IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, Canada, June 2023.

PermutoSDF: Fast Multi-View Reconstruction with Implicit Surfaces using Permutohedral Lattices

Radu Alexandru Rosu Sven Behnke
University of Bonn, Germany
{rosu, 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 visually pleasing 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, especially for 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. Code is publicly available at https://radualexandru.github.io/permuto_sdf

1. Introduction

Accurate reconstruction geometry and appearance of scenes is an important component of many computer vision tasks [16,19,25,31]. Recent Neural Radiance Field (NeRF)-like models [3, 16, 18, 28] represent the scene as a density and radiance field and, when supervised with enough input images, can render photorealistic novel views.

Works like INGP [18] further improve on NeRF by using a hash-based positional encoding which results in fast training and visually pleasing results. However, despite the photorealistic renderings, the reconstructed scene geometry 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; untextured regions can have arbitrary density in the reconstruction.

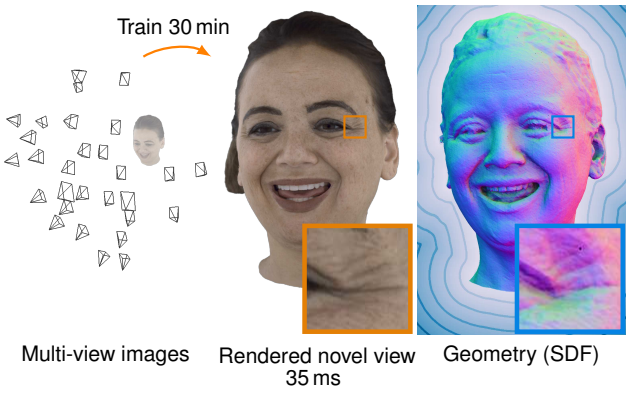

Figure 1. Given multi-view images, we recover both high quality geometry as an implicit SDF and appearance which can be rendered in real-time.

Another line of recent methods tackles the issue of incorrect geometry by representing the surfaces of objects as binary occupancy [22] or Signed Distance Function (SDF) [30, 33]. This representation can also be optimized with volumetric rendering techniques that are supervised with RGB images. However, parametrization of the SDF as a single fully-connected Multi-Layer Perceptron (MLP) often leads to overly smooth geometry and color.

In this work, we propose PermutoSDF, 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 [30]. 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. Attempting to regularize the SDF with a total variation loss or a curvature loss will produce a 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 (triangle, tetrahedron, ...) in this data 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 a 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 [8,9,25,37] or volume-based [6,19,21,31]. Depth map methods like COLMAP [25] 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]. NeRFs represent the scene as density and radiance fields, parameterized by a MLP. Volumetric rendering is used to train them to match posed 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 occupied 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 [27] 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 Neural Graphics Primitives (INGP) [18] proposes a hash-based encoding in which ray samples trilinearly interpolate features between eight 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 [34] 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 binary occupancy function through volumetric rendering. VolSDF [33] extends this idea to SDFs. NeuS [30] 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 reconstruct a scene as SDF and color field 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.

3. Method 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.

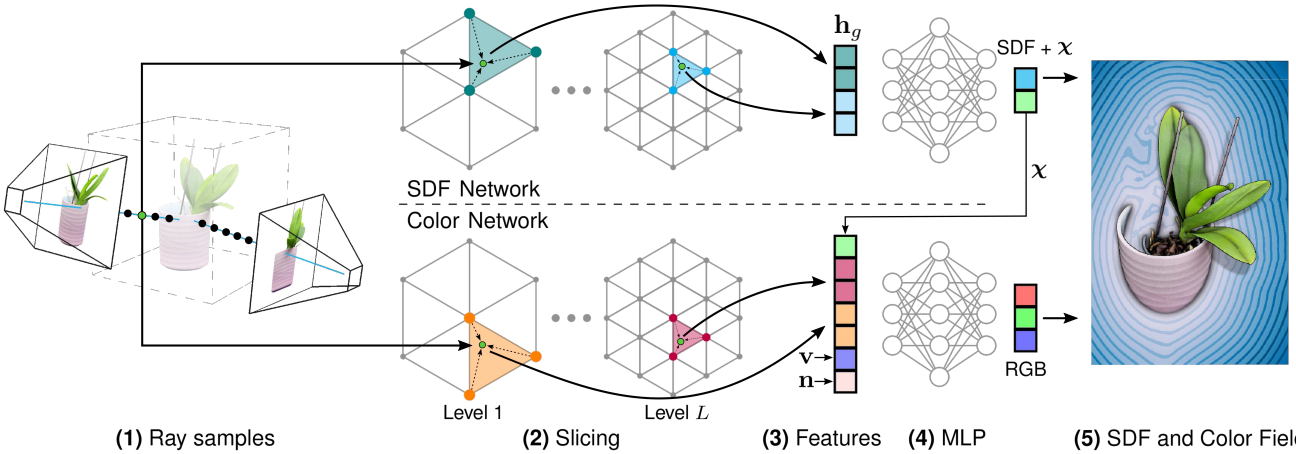

Figure 2. Overview of our PermutoSDF pipeline. **(1)** For a batch of pixels from the posed images, we sample rays inside the volume of interest. **(2)** For each sample, we slice features from a multi-resolution permutohedral lattice. **(3)** The features from all lattice levels are concatenated. For the color network, we also concatenate additional features regarding normal $\mathbf{n}$ of the SDF, view direction $\mathbf{v}$, and learnable features χ from the SDF network. **(4)** Small MLPs decode the SDF and a view-dependent RGB color. **(5)** The output is rendered volumetrically and supervised only with RGB images. We visualize surface color and a 2D slice of the SDF.

