

# HashSDF: Accurate Implicit Surfaces with Fast Local Features on Permutohedral Lattices

Radu Alexandru Rosu  
University of Bonn  
rosu@ais.uni-bonn.de

Sven Behnke  
University of Bonn  
behnke@ais.uni-bonn.de

## Abstract

*Neural radiance-density field methods have become increasingly popular for the task of novel-view rendering. Their recent extension to hash-based positional encoding ensures fast training and inference with state-of-the-art results. However, density-based methods struggle with recovering accurate surface geometry. Hybrid methods alleviate this issue by optimizing the density based on an underlying SDF. However, current SDF methods are overly smooth and miss fine geometric details. In this work, we combine the strengths of these two lines of work in a novel hash-based implicit surface representation. We propose improvements to the two areas by replacing the voxel hash encoding with a permutohedral lattice which optimizes faster in three and higher dimensions. We additionally propose a regularization scheme which is crucial for recovering high-frequency geometric detail. We evaluate our method on multiple datasets and show that we can recover geometric detail at the level of pores and wrinkles while using only RGB images for supervision. Furthermore, using sphere tracing we can render novel views at 30 fps on an RTX 3090.*

## 1. Introduction

Accurate reconstruction of color and geometry of scenes is an important component of many computer vision tasks [16, 19, 23, 28]. Recent NeRF-like models [3, 16, 18, 25] represent the scene as a density and radiance field which, when supervised with enough input images, can render photorealistic novel views.

Works like INGP [18] further improved on NeRF by using a hash-based positional encoding which results in fast training and state-of-the-art results. However, despite the photorealistic renderings, the reconstructed scene geometry

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2070 - 390732324.

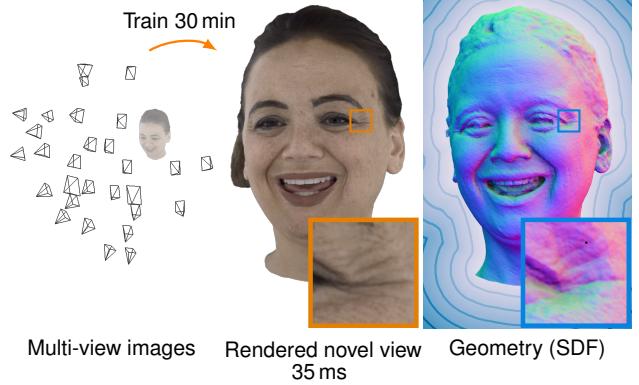


Figure 1. Given multi-view images, we recover both high quality geometry as an implicit SDF and appearance which can be rendered in real-time.

can deviate severally from the ground-truth. For example, objects with high specularity or view-dependent effects are often reconstructed as a cloud of low density and untextured regions can have arbitrary density.

Another line of recent methods tackles the issue of incorrect geometry by representing the surface as occupancy or Signed Distance Function (SDF) [22, 27, 30]. This representation can also be optimized with volumetric rendering techniques and supervised with RGB images. However, their parametrization as a single fully-connected Multi-Layer Perceptron (MLP) leads to overly smooth geometry and color.

In this work, we propose HashSDF, a method that combines the strengths of hash-based encodings and implicit surfaces. We represent the scene as an SDF and a color field, which we render using unbiased volumetric integration [27]. A naive combination of these two methods would fail to reconstruct accurate surfaces however, since it lacks any inductive bias for the ambiguous cases like specular or untextured surfaces. Attempts to regularize the SDF with a total variation loss or a curvature loss will create smoother geometry at the expense of losing smaller details. In this work, we propose a regularization scheme that ensures both smooth geometry where needed and also reconstruction of

fine details like pores and wrinkles. Furthermore, we improve upon the voxel-hashing method of INGP by proposing permutohedral lattice hashing. The number of vertices per simplex in this structure scales linearly with dimensionality instead of exponentially as in the hyper-cubical voxel case. We show that the permutohedral lattice performs better than voxels for 3D reconstruction and 4D background estimation.

In summary our main contributions are:

- a novel framework for optimizing neural implicit surfaces based on hash-encoding,
- an extension of hash encoding to a permutohedral lattice which scales linearly with the input dimensions and allows for faster optimization, and
- a regularization scheme that allows to recover accurate SDF geometry with level of detail at the scale of pores and wrinkles.

## 2. Related Work

### 2.1. Classical Multi-View Reconstruction

Multi-view 3D reconstruction has been studied for decades. The classical methods can be categorized as either depth map-based [8,9,23,34] or volume-based [6,19,21,28]. Depth map methods like COLMAP [23] reconstruct a depth map for each input view by matching photometrically consistent patches. The depth maps are fused to a global 3D point cloud and a watertight surface is recovered using Poisson Reconstruction [12]. While COLMAP can give good results in most scenarios, it yields suboptimal results on non-Lambertian surfaces. Volume-based approaches fuse the depth maps into a volumetric structure (usually a Truncated Signed Distance Function) from which an explicit surface can be recovered via the marching cubes algorithm [15]. Volumetric methods work well when fusing multiple noisy depth maps but struggle with reconstructing thin surfaces and fine details.

### 2.2. NeRF Models

A recent paradigm shift in 3D scene reconstruction has been the introduction of NeRF [16]. NeRF represents the scene as a density and radiance field, parameterized by a MLP. Volumetric rendering is used to supervise using only RGB images. This yields highly photorealistic renderings with view-dependent effects. However, the long training time of the original NeRF prompted a series of subsequent works to address this issue.

### 2.3. Accelerating NeRF

Two main areas were identified as problematic: the large number of ray samples that traverse empty space and the requirement to query a large MLP for each individual sample.

