepSDF	IF-NET	Ours	DIST DeepSDF	Ours DDF	Ours	DeepSDF
Metric	1000× CD ↓					ms/iteration ↓		ms / 256 × 256 frame ↓		
Car	0.55	**0.38**	0.60	0.61	4.09	**17**	282	62	**23**	132
Chair	0.74	**0.64**	1.96	1.92	5.45	**18**	236	58	**23**	118
Lamp	**2.50**	4.81	6.39	7.34	6.05	**15**	281	65	**22**	120
Plane	**0.18**	0.32	0.69	0.94	2.08	**15**	231	56	**22**	116
Sofa	0.77	**0.67**	1.64	1.81	9.43	**18**	238	61	**21**	113
Table	1.28	**0.83**	3.02	2.79	4.67	**18**	283	56	**23**	126

Table 1. Quantitative results of comparisons of our method with DIST [24] and IF-NET [10] (middle-left columns). Our method outperforms DIST and IF-NET in all the shape classes, showing that our models and our depth-fitting algorithm lead to better reconstructions. DIST performs marginally better in most classes with our SDF model compared with DeepSDF's, showing that the majority of improvement is due to our method and not the SDF model. We compare the time per optimization step with DIST, where ours is on an average 15.5× faster than DIST, as shown in the middle-right columns. Finally, in the right-most columns, we show that we can render 256 × 256 images in real-time with just one forward pass using our DDF representation. Rendering times include time for normal computation.

LFNs with just the depth and ray hit supervision so that the model can predict dense depth. We train the three models on the first 256 shapes of the training set and test them on the first 64 shapes of the test set, of each shape class. We follow this split to closely replicate the number of train and test shapes in PRIF. We evaluate the methods quantitatively using chamfer distance between the predicted and ground truth shapes.

We optimize for latent code using the algorithm in Sec. 4.1 during inference with our method. For other methods, we optimize for the latent code with the losses from Eqs. (14) and (8). Our method marries the best of both the models, view-consistent geometric details from the SDF model and 1 forward pass rendering from the DDF model. Owing to this, our model outperforms the state-of-the-art DDF models by a large margin quantitatively, as seen in Tab. 2, and qualitatively (see supp. mat. Fig. 3).

4.2. Ablations

In this experiment, we evaluate the design choices in our method. We train the models with the first 256 shapes from the training split and test on the first 64 test shapes of the sofas class from the ShapeNet dataset. We report 1000× the chamfers distance between reconstructions and ground truth in Tab. 3.

Reconstruction Algorithm: We train our models as described in Sec. 3.3. We reconstruct 3D shapes from depth maps (see Sec. 4.1) using our learned models. We ablate the components of losses introduced in Sec. 3.4. Quantitative results are shown in Tab. 3 and qualitative results are shown in Fig. 4. *(DIST)* We run the single view reconstruction algorithm with DIST on our trained model. As DDF and SDF share a latent space, we can also evaluate DDF. Quantitatively DIST underperforms as we have also seen in Tab. 1. *(wo $\mathcal{L}_S$)* Without any silhouette losses, our method performs poorly, showing the impact of silhouette

Class	Ours SDF	Ours DDF	PRIF	LFN	Neural ODF
	1000× CD ↓ (Mean)				
Cars	0.71	**0.57**	0.85	0.66	0.83
Chairs	1.30	**1.15**	1.83	1.56	1.78
Lamps	**4.98**	6.52	9.22	DNC	7.68
Planes	**0.26**	0.51	0.78	0.59	0.68
Sofas	0.78	**0.78**	1.57	1.08	1.57
Tables	1.49	**1.40**	2.30	1.63	2.60

Table 2. Quantitative comparison of our model with PRIF [16], Depth-LFNs [40], and NeuralODF [19] on reconstruction from depth maps. We train the models on different classes of shapes and utilize them in the autodecoder framework to optimize for shapes from a given depth map during inference. We report the mean chamfer distance between the reconstructed and ground truth shapes. Our model outperforms competitive DDF models in all the classes. While our model maintains the salient features of DDF, such as 1 forward pass rendering, it can also represent view-consistent geometric details using the SDF model. (Depth-LFN did not converge for lamps class with 256 shapes.)

