

Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision

Michael Niemeyer^{1,2} Lars Mescheder^{1,2,3†} Michael Oechsle^{1,2,4} Andreas Geiger^{1,2}

¹Max Planck Institute for Intelligent Systems, Tübingen ²University of Tübingen

³Amazon, Tübingen ⁴ETAS GmbH, Bosch Group, Stuttgart

{firstname.lastname}@tue.mpg.de

Abstract

Learning-based 3D reconstruction methods have shown impressive results. However, most methods require 3D supervision which is often hard to obtain for real-world datasets. Recently, several works have proposed differentiable rendering techniques to train reconstruction models from RGB images. Unfortunately, these approaches are currently restricted to voxel- and mesh-based representations, suffering from discretization or low resolution. In this work, we propose a differentiable rendering formulation for implicit shape and texture representations. Implicit representations have recently gained popularity as they represent shape and texture continuously. Our key insight is that depth gradients can be derived analytically using the concept of implicit differentiation. This allows us to learn implicit shape and texture representations directly from RGB images. We experimentally show that our single-view reconstructions rival those learned with full 3D supervision. Moreover, we find that our method can be used for multi-view 3D reconstruction, directly resulting in watertight meshes.

1. Introduction

In recent years, learning-based 3D reconstruction approaches have achieved impressive results [12, 13, 17, 24, 41, 48, 49, 56, 64, 80]. By using rich prior knowledge obtained during the training process, they are able to infer a 3D model from as little as a single image. However, most learning-based methods are restricted to synthetic data, mainly because they require accurate 3D ground truth models as supervision for training.

To overcome this barrier, recent works have investigated approaches that require only 2D supervision in the form of depth maps or multi-view images. Most existing approaches achieve this by modifying the rendering process

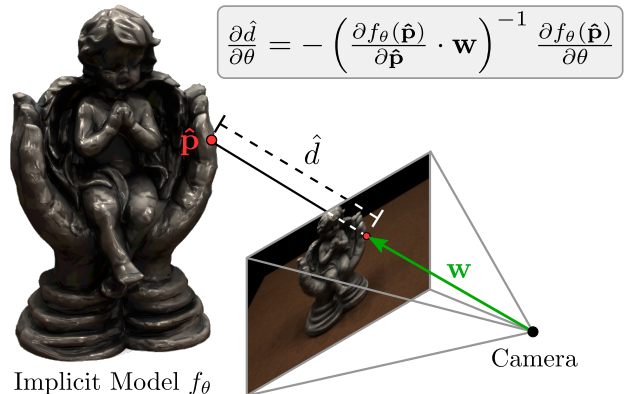


Figure 1: **Overview.** We show that volumetric rendering is inherently differentiable for implicit shape and texture representations. Using an analytic expression for the gradient of the depth $\frac{\partial \hat{d}}{\partial \theta}$ wrt. the network parameters θ , we learn implicit 3D representations f_{θ} from 2D images.

to make it differentiable [4, 11, 15, 21, 33, 36, 43, 44, 47, 50, 58, 59, 62, 75, 76, 79, 88]. While yielding compelling results, they are restricted to specific 3D representations (e.g. voxels or meshes) that suffer from discretization artifacts and the computational cost limits them to small resolutions or deforming a fixed template mesh. At the same time, implicit representations [12, 48, 56] for shape and texture [54, 66] have been proposed which do not require discretization during training and have a constant memory footprint. However, existing approaches using implicit representations require 3D ground truth for training and it remains unclear how to learn implicit representations from image data alone.

Contribution: In this work, we introduce *Differentiable Volumetric Rendering (DVR)*. Our key insight is that we can derive analytic gradients for the predicted depth map with respect to the network parameters of the implicit shape and texture representation (see Fig. 1). This insight enables us to design a differentiable renderer for implicit shape and texture representations and allows us to learn these representations solely from multi-view images and object masks.

[†]This work was done prior to joining Amazon.

Since our method does not have to store volumetric data in the forward pass, its memory footprint is independent of the sampling accuracy of the depth prediction step. We show that our formulation can be used for various tasks such as single- and multi-view reconstruction, and works with synthetic and real data. In contrast to [54], we do not need to condition the texture representation on the geometry, but learn a *single model* with shared parameters that represents both geometry and texture. Our code and data are provided at https://github.com/autonomousvision/differentiable_volumetric_rendering.

2. Related Work

3D Representations: Learning-based 3D reconstruction approaches can be categorized wrt. the representation they use as voxel-based [8, 13, 19, 61, 64, 73, 82, 83], point-based [2, 17, 31, 40, 77, 85], mesh-based [24, 32, 41, 55, 80], or implicit representations [3, 12, 22, 30, 48, 49, 56, 66, 81].

Voxels can be easily processed by standard deep learning architectures, but even when operating on sparse data structures [23, 64, 74], they are limited to relatively small resolution. While point-based approaches [2, 17, 40, 77, 85] are more memory-efficient, they require intensive post-processing because of missing connectivity information. Most mesh-based methods do not perform post-processing, but they often require a deformable template mesh [80] or represent geometry as a collection of 3D patches [24] which leads to self-intersections and non-watertight meshes.

To mitigate these problems, implicit representations have gained popularity [3, 12, 22, 30, 48, 49, 53, 54, 56, 66, 81]. By describing 3D geometry and texture implicitly, e.g., as the decision boundary of a binary classifier [12, 48], they do not discretize space and have a fixed memory footprint.

In this work, we show that the volumetric rendering step for implicit representations is inherently differentiable. In contrast to previous works, this allows us to learn implicit 3D shape and texture representations using 2D supervision.

3D Reconstruction: Recovering 3D information which is lost during the image capturing process is one of the long-standing goals of computer vision [25]. Classic multi-view stereo (MVS) methods [5–7, 20, 37, 60, 68–70] usually match features between neighboring views [5, 20, 68] or reconstruct the 3D shape in a voxel grid [6, 7, 37, 60, 70]. While the former methods produce depth maps as output which have to be fused in a lossy post-processing step, e.g., using volumetric fusion [14], the latter approaches are limited by the excessive memory requirements of 3D voxel grids. In contrast to these highly engineered approaches, our generic method directly outputs a consistent representation in 3D space which can be easily converted into a watertight mesh while having a constant memory footprint.

