# MatchSDF: Learning Generalizable 3D Reconstruction Using Correspondence Matching

Eren Cetin Nikolas Hars Elias Salameh Shi Chen
ETH Zürich [*]
Rämistrasse 101, 8092 Zurich
{ecetin, nihars, esalameh, shichen}@ethz.ch

## Abstract

*Recent advancements, particularly in novel view synthesis, catalyzed the pursuit of 3D reconstruction using neural implicit representations. However, per-scene optimized models do not emerge as feasible options in many real-life applications and suggest the development of generalizable 3D reconstruction models. In this paper, we present MatchSDF, a novel 3D reconstruction model that uses pairwise cosine similarity for views provided and additional geometry encoding features such as groupwise variance to improve geometric cues. In addition, MatchSDF benefits from cross-point correlation by incorporating Ray Transformer into generic SDF networks. MatchSDF is capable of performing novel view synthesis by using the recent idea of color blending to improve the appearance of views. On the DTU benchmark, MatchSDF achieves comparable and slightly better results against SparseNeuS while being reference-view agnostic and not limited by volumetric features. Our code is publicly available at* `https://github.com/EliasSalameh/SparseNeuS--MatchSDF`

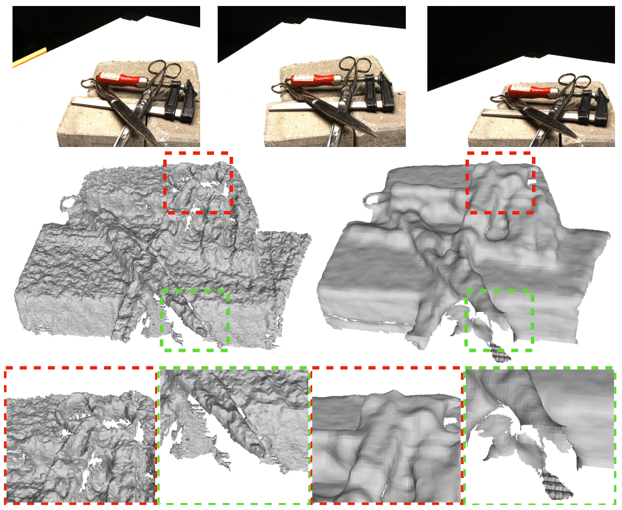


Figure 1. 3D surface reconstruction using three views from DTU benchmark [6] (top). SparseNeuS generates oversmooth and cluttered surfaces (right), while MatchSDF captures finer details (left).

## 1. Introduction

3D reconstruction using multiple views is a fundamental and widely applicable task in computer vision with relevance in robotics, graphics, mixed reality, as well as other fields. Recently, neural implicit scene representations have emerged as a powerful approach for handling 3D geometry and appearance. The mainstream approach in 3D reconstruction revolves around employing multi-layer perceptrons (MLPs) to parameterize shape representations, such as occupancy [12] or signed distance fields [10, 15, 18, 20]. Typical methods to train the neural models utilize differentiable surface rendering [19] or volume rendering [12, 15, 18, 20]. These rendering techniques project the implicit shape representations onto 2D image planes from specified views, allowing for differentiation and gradient-descent-based optimization during training. One notable example among these methods is MonoSDF [20], which further incorporates monocular geometric cues to enhance the quality of multi-view 3D reconstruction.

Despite the satisfactory results achieved by the aforementioned reconstruction methods capitalizing on the progress in implicit 3D representations, most of these methods are proposed in the context of per-scene optimization. This entails optimizing a dedicated model for each scene, which limits efficient training on large-scale datasets. Therefore, the primary objective of this project is to explore how to learn generalizable models for 3D reconstruction. While the exploration of generalizable 3D reconstruction remains limited in the existing literature, significant attempts have been made to address this challenge. One

[*] This project is supervised by Haofei Xu and Zehao Yu.

such approach is SparseNeuS [10], which offers a generalizable 3D reconstruction method requiring only a sparse set of input images (as few as 2 or 3). SparseNeuS constructs a geometry encoding volume, utilizing features extracted from the 2D images. Subsequently, a multi-level geometry reasoning scheme is employed to refine the geometry progressively. Additionally, SparseNeuS employs a multi-scale color blending scheme to render appearance from novel views, facilitating the use of images for supervision. However, SparseNeuS is essentially based on cost volume construction, which relies on a predefined reference view. Consequently, rendering quality tends to degrade significantly when the target view lacks sufficient overlap with the reference view. This limitation is a common drawback of cost-volume-based methods, potentially compromising the accuracy and robustness of the reconstruction process.

To address the inherent limitations of cost volume-based approaches, MatchNeRF [3] proposed a novel strategy that leverages 2D feature matching information between multi-view images, yielding promising results in the domain of novel view synthesis. In this study, we extend a similar strategy for 3D reconstruction while utilizing the SDF representation.

Our proposed framework, named MatchSDF, fuses the frameworks of MatchNeRF and SparseNeuS [10] by providing the SparseNeuS decoder with a pairwise cosine similarity computed by the MatchNeRF encoder as well as additional geometry encoding features. Through this integration, we aim to establish MatchSDF as a novel framework capable of achieving generalizable 3D reconstruction that works well regardless of the selection of the reference view. In addition, unlike recent volumetric feature-based models, MatchSDF does not suffer from being limited to small scenes.

Furthermore, we explore the application of the Ray Transformer [16] in the mesh generation phase in a similar way as VolRecon [14]. A series of experiments have been conducted to evaluate the performance of our method in surface reconstruction and to investigate the impacts of the Ray Transformer, mesh generation method, architectural changes, additional features, and loss function on the overall results. These investigations offer valuable insights into the effectiveness and potential enhancements of our proposed MatchSDF framework.

Our major contributions can be summarized as follows:

- We introduce MatchSDF, a novel and generalizable SDF-based 3D reconstruction pipeline that leverages feature matching statistics as the geometry prior, resulting in a view-agnostic approach.
- Unlike other state-of-the-art models, MatchSDF uses explicit correspondence matching among views and does not require volumetric features, which limits to smaller scenes.
- We conduct thorough and extensive ablation studies to investigate the effectiveness of various components within our MatchSDF framework and provide a codebase for highly customizable model training.