losses on reconstruction quality. *(wo $\mathcal{L}_{s_+}$)* Without the foreground SDF silhouette loss, the background silhouette loss overpowers the reconstruction and leads to missing regions, as shown in Fig. 4, 4$^{\text{th}}$ column. *(wo $\mathcal{L}_{s_-}$)* Without a background silhouette loss, the SDF reconstruction can be larger than the masks, leading to poor accuracy. *(wo $\mathcal{L}_{s_+} + \mathcal{L}_{s_-}$)* Without any SDF silhouette losses, the SDF reconstructions miss structures leading to poor accuracy. *(wo $\mathcal{L}_\sigma$)* Without the DDF silhouette loss the DDF renders are inconsistent with SDF reconstructions. Since we use DDF for rendering the SDF, this also leads to a decrease in the reconstruction quality of the SDF.

Model: We perform ablation studies on our models and losses presented in Sec 3. Qualitative results are shown in Fig. 4, and quantitative results are shown in Tab. 3. *(wo shared latent space)* Independent optimization for latent

	Reconstruction Algorithm							Model and Losses					
	DIST	**wo $\mathcal{L}_S$ Eq. (15)**	**wo $\mathcal{L}_{S_{s_+}}$ Eq. (16)**	**wo $\mathcal{L}_{S_{s_-}}$ Eq. (17)**	**wo $\mathcal{L}_{S_{s_+}}$ $+\mathcal{L}_{S_{s_-}}$**	**wo $\mathcal{L}_\sigma$ Eq. (8)**	**Ours**	**wo sh. lats.**	**w $\mathcal{L}_{ts}$ to SDF**	**wo $\mathcal{L}_{ts}$ Eq. (10)**	**wo $\mathcal{L}_{tv}$ Eq. (9)**	**wo DDF σ preds.**	**w SDF 3D Grid**
SDF	1.56	2.74	7.92	1.38	2.62	0.96	0.78	0.98	1.48	0.93	0.77	0.96	0.87
DDF	1.72	1.37	0.89	1.13	1.00	1.00	0.78	0.95	1.49	0.79	1.26	1.00	0.84

Table 3. Quantitative results of ablation study. Ablations on reconstruction algorithm (left columns), and ablations on models (right columns). *Reconstruction algorithm*: from left to right, DIST algorithm with our model, without any silhouette losses $\mathcal{L}_S$, without foreground SDF silhouette loss $\mathcal{L}_{S_{s_+}}$, without background SDF silhouette loss $\mathcal{L}_{S_{s_-}}$, without any SDF silhouette losses, without the DDF silhouette loss $\mathcal{L}_\sigma$, and ours. *Model*: From left to right: without a shared latent space for SDF and DDF models, with gradients from Track-SDF regularizer $\mathcal{L}_{ts}$ to SDF model, without Track-SDF regularizer $\mathcal{L}_{ts}$, without the TV regularizer $\mathcal{L}_{tv}$, without DDF ray hit predictions σ, and with a 3D feature grid instead of a 2D grid for SDF model. Middle: Our proposed algorithm.