We define the surface $\mathcal{S}$ as the zero level set of an SDF:

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

The SDF is parameterized by a fully connected neural network $g(\mathbf{h}_g; \Phi_g)$ that processes an encoding $\mathbf{h}_g = \text{enc}(\mathbf{x}; \theta_g)$ of the input position $\mathbf{x}$. 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, we define an MLP $c(\mathbf{h}_c, \mathbf{v}, \mathbf{n}, \boldsymbol{\chi}; \Phi_c)$ for the color which processes an encoded position $\mathbf{h}_c = \text{enc}(\mathbf{x}; \theta_c)$, a view direction $\mathbf{v}$, the normal vector of the SDF $\mathbf{n}$, and a learnable geometric feature $\boldsymbol{\chi}$ 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-resolutional hash-based permutohedral lattice (cf. Sec. 4.2). The lattice features from different levels are concatenated and processed by the color and SDF MLPs. The SDF values are mapped to density (cf. Sec. 4.1) and the sample colors are rendered volumetrically 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. Permutohedral Lattice SDF Rendering

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

4.1. Volumetric Rendering

We denote the ray emitted from a 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

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

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

In NeuS [30], 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 $\rho(t)$:

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

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

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

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 encoding maps a spatial position $\mathbf{x}$ into a cubical grid and linearly interpolates features from the hashed eight corners

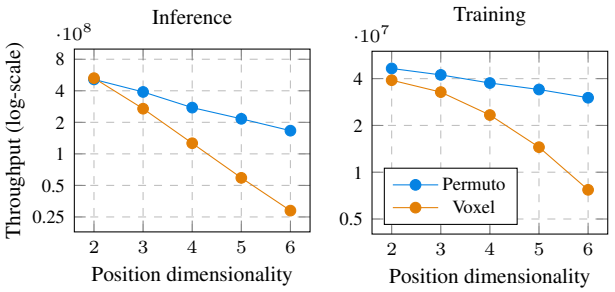

Figure 3. We use a permutohedral lattice instead of hyper-cubical voxels since the number of vertices per simplex scales linearly with the dimensionality instead of exponentially. The permutohedral lattice trains faster and encodes points faster for dim. ≥ 3.

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. This advantage grows for higher dimensions when 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 $\mathbf{h} = \text{enc}(\mathbf{x}; \theta)$ of the input $\mathbf{x}$ given lattice features θ.

4.3. 4D Background Estimation

For modeling the background, we follow the formulation of NeRF++ [36] 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-length 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. PermutoSDF Training and Regularization

Given the permutohedral lattice hash encoding and the unbiased volumetric rendering scheme, we have all the tools to train our model. We sample pixels from the input images and infer SDF and color for positions along their rays. Through volumetric rendering (Eq. 2) we obtain the pixel color $\hat{C}(\mathbf{p})$. We optimize an L2 loss on the RGB pixels:

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

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

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

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 3×3 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 $\boldsymbol{\eta}$ by cross product with a random unit vector $\boldsymbol{\tau}$ such that $\boldsymbol{\eta} = \mathbf{n} \times \boldsymbol{\tau}$. Given this random vector in the tangent plane, we slightly perturb our sample $\mathbf{x}$ to obtain $\mathbf{x}_\epsilon = \mathbf{x} + \epsilon\boldsymbol{\eta}$. 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 points:

$$\mathcal{L}_{\text{curv}} = \sum_{x} (\mathbf{n} \cdot \mathbf{n}_\epsilon - 1)^2. \tag{7}$$

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 $\mathcal{L}_{\text{rgb}}$ to zero. Despite lowering $\mathcal{L}_{\text{curv}}$ 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 $c(\mathbf{h}, \mathbf{v}, \mathbf{n}, \boldsymbol{\chi}; \Phi_c)$, with an input encoding of $\mathbf{h} = \text{enc}(\mathbf{x}; \theta_c)$. We observe that all the high-frequency detail learned by the color network has to be present in the weights of the MLP Φ_c or the hash-map table θ_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:

$$\underbrace{\|f(d) - f(e)\|}_{\text{change in the output}} \leq k \underbrace{\|d - e\|}_{\text{change in the input}}. \tag{8}$$

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, 29, 35]. 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 of 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:

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

where $\text{softplus}\left(k_i\right) = \ln(1+e^{k_i})$ 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 $\text{softplus}\left(k_i\right)$. Since the product of per-layer Lipschitz constants k_i is the Lipschitz bound for the whole network, we can regularize it using:

$$\mathcal{L}_{\text{Lipschitz}} = \prod_{l}^{i=1} \text{softplus}\left(k_i\right). \tag{10}$$

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

5.3. Training Schedule

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

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

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 a. Instead, we use a scheduled linear decay $1/a$ over 30 k iterations which we found to be robust for all the objects we tested.

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

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_1 \mathcal{L}_{\text{eik}} + \lambda_2 \mathcal{L}_{\text{curv}}. \tag{11}$$

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

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 10 k iterations. We refer to the supplementary material for more details regarding the hyperparameters λ_{1-3}.

6. Acceleration

Similar to other volumetric rendering methods, a major bottleneck for the speed is the number of position 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.