Recently, learning-based approaches [16, 29, 39, 58, 63,

86, 87] have been proposed that either learn to match image features [39], refine or fuse depth maps [16, 63], optimize parts of the classical MVS pipeline [57], or replace the entire MVS pipeline with neural networks that are trained end-to-end [29, 86, 87]. In contrast to these learning-based approaches, our method can be supervised from 2D images alone and outputs a consistent 3D representation.

Differentiable Rendering: We focus on methods that learn 3D geometry via differentiable rendering in contrast to recent neural rendering approaches [42, 51, 52, 71] which synthesize high-quality novel views but do not infer the 3D object. They can again be categorized by the underlying representation of 3D geometry that they use.

Loper et al. [47] propose OpenDR which approximates the backward pass of the traditional mesh-based graphics pipeline and has inspired several follow-up works [11, 21, 27, 28, 33, 44, 88]. Liu et al. [44] replace the rasterization step with a soft version to make it differentiable. While yielding compelling results in reconstruction tasks, these approaches require a deformable template mesh for training, restricting the topology of the output.

Another line of work operates on voxel grids [46, 50, 57, 79]. Paschalidou et al. [57] and Tulsiani et al. [79] propose a probabilistic ray potential formulation. While providing a solid mathematical framework, all intermediate evaluations need to be saved for backpropagation, restricting these approaches to relatively small-resolution voxel grids.

Liu et al. [45] propose to infer implicit representations from multi-view silhouettes by performing max-pooling over the intersections of rays with a sparse number of supporting regions. In contrast, we use texture information enabling us to improve over the visual hull and to reconstruct concave shapes. Sitzmann et al. [72] infer implicit scene representations from RGB images via an LSTM-based differentiable renderer. While producing high-quality renderings, the geometry cannot be extracted directly and intermediate results need to be stored for computing gradients. In contrast, we show that volumetric rendering is *inherently differentiable* for implicit representations. Thus, no intermediate results need to be saved for the backward pass.

3. Method

In this section, we describe our Differentiable Volumetric Rendering (DVR) approach. We first define the implicit neural representation which we use for representing 3D shape and texture. Next, we provide a formal description of DVR and all relevant implementation details. An overview of our approach is provided in Fig. 2.

3.1. Shape and Texture Representation

Shape: In contrast to discrete voxel- and point-based representations, we represent the 3D shape of an object *implicitly*

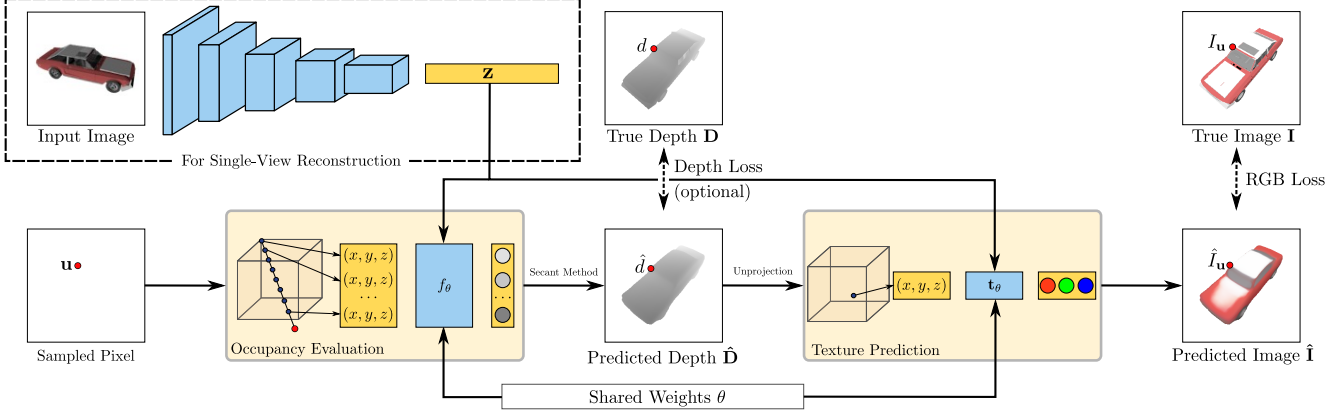


Figure 2: **Differentiable Volumetric Rendering.** We first predict the surface depth \hat{d} by performing occupancy evaluations for a given camera matrix. To this end, we project sampled pixel \mathbf{u} to 3D and evaluate the occupancy network at fixed steps on the ray cast from the camera origin towards this point. We then unproject the surface depth into 3D and evaluate the texture field at the given 3D location. The resulting 2D rendering $\hat{\mathbf{I}}$ can be compared to the ground truth image. When we also have access to ground truth depth maps, we can define a loss directly on the predicted surface depth. We can make our model conditional by incorporating an additional image encoder that predicts a global descriptor \mathbf{z} of both shape and texture.

using the occupancy network introduced in [48]:

$$f_{\theta} : \mathbb{R}^3 \times \mathcal{Z} \rightarrow [0, 1] \quad (1)$$

An occupancy network $f_{\theta}(\mathbf{p}, \mathbf{z})$ assigns a probability of occupancy to every point $\mathbf{p} \in \mathbb{R}^3$ in 3D space. For the task of single-view reconstruction, we process the input image with an encoder network $\mathbf{g}_{\theta}(\cdot)$ and use the output $\mathbf{z} \in \mathcal{Z}$ to condition f_{θ} . The 3D surface of an object is implicitly determined by the level set $f_{\theta} = \tau$ for a threshold parameter $\tau \in [0, 1]$ and can be extracted at arbitrary resolution using isosurface extraction techniques.¹

Texture: Similarly, we can describe the texture of a 3D object using a texture field [54]

$$\mathbf{t}_{\theta} : \mathbb{R}^3 \times \mathcal{Z} \rightarrow \mathbb{R}^3 \quad (2)$$

which regresses an RGB color value for every point $\mathbf{p} \in \mathbb{R}^3$ in 3D space. Again, \mathbf{t}_{θ} can be conditioned on a latent embedding \mathbf{z} of the object. The texture of an object is given by the values of \mathbf{t}_{θ} on the object’s surface ($f_{\theta} = \tau$). In this work, we implement f_{θ} and \mathbf{t}_{θ} as a single neural network with two shallow heads.

