Generalizable Neural Reconstruction of High-Fidelity Surfaces via Sparse Volumetric Representations

Aoxiang Fan[1], Corentin Dumery[1], Nicolas Talabot[1], Ming Xu[1], Hieu Le[2], and Pascal Fua[1]

[1] CVLab, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{aoxiang.fan, corentin.dumery, nicolas.talabot, mingda.xu, pascal.fua}@epfl.ch
[2] University of North Carolina at Charlotte, USA
hle40@charlotte.edu

Abstract. Neural implicit representations have recently achieved impressive results in novel view synthesis and multi-view 3D reconstruction, yet both NeRF- and Gaussian Splatting-based methods require per-scene optimization, which makes them inefficient. Generalizable Neural Surface Reconstruction (GNSR) methods have been proposed to remove this need by learning feature representations directly predicted from input images. However, their typical reliance on dense feature volumes severely limits achievable resolution and fidelity due to prohibitive memory costs. We introduce Sparse Volumetric Reconstruction (SVRecon), a new GNSR framework that unlocks high-resolution, memory-efficient reconstruction through learned occupancy-driven sparsity, in a more effective way than earlier approaches to introducing sparsity in GNSRs. Our approach uses a nested two-stage architecture: (1) an occupancy prediction network that identifies surface-containing voxels, and (2) a high-resolution sparse volume rendering framework defined only within these occupied regions, together with specialized sparsified algorithms for ray sampling, feature aggregation, and querying. This design enables fine-grained surface reconstruction while avoiding the heavy memory footprint of dense grids. SVRecon operates at resolutions up to 512^3 on standard 32GB hardware—substantially higher than prior generalizable methods—and delivers smoother and more precise reconstructions across diverse datasets, particularly in sparse-view settings.

1 Introduction

The emergence of neural implicit representation techniques, starting with the seminal Neural Radiance Fields (NeRF) [25] and continuing with Neural Implicit Surfaces (NeuS) [19, 36, 43, 44], has greatly boosted novel view synthesis and multi-view geometry reconstruction. These implicit surface reconstruction algorithms take posed images as input, and optimize a Signed Distance Function (SDF) to minimize a volume rendering loss, which yields accurate 3D reconstructions given enough views. A major drawback, however, is that the optimization

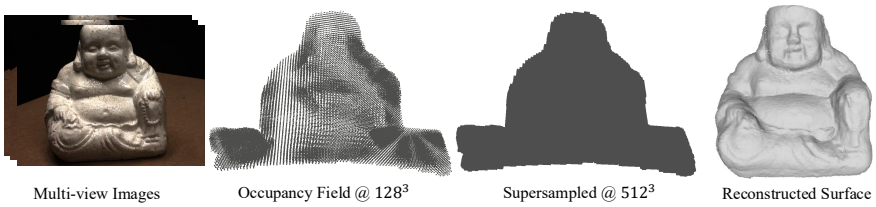

Fig. 1: Two-Stage Approach. From multiple images, we predict occupancy field that is then supersampled to build a sparse scene representation. This lets us reconstruct high-fidelity surfaces by operating at previously unattainable 512^3 resolution.

must be performed from scratch for every new set of views, which makes these approaches computationally demanding. Furthermore, accuracy tends to decline when only a few views are available. Newer 3D Gaussian Splatting-based methods [14] are subject to the same limitations [10, 47].

Generalizable approaches to Neural Surface Reconstruction (GNSRs) [20, 23, 26, 29, 38, 45] are designed to remove these restrictions. These methods are pre-trained on a large number of scenes. As a result, the rendering of a novel scene simply becomes a test-time inference procedure, without any optimization required. Volume rendering and scene reconstruction are performed by relying on a 3D scene representation produced by a neural network that takes as input the images. This representation is typically in the form of a dense volume of high-dimensional features. However, even though they are much faster, these methods have yet to achieve their full potential because the necessary volumetric feature representation is extremely memory-intensive. This drastically constrains their operational resolution, and adversely impacts reconstruction quality and detail preservation.

Sparse scene representation is a natural answer to this problem. Since surfaces are mathematically of measure zero, they occupy only a small fraction of voxels in a discretized 3D space, which may be leveraged for high-resolution, memory-efficient reconstruction. However, this brings significant challenges. Existing approaches to sparsification [23, 27] require a complex and cumbersome pre-rendering process to infer sparsity followed by using densified feature volumes in the final rendering stage, which negates many of the benefits of sparsification. In contrast, our occupancy-prediction network offers fine-scale sparsity prediction in a simple feed-forward process. Moreover, our single-scale occupancy field can be combined with our specialized algorithms for efficient sparse volume rendering. As a result, our model can operate at a 512^3 resolution, which existing methods cannot do as shown in Tab. 1.

In general terms, we propose a novel and effective way of bringing sparsity into GNSRs with occupancy predictions. Specifically, we introduce a nested two-stage architecture that enables our GNSR to operate at significantly higher resolutions than previous ones, while maintaining both generalizability across scenes and the ability to handle cases where only a limited number of views are available. As illustrated by Fig. 1, the two stages perform the following tasks:

Methods	128^3	192^3	256^3	512^3
SparseNeuS [23]	✓	✓	✗	✗
VolRecon [29]	✓	✗	✗	✗
ReTR [20]	✓	✗	✗	✗
Ours (*SVRecon*)	✓	✓	✓	✓

Table 1: Resolutions handled by GNSR algorithms at training time. Crosses indicate that the memory requirements were too large on a 32GB NVIDIA Tesla V100 GPU, with the same setting of batch size 1 and 1024 sampled rays for all methods.

1. **Computing occupancy at a lower resolution.** Voxels from a coarser grid are labeled as containing a surface element or not, by training a network using posed images as input. We implement this network to significantly sparsify the 3D volume while achieving near-perfect recall.
2. **Volume rendering with sparse representations at a high resolution.** High-dimensional features are constructed at a higher-resolution but only within the occupied voxels, where volume rendering can be performed. This allows for fine-grained neural surface reconstruction while keeping memory and computational costs at a minimum.

Fig. 2 summarizes our *Sparse Volumetric Reconstruction* approach, which we dub *SVRecon*. By scaling up the operating resolution, our main contribution firstly includes demonstrating the significance of 3D sampling resolutions for higher-fidelity reconstruction. In addition, our new method requires rethinking and reformulating the algorithms used by existing methods, which are designed for handling dense representations. Thus, our contribution also includes a dedicated framework for sparse volume rendering, supporting sparse operations including ray sampling, feature aggregation, and querying. These technical innovations are key to realizing the theoretical memory advantages of our approach while maintaining computational efficiency.