Neural Sparse Voxel Fields [14] uses an octree to model only the occupied space and restricts samples to be generated only inside the voxels. Features from the voxels are interpolated and a shallow MLP predicts color and density. This achieves significant speedups but requires complicated pruning and updating of the octree structure.

DVGO [24] similarly models the scene with local features which are stored in a dense grid that is decoded by an MLP into view-dependent color.

Plenoxels [7] completely removes any MLP and instead stores opacity and spherical harmonics (SH) coefficients at sparse voxel positions.

Instant NGP (INGP) [18] proposes a hash-based encoding in which ray samples trilinearly interpolate features between eight hash positions from a hash-map. A shallow MLP implemented as a fully fused CUDA kernel predicts color and density. Using a hash-map for encoding has the advantage of not requiring complicated mechanisms for pruning or updating like in the case of octrees.

In our work, we improve upon INGP by proposing a novel permutohedral lattice-based hash encoding, which is better suited for interpolating in high-dimensional spaces. We use our new encoding to reconstruct accurate 3D surfaces and model background as a 4D space.

### 2.4. Implicit Representation

Other works have focused on reconstructing the scene geometry using implicit surfaces.

SDFDiff [11] discretizes SDF values on a dense grid and by defining a differentiable shading function can optimize the underlying geometry. However, their approach can neither recover arbitrary color values nor can it scale to higher-resolution geometry.

IDR [31] and DVR [20] represent the scene as SDF and occupancy map respectively, and by using differentiable rendering can recover high frequency geometry and color. However, both methods require 2D mask supervision for training which is not easy to obtain in practice.

In order to remove the requirement of mask supervision, UNISURF [22] optimizes an occupancy function through volumetric rendering. VolSDF [30] further extends this idea to SDFs. NeuS [27] further analyzes the biases caused by using volumetric rendering for optimizing SDFs and introduces an unbiased and occlusion-aware weighting scheme which allows to recover more accurate surfaces.

In this work, we aim to reconstruct a scene as SDF and color fields without using any mask supervision. We model the scene using locally hashed features in order to recover finer detail than previous works. We also propose several regularizations which help to recover geometric detail at the level of pores and wrinkles.

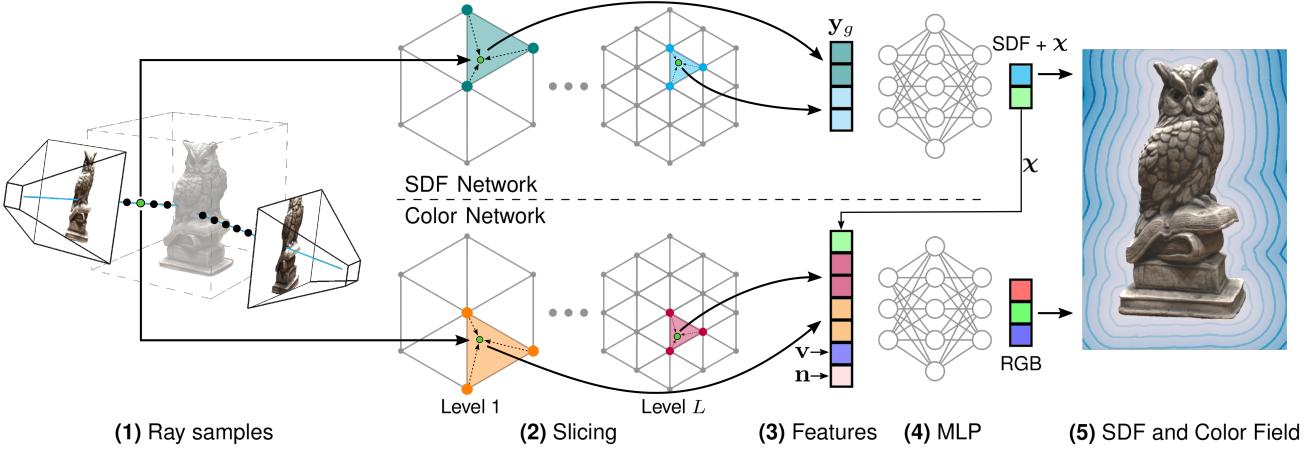


Figure 2. Overview of our HashSDF pipeline. (1) For a batch of pixels from the posed images, we sample rays inside the volume of interest. (2) For each sample we slice features from a multi-resolution permutohedral lattice. (3) The features from all lattice levels are concatenated. For the color network, we also concatenate additional features regarding normal  $\mathbf{n}$  of the SDF, view direction  $\mathbf{v}$  and learnable features  $\chi$  from the SDF network. (4) A small MLP decodes the SDF or a view-dependent RGB color. (5) The final output is an SDF and color field which is volumetrically rendered and supervised only with RGB images.

### 3. Overview

Given a series of images with poses  $\{\mathcal{I}_k\}$  our task is to recover both surface  $\mathcal{S}$  and appearance of the objects within. We define the surface  $\mathcal{S}$  as the zero-level set of an SDF:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 | g(\mathbf{x}) = 0\} \quad (1)$$

The SDF is parameterized by a fully connected neural network  $g(\mathbf{y}_g; \Phi_g)$  together with an encoding of its input  $\mathbf{y}_g = \text{enc}(\mathbf{x}; \theta_g)$ . We refer to the composition  $g(\text{enc}(\mathbf{x}; \theta_g); \Phi_g)$  as the SDF network which outputs SDF values for a given spatial 3D position  $\mathbf{x}$ .