## 2. Related Work

### 2.1. Generalizable NeRF

Conventional NeRF methods often encounter issues of overfitting to specific scenes, leading to inferior performance when applied to unseen scenes. This problem is targeted by several approaches [2, 7, 9] to demonstrate the effectiveness of introducing a geometric prior to generalization. For instance, MVSNeRF [2] employs a 3D cost volume followed by a 3D CNN for post-regularization. Nevertheless, the efficacy of this cost volume-based geometric prior is contingent on selecting the reference view. To improve upon such limitations, GeoNeRF [7] refines the approach by fusing multiple cascaded cost volumes using attention modules. However, even with this improvement, GeoNeRF may still be constrained by the inherent limitations of the cost volume representation. In contrast, MatchNeRF [3] introduces a correspondence matching-based strategy to incorporate the geometry prior without relying on a 3D cost volume or subsequent 3D CNN, thereby possessing a valuable view-agnostic property. While these methods achieve satisfactory generalizability, their primary focus is often on novel view synthesis and may not explicitly represent 3D surfaces. On the other hand, our proposed method, MatchSDF, extends the MatchNeRF framework to achieve generalizable 3D reconstruction while retaining the advantageous view-agnostic feature. This extension enables our approach to provide an explicit representation of 3D surfaces, offering further potential benefits in addressing the challenges faced by traditional NeRF methods.

### 2.2. SDF-based Neural Surface Reconstruction

Recently, neural implicit representations have emerged as the prevailing approach for multi-view reconstruction tasks. Among these methods, a common practice involves decoding neural implicit encodings into either occupancy fields or SDF to represent 3D geometry. SDF-based methods [10, 15, 18, 20] are often favored over occupancy-based alternatives [12] due to their convenience in extracting clean and high-fidelity surfaces as the zero-level set. While some approaches use surface rendering [19] for multi-view reconstruction from predicted SDFs, they typically require additional object masks [11, 19] or depth priors [21], which can be impractical in real-world applications. Certain methods [12, 15, 18, 20] leverage volume rendering for reconstruction to eliminate the need for extra masks and depth priors. However, these methods heavily rely on a substantial number of images and require time-consuming per-scene optimization, making them unsuitable for generalizing to new scenes. SparseNeuS [10], a subsequent work, facilitates generalization to new scenes for 3D reconstruction by learning geometric priors across scenes from a sparse set of input images. Despite its improved generalizability, SparseNeuS still relies on cost volume representations, similar to MVSNeRF [2] and GeoNeRF [7], which may lead to deteriorated reconstruction results based on the selection of reference views. In addition to the cost volume representation, VolRecon [14] also utilizes projected multi-view features. Although VolRecon uses signed ray distance functions, it adopts TSDF fusion [4] to reconstruct 3D objects via depth map fusion. In the design of our MatchSDF architecture, we aim to enhance the reconstruction by introducing a view-agnostic feature by incorporating a pairwise-matching strategy, as seen in MatchNeRF [3], for scene encoding.

## 3. Method

We extend MatchNeRF [3], which is developed solely for Novel-View Synthesis, to perform 3D Surface Reconstruction. We utilize tools from MatchNeRF [3], SparseNeuS [10], and VolRecon [14] and improve in certain aspects to build the novel vanilla-MatchSDF and MatchSDF models.

A high-level overview of our pipeline can be seen in Figure 2.

### 3.1. Feature Encoder

Our feature encoder is similar to the one employed in MatchNeRF [3], with some modifications to the output of the feature encoder. Using a weight-sharing CNN, MatchNeRF's encoder first extracts 8x downsampled features for each view separately. Then, a Transformer with cross-attention built on top of GMFlow [17] is used. Each possible 2-view pair of convolutional features is injected with positional encodings and subsequently fed into a weight-sharing Transformer jointly to model the cross-view interaction. The output features are upsampled with a network to a resolution of 1/4th of the original resolution. That way, the output features at this stage have a resolution of 1/4th and 1/8th of the original images. Each 3D point on a ray is then projected onto the pair-wise 2D Transformer features $F_i$ and $F_j$. To obtain the features $f_i$ and $f_j$ at each specific 3D point, the Transformer features $\{F_i,\ F_j\} \in \mathbb{R}^{256}$ are bilinearly sampled at the projected 2D locations. The model then computes the group-wise cosine similarity for each resolution to increase the expressiveness compared to computing a single value for the whole feature representation. The final cosine similarity in MatchNeRF [3] is obtained using an element-wise average over all pairs. The cosine similarity obtained in this way is shown to have a high correlation with the SDF value of each point [3].

In contrast to SparseNeuS [10], our feature encoding module does not require the construction of cost volume and thus avoids expensive 3D convolutional architecture.

Further improvements on our feature encoder module are discussed in the following sections.

#### 3.1.1 Additional Features

We explore the possibility of not just using the pair-wise cosine similarity and other features to give the SDF network more information to process. For this reason, we use the raw sampled Transformer features from the MatchNeRF [3] encoder as additional latent features for the SDF network. As these features are very high-dimensional (256 channels), the features are compressed with a 2-layer MLP with ReLU activation. In Figure 2, these features are represented as $q$.

#### 3.1.2 Modified Pairwise Cosine Similarity

The pair-wise cosine similarity of the Transformer features $f_i$ and $f_j$ between 2 points of views $i$ and $j$ in MatchNeRF [3] is computed without verifying if one or both points are not visible from one of the two views. The features $f_i$ and $f_j$ at each 3D point are sampled using the border constraint, which means that the feature of the closest point at the border of the viewing frustum is used instead. This can lead to features having a high cosine similarity, although one or both could potentially not be visible. After the computations between each viewing pair are completed, the original method takes an unweighted average. In this setting, a point unobserved by even a single view will lead to unpredictable results in 2 out of 3 viewing pairs. To avoid this from happening, we propose to replace the non-weighted average with a weighted one with an extra filtering step:

$$\hat{z}_{\mathbf{s}} = \frac{1}{\hat{P}_{val}} \sum_{p=1}^{P} m_{p,val} \cdot z_{p,\mathbf{s}} \tag{1}$$

where $\hat{z}_{\mathbf{s}}$ is the weighted cosine similarity of the features at point $\mathbf{s}$, $z_{p,\mathbf{s}}$ is the cosine similarity of the viewing pair $p$ at point $\mathbf{s}$, $P$ is the number of viewing pairs and $\hat{P}_{val} = \max(P_{val}, 1)$ is the number of viewing pairs this point is observed, but lower-bounded by 1 to avoid unboundedness of the expression from above. $m_{p,val,\mathbf{s}} \in \{0, 1\}$ is a mask that describes if the current point $\mathbf{s}$ is observed by both views of viewing pair $p$. .