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

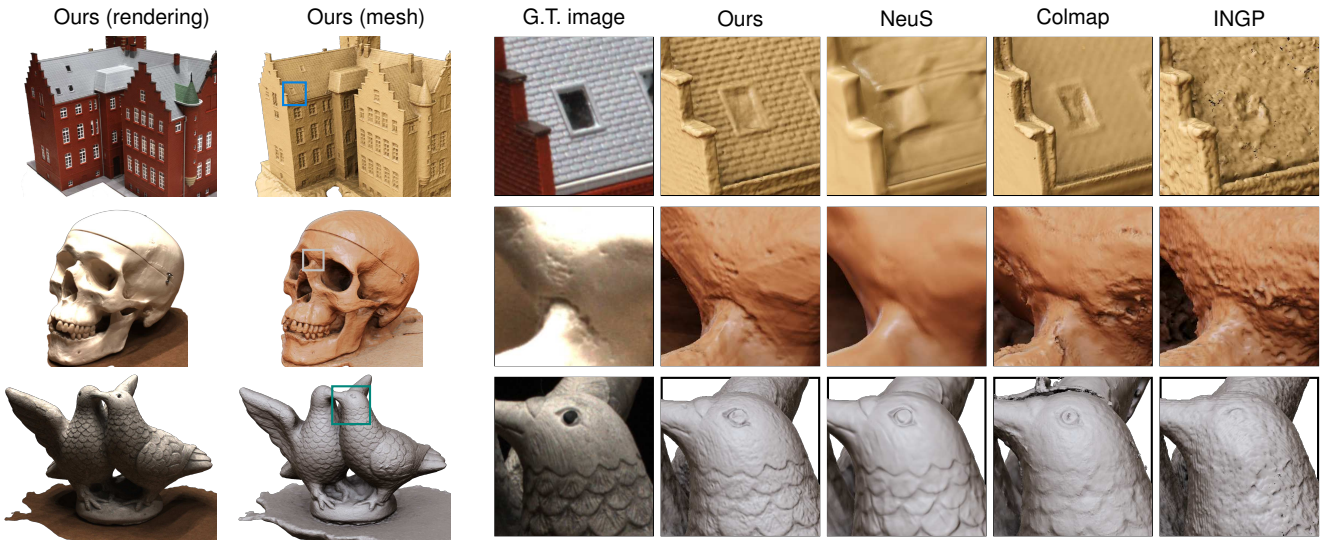

Figure 5. Qualitative comparison of the geometry reconstructed by our method compared to the baselines. Note that our method recovers significantly higher geometrical detail.

equations to determine if a certain voxel has enough 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 Details

We implement the encoding $\mathbf{h} = \text{enc}(\mathbf{x}; \theta)$ using permutohedral lattices in a custom CUDA kernel which slices in parallel from all resolutions. The backward pass for updating the hashmap $\frac{\partial \mathbf{h}}{\partial \theta}$ is also implemented in an optimized CUDA kernel. We use the chain rule to backpropagate the upstream gradients as: $\frac{\partial \mathcal{L}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \theta}$.

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 $\mathbf{h} = \text{enc}(\mathbf{x}; \theta_g)$ w.r.t. to spatial position $\mathbf{x}$, i.e. $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$. Again, the chain rule is applied with the autograd partial derivative of $g(.)$ as: $\frac{\partial g}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ to obtain the normal.

Furthermore, since we use this normal as part of our loss function $\mathcal{L}_{\text{eik}}$, 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 $\partial(\frac{\partial \mathcal{L}}{\partial \mathbf{x}})/\partial(\frac{\partial \mathcal{L}}{\partial \mathbf{h}})$ 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 Data Set

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 a wide 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 [30] which is our baseline, INGP [18] which is state-of-the-art in NVS rendering, and COLMAP [25], a classical multi-view stereo method.

We show qualitative results of the extracted meshes 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 supervi-

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

	w/ mask			**w/o** mask			
ScanID	INGP [18]	NeuS [30]	Ours	COLMAP [25]	INGP [18]	NeuS [30]	Ours
scan24	1.73	0.83	**0.53**	0.81	1.56	1.00	**0.52**
scan37	1.79	0.98	**0.67**	2.05	2.15	1.37	**0.75**
scan40	1.46	0.56	**0.34**	0.73	1.45	0.93	**0.41**
scan55	0.86	**0.37**	0.37	1.22	0.76	0.43	**0.37**
scan63	1.70	1.13	**0.94**	1.79	1.62	1.10	**0.90**
scan65	1.57	**0.59**	0.59	1.58	1.33	**0.65**	0.66
scan69	1.66	0.60	**0.57**	1.02	1.63	**0.57**	0.59
scan83	1.56	1.45	**1.22**	3.05	1.79	1.48	**1.37**
scan97	1.83	0.95	**0.78**	1.40	2.16	1.09	**1.07**
scan105	1.55	0.78	**0.66**	2.05	1.45	**0.83**	0.85
scan106	1.23	0.52	**0.49**	1.00	1.25	0.52	**0.46**
scan110	1.75	1.43	**0.73**	1.32	1.91	1.20	**0.98**
scan114	1.71	0.36	**0.35**	0.49	1.76	0.35	**0.33**
scan118	1.44	0.45	**0.41**	0.78	1.24	0.49	**0.39**
scan122	1.31	0.49	**0.47**	1.17	1.47	0.54	**0.50**
mean	1.54	0.77	**0.61**	1.36	1.57	0.84	**0.68**

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

ScanID	NeuS [30]	NeRF [16]	INGP [18]	Ours	ScanID	NeuS [30]	NeRF [16]	INGP [18]	Ours
scan24	26.76	27.54	28.77	**30.06**	scan97	29.30	**30.46**	29.43	30.45
scan37	25.84	26.54	26.34	**27.29**	scan105	34.50	35.51	36.20	**36.85**
scan40	27.25	28.53	28.97	**30.43**	scan106	34.12	34.86	35.05	**36.27**
scan55	28.09	30.39	31.20	**32.45**	scan110	32.46	32.87	32.16	**34.52**
scan63	34.24	35.25	**36.72**	36.32	scan114	30.01	30.82	31.04	**31.26**
scan65	33.83	33.42	**34.13**	34.00	scan118	36.73	36.87	37.91	**38.70**
scan69	29.94	30.22	29.63	**30.49**	scan122	37.89	37.77	38.64	**39.74**
scan83	39.02	40.12	40.29	**40.81**	Mean	31.99	32.74	33.10	**33.97**

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

sion. In both cases, our method significantly outperforms the competing approaches.