Similarly, for the color we define an MLP  $c(\mathbf{y}_c, \mathbf{v}, \mathbf{n}, \chi; \Phi_c)$  which takes as input an encoded position  $\mathbf{y}_c = \text{enc}(\mathbf{x}; \theta_c)$ , a view direction  $\mathbf{v}$ , the normal vector of the SDF  $\mathbf{n}$ , and an learnable geometric feature  $\chi$  which is output by the SDF network.

Fig. 2 given an overview of our two-network pipeline. Rays from the input images are cast into the scenes and multiple samples are created along each ray. Each sample  $\mathbf{x}$  on the ray is encoded using a multi-resolutonal hash-based permutohedral lattice (cf. Sec. 4.2). The lattice features from different levels are concatenated and input to the color and SDF MLPs. The SDF values are mapped to density (cf. Sec. 4.1) and the sample colors are volumetrically rendered to yield the final pixel value.

Note, that using two separate networks is crucial as we want to regularize each one individually in order to recover high-quality geometry.

### 4. HashSDF

We now detail each network, the permutohedral lattice, and our training methodology.

#### 4.1. Volume Rendering

Given a pixel, we denote the ray emitted from this pixel by  $\mathbf{p}(t) = \mathbf{o} + t\mathbf{v}$ , where  $\mathbf{o}$  is the camera origin and  $\mathbf{v}$  is the view direction. Colors along the ray are accumulated according to

$$\hat{C}(\mathbf{p}) = \int_{t=0}^{+\infty} w(t)c(\mathbf{p}(t), \mathbf{v}, \mathbf{n}, \chi; \Phi_c), \quad (2)$$

where  $w(t)$  is a weighting function for the point at  $\mathbf{p}(t)$ .

In NeuS [27], Wang et al. show that in order to learn an SDF of the scene, it is crucial to derive an appropriate weighting function based on the SDF.

They propose an unbiased and occlusion-aware weighting function based on an opaque density function  $\rho(t)$ :

$$\rho(t) = \max \left( -\frac{\frac{d\psi_s}{dt}(g(\mathbf{p}(t); \Phi_g))}{\psi_s(g(\mathbf{p}(t); \Phi_g))}, 0 \right), \quad (3)$$

where  $g(\mathbf{p}(t))$  outputs the SDF for the point at  $t$  and  $\psi$  is the sigmoid function defined as  $\psi_s(x) = (1 + e^{-hx})^{-1}$  with slope  $h$ . This can be used directly in a volumetric rendering scheme:

$$w(t) = T(t)\rho(t), \text{ where } T(t) = \exp \left( - \int_0^t \rho(u)du \right) \quad (4)$$

#### 4.2. Hash Encoding with Permutohedral Lattice

In order to facilitate learning of high-frequency details INGP [18] proposed a hash-based encoding which maps a 3D spatial coordinate to a higher-dimensional space. The

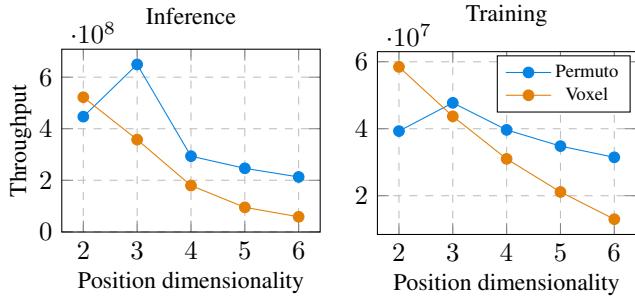


Figure 3. We propose to use a permutohedral lattice instead of hyper-cubical voxels since the number of vertices per simplex scales linearly with the dimensionality instead of exponentially. The permutohedral lattice performs better at dimensions 3 and above in both, training and inference.

encoding maps a spatial position  $\mathbf{x}$  into a cubical grid and linearly interpolate features from the hashed eight corners of the containing cube. A fast CUDA implementation interpolates over various multi-resolutional grids in parallel. The hash-map is stored as a tensor of  $L$  levels, each containing up to  $T$  feature vectors with dimensionality  $F$ .

The speed of the encoding function is mostly determined by the number of accesses to the hash-map as the operations to determine the eight hash indices are fast. Hence, it is of interest to reduce the memory accesses required to linearly interpolate features for position  $\mathbf{x}$ . By using a tetrahedral lattice instead of a cubical one, memory accesses can be reduced by a factor of two as each simplex has only four vertices instead of eight. Going one step further, we consider even higher dimensions by using a permutohedral lattice [1].

The permutohedral lattice divides the space into uniform simplices which form triangles and tetrahedra in 2D and 3D, respectively. The main advantage of this lattice is that given dimensionality  $d$  the number of vertices per simplex is  $d+1$ , which scales linearly instead of the exponential growth  $2^d$  for hyper-cubical voxels. This ensures a low number of memory accesses to the hashmap and therefore fast optimization.

Given a position  $\mathbf{x}$ , the containing simplex can be obtained in  $\mathcal{O}(d^2)$ . Within the simplex, barycentric coordinates are calculated and d-linear interpolation is performed similar to INGP. For more details regarding calculating the containing simplex, we refer to [2].

Similarly to INGP, we slice from lattices at multiple resolutions and concatenate the results. The final output is a high-dimensional encoding  $\mathbf{y} = \text{enc}(\mathbf{x}; \theta)$  of the input  $\mathbf{x}$  given lattice features  $\theta$ .

#### 4.3. 4D Background Estimation

For modeling the background, we follow the formulation of NeRF++ [33] which represents foreground volume as a unit sphere and background volume by an inverted sphere.

Points in the outer volume are represented using 4D positions  $(x', y', z', 1/r)$  where  $(x', y', z')$  is a unit directional vector and  $1/r$  is the inverse distance.