#### 3.1.3 Grouped Variance

Similarly to the cosine similarity, we compute the grouped variance in a pairwise manner. We divide the feature of each projected point on each view into $n$ groups, expressed as $F_{p,k} = \{F^1_{p,k}, ..., F^n_{p,k}\}$ where $p$ is the point of interest and

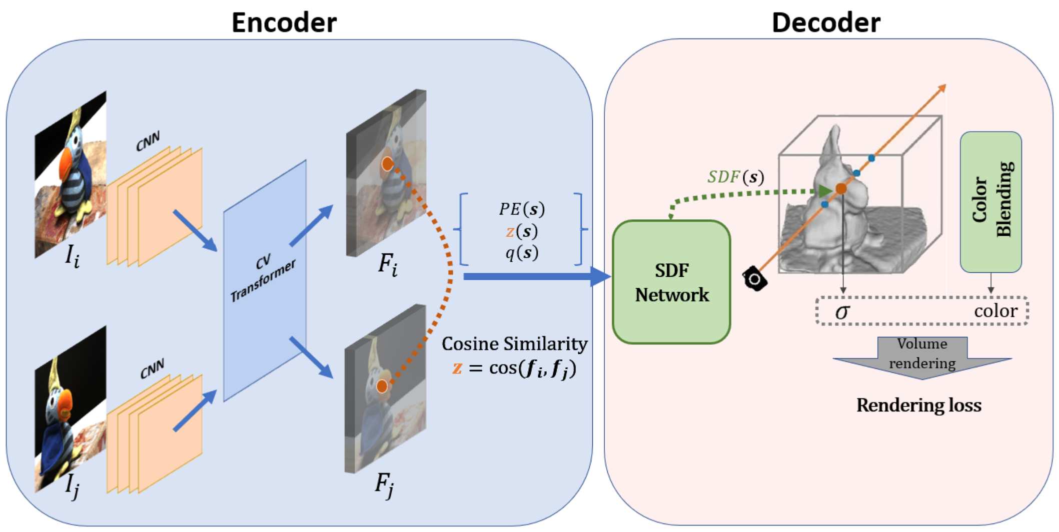


Figure 2. Overview of MatchSDF architecture. $N = 3$ views are fed into our derivative of the MatchNeRF [3] encoder. Later on, the extracted features are used in the SDF network with a 4-layer MLP and a Ray Transformer in order to generate final SDF values per point. A compound loss function is used after applying volume rendering on the final SDF values and estimated appearance per point.

$k$ is the view onto which the point is projected. Next, the variance between the group pair is computed as $F_{p,(i,j)} = \{Var(F^1_{p,i}, F^1_{p,j}), ..., Var(F^n_{p,i}, F^n_{p,j})\}$, where $i$ and $j$ are 2 distinct views. Afterward, we compute the average of each group to get a new feature vector of size $n$ for each view-pair: $F'_{p,(i,j)} = \{Mean(F^1_{p,(i,j)}), ..., Mean(F^n_{p,(i,j)})\}$. Finally, we compute the average of each $n$-component of all view-pairs that are overseen by the point:

$$F_p = \left\{ \frac{\sum_{i,j} Mean(F^1_{p,(i,j)})}{N(N-1)/2}, ..., \frac{\sum_{i,j} Mean(F^n_{p,(i,j)})}{N(N-1)/2} \right\}$$

$$F_p = \frac{\sum_{i,j} F'_{p,(i,j)}}{N(N-1)/2}$$

The grouped variance is computed using 8x downsampled as well as 4x downsampled features. We then concatenate the resulting grouped variance feature yielded from each feature map. The final feature is then added to the conditional features used by the SDF network.

## 3.2. Surface Extraction

To represent the surface, we use a 4-layer MLP network $f_\theta$ similar to [10, 15] to predict the SDF value of a sampled 3D point $\mathbf{s}$. The network is fed at its input the positional encoding ($PE$) of the 3D coordinates of $\mathbf{s}$ as well as conditional features $Cond(\mathbf{s})$ described in Section 3.1. The network is then expressed as:

$$sdf_1(\mathbf{s}) = f_\theta(PE(\mathbf{s}), Cond_1(\mathbf{s}))$$

To get better surfaces, we go beyond the MLP block by incorporating a Ray Transformer at its output, resulting in an improved SDF prediction. However, following [1], the initialization of the volume rendering network appears as an important regularizer for surface extraction. Although the efficacy of the Ray Transformer is vividly demonstrated in [3, 14], it breaks the initialization mechanism. For that reason, we propose the use of an optional second SDF network with proper geometric initialization as described in [1, 10, 16] to use the Ray Transformer output as additional features instead of the final SDF values. The output of the Ray Transformer is then concatenated with other features outputted from the first 4-layer MLP block $s_1$ to form the second set of conditional features $Cond_2(\mathbf{s})$. The final predicted SDF value by our network is,

$$sdf_2(\mathbf{s}) = f_\theta(PE(\mathbf{s}), Cond_2(\mathbf{s}))$$

. The full SDF architecture used is visualized in Figure 3.

### 3.2.1 Color Blending

To get improved color samples, we additionally employ the color blending scheme introduced in SparseNeuS [10] to make up for the fact that we oversee few images. The color blending scheme consists of predicting the color of a point $\mathbf{s}$ by aggregating appearance information from the input images. Initially, point $\mathbf{s}$ undergoes projection onto the input images, yielding its respective colors $\{I_i(\mathbf{s})\}_{i=0}^{N-1}$. Subsequently, these colors originating from various perspectives are blended together using blending weights. This fusion culminates in the predicted color of $q$. The blending weight $\{w_i^{\mathbf{s}}\}_{i=0}^{N-1}$ is computed by considering the photographic consistency of the input images. Full details on how to obtain the blending weights can be followed in [10].