Supervision: Recent works [12, 48, 54, 56, 66] have shown that it is possible to learn f_{θ} and \mathbf{t}_{θ} with 3D supervision (i.e., ground truth 3D models). However, ground truth 3D data is often very expensive or even impossible to obtain for real-world datasets. In the next section, we introduce DVR, an alternative approach that enables us to learn both f_{θ} and \mathbf{t}_{θ} from 2D images alone. For clarity, we drop the condition variable \mathbf{z} in the following.

¹See Mescheder et al. [48] for details.

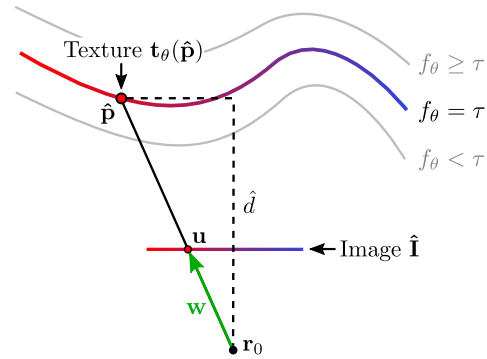


Figure 3: **Notation.** To render an object from the occupancy network f_{θ} and texture field \mathbf{t}_{θ} , we cast a ray with direction \mathbf{w} through a pixel \mathbf{u} and determine the intersection point $\hat{\mathbf{p}}$ with the isosurface $f_{\theta}(\mathbf{p}) = \tau$. Afterwards, we evaluate the texture field \mathbf{t}_{θ} at $\hat{\mathbf{p}}$ to obtain the color prediction $\hat{\mathbf{I}}_{\mathbf{u}}$ at \mathbf{u} .

3.2. Differentiable Volumetric Rendering

Our goal is to learn f_{θ} and \mathbf{t}_{θ} from 2D image observations. Consider a single image observation. We define a photometric reconstruction loss

$$\mathcal{L}(\hat{\mathbf{I}}, \mathbf{I}) = \sum_{\mathbf{u}} \|\hat{\mathbf{I}}_{\mathbf{u}} - \mathbf{I}_{\mathbf{u}}\| \quad (3)$$

which we aim to optimize. Here, \mathbf{I} denotes the observed image and $\hat{\mathbf{I}}$ is the image rendered by our implicit model.² Moreover, $\mathbf{I}_{\mathbf{u}}$ denotes the RGB value of the observation \mathbf{I}

²Note that the rendered image $\hat{\mathbf{I}}$ depends on θ through f_{θ} and \mathbf{t}_{θ} . We have dropped this dependency here to avoid clutter in the notation.

at pixel \mathbf{u} and $\|\cdot\|$ is a (robust) photo-consistency measure such as the ℓ_1 -norm. To minimize the reconstruction loss \mathcal{L} wrt. the network parameters θ using gradient-based optimization techniques, we must be able to (i) **render** $\hat{\mathbf{I}}$ given f_θ and \mathbf{t}_θ and (ii) compute **gradients** of \mathcal{L} wrt. the network parameters θ . Our core contribution is to provide solutions to both problems, leading to an efficient algorithm for learning implicit 3D representations from 2D images.

Rendering: For a camera located at \mathbf{r}_0 we can predict the color $\hat{\mathbf{I}}_{\mathbf{u}}$ at pixel \mathbf{u} by casting a ray from \mathbf{r}_0 through \mathbf{u} and determining the first point of intersection $\hat{\mathbf{p}}$ with the isosurface $\{\mathbf{p} \in \mathbb{R}^3 | f_\theta(\mathbf{p}) = \tau\}$ as illustrated in Fig. 3. The color value $\hat{\mathbf{I}}_{\mathbf{u}}$ is then given by $\hat{\mathbf{I}}_{\mathbf{u}} = \mathbf{t}_\theta(\hat{\mathbf{p}})$. We refer the reader to Section 3.3 for details on the ray casting process.

Gradients: To obtain gradients of \mathcal{L} with respect to θ , we first use the multivariate chain rule:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{\mathbf{u}} \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{I}}_{\mathbf{u}}} \cdot \frac{\partial \hat{\mathbf{I}}_{\mathbf{u}}}{\partial \theta} \quad (4)$$

Here, $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ denotes the Jacobian matrix for a vector-valued function \mathbf{g} with vector-valued argument \mathbf{x} and \cdot indicates matrix multiplication. By exploiting $\hat{\mathbf{I}}_{\mathbf{u}} = \mathbf{t}_\theta(\hat{\mathbf{p}})$, we obtain

$$\frac{\partial \hat{\mathbf{I}}_{\mathbf{u}}}{\partial \theta} = \frac{d\mathbf{t}_\theta(\hat{\mathbf{p}})}{d\theta} = \frac{\partial \mathbf{t}_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial \mathbf{t}_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \frac{\partial \hat{\mathbf{p}}}{\partial \theta} \quad (5)$$

since both \mathbf{t}_θ as well as $\hat{\mathbf{p}}$ depend on θ . Because $\hat{\mathbf{p}}$ is defined implicitly, calculating $\frac{\partial \hat{\mathbf{p}}}{\partial \theta}$ is non-trivial. We first exploit that $\hat{\mathbf{p}}$ lies on the ray from \mathbf{r}_0 through \mathbf{u} . For any pixel \mathbf{u} , this ray can be described by $\mathbf{r}(d) = \mathbf{r}_0 + d\mathbf{w}$ where \mathbf{w} is the vector connecting \mathbf{r}_0 and \mathbf{u} (see Fig. 3). Since $\hat{\mathbf{p}}$ must lie on \mathbf{r} , there exists a depth value \hat{d} , such that $\hat{\mathbf{p}} = \mathbf{r}(\hat{d})$. We call \hat{d} the *surface depth*. This enables us to rewrite $\frac{\partial \hat{\mathbf{p}}}{\partial \theta}$ as

$$\frac{\partial \hat{\mathbf{p}}}{\partial \theta} = \frac{\partial \mathbf{r}(\hat{d})}{\partial \theta} = \mathbf{w} \frac{\partial \hat{d}}{\partial \theta} \quad (6)$$