We directly use this 4D coordinate to slice from a 4-dimensional lattice and obtain multi-resolutional features. A small MLP outputs the radiance and density which are volumetrically rendered and blended with the foreground. Please note that in 4D, the permutohedral lattice only needs to access five vertices for each simplex while a cubical voxel would need 16. Our linear scaling with dimensionality is of significant advantage in this use-case.

### 5. HashSDF Training

Given the new encoding and the unbiased volumetric rendering scheme, we have all the tools to train our model. We sample random rays from the input images, create samples on them for which we infer SDF and color. Through volumetric rendering we obtain the per-pixel color  $\hat{C}$ .

We optimize an L2 loss on the RGB pixels:

$$\mathcal{L}_{rgb} = \sum_p \|\hat{C}(\mathbf{p}) - C(\mathbf{p})\|_2^2 \quad (5)$$

and an Eikonal loss which prevents the zero-everywhere solution for the SDF:

$$\mathcal{L}_{eik} = \sum_x (\|\nabla g(\text{enc}(\mathbf{x}))\| - 1)^2, \quad (6)$$

where the gradient  $\nabla g(\text{enc}(\mathbf{x}))$  of the SDF is obtained through automatic differentiation.

A naive combination of hash-based encoding and implicit surfaces can yield undesirable surfaces, though. While the model is regularized by the Eikonal loss, there are many surfaces that satisfy the Eikonal constraint. For specular or untextured areas, the Eikonal regularization doesn't provide enough information to properly recover the surface. To address this issue, we propose several regularizations that serve to both recover smoother surfaces and more detail.

#### 5.1. SDF Regularization

In order to aid the network in recovering smoother surfaces in reflective or untextured areas, we add a curvature loss on the SDF. Calculating the full  $3 \times 3$  Hessian matrix can be expensive, so we approximate curvature as local deviation of the normal vector. Recall that we already have the normal  $\mathbf{n} = \nabla g(\text{enc}(\mathbf{x}))$  at each ray sample since it was required for the Eikonal loss. With this normal we define a tangent vector  $\eta$  by cross product with a random unit vector  $\tau$  such that  $\eta = \mathbf{n} \times \tau$ . Given this random vector in the tangent plane, we slightly perturb our sample  $\mathbf{x}$  to obtain  $\mathbf{x}_\epsilon = \mathbf{x} + \epsilon\eta$ . We obtain the normal at the new perturbed point as  $\mathbf{n}_\epsilon = \nabla g(\text{enc}(\mathbf{x}_\epsilon))$  and define a curvature

loss based on the dot product between the normals at the original and perturbed point:

$$\mathcal{L}_{curv} = \sum_x (\mathbf{n} \cdot \mathbf{n}_\epsilon - 1)^2. \quad (7)$$

## 5.2. Color Regularization

While the curvature regularization helps in recovering smooth surfaces, we observe that the network converges to an undesirable state where the geometry gets increasingly smoother while the color network learns to model all the high-frequency detail in order to drive the  $\mathcal{L}_{rgb}$  to zero. Despite lowering  $\mathcal{L}_{curv}$  during optimization, the SDF doesn't regain back the lost detail. We show this behavior in Fig. 8. Recall that the color network is defined as  $c(\mathbf{y}, \mathbf{v}, \mathbf{n}, \chi; \Phi_c)$ , with an input encoding of  $\mathbf{y} = \text{enc}(\mathbf{x}; \theta_c)$ . We observe that all the high-frequency detail learned by the color network has to be present in the weights of the MLP  $\Phi_c$  or the hashmap table  $\theta_c$  as all the other inputs are smooth.

In order to recover fine geometric detail, we propose to learn a color mapping network that is itself smooth w.r.t. to its input such that large changes in color are matched with large changes in surfaces normal. Function smoothness can be studied in the context of Lipschitz continuous networks. A function  $f$  is  $k$ -Lipschitz continuous if it satisfies:

$$\underbrace{\|f(a) - f(b)\|}_{\text{change in the output}} \leq k \underbrace{\|a - b\|}_{\text{change in the input}} \quad (8)$$

Intuitively, it sets  $k$  as an upper bound for the rate of change of the function. We are interested in the color network being a smooth function (small  $k$ ) such that high-frequency color is also reflected in high-detailed geometry.

There are several ways to enforce Lipschitz smoothness on a function [5, 17, 26, 32]. Most of them impose a hard 1-Lipschitz requirement or ignore effects such as network depth which makes them difficult to tune for our use-case.

The recent work from Liu *et al.* [13] provides a simple and interpretable framework for softly regularizing the Lipschitz constant of a network. Given an MLP layer  $y = \sigma(W_i x + b_i)$  and a trainable Lipschitz bound  $k_i$  for the layer, they replace the weight matrix  $W_i$  with:

$$y = \sigma(\widehat{W}_i x + b_i), \quad \widehat{W}_i = m(W_i, \text{softplus}(k_i)), \quad (9)$$

where  $\text{softplus}(k_i) = \ln(1 + e^{k_i})$  and the function  $m()$  normalizes the weight matrix by rescaling each row of  $W_i$  such that the absolute value of the row-sum is less than or equal to  $\text{softplus}(k_i)$ .

Finally, since the product of per-layer Lipschitz constants  $k_i$  is the Lipschitz bound for the whole network, we can regularize it using:

$$\mathcal{L}_{Lipschitz} = \prod_l^i \text{softplus}(k_i) \quad (10)$$



Figure 4. We train for 30 min using posed images and afterwards render novel views in real-time using sphere tracing.