Testing on public datasets demonstrates that our sparse-representation method is highly effective, enabling us to operate at a high resolution of 512^3 on standard hardware with 32GB of VRAM. Consequently, our method produces the smoothest and most precise surfaces of all state-of-the-art methods.

2 Related Works

Multi-View Stereo. Multi-view stereo (MVS) has long been known to be effective for 3D reconstruction [6, 7, 9, 16, 17, 18, 32, 33]. Modern multi-view stereo methods can be categorized into depth-map ones [2, 5, 8, 35, 40, 41] and volumetric ones [11, 12, 13]. Depth-map methods rely on view-dependent frustums as an indirect model of 3D space, which limits the reconstruction accuracy without providing novel view rendering capabilities. Volumetric methods are directly related to our method, as they also operate on a volumetric 3D space. Comparatively, our scene representation and volume rendering formulation make it possible to recover finer details.

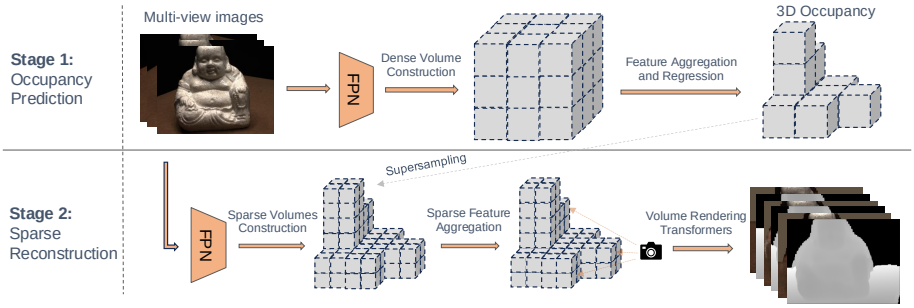

Fig. 2: *SVRecon* **pipeline.** First, we build a low-resolution dense 3D feature volume from multi-view features and predict 3D occupancies. Second, we construct high-resolution sparse feature volumes where the predicted occupancy warrants it. Finally, volume rendering Transformers are used to aggregate per-ray sampled features and infer color and depth, enabling scene reconstruction.

Neural Scene Reconstruction. The success of NeRFs [25] has prompted researchers to examine the use of Neural Implicit Representations for surface reconstruction as an alternative to MVS. This line of work includes IDR [44], VolSDF [43], NeuS [36], and Neuralangelo [19], which are able to recover fine geometry given dense input views. More recently, SparseCraft [45] proposes to use stereopsis cues regularization to enhance surface reconstruction qualities under sparse input views. However, all these methods are not generalizable and need to be trained per scene.

Generalizable Surface Reconstruction. Many generalizable approaches have been proposed to dispense with intensive per-scene optimization. These methods include volumetric methods such as VolRecon [29], ReTR [20]; and frustum-based 3D representations [26, 38]. However, all these methods have to balance reconstruction quality against storage overhead due to the cost of their global scene representations, which is what our method improves upon. Some of these methods [23, 27] incorporate some sparsity handling. However, they depend on a cumbersome pre-rendering process, and still rely on densified feature volumes in the fine-rendering stage, which mostly negates the benefits of sparsification. As a result, our more streamlined approach can achieve a much higher resolution, as shown by Tab. 1, which lists the admissible resolutions the various algorithms can handle.

Gaussian Splatting. Gaussian splatting [14] has emerged as a powerful alternative to NeRF methods for novel view synthesis. The same trend is at work for 3D reconstruction, with the advent of methods such as 2D Gaussian Splatting [10] and Gaussian Opacity Fields [47]. There are also recent attempts at making Gaussian Splatting generalizable [39, 48]. However, they typically struggle to reconstruct smooth surfaces due to the inherent difficulty of modeling surfaces with Gaussians.

3 Background: Generalizable NeRFs

We now briefly review the general formulation of the original and generalizable NeRFs. In both cases, the goal is to reconstruct a scene captured by a set of posed images $\mathcal{I} = \{(\mathbf{I}_i, \mathbf{P}_i)\}_{i=1}^{M}$, where $\mathbf{I}_i \in \mathbb{R}^{H \times W \times 3}$ and $\mathbf{P}_i \in \mathbb{R}^{3 \times 4}$ denote the projection matrix associated with the i-th view.

Given $\mathcal{I}$ as input, the aim is to learn a scene representation function F and a rendering function R

$$F : (\mathbf{x}, \mathbf{r}) \rightarrow \mathbf{f}, \tag{1}$$

$$R : \{(\mathbf{x}_i, \mathbf{f}_i)\}_{i=1}^{N} \rightarrow (\mathbf{c}_i, d_i), \tag{2}$$

where for the inputs, $\mathbf{x}_i \in \mathbb{R}^3$ are 3D locations (typically sampled along a ray extending from the camera frustum) and $\mathbf{r} \in \mathbb{S}^2$ is a viewing direction, and for the outputs, $\mathbf{f} \in \mathbb{R}^C$ is a feature vector, $\mathbf{c}_i \in \mathbb{R}^3$ is a color and $d_i \in \mathbb{R}^+$ is (optional) depth.

In the original NeRF, $\mathbf{f}$ directly encodes an emitted color and a volume density, R is a handcrafted volume rendering function. The scene representation function F is implemented by a network with weights λ. Learning them requires per-scene minimization of the photometric loss

$$\arg\min_{\lambda} \sum_{i=1}^{M} \sum_{\mathbf{p} \in \mathbf{I}_i} L(\mathbf{c}_i, \hat{c}(I_i, \mathbf{p})) \ , \tag{3}$$

where L is usually taken to be square norm of the difference between the arguments, $\mathbf{p}$ are pixel coordinates and $\hat{c}(I, \mathbf{p})$ yields the color at $\mathbf{p}$ in image I.

In a generalizable NeRF, the feature vector $\mathbf{f}$ is high-dimensional and does not have physical meaning. The handcrafted rendering function R is replaced by one implemented by a network with weights ϕ, while F uses a different architecture that allows for generalization and has weights θ. The weights θ and ϕ are learned by training on a large corpus of scenes $\mathcal{D} = \{(\mathcal{I}_i, \mathbf{D}_i)\}_{i=1}^{J}$, where $\mathbf{D}_i \in \mathbb{R}^{H \times W}$ denotes the ground-truth depth map corresponding to image $\mathcal{I}_i$. We minimize

$$\arg\min_{\theta, \phi} \sum_{i=1}^{J} \sum_{j=1}^{|\mathcal{I}_i|} \sum_{\mathbf{p} \in \mathbf{I}_j} \mathcal{L}(\mathbf{c}_{ij}, \hat{c}(\mathbf{I}_{ij}, \mathbf{p})) + \mathcal{L}(d_{ij}, \hat{d}(\mathbf{D}_{ij}, \mathbf{p})). \tag{4}$$

The key advantage over a vanilla NeRF is that, given a novel scene $\mathcal{I}$, the scene representation function F can be obtained via direct inference without the need of per-scene optimization as in Eq. 3. We adopt this general formulation for our method. The main drawback is that having to store a dense feature volume as scene representation and having to perform computations on it is memory intensive, hence the resolution limitations of Tab. 1.