For computing the gradient of the surface depth \hat{d} with respect to θ we exploit *implicit differentiation* [3, 65]. Differentiating $f_\theta(\hat{\mathbf{p}}) = \tau$ on both sides wrt. θ , we obtain:

$$\begin{aligned} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \frac{\partial \hat{\mathbf{p}}}{\partial \theta} &= 0 \\ \Leftrightarrow \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \frac{\partial \hat{d}}{\partial \theta} &= 0 \end{aligned} \quad (7)$$

Rearranging (7), we arrive at the following closed form expression for the gradient of the surface depth \hat{d} :

$$\frac{\partial \hat{d}}{\partial \theta} = - \left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \right)^{-1} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \quad (8)$$

We remark that calculating the gradient of the surface depth \hat{d} wrt. the network parameters θ only involves calculating

the gradient of f_θ at $\hat{\mathbf{p}}$ wrt. the network parameters θ and the surface point $\hat{\mathbf{p}}$. Thus, in contrast to voxel-based approaches [58, 79], we do not have to store intermediate results (e.g., volumetric data) for computing the gradient of the loss wrt. the parameters, resulting in a memory-efficient algorithm. In the next section, we describe our implementation of DVR which makes use of reverse-mode automatic differentiation to compute the full gradient (4).

3.3. Implementation

To use automatic differentiation, we have to implement the forward and backward pass for the surface depth prediction step $\theta \rightarrow \hat{d}$. In the following, we describe how both passes are implemented. For more details, we refer the reader to the supplementary material.

Forward Pass: As visualized in Fig. 3, we can determine \hat{d} by finding the first occupancy change on the ray \mathbf{r} . To detect an occupancy change, we evaluate the occupancy network $f_\theta(\cdot)$ at n equally-spaced samples on the ray $\{\mathbf{p}_j^{\text{ray}}\}_{j=1}^n$. Using a step size of Δs , we can express the coordinates of these point in world-coordinates as

$$\mathbf{p}_j^{\text{ray}} = \mathbf{r}(j\Delta s + s_0) \quad (9)$$

where s_0 determines the closest possible surface point. We first find the smallest j for which f_θ changes from free space ($f_\theta < \tau$) to occupied space ($f_\theta \geq \tau$):

$$j = \underset{j'}{\operatorname{argmin}} \left(f_\theta(\mathbf{p}_{j'+1}^{\text{ray}}) \geq \tau > f_\theta(\mathbf{p}_{j'}^{\text{ray}}) \right) \quad (10)$$

We obtain an approximation to the surface depth \hat{d} by applying the iterative secant method to the interval $[j\Delta s + s_0, (j+1)\Delta s + s_0]$. In practice, we compute the surface depth for a batch of N_p points in parallel. It is important to note that we do not need to unroll the forward pass or store any intermediate results as we exploit implicit differentiation to directly obtain the gradient of \hat{d} wrt. θ .

Backward Pass: The input to the backward pass is the gradient $\lambda = \frac{\partial \mathcal{L}}{\partial \hat{d}}$ of the loss wrt. a single surface depth prediction. The output of the backward pass is $\lambda \frac{\partial \hat{d}}{\partial \theta}$, which can be computed using (8). In practice, however, we would like to implement the backward pass not only for a single surface depth \hat{d} but for a whole batch of depth values.

We can implement this efficiently by rewriting $\lambda \frac{\partial \hat{d}}{\partial \theta}$ as

$$\mu \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \quad \text{with} \quad \mu = - \left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \right)^{-1} \lambda \quad (11)$$

Importantly, the left term in (11) corresponds to a normal backward operation applied to the neural network f_θ and the right term in (11) is just an (element-wise) scalar multiplication for all elements in the batch. We can hence conveniently compute the backward pass of the operator $\theta \rightarrow \hat{d}$

by first multiplying the incoming gradient λ element-wise with a factor and then backpropagating the result through the operator $\theta \rightarrow f_\theta(\hat{\mathbf{p}})$. Both operations can be efficiently parallelized in common deep learning frameworks.

3.4. Training

During training, we assume that we are given N images $\{\mathbf{I}_k\}_{k=1}^N$ together with corresponding camera intrinsics, extrinsics, and object masks $\{\mathbf{M}_k\}_{k=1}^N$. As our experiments show, our method works with as little as one image per object. In addition, our method can also incorporate depth information $\{\mathbf{D}_k\}_{k=1}^N$, if available.

For training f_θ and \mathbf{t}_θ , we randomly sample an image \mathbf{I}_k and N_p points \mathbf{u} on the image plane. We distinguish the following three cases: First, let \mathcal{P}_0 denote the set of points \mathbf{u} that lie inside the object mask \mathbf{M}_k and for which the occupancy network predicts a finite surface depth \hat{d} as described in Section 3.3. For these points we can define a loss $\mathcal{L}_{\text{rgb}}(\theta)$ directly on the predicted image $\hat{\mathbf{I}}_k$. Moreover, let \mathcal{P}_1 denote the points \mathbf{u} which lie outside the object mask \mathbf{M}_k . While we cannot define a photometric loss for these points, we can define a loss $\mathcal{L}_{\text{freespace}}(\theta)$ that encourages the network to remove spurious geometry along corresponding rays. Finally, let \mathcal{P}_2 denote the set of points \mathbf{u} which lie inside the object mask \mathbf{M}_k , but for which the occupancy network does not predict a finite surface depth \hat{d} . Again, we cannot use a photometric loss for these points, but we can define a loss $\mathcal{L}_{\text{occupancy}}(\theta)$ that encourages the network to produce a finite surface depth.

RGB Loss: For each point in \mathcal{P}_0 , we detect the predicted surface depth \hat{d} as described in Section 3.3. We define a photo-consistency loss for the points as

$$\mathcal{L}_{\text{rgb}}(\theta) = \sum_{\mathbf{u} \in \mathcal{P}_0} \|\xi(\mathbf{I})_{\mathbf{u}} - \xi(\hat{\mathbf{I}})_{\mathbf{u}}\| \quad (12)$$

where $\xi(\cdot)$ computes image features and $\|\cdot\|$ defines a robust error metric. In practice, we use RGB-values and (optionally) image gradients as features and an ℓ_1 -loss for $\|\cdot\|$.