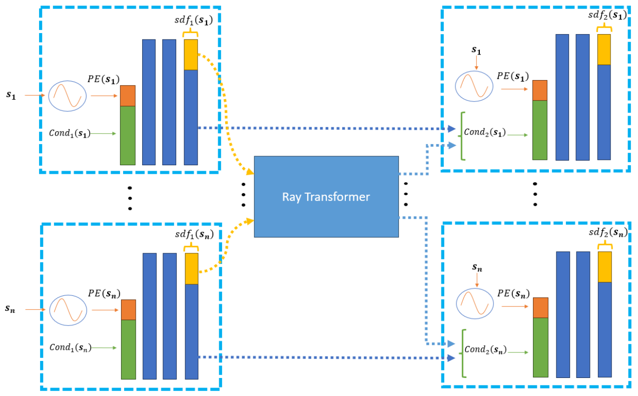


Figure 3. Overall SDF network architecture utilizing the Ray Transformer. Two 4-layer MLPs are used on both sides of the Ray Transformer to allow a spherical initial surface.

After having obtained the blending weights, the color of a 3D point $\mathbf{s}$ is predicted as the weighted sum of its projected colors $\{I_i(\mathbf{s})\}_{i=0}^{N-1}$ on the input images. To render the final color of the query ray, the color and SDF values of 3D points sampled on the ray are fir predicted. The color- and SDF values of the sampled points are then aggregated to obtain the final colors of the ray using SDF-based volume rendering, as done in [15].

## 3.3. Ray Transformer

MatchNeRF [3] uses a small MLP followed by Ray Transformer, a multi-head attention module, to compute the opacity. Similarly, VolRecon [14] uses a Ray Transformer for the purpose of depth estimation.

Inspired by both models, we utilize a Ray Transformer to estimate, refine and improve per-point SDF features by utilizing cross-point attention. After the first SDF network described in Section 3.2 extracts per-point SDF features, the Ray Transformer refines the features using the cross-attention mechanism among the points along the same ray. Later on, an optional second SDF network generates the final SDF values by using conditioning information from the Ray Transformer as depicted in Figure 3.

### 3.3.1 Modified Volume Sampling Process

The mesh generation process from SparseNeuS [10] creates a dense 3D cube centered around the origin with side lengths of 2 units. This cube consists of 3D voxels which contain the SDF values for this partition of the 3D space. The SDF values are then computed in batches where each consists of points from disjunct subcubes, meaning that rays in 3D space could be split into shorter rays. Their SDF network does not consider any point-wise correlations along a ray and thus does not suffer from poor directionality of rays or incomplete rays.

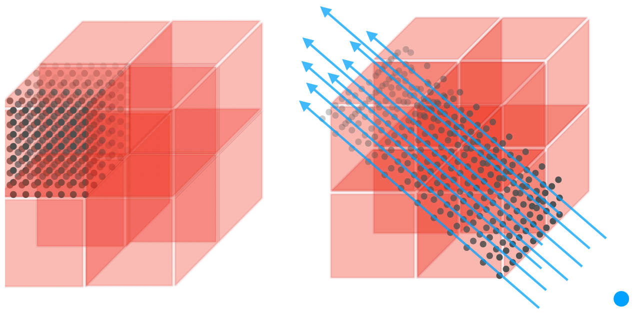


Figure 4. Two types of sampling methods used for mesh generation. Grid-like, *cubes* sampling method (left) and *rays* sampling method (right). Gray points depict the sampled points in each batch, while blue point represents the reference camera origin.

However, the SDF values of each point from our Ray Transformer depend on the features and positions of all other points on the same ray. Additionally, we postulate that rays cast in directions perpendicular to the viewing direction of any of the reference views might extract less rich information compared to the rays cast quasi-parallel to one of the viewing directions of one of the reference views. This implies that we need to modify the implementation of the mesh generation to necessitate the problems that could arise for the utilization of the Ray Transformer.
To account for the directionality of the Ray Transformer, we modify the mentioned mesh generation of SparseNeuS [10], which we will refer to as *cubes* method, to cast rays parallel to one of the three base vectors of the 3D space; for simplicity, we use the x-direction. We then compute the angle between the ray and the target view and rotate each ray to be parallel to the viewing direction of the target view.
We change the batching process of the SparseNeuS [10] mesh generation to partition the 3D space into disjunct rectangles, which span the whole cube in at least one generation to ensure that we do not partition rays into multiple disjunct rays as it is done in SparseNeuS. We will refer to this method as *rays* method. The difference between the two methods can be observed in Figure 4.

### 3.3.2 TSDF Fusion for further Enhancement of View-Dependent Rays

It is important for neural implicit surface reconstruction methods to initialize the SDF network in a geometrically plausible way [1, 10, 16]. Failing to do so can lead to degraded gradients and lessen constraints on surface generation, worsening the generated surface. Due to difficulties in initializing the Ray Transformer in a geometrically plausible way, we additionally explore the method of rendering depth maps from multiple virtual views and then fusing the depth map into a consistent mesh. Truncated SDF (TSDF) fusion [4] is adapted from VolRecon [14] and integrated into our codebase. TSDF fusion method requires the generation of depth maps instead of using direct SDF values of 3D-point samples. The depth maps are fused together to estimate the final surface.

Since the TSDF fusion method does not use simple 3D-point samples but requires casting rays from the camera origin, the validation takes longer than sampling-based surface generation. On the other hand, training is computationally more efficient as the additional surface normal required for Eikonal loss is saved. This also implies that we can no longer use the Eikonal loss and have to resort to using the depth loss with a higher weight.

## 3.4. Loss Functions

Our overall loss function is adapted from SparseNeuS [10] and VolRecon [14] and NeuS [15]. Our loss function is defined as a combination of four components,

$$\mathcal{L} = \mathcal{L}_{rgb} + \alpha\mathcal{L}_{eik} + \beta\mathcal{L}_{sparse} + \gamma\mathcal{L}_{fgbg} + \epsilon\mathcal{L}_{depth} \quad (2)$$

The primary component is the color loss, $\mathcal{L}_{rgb}$, defined as

$$\mathcal{L}_{rgb} = \frac{1}{N}\sum_{i=1}^{N}\left\|C_i - \hat{C}_i\right\|_1 \quad (3)$$

where $\hat{C}_i$ is the ground-truth color per pixel $i$. Secondly, the Eikonal loss $\mathcal{L}_{eik}$ [5] is optionally applied as the surface regularizer to extract a smooth surface. The Eikonal loss is calculated as,

$$\mathcal{L}_{eik} = \frac{1}{\|\mathbb{S}\|}\sum_{\mathbf{s}\in\mathbb{S}}(\|\nabla f_\theta(\mathbf{s})\|_2 - 1)^2 \quad (4)$$

where $\mathbf{s}$ is a sampled 3D point from the set of all sampled points $\mathbb{S}$ along the rays. To induce sparseness in the hindsight of the reconstructed geometry, a sparseness-inducing loss component $\mathcal{L}_{sparse}$ from SparseNeuS [10] is used,

$$\mathcal{L}_{sparse} = \frac{1}{\|\mathbb{S}\|}\sum_{\mathbf{s}\in\mathbb{S}}\exp\left(-\tau\cdot|sdf(\mathbf{s})|\right), \quad (5)$$

where $|sdf(\mathbf{s})|$ is the absolute SDF value of the point $\mathbf{s}$ and $\tau$ is a temperature scalar for the SDF value.