4 Methodology

We now present our *SVRecon* approach to generalizable NeRFs that requires far less memory and can therefore handle larger resolutions. Our main contribution

lies in replacing the scene representation function F of Eq. 1 by one that produces a sparse representation and that we denote as $\mathsf{S} : (\mathbf{x}, \mathbf{r}) \rightarrow \mathbf{f}$. As discussed in the introduction, S operates in two stages. It first estimates occupancy in a low-resolution—-typically 128^3—-volume. The occupied voxels are then supersampled to a higher resolution, typically 512^3, and the high-dimensional features $\mathbf{f}$ are computed within the resulting high-resolution voxels, as depicted by Fig. 2. The sparse representation creates challenges in volume rendering because the ray sampling, querying, and feature aggregations have to be redesigned to handle sparse volumes instead of dense ones, which will also be discussed.

Note that, in theory at least, instead of adopting the simple occupancy prediction described above, we could have used a more sophisticated octree-like structure, a popular choice to represent 3D geometry sparsely. However, the variable-resolution structure of octrees would have made volume rendering even more complex for no obvious gain.

In the remainder of this section, we first introduce the representation we use in both stages for individual voxels and the corresponding features. We then present the two stages to compute our sparse scene representation S. Finally, we discuss the learning of the θ and ϕ parameters by minimizing Eq. 4 given such a sparse representation, under a volume rendering framework.

4.1 Representing Voxels and Features

At low-resolution we use a dense volume while we use a sparse one at high-resolution. Yet, we represent voxels in the same way in both cases. Given the center point $\mathbf{x}$ of a voxel, we first project it onto the images to obtain M-view features

$$\{\mathbf{f}_{I_i}(\pi_i(\mathbf{x}))\}_{i=1}^{M}, \tag{5}$$

by bilinear interpolation, where $\mathbf{f}_{I_i}$ denotes image features computed by using the Feature Pyramid Network [21]. We then write the feature vector associated with $\mathbf{x}$ as

$$\mathbf{V}(\mathbf{x}) = \text{MeanVar}(\{\mathbf{f}_{I_i}(\pi_i(\mathbf{x}))\}_{i=1}^{M}) \ , \tag{6}$$

where MeanVar denotes the concatenation of per-channel mean and variance computed from the M-view features, and $\mathbf{V}$ denotes the raw dense/sparse feature volumes before a 3D global aggregation process. We take the dimension of $\mathbf{V}(\mathbf{x})$ to be the base feature channel C_f.

4.2 Sparse Scene Representation

We now describe the two stages—first occupancy prediction and then supersampling the occupied voxels—involved in building a sparse scene representation.

Occupancy Prediction Assuming that a scene bounding box is given, we voxelize it using a predefined voxel resolution. Due to memory constraints, the resolution cannot be too high. We compute the feature vector using Eq. 6 for each voxel,

resulting in a dense feature volume $\mathbf{V}_d \in \mathbb{R}^{C \times K^3}$, where K denotes the resolution. We then use a 3D U-Net [30] Ψ followed by a linear regression head $\mathbf{U}$ to go from raw features $\mathbf{V}_d$ to the occupancy prediction $\mathbf{O} \in \mathbb{R}^{K^3}$. We write

$$\mathbf{O} = \mathbf{U}(\Psi(\mathbf{V}_d)) \tag{7}$$

To train Ψ and $\mathbf{U}$, we rely on the ground-truth surfaces to estimate ground-truth occupancies $\mathbf{O}^{gt}$. In practice, we generate point clouds by merging the ground truth depth maps of each input view. This works better than using the ground-truth point cloud of the scene, which may contain points that are occluded in the input views and thereby impossible to predict. Since the overwhelming majority of voxels are empty in 3D space, we minimize a focal loss [22] with focusing parameter $\gamma = 2$ during training to counteract the large class-imbalance. We take it to be

$$\mathcal{L}_{fc} = -\sum_{i,j,k} (1 - p_{ijk})^{\gamma} \log(p_{ijk}) \tag{8}$$

where

$$p_{ijk} = \begin{cases} \mathbf{O}_{ijk} & \text{if } \hat{\mathbf{O}}_{ijk} = 1 \\ 1 - \mathbf{O}_{ijk} & \text{if } \hat{\mathbf{O}}_{ijk} = 0 \end{cases}$$

At inference time, we simply threshold the occupancy predictions and apply a dilation process to yield the final binary predictions $\widetilde{\mathbf{O}} = \text{dilate}(\mathbb{I}(\mathbf{O} \geq \tau))$. The predicted occupancy is critical to the performance of our method. We prioritize recall in our method by using a small τ and employing a dilation process to ensure minimal geometry loss. See our supplementary material for more details.

Supersampling. The output $\widetilde{\mathbf{O}} \in \mathbb{R}^{K^3}$ is a low-resolution occupancy field. For scene representation purposes, we *supersample* the occupied voxels by increasing the resolution of them while ignoring the empty ones. Specifically, for a given voxel, the supersampling operation is performed by placing $s \times s \times s$ regular-grid samples within it, lifting the resolution from K^3 to $(sK)^3$. After supersampling, each voxel will be divided into mini-voxels, from which we follow Eq. 6 to construct volumetric representations, resulting in raw 3D sparse feature volumes $\mathbf{V}_s \in \mathbb{R}^{N \times C \times s^3}$, where N denotes the number of occupied voxels. We denote each $s \times s \times s$ block as a mini-volume.

We use the SparseUNet implementation from the sparse convolution library `torchsparse` [34] to perform 3D feature aggregation, as shown in the middle of the bottom row of Fig. 2. The sparse scene representation obtained is $\mathbf{S} = \Psi^{sp}(\mathbf{V}_s)$, where $\mathbf{S} \in \mathbb{R}^{N \times C \times s^3}$ and Ψ^{sp} denotes the 3D SparseUNet.