In addition to regularizing the color MLP, we also apply weight decay of 0.01 to the color hashmap  $\theta_c$ .

## 5.3. Training Schedule

Several scheduling considerations must be observed for our method. In Eq. 3 the sigmoid function  $\psi_s(x)$  is parametrized with  $1/h$  which is the standard deviation that controls the range of influence of the SDF towards the volume rendering. In NeuS  $1/h$  is considered a learnable parameter which starts at a high value and decays towards zero as the network converges.

However, we found out that considering it as a learnable parameter can lead to the network missing thin object features due to the fact that large objects in the scene dominate the gradient towards  $h$ . Instead, we use a manual linear decay  $1/h$  over 30k iterations which we found to be robust for all the objects we tested.

In order to recover smooth surfaces, we train the first 100k iterations using curvature loss:

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_1 \mathcal{L}_{eik} + \lambda_2 \mathcal{L}_{curv}. \quad (11)$$

For further 100k iterations, we recover detail by removing the curvature loss and adding the regularization of the color network  $\lambda_3 \mathcal{L}_{Lipschitz}$ .

In addition, we initialize our network with the SDF of a sphere at the beginning of the optimization and anneal the levels  $L$  of the hash-map in a coarse-to-fine manner over the course of the initial 10k iterations. We refer to the supplementary material for more details regarding the hyperparameters  $\lambda_{1-3}$ .

## 6. Acceleration

Similar to other volumetric rendering methods, a major bottleneck for the speed is the number of samples considered for each ray. We use several methods to accelerate both training and inference.

### 6.1. Occupancy Grid

In order to concentrate more samples near the surface of the SDF and have fewer in empty space, we use an occupancy grid modeled as a dense grid of resolution  $128^3$ .

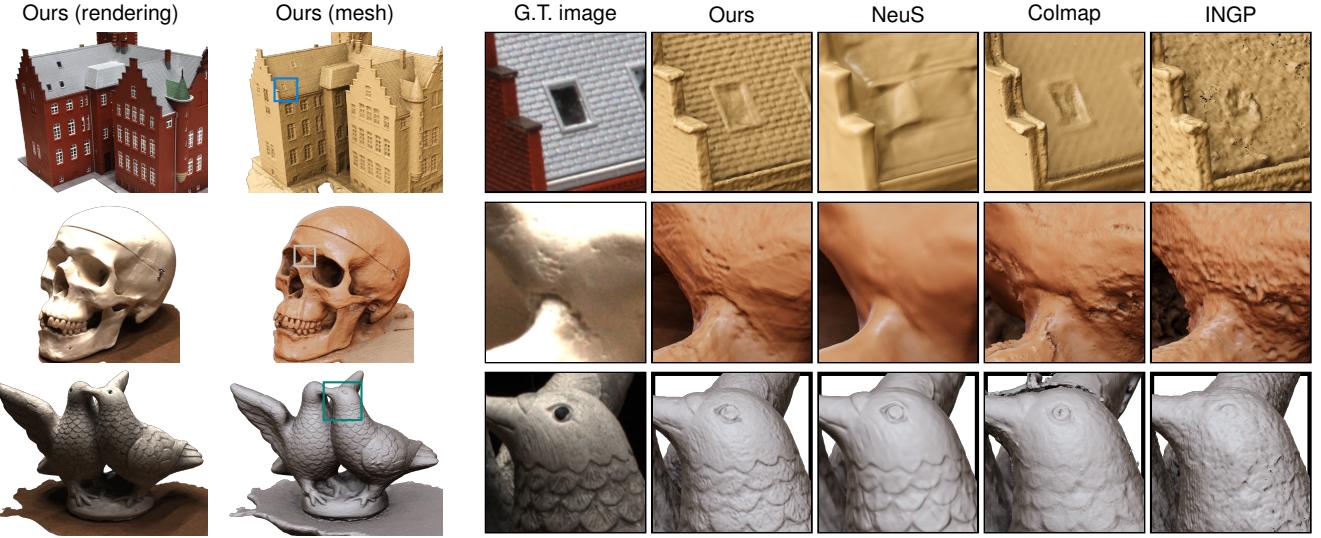


Figure 5. Qualitative comparison of the geometry reconstructed by our method compared to the baselines. Note that our method recovers significantly higher geometrical detail.

We maintain two versions of the grid, one with full precision, storing the SDF value at each voxel, and another containing only a binary occupancy bit. The grids are laid out in Morton order to ensure fast traversal. Note that differently from INGP, we store signed distance and not density in our grid. This allows us to use the SDF volume rendering equations to determine if a certain voxel has enough of a weight that it would meaningfully contribute to the integral Eq. 2 and therefore should be marked as occupied space.

We refer to the supplementary material for more details on the update of the occupancy grid.

## 6.2. Sphere Tracing

Having an SDF opens up a possibility for accelerating rendering at inference time by using sphere tracing. This can be significantly faster than volume rendering as most rays converge in 2-3 iterations towards the surface. We create a ray sample at the first voxel along the ray that is marked as occupied. We run sphere tracing for a predefined number of iterations and march not-yet-converged samples towards the surface (indicated by their SDF being above a certain threshold). Once all samples have converged or we reached a maximum number of sphere traces, we sample the color network once and render.

We show in Fig. 4 that by using sphere tracing we can render in real-time and can also trade-off rendering speed against accuracy by varying the number of iterations.