Additionally, we evaluate the quality of novel-view synthesis on the same dataset in Tab. 2. One can observe that we surpass NeuS, NeRF, and in most cases also 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 Data Set

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 datset [32]. It consists of human subjects that were captured with a dome of $\approx$ 40 high-resolution cameras while performing various facial expressions.

Fig. 7 shows a qualitative comparison between our reconstruction and NeuS. 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. Rendered Head Images

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 realistic images of a head figure [4] using the EasyPBR renderer [24]. 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 for inference and training and compare it to the cubical voxels used in INGP. We encode a batch of 2^{19} random points and set both hash maps to a capacity $T = 2^{18}$, $L = 24$ levels, and $F = 2$ features per entry. In Fig. 3 one can observe that our permutohedral lattice outperforms the cubical lattice during training and for dim. >2 during inference; with a larger gap for higher dimensions. This is to be expected since the number of vertices per simplex scales linearly with the dimensionality instead of exponentially. The only exception is in 2D, where a square lattice accesses four vertices per simplex while we access three—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 PermutoSDF in Fig. 8. While sphere initialization and coarse-to-fine optimization help 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 issues but also severely over-smoothes the object and results in a higher Chamfer distance. Adding RGB regularization via the Lipschitz loss is crucial to recover high-frequency details and obtaining the lowest Chamfer distance.

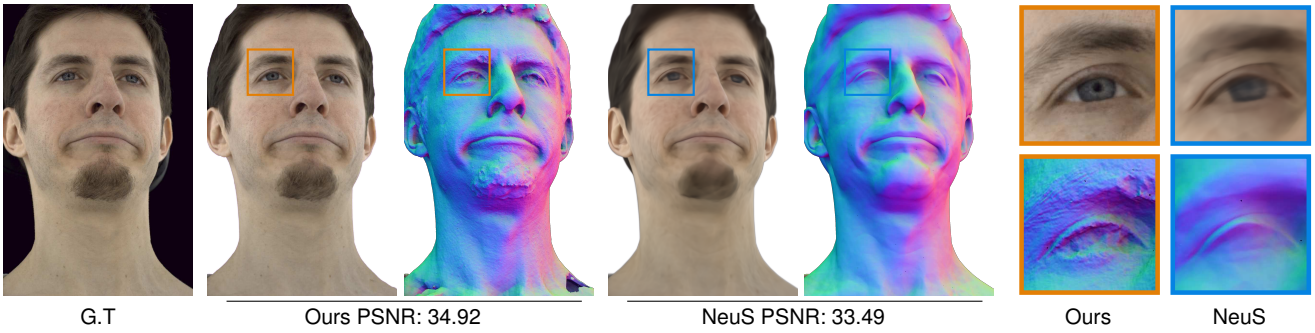

Figure 7. Qualitative comparison between our method and NeuS on the Multiface dataset. We can recover finer detail with higher PNSR.

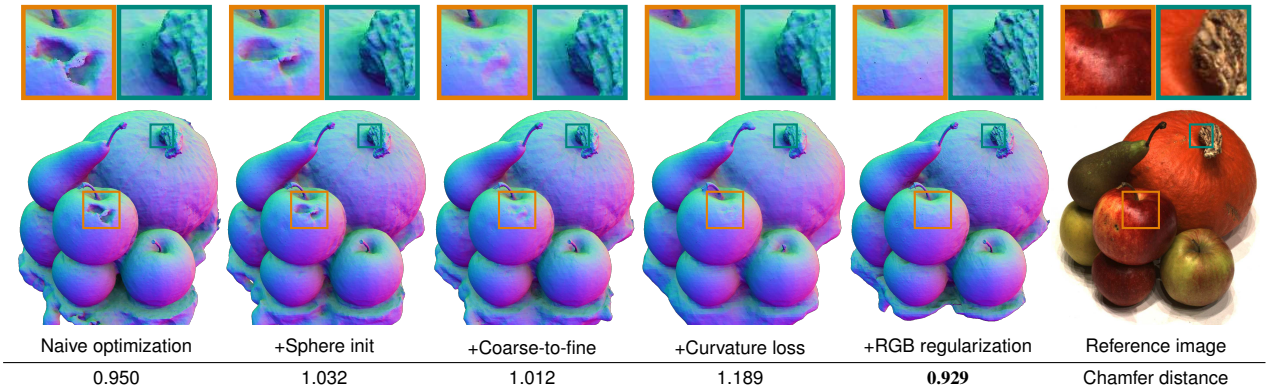

Figure 8. Ablation study of the various components of our method. A naive combination of hash-based encoding and implicit surfaces can lead to undesirable holes and overly smooth geometry. Adding sphere initialization and coarse-to-fine optimization helps with recovering smoother surfaces especially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

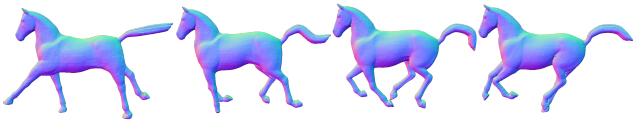

Figure 9. 4D surface. Using our lattice we can efficiently learn surfaces that evolve in time. Here we visualize the learned geometry of a 4D model while we sweep through the time dimension.

7.6. 4D Spatio-temporal Surface

Since our lattice representation scales better in higher dimensions, we also include an experiment of encoding the surface of an object evolving through time as shown in Fig. 9. We directly fit the geometry of the 4D model by supervising with batches of orientated point samples from the surfaces of animated meshes. The loss function is similar to the one introduced in SIREN [26] and we refer to the supplemental for more details. Our method successfully learned the evolving shape and can generate intermediate shapes by sweeping through time.