4.3 Sparse Volumetric Reconstruction

Minimizing the losses of Eq. 4, requires repeatedly evaluating R given sparse scene representation $\mathbf{S}$, which means that R must efficiently handle the sparse nature of our representation. This means that the mechanisms for ray sampling, querying volume locations, and volume rendering used by R have to be redesigned to handle sparse volumes instead of dense ones.

Ray Sampling. Sampling along the ray has to be adapted because the only meaningful samples are those within occupied voxels. For any given ray, we detect its intersections with every occupied voxel and confine the sampling to ray fragments within occupied ones. We take an arbitrary sampled ray to be $\mathbf{r}(t_i) = \mathbf{o} + t_i\mathbf{d}$, $i = 1, 2, ..., N_s$, where t_i denotes the samples, $\mathbf{o}$ and $\mathbf{d}$ denote ray origin and ray direction respectively.

Querying $\mathbf{S}$. The volume rendering process is predicated on the ability to query at arbitrary continuous locations along 3D rays, which is easy to achieve in a dense volume but not in sparse ones due to their irregularity. Here we propose a simple yet effective algorithm to query the sparse feature volumes efficiently as shown in Alg. 1. The key to making our query algorithm very efficient is our specific data structure: our sparse volumes are a collection of small regular volumes spanning the occupied voxels. We pre-compute a dense lookup table $\mathbf{H}$ to encode the order of stored small volumes in memory and then perform the operations below. Please refer to our supplementary material for more details.

Algorithm 1 Querying Sparse Volumes

1: **Input:** Querying point $\mathbf{x}$, lookup table $\mathbf{H} \in K^3$, sparse feature volumes $\mathbf{S} \in \mathbb{R}^{N \times s^3 \times C}$.
2: Find corner points surrounding $\mathbf{x}$, denote their coordinates $\{\mathbf{v}_g^i\}_{i=1}^8$ at scale $(sK)^3$.
3: Convert $\{\mathbf{v}_g^i\}_{i=1}^8$ at scale $(sK)^3$ to $\{\mathbf{v}_o^i\}_{i=1}^8$ at scale K^3 plus local shift $\{\mathbf{v}_l^i\}_{i=1}^8$ at scale s^3.
4: Get corner features from memory by $\mathbf{S}[\mathbf{H}[\mathbf{v}_o^i], \mathbf{v}_l^i, :]$.
5: Perform trilinear interpolation with corner features.
6: **Output:** Interpolated feature $\mathbf{f}_{vol}$.

Rendering Function R. We adopt the generalized volume rendering equation of [20] to model R. Specifically, the augmented feature representation for a ray sample t_i is $\mathbf{f}_i^r = \text{cat}(\mathbf{f}_{vol}, \mathbf{f}_{proj}, \beta)$, where β denotes positional encoding [37], $\text{cat}(\cdot)$ denotes concatenation, $\mathbf{f}_{vol}$ denotes queried feature from $\mathbf{S}$, and $\mathbf{f}_{proj}$ denotes the aggregated projection features of Eq. 5 using a Transformer network [29]. We write

$$\mathbf{c}(\mathbf{r}) = \mathcal{C}\left(\sum_{i=1}^{N_s} \sigma\left(\frac{q(\mathbf{f}^{tok})k(\mathbf{f}_i^{occ})^\top}{\sqrt{C}}\right) v(\mathbf{f}_i^r)\right) \tag{9}$$

$$d\,(\mathbf{r}) = \sum_{i=1}^{N} \sigma\left(\frac{q(\mathbf{f}^{tok})k(\mathbf{f}_i^r)^\top}{\sqrt{C}}\right) t_i, \tag{10}$$

where σ denotes the *softmax* operation, $\mathbf{f}^{tok}$ is a learnable token, $\mathbf{f}_i^{occ}$ denotes occlusion-aware feature derived from $\mathbf{f}_i^r$, $q(\cdot), k(\cdot), v(\cdot)$ are linear layers, C denotes feature dimension, and $\mathcal{C}(\cdot)$ is an MLP. To learn the network weights, we

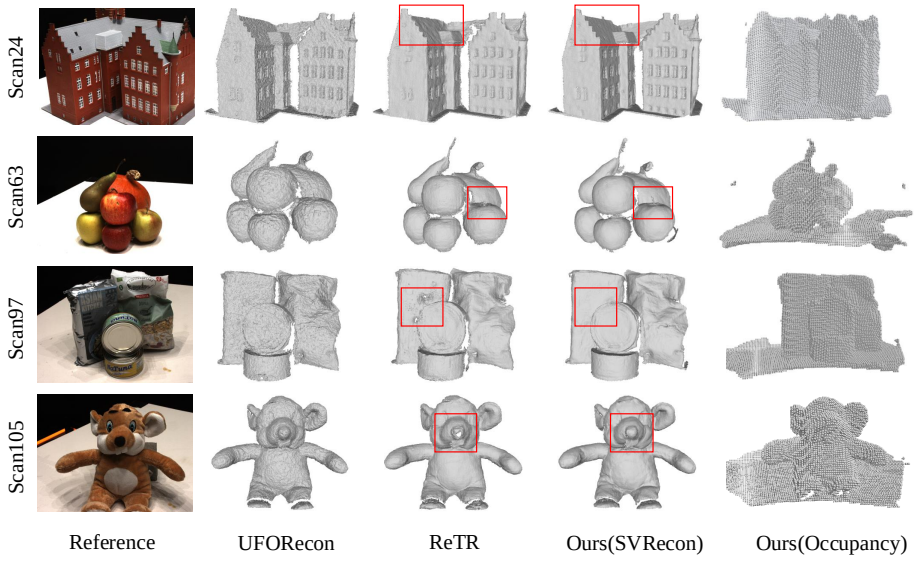

Fig. 3: Sparse-view surface reconstructions on DTU test scenes. In the middle columns, we show shaded surfaces for UFORecon, ReTR, and our approach. In the rightmost column, we show the occupancies predicted by the first stage of our method at resolution 128^3. Voxels are shrunk slightly for visualization purposes. The red rectangles highlight the region with visible differences.

minimize the loss function

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{color}} + \alpha \mathcal{L}_{\text{depth}} \\ &= \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \mathbf{c}\left(\mathbf{r}\right) - \mathbf{c}_g\left(\mathbf{r}\right) \right\|_2 + \frac{\alpha}{N_d} \sum_{i=1}^{N_d} \left| d\left(\mathbf{r}\right) - d_g\left(\mathbf{r}\right) \right|,\end{aligned} \tag{11}$$

where $\mathbf{c}_g\left(\mathbf{r}\right)$ and $d_g\left(\mathbf{r}\right)$ are ground truth color and depth, respectively, α denotes a weight coefficient to balance the two terms, N_s the number of sampled rays and N_d the number of rays with valid depth.