On the other hand, the foreground-background loss $\mathcal{L}_{fgbg}$ [15] penalizes the model for giving high weights to rays that are part of the background. This prior is optionally used in the first 50,000 iterations to make the foreground cleaner. The equation for the $\mathcal{L}_{fgbg}$ is formulated as follows:

$$\mathcal{L}_{fgbg} = \frac{1}{K}\sum_{k=1}^{K}(1 - M_k)\cdot\left\|\hat{W}_k - M_k\right\|_1 \quad (6)$$

where $M_k \in \{0,1\}$ is the mask value of pixel $k$ out of the set of pixels with cardinality $K$, $\hat{W}_k = \sum_{i=1}^{n} T_{k,i}\alpha_{k,i}$ is the sum of weights along the camera ray [15].

Finally, the depth loss $\mathcal{L}_{depth}$ appears as an important guideline for the geometry learning process. The depth loss is calculated as,

$$\mathcal{L}_{depth} = \frac{1}{\|\mathbb{S}\|}\sum_{\mathbf{s}\in\mathbb{S}}\left\|D_{\mathbf{s}} - \hat{D}_{\mathbf{s}}\right\|_1 \quad (7)$$

where $\mathbf{s}$ is a sampled point with valid depth, $D_s$ is the predicted depth and $\hat{D}_s$ is the ground-truth depth of the corresponding point $\mathbf{s}$. While the network can learn the novel-view synthesis generation process without the depth loss, the reconstructed 3D object falls far behind when depth loss is not introduced. As a result, our loss function for vanilla-MatchSDF uses loss component weights $\alpha = 0.2$, $\beta = 0.02$, $\gamma = 0.01$, $\epsilon = 0.3$.

However, a significant observation of our work is the trivial but vital challenge of adapting single loss function for all architectures. Although [10,15,20] suggest using the Eikonal loss is helpful for geometry regularization, computation of second-order derivatives is computationally expensive due to the use of Ray Transformer in MatchSDF architecture. Therefore, we use a non-zero $\alpha$ coefficient only for vanilla-MatchSDF without Ray Transformer. Specifically, our loss function for MatchSDF uses the loss component weights $\alpha = 0$, $\beta = 0.02$, $\gamma = 0.01$, $\epsilon = 1$.

## 4. Experiments

### 4.1. Experimental Setup

#### 4.1.1 Dataset

Following recent works in 3D reconstruction [10, 19, 20] and Novel-View Synthesis [2, 3, 7], we train our model on the DTU dataset [6].It is a large-scale Multi-View Stereopsis dataset comprising 124 diverse scenes recorded in 7 different lighting conditions. We use the same 15 scans as SparseNeus [10] and VolRecon [14] to evaluate our model and use the remaining scans for training. Each scan has a ground-truth mesh, RGB images, depth maps, and foreground-background masks for each captured pose. Each scan contains one specific object or a combination of objects, as depicted in Figure 5.

For testing our model on the task of 3D reconstruction, we follow the same evaluation method as SparseNeuS [10] and VolRecon [14]. We evaluate our model on two different sets of the same 15 scans from the DTU benchmark [6]. Later on, we report the average of results from two sets.

#### 4.1.2 Implementation Details

We build our model as a merged and adapted version of the code of SparseNeuS [10] and MatchNeRF [3] in Py-

(a) Multiple objects (b) Single object

Figure 5. Example RGB images extracted from the DTU dataset [6]

Torch [13]. We are constrained by the VRAM of the used GPUs, which is why we train our model using an image resolution of 640 x 512 with 3 reference images. The training is performed on single external Tesla V100 GPU. We train our model for 200,000 iterations using Adam [8] with different learning rates for our encoder and decoder. Specifically, learning rates of $2e-4$ and $5e-5$ are used for the decoder and the encoder, respectively. In addition, the learning rate is halved at [100K, 150K, 200K] iteration milestones following SparseNeuS [10]. We perform experiments with a pre-trained MatchNeRF [3] encoder and from scratch. If not specified otherwise, the pre-trained version is used for the experiments. We vary the loss weights described in Section 3.4 when switching between our baseline model and the model with ray-transformer by setting the Eikonal loss weight to 0. We sample 512 rays per batch which we set to a size of $N = 1$. Following SparseNeuS [10], we use hierarchical sampling to resample points on each ray that are closer to the surface. For that, we sample $N = 64$ points on each ray uniformly and then use importance sampling to resample another $N = 64$ points, which we split up into 4 steps of $N = 16$. During testing, we set the image resolution to 800 x 576 to preserve the aspect ratio of images with a resolution of 1600 x 1152. We compare the TSDF fusion process and the mesh generation processes described in Section 3.3.1 with each other.

#### 4.1.3 Baseline

We compare our model mainly against SparseNeuS [10] and include the scores from MVSNeRF [2], IBRNet [16] and VolRecon [14], which we borrow from [14]. All listed models, including ours, were trained on- and evaluated on 3 views for 3D reconstruction.

#### 4.1.4 Architectural Choices

We evaluate the performance of our model without Ray Transformer, which will subsequently be called vanilla-MatchSDF, versus our model, MatchSDF, which includes the Ray Transformer by default. MatchSDF's Ray Transformer introduces cross-point interactions by fusing the rendered information along a ray via a Transformer, unlike vanilla-MatchSDF's decoder, which processes all points on a ray independently. Furthermore, we conduct experiments on the integration of the Ray Transformer. These experiments include a model where the Ray Transformer is sandwiched between 2 small MLPs to leverage the geometric initialization of the MLPs [10, 16], as well as one model where the cosine similarity is fed as a residual connection to the second to last layer of the SDF network to leverage the strong correlation between the SDF and the cosine similarity found in MatchNeRF [3].