## 6.3. Implementation

We implement the encoding  $\mathbf{y} = \text{enc}(\mathbf{x}; \theta)$  using permutohedral lattices in a custom CUDA kernel which slices in parallel from all resolutions. The backward pass for updating the hashmap  $\frac{\partial \mathbf{y}}{\partial \theta}$  is also implemented in an optimized

CUDA kernel. We use the chain rule to backpropagate the upstream gradients as:  $\frac{\partial \mathcal{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \theta}$ .

Additionally, since we require the normals  $\mathbf{n} = \nabla g(\text{enc}(\mathbf{x}))$  for fitting the SDF, we also implement a kernel for calculating the partial derivative of encoding  $\mathbf{y} = \text{enc}(\mathbf{x}; \theta_g)$  w.r.t. to spatial positon  $\mathbf{x}$ , ie.  $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ . Again, the chain rule is applied with the autograd partial derivative of  $g()$  as:  $\frac{\partial g}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$  to obtain the normal.

Furthermore, since we use this normal as part of our loss function  $\mathcal{L}_{eik}$ , we support also double backward operations, i.e., we also implement CUDA kernels for  $\partial(\frac{\partial \mathcal{L}}{\partial \mathbf{x}})/\partial \theta$  and  $\partial(\frac{\partial \mathcal{L}}{\partial \mathbf{x}})/\partial(\frac{\partial \mathcal{L}}{\partial \mathbf{y}})$ . Hence, we can run our optimization entirely within PyTorch’s autograd engine, without requiring any finite differences.

## 7. Results

We evaluate our method on multiple datasets and report metrics on the accuracy of both, the 3D reconstruction and novel-view synthesis (NVS).

### 7.1. DTU

We evaluate the quality of 3D reconstruction on the DTU [10] dataset which consists of 2D images of objects and ground-truth 3D point clouds. The objects span different range of materials with different specularities which can pose a challenge for classical multi-view reconstruction methods. Each object is captured with  $\approx 50$  images and we use every 8-th image for testing and the rest for training. We evaluate against NeuS [27] which is our baseline, INGP [18] which is state-of-the-art in NVS rendering, and COLMAP [23], a classical MVS method.

We show qualitative results of the extracted meshes

ScanID	w/ mask			w/o mask			
	INGP [18]	NeuS [27]	Ours	COLMAP [23]	INGP [18]	NeuS [27]	Ours
scan24	1.73	0.83	<b>0.53</b>	0.81	1.56	1.00	<b>0.52</b>
scan37	1.79	0.98	<b>0.64</b>	2.05	2.15	1.37	<b>0.69</b>
scan40	1.46	0.56	<b>0.34</b>	0.73	1.45	0.93	<b>0.38</b>
scan55	0.86	<b>0.37</b>	0.37	1.22	0.76	0.43	<b>0.36</b>
scan63	1.70	1.13	<b>0.96</b>	1.79	1.62	1.10	<b>0.93</b>
scan65	1.57	<b>0.59</b>	0.71	1.58	1.33	<b>0.65</b>	0.74
scan69	1.66	0.60	<b>0.58</b>	1.02	1.63	<b>0.57</b>	0.58
scan83	1.56	1.45	<b>1.22</b>	3.05	1.79	1.48	<b>1.37</b>
scan97	1.83	0.95	<b>0.82</b>	1.40	2.16	1.09	<b>1.03</b>
scan105	1.55	0.78	<b>0.66</b>	2.05	1.45	<b>0.83</b>	0.84
scan106	1.23	0.52	<b>0.51</b>	1.00	1.25	0.52	<b>0.45</b>
scan110	1.75	1.43	<b>0.72</b>	1.32	1.91	1.20	<b>0.97</b>
scan114	1.71	0.36	<b>0.36</b>	0.49	1.76	0.35	<b>0.33</b>
scan118	1.44	0.45	<b>0.42</b>	0.78	1.24	0.49	<b>0.40</b>
scan122	1.31	0.49	<b>0.47</b>	1.17	1.47	0.54	<b>0.44</b>
mean	1.54	0.77	<b>0.62</b>	1.36	1.57	0.84	<b>0.67</b>

Table 1. Quantitative Chamfer distance evaluation on DTU dataset. COLMAP results are achieved by trim=0.

ScanID	ScanID: 24-83				ScanID: 97-122			
	NeuS [27]	NeRF [16]	INGP [18]	Ours	NeuS [27]	NeRF [16]	INGP [18]	Ours
scan24	26.76	27.54	28.77	<b>30.41</b>	29.30	<b>30.46</b>	29.43	30.45
scan37	25.84	26.54	26.34	<b>28.03</b>	34.50	35.51	36.20	<b>37.02</b>
scan40	27.25	28.53	28.97	<b>30.74</b>	34.12	34.86	35.05	<b>35.86</b>
scan55	28.09	30.39	31.20	<b>32.37</b>	32.46	32.87	32.16	<b>35.03</b>
scan63	34.24	35.25	<b>36.72</b>	36.64	30.01	30.82	31.04	<b>31.50</b>
scan65	33.83	33.42	<b>34.13</b>	34.04	36.73	36.87	37.91	<b>38.13</b>
scan69	29.94	30.22	29.63	<b>30.69</b>	37.89	37.77	38.64	<b>39.82</b>
scan83	39.02	40.12	40.29	<b>40.46</b>	Mean	31.99	32.74	33.10
								<b>34.07</b>

Table 2. Quantitative PSNR comparisons on the task of novel view synthesis without mask supervision.

in Fig. 5 where we train all methods without any mask supervision. Our method surpasses the level-of-detail of the other methods and is robust to view-dependent and untextured areas. We report quantitative Chamfer distance results in Tab. 1 comparing reconstruction with and without mask supervision.