We note that learning 4D directly from images, similar to D-Nerf [23], is also possible. However, since the extension of the occupancy grid to 4D is not trivial and additional losses may also be needed to ensure smoothness in the time dimensions, we leave this for future work.

8. 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 always faster in training and faster for inference in three and higher dimensions. Additionally, we proposed a simple regularization scheme that allows to recover fine geometrical detail at the level of pores and wrinkles. Our full system can train in $\approx 30\,\mathrm{min}$ on an RTX 3090 GPU and render in real-time using sphere tracing. We believe this work together with the code release will help the community in a multitude of other tasks that require modeling signals using fast local features.

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

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. *Technical Report, Stanford University*, 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. In *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. Plenoxels: 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] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-Nerf: Neural radiance fields for dynamic scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327, 2021. 8

[24] Radu Alexandru Rosu and Sven Behnke. EasyPBR: A lightweight physically-based renderer. In *16th International Conference on Computer Graphics Theory and Applications (GRAPP)*, 2021. 7

[25] 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

[26] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 7462–7473, 2020. 8

[27] 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

[28] 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

[29] Dávid Terjék. Adversarial Lipschitz regularization. In *8th International Conference on Learning Representations (ICLR)*, 2020. 5

[30] 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

[31] 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

[32] 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, Shoou-I Yu, and Yaser Sheikh. Multiface: A dataset for neural face rendering. *arXiv preprint arXiv:2207.11243*, 2022. 7

[33] 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

[34] 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

[35] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017. 5

[36] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 4

[37] 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

PermutoSDF: Fast Multi-View Reconstruction with Implicit Surfaces using Permutohedral Lattices Supplementary Material

Radu Alexandru Rosu Sven Behnke
University of Bonn, Germany
{rosu, behnke}@ais.uni-bonn.de

S1. Training Details

For the first 100 k iterations, we train using the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_1 \mathcal{L}_{\text{eik}} + \lambda_2 \mathcal{L}_{\text{curv}}, \quad (1)$$

where $\lambda_1 = 0.05$, $\lambda_2 = 1.5$. For the remaining 100 k iteration, we remove $\lambda_2 \mathcal{L}_{\text{curv}}$ and replace it with $\lambda_3 \mathcal{L}_{\text{Lipschitz}}$, where $\lambda_3 = 1\text{e}{-5}$.

For 3D point sampling, we first create 64 uniform samples along each ray. We restrict the samples to be within the region that is defined as occupied by the occupancy grid. Afterwards, we run two iterations of importance sampling, each creating an additional 16 samples in the regions that are close to the surface. Concentrating samples close to the surface is crucial for recovering detail.

S2. Synthetic Data Comparison

We train also NeuS [4] and INGP [2] on the synthetically rendered head dataset described in Sec 7.3. The recovered meshes are shown in Fig. 1.

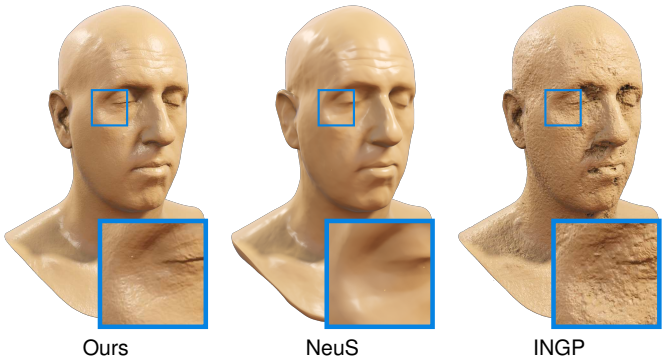

Figure 1. We train our method, NeuS [4], and INGP [2] on the synthetically rendered dataset, described in Sec 7.3. We recover significantly more small detail than the other two methods. Best viewed zoomed-in.

S3. Rendering Strategy

We compare images rendered through volumetric integration to the ones using sphere tracing. We observe that sphere tracing has the advantage of being significantly faster, as most rays converge towards the surface in few iterations. However, grazing surfaces require an arbitrary number of iterations and since we use a maximum of 20 iterations, these grazing surfaces may exhibit artifacts. A comparison between volumetric rendering and sphere tracing is shown in Fig. 2.

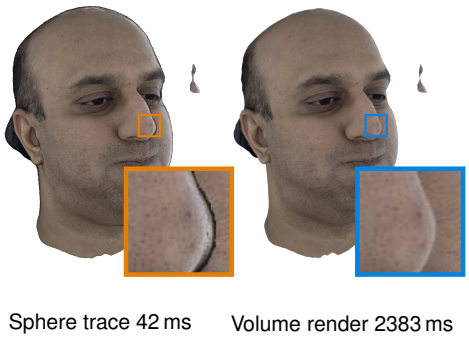

Figure 2. Sphere tracing is significantly faster than volumetric rendering, but it suffers from artifacts at surfaces with a grazing angle.

S4. Tetrahedron vs Cube

Apart from the speed improvements of using a permutohedral lattice instead of a hyper-cubical one, we are also interested on maintaining the encoding quality and therefore the reconstruction details. We reconstruct the same scene with both permutohedral encoding and cubical encoding as described in INGP [2]. We set the hash maps of both approaches to the same number of parameters, features per layer, and levels. We also extended the cubical lattice with the coarse-to-fine optimization in order match the optimiza-

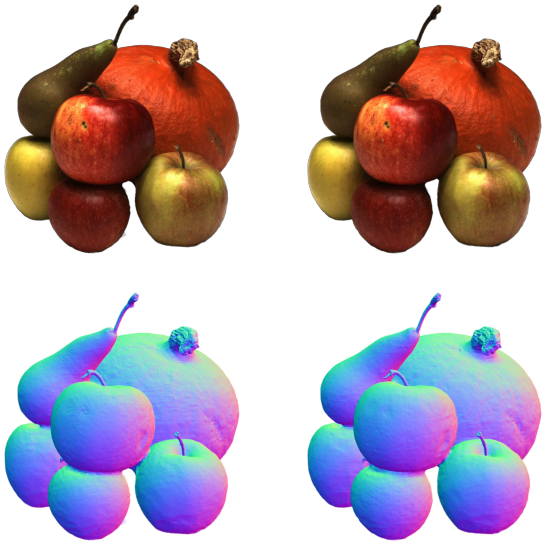

Figure 3. We reconstruct the same scene using cube encoding and permutohedral encoding. We did not observe significant differences in the reconstruction quality.

tion behavior of the permutohedral lattice. In Fig. 3, we show both reconstructions and compare their Chamfer distance and PSNR values for novel-view synthesis. We did not observe a significant difference in the reconstruction quality.