Depth Loss: When the depth is also given, we can directly incorporate an ℓ_1 loss on the predicted surface depth as

$$\mathcal{L}_{\text{depth}}(\theta) = \sum_{\mathbf{u} \in \mathcal{P}_0} |d - \hat{d}|_1 \quad (13)$$

where d indicates the ground truth depth value of the sampled image point \mathbf{u} and \hat{d} denotes the predicted surface depth for pixel \mathbf{u} .

Freespace Loss: If a point \mathbf{u} lies outside the object mask but the predicted surface depth \hat{d} is finite, the network falsely predicts surface point $\hat{\mathbf{p}} = \mathbf{r}(\hat{d})$. Therefore, we penalize this occupancy with

$$\mathcal{L}_{\text{freespace}}(\theta) = \sum_{\mathbf{u} \in \mathcal{P}_1} \text{BCE}(f_\theta(\hat{\mathbf{p}}), 0) \quad (14)$$

where BCE is the binary cross entropy. When no surface depth is predicted, we apply the freespace loss to a randomly sampled point on the ray.

Occupancy Loss: If a point \mathbf{u} lies inside the object mask but the predicted surface depth \hat{d} is infinite, the network falsely predicts no surface points on ray \mathbf{r} . To encourage predicting occupied space on this ray, we uniformly sample depth values d_{random} and define

$$\mathcal{L}_{\text{occupancy}}(\theta) = \sum_{\mathbf{u} \in \mathcal{P}_2} \text{BCE}(f_\theta(\mathbf{r}(d_{\text{random}})), 1) \quad (15)$$

In the single-view reconstruction experiments, we instead use the first point on the ray which lies inside all object masks (depth of the visual hull). If we have additional depth supervision, we use the ground truth depth for the occupancy loss. Intuitively, $\mathcal{L}_{\text{occupancy}}$ encourages the network to occupy space along the respective rays which can then be used by \mathcal{L}_{rgb} in (12) and $\mathcal{L}_{\text{depth}}$ in (13) to refine the initial occupancy.

Normal Loss: Optionally, our representation allows us to incorporate a smoothness prior by regularizing surface normals. This is useful especially for real-world data as training with 2D or 2.5D supervision includes unconstrained areas where this prior enforces more natural shapes. We define this loss as

$$\mathcal{L}_{\text{normal}}(\theta) = \sum_{\mathbf{u} \in \mathcal{P}_0} \|\mathbf{n}(\hat{\mathbf{p}}_{\mathbf{u}}) - \mathbf{n}(\mathbf{q}_{\mathbf{u}})\|_2 \quad (16)$$

where $\mathbf{n}(\cdot)$ denotes the normal vector, $\hat{\mathbf{p}}_{\mathbf{u}}$ the predicted surface point and $\mathbf{q}_{\mathbf{u}}$ a randomly sampled neighbor of $\hat{\mathbf{p}}_{\mathbf{u}}$.³

3.5. Implementation Details

We implement the combined network with 5 fully-connected ResNet [26] blocks and ReLU activation. The output dimension of the last layer is 4, one dimension for the occupancy probability and three dimensions for the texture. For the single-view reconstruction experiments, we encode the input image with an ResNet-18 [26] encoder network \mathbf{g}_ϕ which outputs a 256-dimensional latent code \mathbf{z} . To facilitate training, we start with a ray sampling accuracy of $n = 16$ which we iteratively increase to $n = 128$ by doubling n after 50, 150, and 250 thousand iterations. We choose the sampling interval $[s_0, n\Delta s + s_0]$ such that it covers the volume of interest for each object. We set $\tau = 0.5$ for all experiments. We train on a single NVIDIA V100 GPU with a batch size of 64 images with 1024 random pixels each. We use the Adam optimizer [35] with learning rate $\gamma = 10^{-4}$ which we decrease by a factor of 5 after 750 and 1000 epochs, respectively.

³See supplementary for details.

4. Experiments

We conduct two different types of experiments to validate our approach. First, we investigate how well our approach reconstructs 3D shape and texture from a **single RGB image** when trained on a large collection of RGB or RGB-D images. Here, we consider both the case where we have access to multi-view supervision and the case where we use only a single RGB-D image per object during training. Next, we apply our approach to the challenging task of **multi-view reconstruction**, where the goal is to reconstruct complex 3D objects from real-world multi-view imagery.

4.1. Single-View Reconstruction

First, we investigate to which degree our method can infer a 3D shape and texture representation from single-views. We train a single model jointly on all categories.

Datasets: To adhere to community standards [13, 48, 80], we use the Choy et al. [13] subset (13 classes) of the ShapeNet dataset [10] for 2.5D and 3D supervised methods with training, validation, and test splits from [48]. While we use the renderings from Choy et al. [13] as input, we additionally render 24 images of resolution 256^2 with depth maps and object masks per object which we use for supervision. We randomly sample the viewpoint on the northern hemisphere as well as the distance of the camera to the object to get diverse supervision data. For 2D supervised methods, we adhere to community standards [33, 44, 84] and use the renderings and splits from [33]. Similar to [13, 33, 48], we train with objects in canonical pose.

Baselines: We compare against the following methods which all produce watertight meshes as output: 3D-R2N2 [13] (voxel-based), Pixel2Mesh [80] (mesh-based), and ONet [48] (implicit representation). We further compare against both the 2D and the 2.5D supervised version of Differentiable Ray Consistency (DRC) [79] (voxel-based) and the 2D supervised Soft Rasterizer (SoftRas) [44] (mesh-based). For 3D-R2N2, we use the pre-trained model from [48] which was shown to produce better results than the original model from [13]. For the other baselines, we use the pre-trained models⁴ from the authors.

4.1.1 Multi-View Supervision

We first consider the case where we have access to multi-view supervision with $N = 24$ images and corresponding object masks. In addition, we also investigate the case when ground truth depth maps are given.

Results: We evaluate the results using the Chamfer- L_1 distance from [48]. In contrast to previous works [13, 44, 48,

⁴Unfortunately, we cannot show texture results for DRC and SoftRas as texture prediction is not part of the official code repositories.