Additionally, we evaluate the quality of our novel-view synthesis on the same dataset in Tab. 2. We show that we surpass NeuS, NeRF, and in some cases even INGP. This is due to the fact that our method reconstructs the underlying geometry more faithfully, making it easier to generalize the rendering to novel views.

## 7.2. Multiface

Reconstructing human figures is especially difficult as they exhibit multiple view-dependent effects and fine details that need to be captured. To evaluate this, we use the Multiface dataset [29]. It consists of human subjects that were captured with a dome of  $\approx 40$  high-resolution images performing various expressions. We show in Fig. 7 a qualitative comparison between our reconstruction and NeuS.

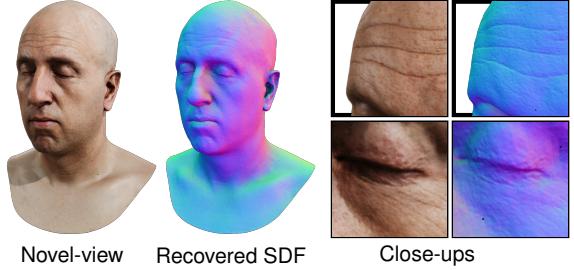


Figure 6. Given high-resolution synthetically rendered images, our approach can recover small details like pores and wrinkles. Please refer to the supplementary material for reconstructions of the other methods.

Our method is more detailed than NeuS but it still struggles with very fine detail like hair where it usually learns to create hair-like streaks in the geometry in order to model eyebrows and beards.

## 7.3. Synthetic

Since the Multiface dataset was captured with real cameras, they exhibit several camera issues like depth-of-field effects and inaccurate color calibration which can prevent our method from learning more detail. To evaluate what can be achieved with perfect camera conditions, we render a head figure [4] using a realistic physically-based renderer as input for reconstruction. Since the virtual cameras are perfectly calibrated and without defects, this can be seen as an upper bound on the quality that can be achieved with our method. We show in Fig. 6 that we are capable of recovering details at the level of pores and wrinkles which cannot be achieved with previous learning-based volumetric rendering methods.

## 7.4. Performance

We evaluate the performance of our proposed permutohedral lattice method for inference and training. In Fig. 3 we compare it to cubical voxels used in INGP by interpolating from a batch of  $2^{19}$  random points. We set both hashmaps to a capacity  $T = 2^{18}$ , a number of levels  $L = 24$  and features per entry  $F = 2$ . Our lattice outperforms the cubical lattice for dimensions 3 and above but it is slower for the 2D case. This is to be expected as in 2D, a square lattice accesses 4 vertices per simplex, while we access 3 which is not much of an improvement and the cost of finding the simplex and calculating barycentric coordinates dominates.

## 7.5. Ablation Study

We perform an ablation study of the components that we propose for HashSDF in Fig. 8. While sphere initialization and coarse-to-fine optimization help slightly in recovering a smoother shape, they don't fully fix the issue of holes in the geometry. Adding the curvature loss solves most of the

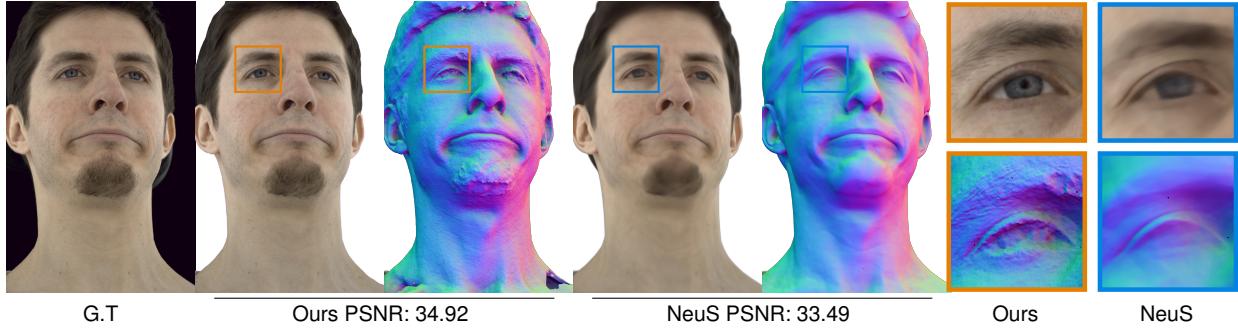


Figure 7. Qualitative comparison between our method and NeuS on the Multiface dataset. We can recover finer detail with higher PNSR.