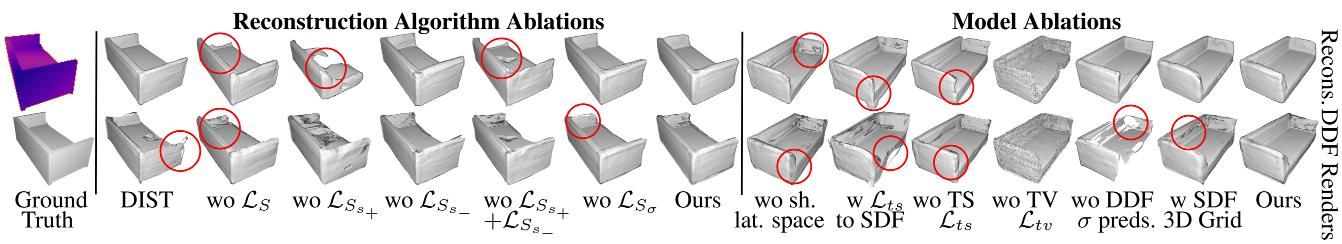

Figure 4. Qualitative results of ablation study. Left: Ground truth depth (top) and geometry (bottom). Middle: *Reconstruction algorithm*: (left to right) with DIST and our SDF, without any silhouette loss $\mathcal{L}_S$, without foreground SDF silhouette loss $\mathcal{L}_{S_{s_+}}$, without background SDF silhouette loss $\mathcal{L}_{S_{s_-}}$, without any SDF silhouette loss $\mathcal{L}_{S_{s_+}} + \mathcal{L}_{S_{s_-}}$, without DDF silhouette loss $\mathcal{L}_\sigma$, and ours. Right: *Model*: (left to right) without a shared latent space between DDF and SDF, track SDF $\mathcal{L}_{ts}$ loss also trains SDF model, without track SDF $\mathcal{L}_{ts}$, without TV regularizer $\mathcal{L}_{tv}$, without DDF ray hit predictions σ, with a 3D feature grid for SDF instead of 2D grids, and ours. Our design choices result in fast and accurate reconstructions.

codes does not let the DDF and SDF models change together, leading to inaccuracies between the reconstructions as shown in Fig. 4. *(with $\mathcal{L}_{ts}$ to SDF)* When the SDF model is allowed to train with the gradients from the Track-SDF regularizer (Eq. (10)), the reconstructions are bad as the SDF model can incorrectly learn to place a surface at DDF's predictions during training. *(wo TS $\mathcal{L}_{ts}$)* As the DDF is unconstrained the accuracy increases, however, since the DDF model does not predict close to the SDF surface, the reconstruction quality of SDFs is lower. *(wo TV $\mathcal{L}_{tv}$)* Without the TV regularizer, the reconstructions are noisy around the surface but sharp for SDF hence leading to high accuracy whereas for DDF this noise leads to higher error as the predicted distances are noisy. *(wo DDF σ preds.)* without DDF ray hit predictions, the DDF renders are incomplete, as we rely on SDF value for the ray hit predictions, leading to poor accuracy. *(w 3D Grid SDF)* As a 3D feature grid with the same number of parameters as a 2D feature grid is of lower resolution, the model performs worse with the same number of features in the grid, leading to smoother reconstructions. Our model performs the best with all the design choices.

5. Future Work

Overall, our method has shown significant improvement in terms of speed while reconstructing from different inputs such as depth maps, silhouettes (supp. mat.), and videos. While our results show consistent renders from different views using the DDF model, 3D inconsistency is a persistent problem with directional representations. The feature grid-based representation achieves high-quality results, however, better regularizers than the TV (Eq. 9) that allow for discontinuities while suppressing noise could help improve the representational capacity of DDF models.

6. Conclusion

We presented a novel 3D representation, DDF, that enables us to replace sphere tracing for rendering SDFs with just 1 network evaluation per camera ray. Based on the learned DDF and SDF models, we introduced a fast algorithm (FIRe) to reconstruct shapes with our learned models from depth maps. We experimentally showed that FIRe can reconstruct high-quality 3D shapes given a depth map or a video while achieving an order of magnitude speedup of the optimization algorithm. We believe that the proposed method can play a crucial role in working with learned implicit scene representations for various applications. In order to stimulate follow-up work we plan to make our code publicly available.

7. Acknowledgements

This work has been supported by the ERC Advanced Grant SIMULACRON (884679), and the BMBF-funded Munich Center for Machine Learning. Thanks to Mariia Gladkova for helping set up the camera pipelines.

References

[1] Tristan Aumentado-Armstrong, Stavros Tsogkas, Sven Dickinson, and Allan D. Jepson. Representing 3d shapes with probabilistic directed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19343–19354, June 2022. 2

[2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999. 2

[3] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision*, 2016. 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 *European Conference on Computer Vision*, pages 608–625. Springer, 2020. 2

[5] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv*, 2020. 3

[6] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022. 2, 3

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015. 4

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 3

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[10] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020. 2, 5, 6, 7

[11] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 5

[12] Daniel Cremers. Dynamical statistical shape priors for level set-based tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1262–1273, 2006. 2

[13] Dawson-Haggerty et al. trimesh. 4

[14] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *arxiv:2205.15285*, 2022. 4

[15] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. *IEEE TIP*, 7(3):336–344, Mar. 1998. 2