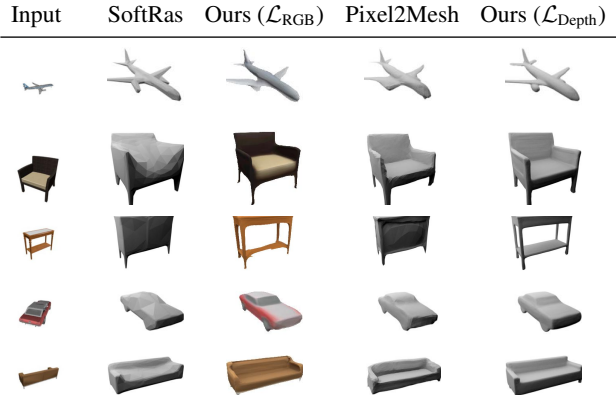


Figure 4: **Single-View Reconstruction.** We show the input renderings from [13] and the output of our 2D supervised (\mathcal{L}_{RGB}) and 2.5D supervised (\mathcal{L}_{Depth}) model, Soft Rasterizer [44] and Pixel2Mesh [80]. For 2D supervised methods we use a corresponding view from [33] as input.

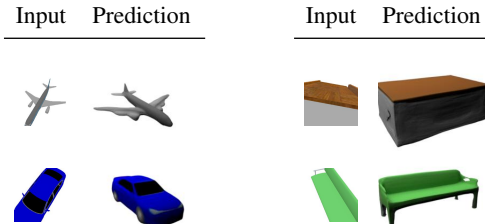


Figure 5: **Single-View Reconstruction with Single-View Supervision.** While only trained with a single-view per object, our model predicts accurate 3D geometry and texture.

[79], we compare directly wrt. to the ground truth shape models, not the voxelized or watertight versions.

In Table 1 and Fig. 4 we show quantitative and qualitative results for our method and various baselines. We can see that our method is able to infer accurate 3D shape and texture representations from single-view images when only trained on multi-view images and object masks as supervision signal. Quantitatively (Table 1), our method performs best among the approaches with 2D supervision and rivals the quality of methods with full 3D supervision. When trained with depth, our method performs comparably to the methods which use full 3D information. Qualitatively (Fig. 4), we see that in contrast to the mesh-based approaches, our method is not restricted to certain topologies. When trained with the photo-consistency loss \mathcal{L}_{RGB} , we see that our approach is able to predict accurate texture information in addition to the 3D shape.

4.1.2 Single-View Supervision

The previous experiment indicates that our model is able to infer accurate shape and texture information without 3D

category	2D Supervision			2.5D Supervision		3D Supervision		
	DRC (Mask) [79]	SoftRas [44]	Ours (\mathcal{L}_{RGB})	DRC (Depth) [79]	Ours (\mathcal{L}_{Depth})	3D R2N2 [13]	ONet [48]	Pixel2Mesh [80]
airplane	0.659	0.149	0.190	0.377	0.143	0.215	0.151	0.183
bench	-	0.241	0.210	-	0.165	0.210	0.171	0.191
cabinet	-	0.231	0.220	-	0.183	0.246	0.189	0.194
car	0.340	0.221	0.196	0.316	0.179	0.250	0.181	0.154
chair	0.660	0.338	0.264	0.510	0.226	0.282	0.224	0.259
display	-	0.284	0.255	-	0.246	0.323	0.275	0.231
lamp	-	0.381	0.413	-	0.362	0.566	0.380	0.309
loudspeaker	-	0.320	0.289	-	0.295	0.333	0.290	0.284
rifle	-	0.155	0.175	-	0.143	0.199	0.160	0.151
sofa	-	0.407	0.224	-	0.221	0.264	0.217	0.211
table	-	0.374	0.280	-	0.180	0.247	0.185	0.215
telephone	-	0.131	0.148	-	0.130	0.221	0.155	0.145
vessel	-	0.233	0.245	-	0.206	0.248	0.220	0.201
mean	0.553	0.266	0.239	0.401	0.206	0.277	0.215	0.210

Table 1: **Single-View Reconstruction.** We report Chamfer- L_1 distances wrt. the ground truth meshes for the single-view experiment. We compare against Differentiable Ray Consistency (DRC) [79] (2D and 2.5D supervision), Soft Rasterizer [44] (2D supervision), 3D-R2N2 [13], Occupancy Networks (ONet) [48], and Pixel2Mesh [80] (all 3D supervision).

supervision. A natural question to ask is how many images are required during training. To this end, we investigate the case when only *a single image* with depth and camera information is available. Since we represent the 3D shape in a canonical object coordinate system, the hypothesis is that the model can aggregate the information over multiple training instances, although it sees every object only from one perspective. As the same image is used both as input and supervision signal, we now condition on our renderings instead of the ones provided by Choy et al. [13].

Results: Surprisingly, Fig. 5 shows that our method can infer appropriate 3D shape and texture when only a single-view is available per object, confirming our hypothesis. Quantitatively, the Chamfer distance of the model trained with \mathcal{L}_{RGB} and \mathcal{L}_{Depth} with only a single view (0.410) is comparable to the model trained with \mathcal{L}_{Depth} with 24 views (0.383). The reason for the numbers being worse than in Section 4.1 is that for our renderings, we do not only sample the viewpoint, but also the distance to the object resulting in a much harder task (see Fig. 5).

4.2. Multi-View Reconstruction

Finally, we investigate if our method is also applicable to multi-view reconstruction in real-world scenarios. We investigate two cases: First, when multi-view images and object masks are given. Second, when additional sparse depth maps are given which can be obtained from classic multi-view stereo algorithms [67]. For this experiment, we do not condition our model and train one model per object.

Dataset: We conduct this experiment on scans 65, 106, and 118 from the challenging real-world DTU dataset [1]. The dataset contains 49 or 65 images with camera information for each object and baseline and structured light ground

truth data. The presented objects are challenging as their appearance changes in different viewpoints due to specularities. Our sampling-based approach allows us to train on the full image resolution of 1200×1600 . We label the object masks ourselves and always remove the same images with profound changes in lighting conditions, e.g., caused by the appearance of scanner parts in the background.