### 4.2. Evaluation Results

#### 4.2.1 Ablation on Mesh Generation Method

We evaluate the 3D reconstruction capability of vanilla-MatchSDF and MatchSDF with the described mesh generation modes from Section 3.3.1. Because of the importance of the directionality of MatchSDF, we only evaluate the **rays** method and TSDF fusion for it. Table 1 shows a summary of the results of this experiment. Vanilla-MatchSDF with the *rays* method performs better than the *cubes* method, although vanilla-MatchSDF considers no spatial relationship between any other points. We conjure that this slight improvement is due to different sampling locations arising from the rotation of points around the origin. MatchSDF performs best with the TSDF fusion method. In contrast, vanilla-MatchSDF performs worse using TSDF Fusion, which can be explained by the fact that vanilla-MatchSDF relies more on the Eikonal loss, whereas MatchSDF puts a higher weight on the depth loss, making the depth predictions more accurate. Overall, the table shows that MatchSDF shows more promising results than vanilla-MatchSDF, which is why we decide to continue the evaluation using MatchSDF as our standard model.

| Scan | | Mean↓ | 40 | 55 | 69 | 110 | 114 |
|---|---|---|---|---|---|---|---|
| vanilla- | TSDF | 3.62 | 4.53 | 2.77 | 3.95 | 4.24 | 3.08 |
| | *cubes* | 2.81 | 2.83 | 1.39 | 2.23 | 2.50 | 1.61 |
| | *rays* | 2.79 | 2.77 | **1.36** | 2.25 | 2.48 | 1.60 |
| MatchSDF | TSDF | **2.01** | **2.26** | 1.59 | **2.01** | **2.26** | **1.38** |
| | *rays* | 4.05 | 4.42 | 3.75 | 3.52 | 3.31 | 3.11 |

Table 1. Ablation Study on the choice of architecture combined with their respective mesh generation processes. We use vanilla-MatchSDF and MatchSDF for this analysis. We show the representative Chamfer distances of 3-view reconstruction on 15 testing scenes of DTU benchmark [6]. **Bold** font means best score and underlining means second best score. MatchSDF using the TSDF fusion method shows a clear improvement over the best vanilla-MatchSDF model using the *cubes* method.

### 4.2.2 Ablation on Architectural Choices

For the training of MatchSDF, we provide a wide variety of feature choices, mesh generation methods, and SDF network architectures. The experimental comparison of each choice can be viewed in Table 2.

We take the values for SparseNeuS [10], VolRecon [14], IBRNet [16], and MVSNeRF [2] for 3-view 3D reconstruction as reported by VolRecon [14]. For our model, we generate 3 depth maps for each scene. These depth maps are rendered from the 3 source views.

We can clearly see that the model with grouped variance outperforms the standard MatchSDF network by providing a strong geometric prior to the model, letting it express the consistency of points by computing the variance of projected features along different views. It also outperforms SparseNeuS by a small margin in overall mean Chamfer distance on the test split of DTU [6], although SparseNeuS does outperform MatchSDF with grouped variance in various scans.

When we stack a second MLP on top of the existing SDF network with Ray Transformer, we observe a performance deterioration. This is probably because a proper geometric initialization is more important when using the Eikonal loss as a regularizer. Thus, since we do not use the Eikonal loss for these models, the benefit from this initialization is lost. Possibly, the second MLP learns to put a lower weight on the output from the Ray Transformer since it is interpreted as an intermediate feature instead of the final output, which downplays the effectiveness of the Ray Transformer.

Observing Figure 6, it is evident that MatchSDF acquires more accurate yet noisy surfaces compared to SparseNeuS [10] (see Scan 37, 65, and 118). The noise on the surfaces generated by MatchSDF is similar to the results achieved by VolRecon [14] as a common result of using TSDF Fusion. Furthermore, the features generated by MatchSDF are not fully-convolutional features in contrast to SparseNeuS. Thus, the features are more prone to change among neighbors and cause sharper yet noisier results. On the other hand, due to the use of the TSDF Fusion method instead of a sampling-based mesh generation, certain parts behind the visible surfaces are missing for MatchSDF similar to [14].

When we compare MVSNeRF [2] and SparseNeuS [10] with MatchSDF based on their generalization capabilities in Figure 7, we observe that MatchSDF manages to acquire more complete meshes with an object in the foreground similar to SparseNeuS (see row 1), but suffers from noisy surfaces that are resulting from the use of correspondence matching instead of volumetric features. On the other hand, SparseNeuS does not depend on pixel-wise correspondence and thus generates an oversmooth mesh.

### 4.2.3 Influence of Pre-Trained Encoder

We investigate the effect that the pre-trained MatchNeRF [3] encoder has on the final 3D reconstruction quality compared to the model being trained from scratch. It clearly shows the superiority of the pre-trained model over the model trained from scratch. This is expected as the pre-trained encoder was trained for optical flow, so correspondence matching in video frames, which supposedly helps it to better learn correspondences also between images of the same scene taken from different views, as was already shown in MatchNeRF [3]. Although the model trained from scratch is outperformed by the pre-trained model, it still achieves acceptable performance. Table 3 contains the detailed results from this analysis.

## 5. Limitations & Future Work

The most intuitive approach for training MatchSDF is to train the model using the same training procedure as SparseNeuS [10], i.e., with the same loss function and learning rate schedule. However, our experiments showed that the gradient calculation for the Eikonal loss is non-trivial when combined with our Ray Transformer. As the calculation of per 3D point gradient relies on the SDF tensor of all points, the gradient calculation cannot be done with the *autograd* functionality of PyTorch [13]. For that reason, the Eikonal loss has an adverse effect on the MatchSDF network when enabled, as depicted in Figure 8. Suppose the calculation is done manually by preventing the effect of inter-point SDF calculation relation. In that case, the computational complexity of training becomes too high due to the lack of optimization benefits from *autograd*.

Thus, for our future work, we aim to improve the Eikonal loss calculation for MatchSDF and achieve better surface regularization, possibly improving surface normals as well.