[16] Brandon Y. Feng, Yinda Zhang, Danhang Tang, Ruofei Du, and Amitabh Varshney. Prif: Primary ray-based implicit function. In *Proceedings of the European Conference on Computer Vision (ECCV)*, October 2022. 2, 6, 7

[17] Justin Johnson Georgia Gkioxari, Jitendra Malik. Mesh r-cnn. *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 2

[18] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, Dec. 1996. 1

[19] Trevor Houchens, Cheng-You Lu, Shivam Duggal, Rao Fu, and Srinath Sridhar. Neuralodf: Learning omnidirectional distance fields for 3d shape representation, 2022. 2, 6, 7

[20] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. 2

[21] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 57–66, New York, NY, USA, 2001. Association for Computing Machinery. 2

[22] Timo Kohlberger, Daniel Cremers, Mikaël Rousson, Ramamani Ramaraj, and Gareth Funka-Lea. 4d shape priors for a level set segmentation of the left myocardium in spect sequences. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 92–100. Springer, 2006. 2

[23] Michael E Leventon, W Eric L Grimson, and Olivier Faugeras. Statistical shape influence in geodesic active contours. In *5th IEEE EMBS International Summer School on Biomedical Imaging, 2002.*, pages 8–pp. IEEE, 2002. 2

[24] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 5, 6, 7

[25] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics*, Oct. 2015. 2

[26] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[27] 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. 2, 4

[28] 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. 2

[29] Shigeru Muraki. Volumetric shape description of range data using "blobby model". In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 227–235, 1991. 2

[30] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021. 2

[31] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2, 4, 6

[33] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 1, 2

[34] 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. 2

[35] Antonio Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973. 2

[36] Shunsuke Saito, Liwen Hu, Chongyang Ma, Hikaru Ibayashi, Linjie Luo, and Hao Li. 3d hair synthesis using volumetric variational autoencoders. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018. 2

[37] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019. 2

[38] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020. 2

[39] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 3

[40] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Proc. NeurIPS*, 2021. 2, 6, 7

[41] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[42] C Sommer, L Sang, D Schubert, and D Cremers. Gradient-sdf: A semi-implicit surface representation for 3d reconstruction. *arXiv preprint*, 2021. 2

[43] Jürgen Sturm, Erik Bylow, Fredrik Kahl, and Daniel Cremers. Copyme3d: Scanning and printing persons in 3d. In *German Conference on Pattern Recognition*, pages 405–414. Springer, 2013. 2

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

[45] Ayush Tewari, Michael Zollöfer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Theobalt Christian. MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 2

[46] Garvita Tiwari, Dimitrije Antic, Jan Eric Lenssen, Nikolaos Sarafianos, Tony Tung, and Gerard Pons-Moll. Pose-ndf: Modeling human pose manifolds with neural distance fields. In *European Conference on Computer Vision (ECCV)*, October 2022. 2

[47] Garvita Tiwari, Nikolaos Sarafianos, Tony Tung, and Gerard Pons-Moll. Neural-gif: Neural generalized implicit functions for animating people in clothing. In *International Conference on Computer Vision (ICCV)*, October 2021. 2

[48] Eno Toeppe, Claudia Nieuwenhuis, and Daniel Cremers. Volume constraints for single view reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 2

[49] Rui Wang, Nan Yang, Joerg Stueckler, and Daniel Cremers. Directshape: Photometric alignment of shape priors for visual vehicle pose and shape estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020. 2

[50] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[51] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 492–502. Curran Associates, Inc., 2019. 2

[52] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2

[53] Zhenzhang Ye, Tarun Yenamandra, Florian Bernard, and Daniel Cremers. Joint deep multi-graph matching and 3d geometry learning from inhomogeneous 2d image collections. In *AAAI*, 2022. 2

[54] Tarun Yenamandra, Ayush Tewari, Florian Bernard, Hans-Peter Seidel, Mohamed Elgharib, Daniel Cremers, and Christian Theobalt. i3dmm: Deep implicit 3d morphable model of human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12803–12813, 2021. 1, 2

[55] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2020. 2

[56] Ehsan Zobeidi and Nikolay Atanasov. A deep signed directional distance function for object shape representation. *arXiv preprint arXiv:2107.11024*, 2021. 2