Baselines: We compare against classical approaches that have 3D meshes as output. To this end, we run screened Poisson surface reconstruction (sPSR) [34] on the output of the classical MVS algorithms Campbell et al. [9], Furukawa et al. [18], Tola et al. [78], and Colmap [67]. We find that the results on the DTU benchmark for the baselines are highly sensitive to the trim parameter of sPSR and therefore report results for the trim parameters 0 (watertight output), 5 (good qualitative results) and 7 (good quantitative results). For a fair comparison, we use the object masks to remove all points which lie outside the visual hull from the predictions of the baselines before running sPSR.⁵ We use the official DTU evaluation script in “surface mode”.

Results: We show qualitative and quantitative results in Fig. 6 and Table 2. Qualitatively, we find that our method can be used for multi-view 3D reconstruction, directly resulting in watertight meshes. The ability to accurately model cavities of the objects shows that our model uses texture information to improve over the visual hull (Fig. 7). Quantitatively, Table 2 shows that our approach rivals the results from highly tuned MVS algorithms. We note that the DTU ground truth is itself sparse (Fig. 7c) and methods are therefore rewarded for trading off completeness for accuracy, which explains the better quantitative performance of the baselines for higher trim parameters (Fig. 8).

⁵See supplementary material for details.

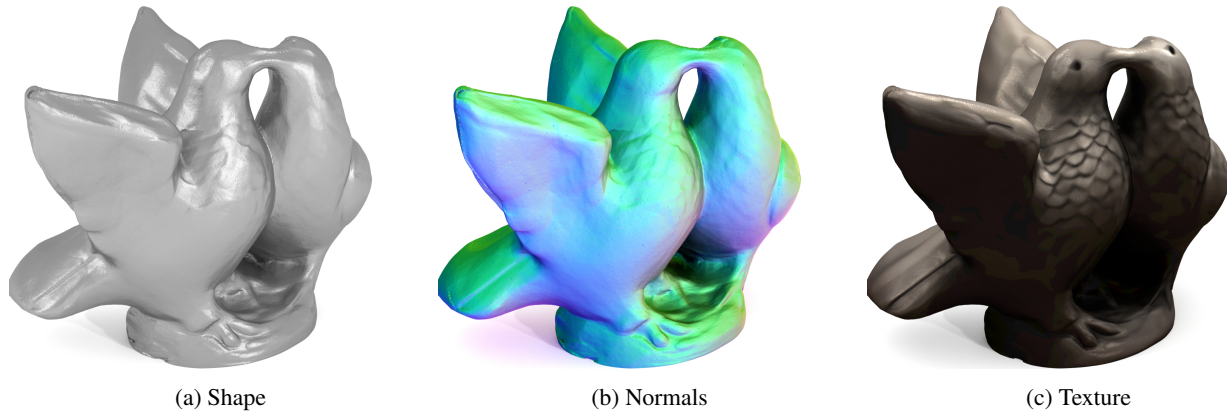


Figure 6: **Multi-View Stereo.** We show the shape, normals, and the textured shape for our method trained with 2D images and sparse depth maps for scan 106 of the DTU dataset [1].

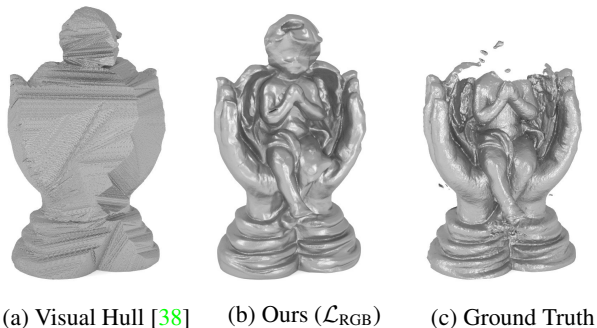


Figure 7: **Comparison against Visual Hull.** We show the visual hull, the shape prediction of our model trained with \mathcal{L}_{RGB} , and the ground truth for scan 118 of the DTU dataset. Our method uses RGB cues to improve over the visual hull and predicts parts that are missing in the ground truth.

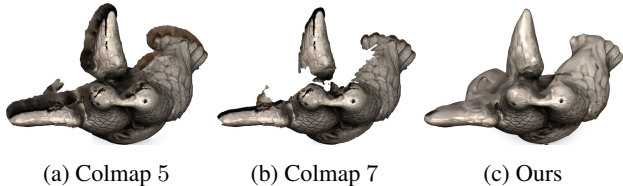


Figure 8: **Effect of Trim Parameter.** We show screened Poisson surface reconstructions [34] with trim parameters 5 and 7 for Colmap [67] and the prediction of our model trained with $\mathcal{L}_{RGB} + \mathcal{L}_{Depth}$ for scan 106 of the DTU dataset.

5. Conclusion and Future Work

In this work, we have presented Differentiable Volumetric Rendering (DVR). Observing that volumetric rendering is inherently differentiable for implicit representations allows us to formulate an analytic expression for the gradients of the depth with respect to the network parameters. Our experiments show that DVR enables us to learn implicit 3D shape representations from multi-view imagery without 3D

	Trim Param.	Accuracy	Completeness	Chamfer- L_1
Tola [78] + sPSR	0	2.409	1.242	1.826
Furu [18] + sPSR	0	2.146	0.888	1.517
Colmap [67] + sPSR	0	1.881	0.726	1.303
Camp [9] + sPSR	0	2.213	0.670	1.441
Tola [78] + sPSR	5	1.531	1.267	1.399
Furu [18] + sPSR	5	1.733	0.888	1.311
Colmap [67] + sPSR	5	1.400	0.782	1.091
Camp [9] + sPSR	5	1.991	0.670	1.331
Tola [78] + sPSR	7	0.396	1.424	0.910
Furu [18] + sPSR	7	0.723	0.955	0.839
Colmap [67] + sPSR	7	0.446	1.020	0.733
Camp [9] + sPSR	7	1.466	0.719	1.092
Ours (\mathcal{L}_{RGB})	-	1.054	0.760	0.907
Ours ($\mathcal{L}_{RGB} + \mathcal{L}_{Depth}$)	-	0.789	0.775	0.782

Table 2: **Multi-View Stereo.** We show quantitative results for scans 65, 106, and 118 on the DTU dataset. For the baselines, we perform screened Poisson surface reconstruction (sPSR) [34] with trim parameters 0, 5, and 7 to obtain the final output. It shows that our generic method achieves results comparable to the highly optimized MVS methods.

supervision, rivaling models that are learned with full 3D supervision. Moreover, we found that our model can also be used for multi-view 3D reconstruction. We believe that DVR is a useful technique that broadens the scope of applications of implicit shape and texture representations.