S5. Occupancy Grid Update

We initialize an occupancy grid with all the voxels being occupied and with an initial SDF that is constant zero. This ensures that we sample everywhere at the beginning of training.

For updating the occupancy grid, we use the following steps:

- Every 8th iteration of training, we sample 2^{18} random points within the bounding box that contains the scene.
- We obtain the SDF value $s_{\mathbf{x}}$ for each point $\mathbf{x}$ by running a forward pass through the model.
- We obtain the old SDF value s_{old} stored for the voxel in which the point falls into.
- We compute a new SDF value for this voxel s_{new} as the exponential average of the old SDF for the voxel and the SDF for the point: $s_{new} = s_{old} + 0.3(s_{\mathbf{x}} - s_{old})$.
- Since we are discretizing the SDF to a grid, and we don't want to miss any possible low SDF values that we would want to sample, we compute the minimum possible SDF that can be reached within this voxel under the assumption of perfect Eikonal loss. For this, we use: $s_{min} = max(0, |s_{new}| - d)$, where d is the length of the voxel diagonal.
- Using the logistic density distribution as described in NeuS [4], we compute the weight that this sample would contribute to the volumetric render—assuming no obstruction from other samples: $w = a \cdot e^{-a \cdot s_{min}} / (1 + e^{-a \cdot s_{min}})^2$.
- If the weight w falls bellow a specified threshold, we set the voxel to unoccupied and therefore don't create samples within it anymore.

S6. Color Calibration

We observe that some datasets exhibit images with different exposure times. This discrepancy between images can influence both the reconstruction and the obtained color field as the network would try to explain the variability with view-dependent effects. We circumvent this by learning a per-camera gain $g = (1 + \Delta g)$ and bias b so that the reconstructed color for each camera is $c = \sigma(\hat{c} \cdot g + b)$, where $\hat{c}$ is the raw color output from the network and σ is a sigmoid function that restricts the color to the correct range. We set a selected camera (usually the first one from the dataset) to have $g = 1$ and $b = 0$ and apply weight decay to Δg and b to further ensure that the calibration doesn't alter the colors unnecessarily.

S7. 4D Spatial-temporal Surface

For fitting a 4D surface, we sample random points from animated 3D meshes. These 3D points are concatenated with a time dimension that ranges from 0 to 1, where 0 is the start time of the animation and 1 is the end. We define these 4D samples at the surface of the mesh as $\mathbf{x}_s$. We also compute the normal $\mathbf{n}_s$ for each on the surface samples. We additionally define random 4D samples in a bounded domain around the animated mesh which we denote with $\mathbf{x}_r$ Afterwards, we learn a model $g(\mathbf{h}; \Phi)$ together with an encoding $\mathbf{h} = \text{enc}(\mathbf{x}; \theta)$ that maps from the 4D coordinate to an SDF value. For this, we follow the approach of SIREN [3] and use a loss of the form:

$$\begin{aligned} \mathcal{L}_{\text{sdf}} = & \sum_{\mathbf{x}_s \cup \mathbf{x}_r} \left(\|\nabla g(\text{enc}(\mathbf{x}))\| - 1 \right)^2 \\ & + \sum_{\mathbf{x}_s} \|g(\text{enc}(\mathbf{x})\| \\ & + \sum_{\mathbf{x}_s} (\nabla g(\text{enc}(\mathbf{x})) \cdot \mathbf{n}_s - 1) \\ & + \sum_{\mathbf{x}_r} exp(-\alpha \cdot |g(\text{enc}(\mathbf{x}))|). \end{aligned} \tag{2}$$

Figure 4. We experiment with the number of input images for our method. We observe significant degradation at around 7 input images and a failure to converge at 3 images.

In this 4D experiment, no explicit smoothness was enforced in the temporal domain since we didn't find it necessary. We sample from an animation of 100 frames so the temporal resolution is relatively high. At lower temporal resolution, smoothness might again become a concern.

Nevertheless, this approach shows that our model can deal with 4D representations onto which further ideas, like dynamic deformation fields, can be built upon.

S8. Number of Cameras

In order to study the robustness of our method to the number of input images, we vary the number of images used for reconstruction as shown in Fig. 4. Due to the curvature loss, our method can recover smooth but plausible surfaces even with as low as seven input images. However, for a smaller number of input images we observe a high likelihood of not converging to the correct surface.

S9. Training schedule

We follow a fixed training schedule over 200k iterations. This includes a phase where we train with curvature loss in order to recover the rough shape, and another phase with RGB regularization to recover detail. In order to study the robustness of our method to this schedule, we expand and contract the schedule to be as long as 300k iteration or as short as 50k iterations and show the results in Fig. 5. By modifying the schedule, we proportionally expand or contract the time that is spent optimizing the sphere, training with high curvature, and training with RGB regularization. We observe that the model is quite robust to different schedules and only for the very short ones it fails to recover some of the geometry. In general, we found that view-dependent effects like the highlight on the apple are the parts that take the longest to converge to good geometry. Most objects are reconstructed well with shorter schedules but our default schedule of 200k iteration is a good trade-off between optimization speed and accuracy.