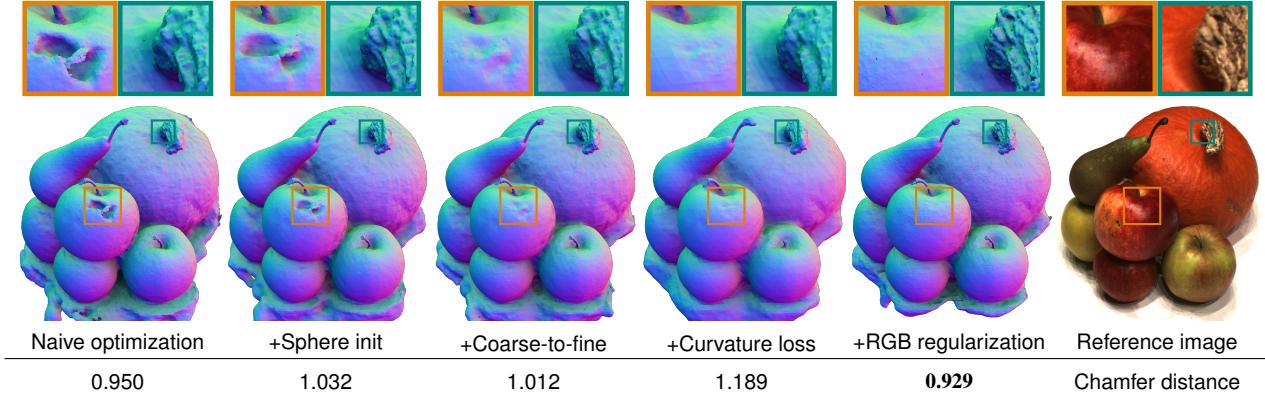


Figure 8. Ablation study of the various components of our method. A naive combination of hash-based encoding and implicit surfaces can lead to undesirable holes and overly smooth geometry. Adding sphere initialization and coarse-to-fine optimization helps with recovering smoother surfaces especially for highly specular areas. Adding a curvature loss helps further in remedying the issue of holes but uniformly smoothes the geometry. Adding RGB regularization forces the network to gain back the fine details.

issues but also severely over-smoothes the rest of the object which results in a higher Chamfer distance. Adding RGB regularization using the Lipschitz loss is crucial to recover high-frequency details and obtaining the lowest Chamfer distance.

## 8. Limitations and Future Work

Here we discuss several future directions to overcome the limitations of our work.

First, we are currently focused on reconstructing a scene that is contained within a predefined bounding volume. Recovering unbounded scenes as an implicit surface is not trivial and is therefore left as future work.

Secondly, in our experiments we only consider static scenes. Reconstructing dynamic scenes is therefore an exciting future direction. Additionally, considering time as a dimension and casting the problem in a 4D space can benefit from the linear scalability of the permutohedral lattice.

## 9. Conclusion

We proposed a combination of implicit surface representations and hash-based encoding methods for the task of reconstructing accurate geometry and appearance from unmasked posed color images. We improved upon the voxel-based hash encoding by using a permutohedral lattice which is faster in three and higher dimensions. Additionally, we propose a simple regularization scheme that allows to recover fine geometrical detail at the level of pores and wrinkles. Our full system can train in  $\approx 30$  min on an RTX 3090 GPU and render in real-time using sphere tracing.

We believe this work together with the (to-be-released) code will help the community in a multitude of other tasks that require modeling signals using fast local features.

## References

- [1] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010. [4](#)

- [2] Jongmin Baek and Andrew Adams. Some useful properties of the permutohedral lattice for Gaussian filtering. *other words*, 10(1):0, 2009. 4
- [3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. *IEEE International Conference on Computer Vision (ICCV)*, 2021. 1
- [4] James Busby. 3D scan store head model. <https://www.3dscanstore.com/blog/Free-3D-Head-Model>. Accessed: 2022-09-30. 7
- [5] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval Networks: Improving robustness to adversarial examples. In *34th International Conference on Machine Learning (ICML)*, pages 854–863. PMLR, 2017. 5
- [6] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *23rd Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 303–312, 1996. 2
- [7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxtels: Radiance fields without neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2022. 2
- [8] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1362–1376, 2009. 2
- [9] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *IEEE International Conference on Computer Vision (ICCV)*, pages 873–881, 2015. 2
- [10] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 406–413, 2014. 6
- [11] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. SDFDiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1261, 2020. 2
- [12] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *4th Eurographics Symposium on Geometry Processing (SGP)*, volume 7, 2006. 2
- [13] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via Lipschitz regularization. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference (SIGGRAPH)*, pages 31:1–31:13. ACM, 2022. 5
- [14] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 15651–15663, 2020. 2
- [15] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987. 2
- [16] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421, 2020. 1, 2, 7
- [17] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations (ICLR)*, 2018. 5
- [18] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (SIGGRAPH)*, 41(4):102:1–102:15, July 2022. 1, 2, 3, 6, 7
- [19] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011. 1, 2
- [20] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3504–3515, 2020. 2
- [21] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013. 2
- [22] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5589–5599, 2021. 1, 2
- [23] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, pages 501–518, 2016. 1, 2, 6, 7
- [24] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5459–5469, 2022. 2
- [25] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8248–8258, 2022. 1
- [26] Dávid Terjék. Adversarial Lipschitz regularization. In *8th International Conference on Learning Representations (ICLR)*, 2020. 5
- [27] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 27171–27183, 2021. 1, 2, 3, 6, 7
- [28] Thomas Whelan, Renato F. Salas-Moreno, Ben Glocker, Andrew J. Davison, and Stefan Leutenegger. ElasticFusion:

- Real-time dense SLAM and light source estimation. *International Journal of Robotics Research (IJRR)*, 35(14):1697–1716, 2016. 1, 2
- [29] Cheng-hsin Wuu, Ningyuan Zheng, Scott Ardisson, Rohan Bali, Danielle Belko, Eric Brockmeyer, Lucas Evans, Timothy Godisart, Hyowon Ha, Alexander Hypes, Taylor Koska, Steven Krenn, Stephen Lombardi, Xiaomin Luo, Kevyn McPhail, Laura Millerschoen, Michal Perdoch, Mark Pitts, Alexander Richard, Jason Saragih, Junko Saragih, Takaaki Shiratori, Tomas Simon, Matt Stewart, Autumn Trimble, Xinshuo Weng, David Whitewolf, Chenglei Wu, Shou-I Yu, and Yaser Sheikh. Multiface: A dataset for neural face rendering. *arXiv preprint arXiv:2207.11243*, 2022. 7
- [30] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 4805–4815, 2021. 1, 2
- [31] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 2492–2502, 2020. 2
- [32] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017. 5
- [33] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 4
- [34] Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. PatchMatch based joint view selection and depthmap estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1510–1517, 2014. 2