In the future, we plan to investigate how to circumvent the need for object masks and camera information, e.g., by predicting soft masks and how to estimate not only texture but also more complex material properties.

Acknowledgments

This work was supported by an NVIDIA research gift. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Michael Niemeyer.

References

- [1] Henrik Aanæs, Rasmus R. Jensen, George Vogiatzis, Engin Tola, and Anders B. Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision (IJCV)*, 120(2):153–168, 2016. 7, 8
- [2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3D point clouds. In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. 2
- [3] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 2, 4
- [4] Bruce G. Baumgart. *Geometric Modeling for Computer Vision*. Stanford University, 1974. 1
- [5] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo - stereo matching with slanted support windows. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2011. 2
- [6] Jeremy S. De Bonet and Paul Viola. Poxels: Probabilistic voxelized volume reconstruction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 1999. 2
- [7] Adrian Broadhurst, Tom W. Drummond, and Roberto Cipolla. A probabilistic framework for space carving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2001. 2
- [8] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv.org*, 1608.04236, 2016. 2
- [9] Neill D.F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2008. 7, 8
- [10] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiang Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. 6
- [11] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaako Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 1, 2
- [12] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3
- [13] 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 *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1, 2, 6, 7
- [14] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM Trans. on Graphics*, 1996. 2
- [15] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. Single-image SVBRDF capture with a rendering-aware deep network. In *ACM Trans. on Graphics*, 2018. 1
- [16] Simon Donné and Andreas Geiger. Learning non-volumetric depth fusion using successive reprojections. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2
- [17] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2
- [18] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1362–1376, 2010. 7, 8
- [19] Matheus Gadelha, Subhansu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017. 2
- [20] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Gipuma: Massively parallel multi-view stereo reconstruction. *Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e. V.*, 25:361–369, 2016. 2
- [21] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. Unsupervised training for 3d morphable model regression. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2
- [22] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [23] Ben Graham. Sparse 3d convolutional neural networks. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2015. 2
- [24] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. AtlasNet: A papier-mâché approach to learning 3d surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2
- [25] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5
- [27] Paul Henderson and Vittorio Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2018. 2
- [28] Paul Henderson and Vittorio Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision (IJCV)*, 2019. 2
- [29] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view

- stereopsis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [30] Zeng Huang, Tianye Li, Weikai Chen, Yajie Zhao, Jun Xing, Chloe LeGendre, Linjie Luo, Chongyang Ma, and Hao Li. Deep volumetric video from very sparse multi-view performance capture. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [31] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. GAL: geometric adversarial loss for single-view 3d-object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [32] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [33] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6
- [34] Michael M. Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics*, 32(3):29, 2013. 7, 8
- [35] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 5
- [36] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1
- [37] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision (IJCV)*, 38(3):199–218, 2000. 2
- [38] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 16(2):150–162, 1994. 8
- [39] Vincent Leroy, Jean-Sébastien Franco, and Edmond Boyer. Shape reconstruction using volume sweeping and learned photoconsistency. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [40] Kejie Li, Trung Pham, Huangying Zhan, and Ian D. Reid. Efficient dense point cloud object reconstruction using deformation vector fields. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [41] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2
- [42] Yiyi Liao, Katja Schwarz, Lars Mescheder, and Andreas Geiger. Towards unsupervised learning of generative models for 3d controllable image synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [43] Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. Material editing using a physically based rendering network. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1
- [44] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2, 6, 7
- [45] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 2
- [46] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. In *ACM Trans. on Graphics*, 2019. 2
- [47] Matthew M. Loper and Michael J. Black. Opendr: An approximate differentiable renderer. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014. 1, 2
- [48] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 6, 7
- [49] Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2
- [50] Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 1, 2
- [51] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [52] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. *arXiv.org*, 2020. 2
- [53] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [54] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2, 3
- [55] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [56] Jeong J. Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3
- [57] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

- [58] Despoina Paschalidou, Ali Osman Ulusoy, Carolin Schmitt, Luc van Gool, and Andreas Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 4
- [59] Felix Petersen, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. Pix2vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *arXiv.org*, 2019. 1
- [60] Andrew Prock and Chuck Dyer. Towards real-time voxel coloring. In *Proceedings of the DARPA Image Understanding Workshop*, 1998. 2
- [61] Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2
- [62] Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. Learning detailed face reconstruction from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1
- [63] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. OctNetFusion: Learning depth fusion from data. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017. 2
- [64] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2
- [65] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964. 4
- [66] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2, 3
- [67] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7, 8
- [68] Johannes L. Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 2
- [69] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006. 2
- [70] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1997. 2
- [71] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2437–2446, 2019. 2
- [72] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 2
- [73] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [74] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 2
- [75] Ayush Tewari, Michael Zollhöfer, Pablo Garrido, Florian Bernard, Hyeonwoo Kim, Patrick Pérez, and Christian Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1
- [76] Ayush Tewari, Michael Zollhöfer, Hyeonwoo Kim, Pablo Garrido, Florian Bernard, Patrick Pérez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1
- [77] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [78] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications (MVA)*, 23(5):903–920, 2012. 7, 8
- [79] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 4, 6, 7
- [80] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1, 2, 6, 7
- [81] Weiyue Wang, Xu Qiangeng, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. DISN: Deep implicit surface network for high-quality single-view 3d reconstruction. *Advances in Neural Information Processing Systems (NIPS)*, 2019. 2
- [82] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2
- [83] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

- [84] Xinchun Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 6
- [85] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [86] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [87] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2
- [88] Jacek Zienkiewicz, Andrew J. Davison, and Stefan Leutenegger. Real-time height map fusion using differentiable rendering. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2016. 1, 2