MatchSDF is capable of generating surfaces using both the 3D-point sampling-based method (as discussed in Section 3.3.1) and the TSDF Fusion method [4] (as discussed in Section 3.3.2). TSDF Fusion improves the surface accuracy for MatchSDF even though it might not be favorable to use TSDF Fusion since missing surfaces are imminent, especially for MatchSDF due to its reliance on correspondence matching and problems with occluded regions. Furthermore, although TSDF Fusion improves the surface accuracy, it is important to note that TSDF Fusion requires the generation of depth maps for a specified number of views. Thus, it takes considerably longer to generate surfaces using TSDF Fusion than 3D-point sampling-based methods.

Although MatchSDF acquires competitive results when compared with SparseNeus [10], and VolRecon [14], using a pre-trained feature encoding module appears as an important component, as depicted in Table 2. Although MatchSDF uses an encoder that is derived from Match-

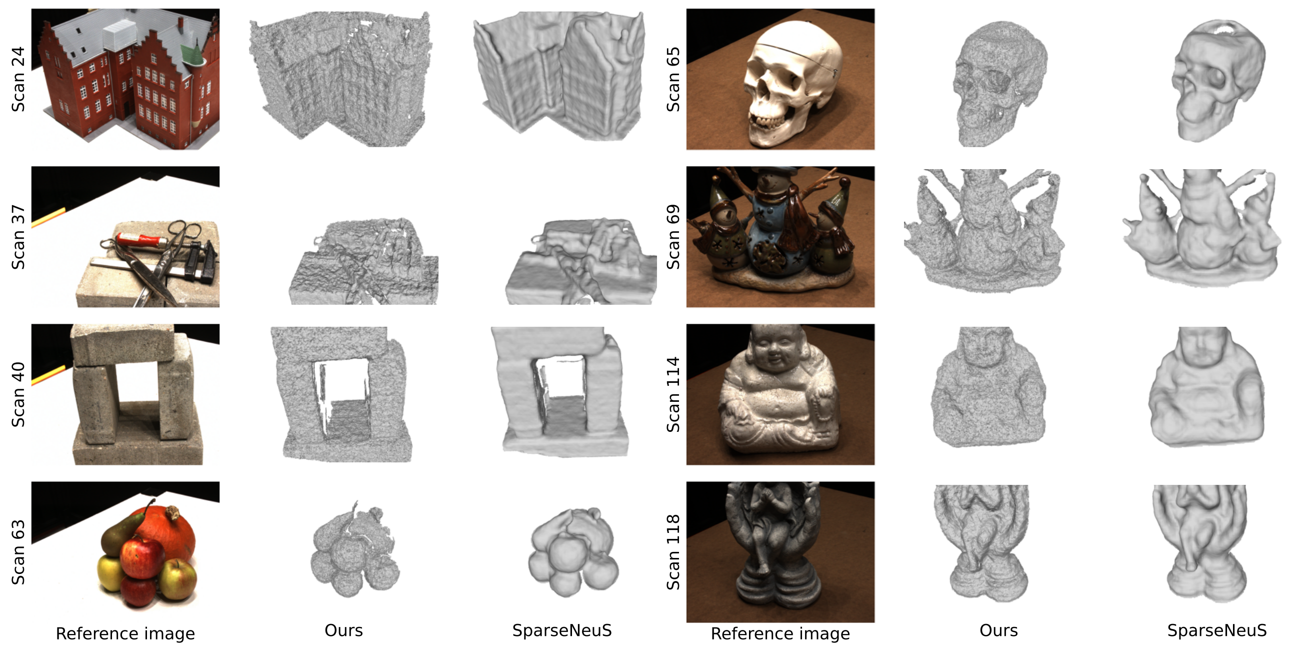


Figure 6. 3D Reconstruction results on testing scenes of DTU benchmark [6] using $N = 3$ views. Our results demonstrate that SparseNeuS leads to oversmoothened (i.e., scan 24) or cluttered/missing surfaces (i.e., scan 37 and scan 65), while MatchSDF yields surfaces with more granular details.

| Scan | Mean↓ | 24 | 37 | 40 | 55 | 63 | 65 | 69 | 83 | 97 | 105 | 106 | 110 | 114 | 118 | 122 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IBRNet [16] | 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 [2] | 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 |
| MatchSDF (Ours) | 2.01 | 2.56 | 3.10 | 2.26 | 1.59 | 2.23 | 2.16 | 2.01 | 1.87 | 2.13 | 1.52 | 1.69 | 2.26 | 1.38 | 1.63 | 1.70 |
| + grouped variance | 1.93 | 2.34 | 2.96 | 2.45 | 1.49 | 2.28 | 2.15 | 1.82 | 1.88 | 2.13 | 1.35 | 1.55 | 2.15 | 1.27 | 1.53 | 1.62 |
| + sandwiched between SDFs | 2.11 | 2.78 | 3.42 | 2.79 | 1.54 | 2.30 | 1.93 | 2.12 | 1.90 | 2.18 | 1.48 | 1.76 | 2.49 | 1.41 | 1.75 | 1.81 |
| SparseNeuS [10] | 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 [14] | **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** |

Table 2. Chamfer distance results of 3D reconstruction on DTU benchmark [6] using 3 views (lower is better). We borrow the reported results for IBRNet [16], MVSNeRF [2], SparseNeuS [10] and VolRecon [14] from [14]. **Bold** font means best score and underlining means second best score. We report the incremental benefit of using grouped variance and an optional second SDF network to allow geometric initialization leading to a spherical initial surface.

| Scan | Mean↓ | 40 | 55 | 69 | 110 | 114 |
|---|---|---|---|---|---|---|
| Pre-trained | 1.93 | 2.45 | 1.49 | 1.82 | 2.15 | 1.27 |
| From Scratch | 2.48 | 3.33 | 2.04 | 2.21 | 2.35 | 1.86 |

Table 3. Ablation Study on the effect of using a pre-trained MatchNeRF [3] encoder vs. training the model from scratch. Shown are representative Chamfer distances of 3-view reconstruction on 15 testing scenes of DTU benchmark [6].

NeRF [3], the 3D reconstruction task requires additional improvement in terms of depth estimation, which is not fully aligned with the novel view synthesis task targeted by MatchNeRF [3]. Therefore, pre-training the feature encoder provides a stronger improvement when compared with MatchNeRF [3].

Furthermore, MatchSDF only relies on matching projected points on 2D feature maps. On the other hand, works such as SparseNeuS [10] and VolRecon [14] also use global feature volumes that provide global shape priors that are helpful for geometry estimation.