5 Experimental Results

In this section, we begin by outlining our experimental setup, including datasets and implementation details. Next, we assess our approach both qualitatively and quantitatively on generalizable surface reconstruction and occupancy prediction tasks. We then evaluate the generalization ability of our method on new datasets. Finally, we conduct analyses for different elements in our method and discuss the limitations.

Datasets. As in previous works [20, 26, 29, 38], we use the DTU [1] dataset for training and main evaluation. It consists of high-resolution images of 124

Scan	Mean (CD) ↓	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
COLMAP [31]	1.52	0.90	2.89	1.63	1.08	2.18	1.94	1.61	1.30	2.34	1.28	1.10	1.42	0.76	1.17	1.14
TransMVSNet [5]	1.35	1.07	3.14	2.39	1.30	1.35	1.61	0.73	1.60	1.15	0.94	1.34	**0.46**	0.60	1.20	1.46
VolSDF [43]	3.41	4.03	4.21	6.12	1.63	3.24	2.73	2.84	1.63	5.14	3.09	2.08	4.81	0.60	3.51	2.18
NeuS [36]	4.00	4.57	4.49	3.97	4.32	4.63	1.95	4.68	3.83	4.15	2.50	1.52	6.47	1.26	5.57	6.11
SparseNeuS-ft [23]	1.27	1.29	2.27	1.57	0.88	1.61	1.86	1.06	1.27	1.42	1.07	0.99	0.87	0.54	1.15	1.18
SparseCraft [45]	1.04	1.17	**1.74**	1.80	**0.70**	1.19	1.53	0.83	**1.05**	1.42	0.78	0.80	0.56	**0.44**	**0.77**	**0.84**
PixelNeRF [46]	6.18	5.13	8.07	5.85	4.40	7.11	4.64	5.68	6.76	9.05	6.11	3.95	5.92	6.26	6.89	6.93
IBRNet [37]	2.32	2.29	3.70	2.66	1.83	3.02	2.83	1.77	2.28	2.73	1.96	1.87	2.13	1.58	2.05	2.09
MVSNeRF [3]	2.09	1.96	3.27	2.54	1.93	2.57	2.71	1.82	1.72	2.29	1.75	1.72	1.47	1.29	2.09	2.26
SparseNeuS [23]	1.96	2.17	3.29	2.74	1.67	2.69	2.42	1.58	1.86	1.94	1.35	1.50	1.45	0.98	1.86	1.87
VolRecon [29]	1.38	1.20	2.59	1.56	1.08	1.43	1.92	1.11	1.48	1.42	1.05	1.19	1.38	0.74	1.23	1.27
ReTR [20]	1.17	1.05	2.31	1.50	0.96	1.20	1.54	0.89	1.34	1.30	0.87	1.06	0.77	0.59	1.06	1.11
C2F2NeuS [38]	1.11	1.12	2.42	1.40	0.75	1.41	1.77	0.85	1.16	1.26	0.76	0.91	0.60	0.46	0.88	0.92
SuRF [27]	1.05	0.85	2.35	1.48	0.88	1.17	1.39	0.74	1.14	1.24	0.84	**0.77**	0.51	0.51	0.86	1.03
UFORecon [26]	**1.00**	0.79	2.03	**1.33**	0.87	**1.11**	**1.19**	0.74	1.22	1.14	**0.71**	0.89	0.59	0.56	0.90	1.02
SVRecon	**1.00**	**0.72**	1.98	1.44	0.83	1.12	1.40	**0.72**	1.27	**1.06**	0.76	0.94	0.56	**0.44**	0.87	0.93

Table 2: Quantitative evaluation results of sparse-view reconstructions on 15 testing scenes of `DTU` dataset using *Chamfer Distances*. We test and report results using released codes for VolRecon, ReTR, and UFORecon, other results are sourced from these papers. From top to bottom, the baseline methods are from different categories: 1) Multi-view Stereo (MVS) methods; 2) neural implicit reconstruction methods (requiring per-scene optimization); 3) generalizable neural rendering methods; 4) generalizable surface reconstruction methods. We use **bold** to indicate best performance, and underline to indicate the second-best. ReTR is the closest baseline and also the one we built our approach upon.

different scenes under 7 lighting conditions captured under controlled laboratory conditions, each accompanied by accurate camera matrix and laser-scanned ground truth depth map. We use the same evaluation protocol as in earlier work and 3 views as input for each one of the 15 test scenes. In addition to `DTU`, we adopt more datasets to evaluate the generality of our approach, including `BlendedMVS` [42], a large-scale synthetic dataset designed for learning-based multi-view stereo, providing rendered images with dense depth maps and camera parameters from diverse 3D scenes; and `Co3D` [28], a large real-world dataset for category-level 3D understanding containing multi-view videos of object instances across many common categories.

Implementation Details. We use $M = 4$ input views with resolution 640×512 in training, and $M = 3$ input views with resolution 800×600 in testing, consistent with previous works. In volumetric feature construction Sec. 4.1, we use $C = 32$ feature channels. Our model is trained for 16 epochs using the Adam optimizer [15]; the learning rate is set to 10^{-4}. Please refer to our supplementary material for more details.

5.1 Main Evaluation Results

In this section, we present the main reconstruction quality evaluation on the `DTU` dataset by introducing the evaluation metrics, and reporting quantitative and

Scan	AUC@15° ↑	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
TransMVSNet [5]	9.2	9.9	9.0	13.2	12.8	3.0	7.5	11.1	5.0	10.9	7.2	14.3	6.2	9.9	10.1	7.7
SparseCraft [45]	19.1	22.7	15.6	26.2	25.9	14.5	14.8	15.9	10.1	10.4	14.7	**26.9**	**20.1**	25.4	21.5	22.4
MVSNeRF [3]	11.4	10.3	10.1	16.5	18.3	9.1	12.1	12.2	8.4	7.8	9.2	14.2	4.9	15.9	10.7	11.2
SparseNeuS [23]	12.1	17.3	10.7	16.5	14.7	9.3	10.9	14.6	9.1	10.4	13.1	11.4	6.3	14.9	10.8	11.8
VolRecon [29]	8.4	8.3	7.0	14.2	12.1	5.7	8.2	7.8	5.0	5.8	6.6	10.8	3.8	11.9	8.5	10.5
ReTR [20]	16.3	20.0	11.5	26.3	20.5	14.7	14.8	15.7	9.4	12.7	12.1	19.9	14.4	21.8	15.2	15.4
SuRF [27]	18.9	25.4	**16.9**	26.5	25.6	17.9	14.0	16.7	11.3	14.4	13.9	23.4	15.9	22.6	21.0	20.0
UFORecon [26]	11.8	14.6	8.3	14.4	15.8	8.1	11.3	11.8	6.5	7.5	9.6	16.4	10.4	15.6	13.5	13.9
SVRecon	**21.0**	**26.3**	14.6	**28.8**	**26.4**	**18.5**	**18.1**	**20.3**	**11.4**	**15.3**	**14.9**	26.3	19.6	**29.3**	**22.2**	**22.9**

Table 3: Quantitative evaluation results of sparse-view reconstructions on 15 testing scenes of DTU dataset using *Normal Consistency*. We test and report results using released codes from the compared methods. We use **bold** to indicate best performance.

qualitative results. We subsequently provide occupancy prediction evaluation results in a similar manner.