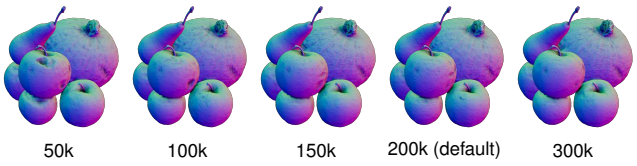

Figure 5. We follow a fixed training schedule that finishes after 200k iteration. We expand and contract this fixed schedule to be shorter or longer in order to test robustness. We see that for a schedule of 50k the method fails to reconstruct the geometry for the highlight of the apple. Our default of 200k can recover good geometry in reasonable time. A longer schedule results in better reconstructions at the cost of more optimization time.

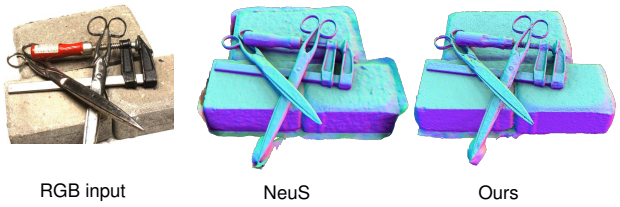

Figure 6. We observe that our model sometimes struggle with very reflective surface like the metal on the scissors. It tends to add noisy geometry to these surfaces in order to explain the view-dependent effects. Object priors or a higher curvature loss for this kind of objects could alleviate the issue.

S10. Reflective Surfaces

Our model tries to explain large color changes with changes in geometry. This behavior can be detrimental in the case of mirror-like surfaces. As we observe in Fig. 6, NeuS recovers a smooth surface on the metal scissors while our method exhibits more noise. This can be seen as a general limitation of RGB reconstruction methods for which it is difficult to know if the changes in color are from view-dependent effects or from high-frequency geometry. A model that learns object priors might perform better in these cases.

S11. Thin Structures

An interesting case to test for our SDF-based method is reconstructing thin structures. For this, we capture 14 images of a plant with relatively complex geometry with many leafs and self occlusions. Fig. 7 shows a rendered novel view and surface normals. Our method reconstructs accurate color and plausible geometry. Despite some errors that are to be expected given the low image count, it can recover thin steams and leaves which shows that our method is robust to this kind of data.

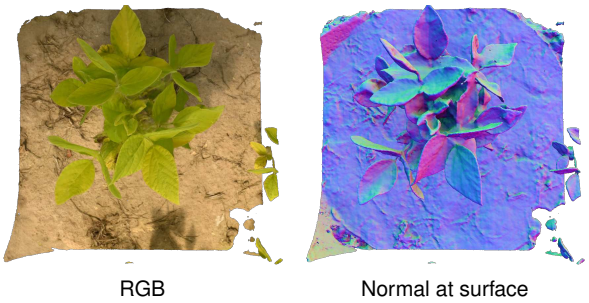

Figure 7. Plant reconstruction is an especially difficult case since it features many self-occlusions and thin structures. We observe that our method can deal well with this kind of data despite using only 14 images as input.

S12. Qualitative DTU Results

In Fig. 8 – Fig. 10, we show additional qualitative results from the DTU dataset [1]. We show extracted meshes and error maps which represent the distance from each mesh vertex towards the nearest point from the ground-truth. Please note that the ground-truth can have holes in areas of high reflectance or self-occlusion and this shows as a bright yellow color in the error map.