## 6. Conclusion

We introduced MatchSDF as a novel generalizable 3D reconstruction architecture. MatchSDF encoder extracts

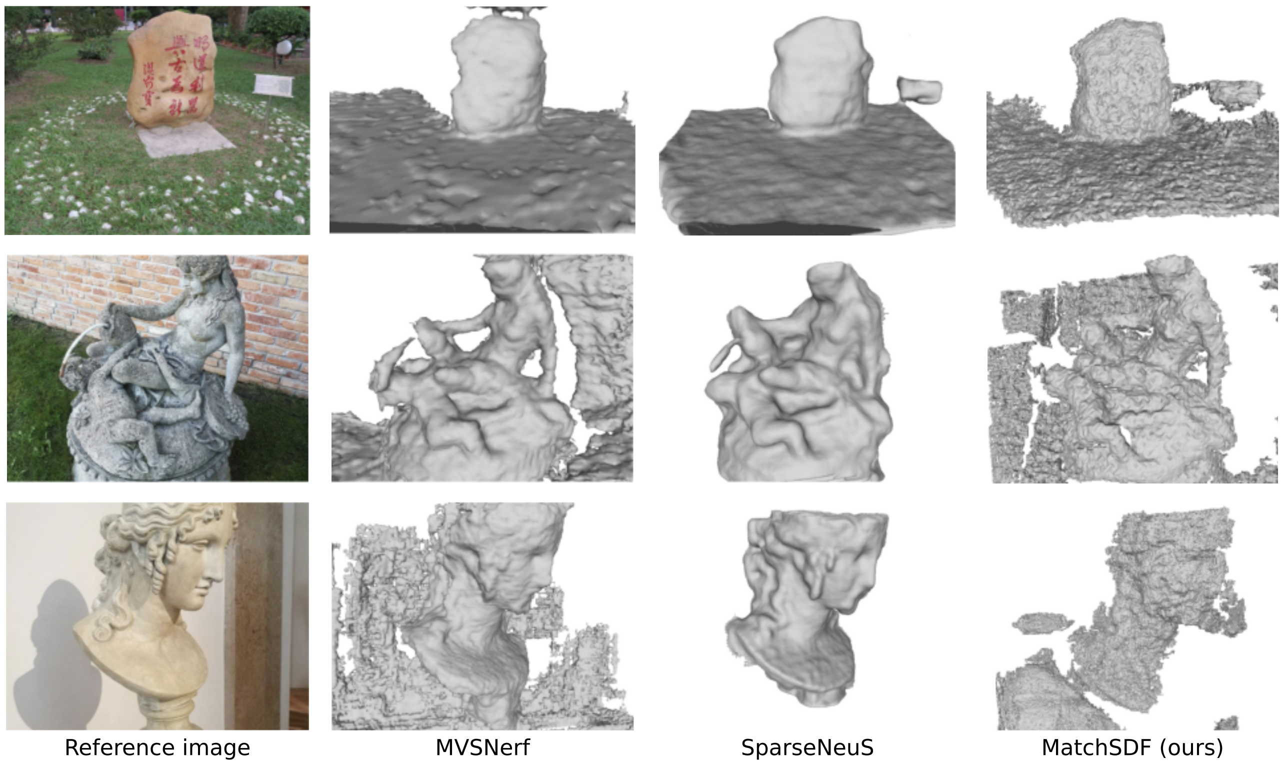
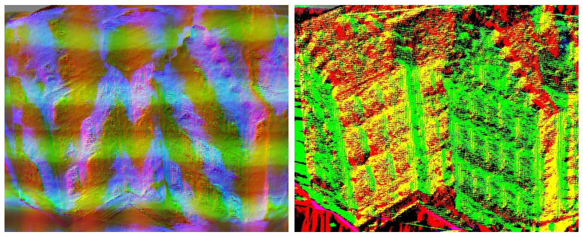


Figure 7. 3D Reconstruction results on testing scenes of BlendedMVS benchmark [6] using $N = 3$ views. All models are *only trained* on DTU and tested on BlendedMVS. We borrow the rendered images of the 3D meshes of MVSNeRF [2] and SparseNeuS [10] from [10].



(a) Intermediate normal map using *autograd* (b) Intermediate normal map using manual gradient calculation

Figure 8. Difference between the result of using *autograd* and manual gradient calculation by preventing inter-point SDF relation born due to Ray Transformer. Green, red and yellow colors correspond to leftward, upward and rightward normal vectors, respectively.

pairwise cosine similarity for views provided as well as additional geometry encoding features such as group-wise variance. Using the extracted features per 3D points, the MatchSDF decoder incorporates a Ray Transformer in order to benefit from inter-point relation between points along the same ray. In addition, MatchSDF also incorporates a color blending scheme to achieve more accurate novel view synthesis in order to improve training signals.

Our method outperforms SparseNeuS [10] by a small margin on the DTU benchmark [6] and does not suffer from limitations related to the requirement of a predefined reference view and construction of a cost volume with a limited size.

## References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4, 5

[2] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 2, 3, 6, 7, 8, 9, 10

[3] Yuedong Chen, Haofei Xu, Qianyi Wu, Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Explicit correspondence matching for generalizable neural radiance fields. *arXiv preprint arXiv:2304.12294*, 2023. 2, 3, 4, 5, 6, 7, 8, 9

[4] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 3, 6, 8

[5] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 6

[6] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanaes. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 1, 6, 7, 8, 9, 10

[7] M. M. Johari, Y. Lepoittevin, and F. Fleuret. Geonerf: Generalizing nerf with geometry priors. *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 6

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 7

[9] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Theobalt Christian, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, 2022. 2

[10] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. *ECCV*, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

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

[12] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021. 1, 2

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 7, 8

[14] Yufan Ren, Fangjinhua Wang, Tong Zhang, Marc Pollefeys, and Sabine Süsstrunk. Volrecon: Volume rendering of signed ray distance functions for generalizable multi-view reconstruction, 2023. 2, 3, 4, 5, 6, 7, 8, 9

[15] 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. *NeurIPS*, 2021. 1, 2, 4, 5, 6

[16] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2, 4, 5, 7, 8, 9

[17] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8121–8130, 2022. 3

[18] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 1, 2

[19] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2, 6

[20] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2, 6

[21] Jingyang Zhang, Yao Yao, and Long Quan. Learning signed distance field for multi-view surface reconstruction. *International Conference on Computer Vision (ICCV)*, 2021. 2