Reconstruction Quality Evaluation. Chamfer Distance between predicted surfaces and ground truth point clouds has been extensively used and is a standard metric to evaluate surface reconstruction quality. Unfortunately, it lacks awareness of local geometry and density and as a result, does not accurately quantify proper recovery of fine-scale details and reconstructed surface smoothness. Thus, we also report a *Normal Consistency* metric that accounts for surface quality. To evaluate it, we first extract 3D meshes from predicted depth maps and ground truth depth maps using TSDF fusion [4] and Marching Cube [24]. We then compute the angular differences between normals at closest vertices in the two meshes, and use Area Under the Curve (AUC) up to 15° in percentage to measure the overall normal consistency robustly. In our supplementary material, we show the angular differences on a specific example.

The quantitative results are shown in Tabs. 2 and 3, and the qualitative ones in Fig. 3. In terms of *Chamfer Distances*, our method is on par with UFORecon and they both outperform all other methods. However, as can be clearly seen in Fig. 3, the surfaces produced by UFORecon are lower-quality and much rougher than ours. The *Normal Consistency* metric quantifies this. In its terms, our method outperforms all the others and especially UFORecon by a large margin. ReTR is the closest baseline to us in methodology and we clearly do better on all metrics and for all scenes. This confirms that our method delivers the most regular and smooth surfaces thanks to our sparse high-resolution volumetric representation.

Occupancy Prediction Evaluation. In essence, occupancy prediction is a classification problem for which *Precision* and *Recall* can be used as evaluation metrics. To quantify the savings of our sparse feature volumes scheme in storage, we also compute a *Space Occupation* metric, taken to be the ratio between number of occupied voxels and number of all voxels. There is in general a trade-off between precision and recall, however in our surface reconstruction problem, we prioritize recall and compromise precision such that surface geometry is preserved as much

Scan	Mean	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
Precision	23.0	28.0	26.2	26.1	23.9	26.7	26.6	23.6	24.7	29.0	26.0	22.5	13.2	18.9	16.4	13.3
Recall	96.8	98.5	91.2	90.4	98.4	92.4	94.7	99.1	97.0	98.1	97.0	97.7	98.3	99.8	99.3	99.9
Space Occupation (Ours)	1.89	2.00	2.06	1.80	1.63	2.13	1.57	1.57	2.35	2.02	2.24	1.58	1.96	1.55	1.91	2.01
Space Occupation (GT)	0.45	0.57	0.59	0.52	0.39	0.62	0.44	0.37	0.60	0.60	0.60	0.36	0.26	0.29	0.32	0.27

Table 4: Evaluation results of occupancy predictions on 15 testing scenes of `DTU` dataset. We use precision and recall to quantify the performance of occupancy prediction. We also provide space efficiency statistics defined as the ratio between number of occupied voxels and number of all voxels. All reported results are in percentage.

as possible. In practice, 1) we use a conservative threshold of 0.1 to determine the occupied voxels from the occupancy prediction from the network $\mathbf{O}$; 2) we then further dilate the occupied voxels using a cubic $3 \times 3 \times 3$ kernel to obtain the final occupancy prediction results.

We report our quantitative results in Tab. 4, and provide qualitative ones in Fig. 3. We achieve a 96.8% average *Recall*, ensuring the preservation of surface geometry. The false negatives mostly come from the textureless table in the scenes, which is very hard to reconstruct and does not take part in the standard evaluation. The *Precision* is on average at 23.0%, resulting in *Space Occupation* at 1.89%, 4.2 times the optimal *Space Occupation* at 0.45%. This demonstrates the strong performance of our occupancy prediction method, which recalls most of the surface by keeping only 1.89% of the voxels on average.

5.2 Generalization Ability

The `DTU` dataset consists of only indoor objects with identical camera poses, which limits generalization. To validate the generalization ability of our method in surface reconstruction, we follow previous work [2] to jointly train on both `DTU` dataset and `BlendedMVS` [42] dataset. For evaluation, we select 10 scenes from `Co3D` [28] dataset to test the generalization ability of our method. We normalize each scene, compute *Chamfer Distance* from predicted points to ground truth points, and use Area Under the Curve (AUC) at threshold 0.05 as the robust metric.

We use MVSFormer++ [2] as the main competitor, which is trained on the same datasets. We use $M = 3$ input views with resolution 640×480. The quantitative results are reported in Tab. 5, and the qualitative results are shown in Fig. 4, including an example of non-object-centric scenes from `BlendedMVS` test set. Our method achieves a better performance in reconstruction quality, demonstrating its strong generalization ability.

5.3 Further Analyses

In this section, we conduct further analyses to study the impact of different settings on the final performance of our method. We also provide some novel view synthesis results using our trained volume rendering network. All experiments are conducted on the `DTU` dataset using the model trained on the same dataset.

Scan	Apple	Backpack	Bench	Bottle	Cake	Couch	Hydrant	Pizza	Bear	Truck	Mean
MVSFormer++ [2]	0.38	0.55	0.59	**0.38**	**0.42**	0.71	0.56	0.49	0.79	0.69	0.60
SVRecon	**0.54**	**0.73**	**0.74**	0.33	0.41	**0.88**	**0.70**	**0.68**	**0.92**	**0.71**	**0.72**

Table 5: Evaluation results of cross-dataset generalization on 10 scenes from unseen `Co3D` dataset. We use Area Under the Curve (AUC) at threshold 0.05 as the robust metric. Our method achieves a better performance compared to the baseline, demonstrating its strong generalization ability.