References

[1] 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. 4

[2] 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

[3] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 7462–7473, 2020. 2

[4] 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

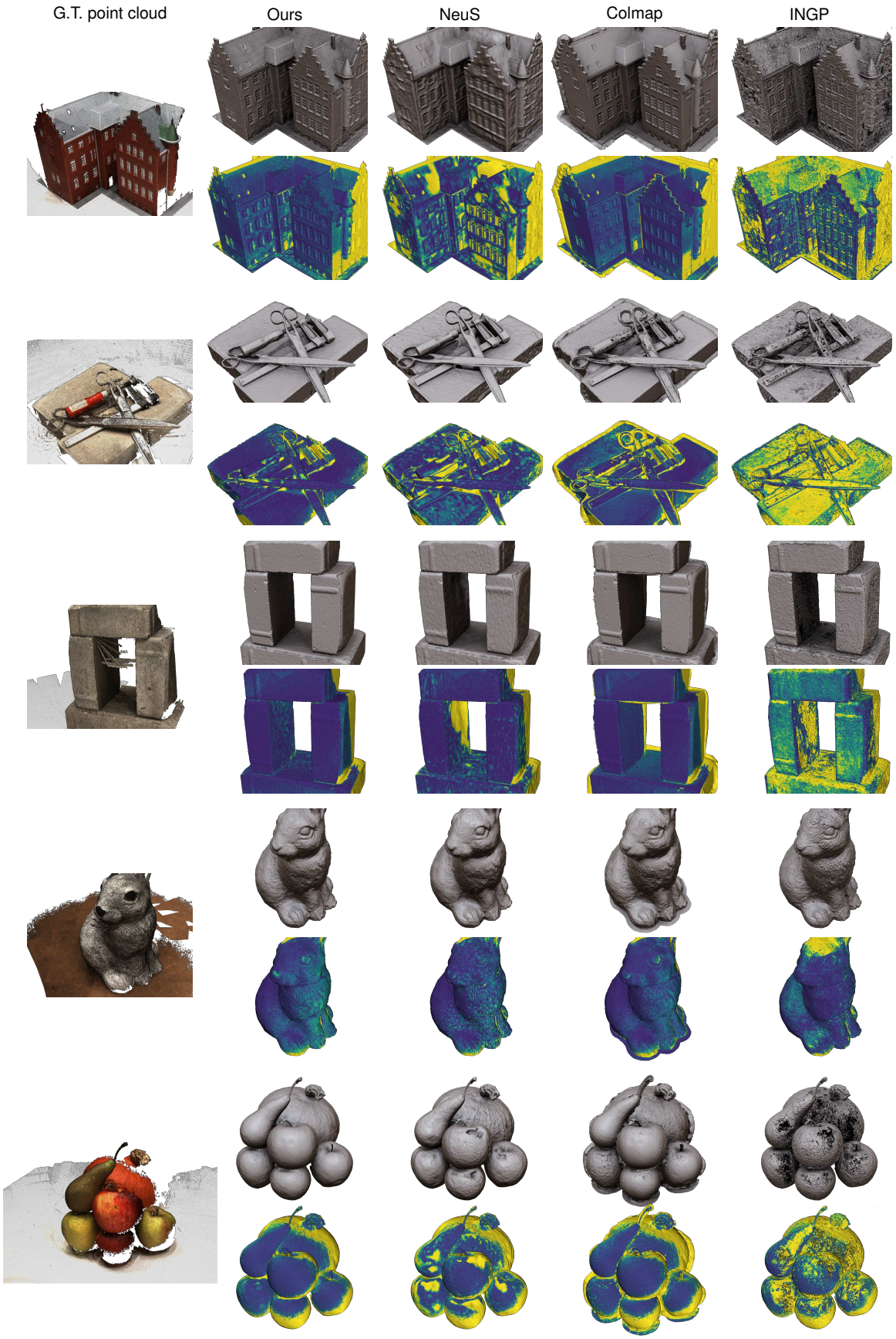

Figure 8. DTU qualitative comparison of extracted meshes and error maps.

Figure 9. DTU qualitative comparison of extracted meshes and error maps.

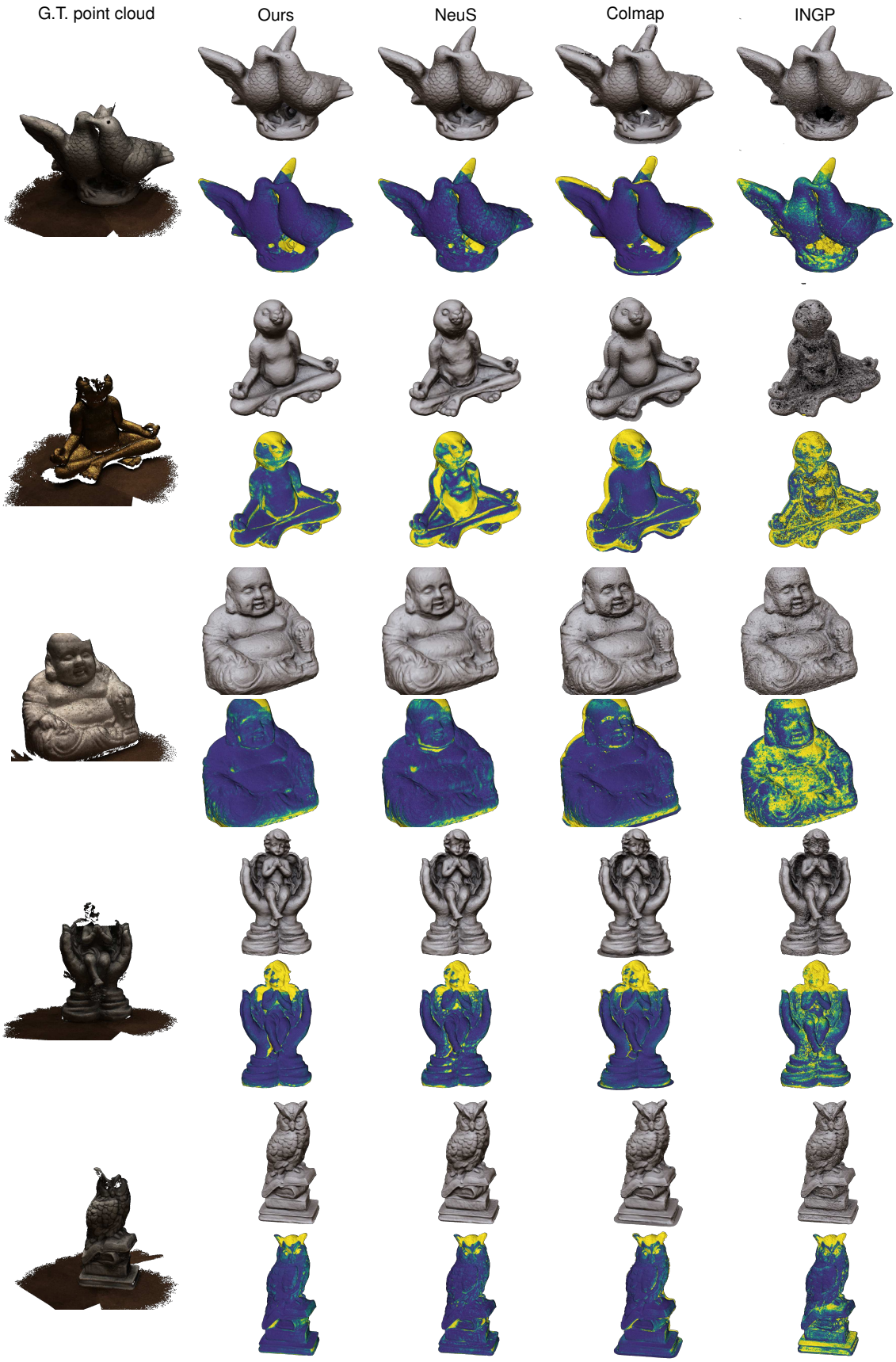

Figure 10. DTU qualitative comparison of extracted meshes and error maps.