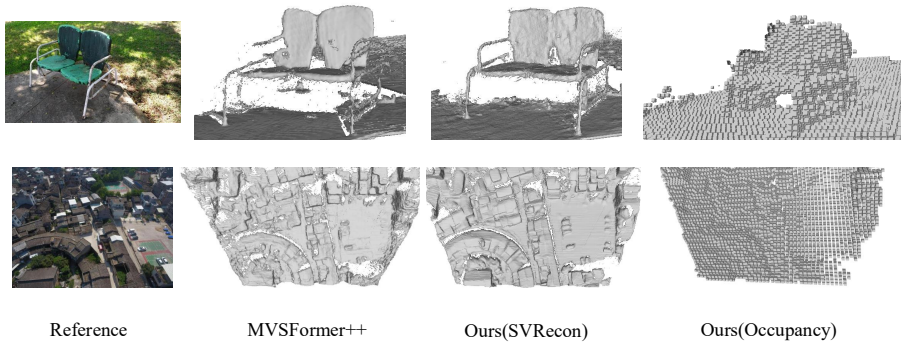

Fig. 4: Visualization of cross-dataset generalization results on *bench* scene from `Co3D` dataset and on a non-object-centric scene from `BlendedMVS` test set.

Memory Consumption. We consider only the second stage here, as the first stage of occupancy prediction can be run separately beforehand and does not consume much memory. Practically, our method consumes around 30 GB memory to train with a batch size of 1, 1024 sampled rays and 32 feature channels in the second stage. This is a moderate requirement for modern GPUs, but also prevents us from lifting the resolution even more. In testing, the memory requirement is reduced even more to around 12 GB with the same setting.

Impact of Different Resolutions. The resolution is of critical importance in the task of surface reconstruction. To verify this point, we use the same occupancy prediction results as in our main experiment and vary the times of supersampling in the second stage of our method. We test our method at $1\times$ and $2\times$ supersampling, leading to resolutions at 128^3 and 256^3. Compared to the adopted resolution in our method at 512^3, the tested resolutions are only lower, because we cannot lift the resolution even more due to memory consumption constraints. The results are presented in Tab. 6. As we can see, there is an abrupt improvement in performance from 128^3 to 256^3, and a mild improvement from 256^3 to 512^3. Overall, this validates the intuition that the performance will increase as the resolution increases.

Number of Views. While our method is trained on $M = 4$ input views, it is not restricted to this setting due to the mean and variance feature construction

Settings	Mean (CD) ↓
Resolution @ 128^3	1.27
Resolution @ 256^3	1.04
Number of Views @ 5	0.96
Number of Views @ 4	0.99
Base Feature Channels @ 16	1.15
Ours (*SVRecon*)	1.00

Table 6: A study on the impact of different settings on the final performance of our method, including **resolution**, **number of input views** and **number of base feature channels**.

operation as in Sec. 4.1. In addition to the $M = 3$ setting in our main experiment, we also test the performance of our method with $M = 4$ and $M = 5$ input views, and the results are given in Tab. 6. It can be observed that the surface reconstruction quality increases with more input views, as more scene information becomes available.

Number of Base Feature Channels. The number of feature channels is crucial in determining the effectiveness of scene feature representations. However, more feature channels will also incur more memory burden. In our main experiment, we use $C_f = 32$ base feature channels in volumetric feature construction. We also test the performance of our method with $C_f = 16$ base feature channels, and the results are provided in Tab. 6. It shows that reducing the number of base channels will degrade the performance.

Novel View Synthesis. Our method can also perform novel view synthesis using the trained volume rendering network, and some visual examples are presented in Fig. 5. Due to our sparse representation, the background is not modeled in the results.

Fig. 5: Novel view synthesis examples of our *SVRecon* method on scan24 and scan105 from the `DTU` dataset. The details are preserved well for the foreground object.

5.4 Limitations

Despite the great performance in high-fidelity reconstruction, our method is limited by the **accuracy of the occupancy prediction stage**. If the occupancy grid is not accurately predicted, it inevitably leads to incomplete or incorrect surface reconstructions.

In our experiments, the occupancy prediction network generally works well, but it can struggle in certain cases with large textureless regions or areas with thin structures, resulting in geometry loss. This limitation stems from the inherent challenges of image feature representation for textureless regions and thin structures, which affects 3D predictions in our model as our 3D representation is constructed from 2D image features. We believe that this limitation can be mitigated by incorporating more tailored and powerful image feature representations, which is a promising direction for future work.

In terms of the generalization ability, we have demonstrated that our method can generalize well to unseen datasets in Tab. 5, even to non-object-centric scenes as shown in Fig. 4, but it is still limited by the diversity of the training data distribution. For example, if the input views are from distant viewpoints, the performance of our method may degrade. We believe that this limitation can be mitigated by incorporating more diverse training data, following previous work [26].

6 Conclusion and Future Work

In this paper, we propose a two-stage neural surface reconstruction method based on the efficient scene representation of sparse feature volumes. In the first stage, our method is capable of performing accurate occupancy prediction, retaining only around 1.9% of all voxels as occupied voxels and greatly reducing the memory burden. In the second stage, our method can reconstruct high-quality surfaces by conducting feature-based volume rendering on the constructed sparse feature volumes at a high resolution of 512^3. Extensive experiments have demonstrated the superiority of our method compared to a variety of existing methods in terms of surface reconstruction quality. In the future, we will explore more efficient schemes that generalize to realistic unbounded scenes with an arbitrary number of views.

Acknowledgements

This work was supported in part by the Swiss National Science Foundation.

Bibliography

[1] Aanæs, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. International Journal of Computer Vision (2016) 9

[2] Cao, C., Ren, X., Fu, Y.: Mvsformer++: Revealing the devil in transformer's details for multi-view stereo. arXiv Preprint (2024) 3, 12, 13

[3] Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: International Conference on Computer Vision (2021) 10, 11

[4] Curless, B., Levoy, M.: A Volumetric Method for Building Complex Models from Range Images. In: Conference on Computer graphics and Interactive Techniques (1996) 11

[5] Ding, Y., Yuan, W., Zhu, Q., Zhang, H., Liu, X., Wang, Y., Liu, X.: Transmvsnet: Global context-aware multi-view stereo network with transformers. In: Conference on Computer Vision and Pattern Recognition (2022) 3, 10, 11

[6] Fua, P.: From Multiple Stereo Views to Multiple 3D Surfaces. International Journal of Computer Vision **24**(1), 19–35 (August 1997) 3

[7] Furukawa, Y., Ponce, J.: Accurate, Dense, and Robust Multi-View Stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence **99** (2009) 3

[8] Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., Tan, P.: Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching. In: Conference on Computer Vision and Pattern Recognition (2020) 3

[9] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2000) 3

[10] Huang, B., Yu, Z., Chen, A., Geiger, A., Gao, S.: 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In: ACM SIGGRAPH (2024) 2, 4

[11] Ji, M., Gall, J., Zheng, H., Liu, Y., Fang, L.: Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In: International Conference on Computer Vision (2017) 3

[12] Ji, M., Zhang, J., Dai, Q., Fang, L.: Surfacenet+: An end-to-end 3d neural network for very sparse multi-view stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 3

[13] Kar, A., Häne, C., Malik, J.: Learning a Multi-View Stereo Machine. In: Advances in Neural Information Processing Systems. pp. 364–375 (2017) 3

[14] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian Splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4), 139:1–139:14 (2023) 2, 4

[15] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: arXiv Preprint (2014) 10

[16] Kostrikov, I., Gall, J.: Depth Sweep Regression Forests for Estimating 3D Human Pose from Images. In: British Machine Vision Conference (2014) 3

[17] Kutulakos, K., Seitz, S.: A Theory of Shape by Space Carving. International Journal of Computer Vision **38**(3), 197–216 (July 2000) 3
[18] Lhuillier, M., Quan, L.: A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images. IEEE Transactions on Pattern Analysis and Machine Intelligence (2005) 3
[19] Li, Z., Müller, T., Evans, A., Taylor, R., Unberath, M., Liu, M., Lin, C.: Neuralangelo: High-Fidelity Neural Surface Reconstruction. In: Conference on Computer Vision and Pattern Recognition (2023) 1, 4
[20] Liang, Y., He, H., Chen, Y.: RETR: Modeling Rendering via Transformer for Generalizable Neural Surface Reconstruction. In: Advances in Neural Information Processing Systems (2024) 2, 3, 4, 8, 9, 10, 11
[21] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature Pyramid Networks for Object Detection. In: Conference on Computer Vision and Pattern Recognition (2017) 6
[22] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal Loss for Dense Object Detection. In: International Conference on Computer Vision (2017) 7
[23] Long, X., Lin, C., Wang, P., Komura, T., Wang, W.: Sparsneus: Fast Generalizable Neural Surface Reconstruction from Sparse Views. In: European Conference on Computer Vision (2022) 2, 3, 4, 10, 11
[24] Lorensen, W., Cline, H.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: ACM SIGGRAPH. pp. 163–169 (1987) 11
[25] Mildenhall, B., P., S.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In: European Conference on Computer Vision (2020) 1, 4
[26] Na, Y., Kim, W.J., Han, K.B., Ha, S., Yoon, S.E.: Uforecon: Generalizable sparse-view surface reconstruction from arbitrary and unfavorable sets. In: Conference on Computer Vision and Pattern Recognition (2024) 2, 4, 9, 10, 11, 15
[27] Peng, R., Shen, S., Xiong, K., Gao, H., Jiao, J., Gu, X., Wang, R.: Surface-centric modeling for high-fidelity generalizable neural surface reconstruction. In: European Conference on Computer Vision. pp. 183–200. Springer (2024) 2, 4, 10, 11
[28] Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., Novotny, D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10901–10911 (2021) 10, 12
[29] Ren, Y., Wang, F., Zhang, T., Pollefeys, M., Süsstrunk, S.: Volrecon: Volume Rendering of Signed Ray Distance Functions for Generalizable Multi-View Reconstruction. In: Conference on Computer Vision and Pattern Recognition (2023) 2, 3, 4, 8, 9, 10, 11
[30] Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Conference on Medical Image Computing and Computer Assisted Intervention. pp. 234–241 (2015) 7
[31] Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (2016) 10

[32] Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: Conference on Computer Vision and Pattern Recognition. pp. 519–528 (2006) 3
[33] Shum, H., Kang, S.B.: Review of Image-Based Rendering Techniques. In: Visual Communications and Image Processing. pp. 2–13 (2000) 3
[34] Tang, H., Yang, S., Liu, Z., Hong, K., Yu, Z., Li, X., Dai, G., Wang, Y., Han, S.: Torchsparse++: Efficient training and inference framework for sparse convolution on gpus. In: IEEE/ACM International Symposium on Microarchitecture (2023) 7
[35] Wang, F., Galliani, S., Vogel, C., Pollefeys, M.: Itermvs: Iterative probability estimation for efficient multi-view stereo. In: Conference on Computer Vision and Pattern Recognition (2022) 3
[36] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning Neural Implicit Surfaces by Volume Rendering for Multi-View Reconstruction. In: Advances in Neural Information Processing Systems (2021) 1, 4, 10
[37] Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: Conference on Computer Vision and Pattern Recognition (2021) 8, 10
[38] Xu, L., Guan, T., Wang, Y., Liu, W., Zeng, Z., Wang, J., Yang, W.: C2f2neus: Cascade Cost Frustum Fusion for High Fidelity and Generalizable Neural Surface Reconstruction. In: International Conference on Computer Vision (2023) 2, 4, 9, 10
[39] Yang, H., Hui, L., Qian, J., Xie, J., Yang, J.: Gsrecon: Efficient generalizable gaussian splatting for surface reconstruction from sparse views. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 25346–25356 (2025) 4
[40] Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: Mvsnet: Depth inference for unstructured multi-view stereo. In: European Conference on Computer Vision (2018) 3
[41] Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T., Quan, L.: Recurrent mvsnet for high-resolution multi-view stereo depth inference. In: Conference on Computer Vision and Pattern Recognition (2019) 3
[42] Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., Quan, L.: BlendedMVS: A Large-Scale Dataset for Generalized Multi-View Stereo Networks. In: Conference on Computer Vision and Pattern Recognition (2020) 10, 12
[43] Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume Rendering of Neural Implicit Surfaces. In: Advances in Neural Information Processing Systems (2021) 1, 4, 10
[44] Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In: Advances in Neural Information Processing Systems (2020) 1, 4

[45] Younes, M., Ouasfi, A., Boukhayma, A.: Sparsecraft: Few-shot neural reconstruction through stereopsis guided geometric linearization. In: European Conference on Computer Vision (2024) 2, 4, 10, 11
[46] Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Conference on Computer Vision and Pattern Recognition (2021) 10
[47] Yu, Z., Sattler, T., Geiger, A.: Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. ACM Transactions on Graphics (2024) 2, 4
[48] Zhang, C., Zou, Y., Li, Z., Yi, M., Wang, H.: Transplat: Generalizable 3D Gaussian Splatting from Sparse Multi-View Images with Transformers. In: AAAI Conference on Artificial Intelligence (2025) 4