Neural RGB-D Surface Reconstruction

Dejan Azinović1  Ricardo Martin-Brualla2  Dan B Goldman2  Matthias Nießner1  Justus Thies1,3
1Technical University of Munich  2Google Research  3Max Planck Institute for Intelligent Systems

Figure 1. Our method obtains a high-quality 3D reconstruction from an RGB-D input sequence by training a multi-layer perceptron. The core idea is to reformulate the neural radiance field definition in NeRF [48], and replace it with a differentiable rendering formulation based on signed distance fields which is specifically tailored to geometry reconstruction.

Abstract

Obtaining high-quality 3D reconstructions of room-scale scenes is of paramount importance for upcoming applications in AR or VR. These range from mixed reality applications for teleconferencing, virtual measuring, virtual room planning, to robotic applications. While current volume-based view synthesis methods that use neural radiance fields (NeRFs) show promising results in reproducing the appearance of an object or scene, they do not reconstruct an actual surface. The volumetric representation of the surface based on densities leads to artifacts when a surface is extracted using Marching Cubes, since during optimization, densities are accumulated along the ray and are not used at a single sample point in isolation. Instead of this volumetric representation of the surface, we propose to represent the surface using an implicit function (truncated signed distance function). We show how to incorporate this representation in the NeRF framework, and extend it to use depth measurements from a commodity RGB-D sensor, such as a Kinect. In addition, we propose a pose and camera refinement technique which improves the overall reconstruction quality. In contrast to concurrent work on integrating depth priors in NeRF which concentrates on novel view synthesis, our approach is able to reconstruct high-quality, metrical 3D reconstructions.

1. Introduction

Research on neural networks for scene representations and image synthesis has made impressive progress in recent years [73]. Methods that learn volumetric representations [42, 48] from color images captured by a smartphone camera can be employed to synthesize near photorealistic images from novel viewpoints. While the focus of these methods lies on the reproduction of color images, they are not able to reconstruct metric and clean (noise-free) meshes. To overcome these limitations, we show that there is a significant advantage in taking additional range measurements from consumer-level depth cameras into account. Inexpensive depth cameras are broadly accessible and are also built into modern smartphones. While classical reconstruction methods [9, 33, 53] that purely rely on depth measurements struggle with the limitations of physical sen-
construction quality which we demonstrate on synthetic as well as on real data from ScanNet [12].

In summary, we propose an RGB-D based scene reconstruction method that leverages both dense color and depth observations. It is based on an effective incorporation of depth measurements into the optimization of a neural radiance field using a signed distance-based surface representation to store the scene geometry. It is able to reconstruct geometry detail that is observed by the color images, but not visible in the depth maps. In addition, our pose and camera refinement technique is able to compensate for misalignments in the input data, resulting in state-of-the-art reconstruction quality which we demonstrate on synthetic as well as on real data from ScanNet [12].

2. Related Work

Our approach reconstructs geometry from a sequence of RGB-D frames, leveraging both dense color and depth information. It is related to classical fusion-based 3D reconstruction methods [9, 14, 50, 53, 93], learned 3D reconstruction [7, 15, 47, 58, 79], as well as to recent coordinate-based scene representation models [48, 69, 74].

**Classical 3D Reconstruction.** There exists a wide range of methods for RGB and RGB-D based 3D reconstruction that are not based on deep learning. Reconstructing objects and scenes can be done using passive stereo systems that rely on stereo matching from two or multiple color views [29, 62], Structure-from-Motion [63], or SLAM-based [20, 21, 23] methods. These approaches may use disjoint representations, like oriented patches [25], volumes [38], or meshes [32] to reconstruct the scene or object. Zollhöfer et al. [93] review the 3D reconstruction methods that rely on range data from RGB-D cameras like the Kinect. Most of these methods are based on [9], where multiple depth measurements are fused using a signed distance function (SDF) which is stored in a uniform 3D grid. E.g., KinectFusion [50] combines such representation with real-time tracking to reconstruct objects and small scenes in real-time. To handle large scenes Nießner et al. [53] propose a memory-efficient storage of the SDF grid using spatial hashing. To handle the loop closure problem when scanning large-scale scenes, bundle adjustment can be used to refine the camera poses [14]. In addition, several regularization techniques have been proposed to handle outliers during reconstruction [19, 61, 89].

**Deep Learning for 3D Reconstruction.** To reduce artifacts from classical reconstruction methods, a series of methods was proposed that use learned spatial priors to predict depth maps from color images [24, 28, 39], to learn multi-view stereo using 3D CNNs on voxel grids [34, 68, 82], or multi-plane images [22], to reduce the influence of noisy depth values [79], to complete incomplete scans [13, 15], to learn image features for SLAM [2, 10, 91] or feature fusion [4, 71, 80], to predict normals [90], or to predict objects or parts of a room from single images [11, 18, 27, 51, 75]. Most recently coordinate-based models have become popular [74]. These models use a scene representation that is based on a deep neural network with fully connected layers, i.e., a multi-layer perceptron (MLP) [73, 74]. As input the MLP takes a 3D location in the model space and outputs for example, occupancy [7, 47, 52, 55, 58–60], density [48], radiance [48], color [54], or the signed distance to the surface [56, 83, 84]. Scene Representation Networks [69] combine such a representation with a learned renderer which is inspired by classical sphere tracing, to reconstruct objects from single RGB images. Instead, Mildenhall et al. [48] propose a method that represents a scene as a neural ra-
Figure 2. Differentiable volumetric rendering is used to reconstruct a scene that has been captured using an RGB-D camera. The scene is represented using multi-layer perceptrons (MLPs), encoding a signed distance value $D_i$ and a viewpoint-dependent radiance value $c_i$ per point $p_i$. We perform volumetric rendering by integrating the radiance along a ray, weighing the samples as a function of their signed distance $D_i$ and their visibility. We also learn a per-frame latent corrective code to account for exposure or white balance changes throughout the capture, which is passed to the radiance MLP alongside the ray direction $d$. We optimize the scene representation's MLPs, together with the per-frame corrective codes, the input camera poses, and an image-plane deformation field (not shown) by computing losses for the signed distance $D_i$ of the samples, and the final integrated color $C$ with respect to the input depth and color views.

Our method is built upon a hybrid scene representation which combines an implicit surface representation with a volumetric radiance field (NeRF) using a coordinate-based model, and a classical, fixed volumetric rendering formulation [46]. Based on this representation, they show impressive novel view synthesis results, while only requiring color input images with corresponding camera poses and intrinsics. Besides the volumetric image formation, a key component of the NeRF technique is a positional encoding layer, that uses sinusoidal functions to improve the learning properties of the MLP. In follow-up work, alternatives to the positional encoding were proposed, such as Fourier features [72] or sinusoidal activation layers [67]. NeRF has been extended to handle in-the-wild data with different lighting and occluders [45], dynamic scenes [40, 57], avatars [26], and adapted for generative modeling [5, 66] and image-based rendering [77, 87]. Others have focused on resectioning a camera given a learned NeRF [85], and optimizing for the camera poses while learning a NeRF [41, 78].

In our work, we take advantage of the volumetric rendering of NeRF and propose the usage of a hybrid scene representation that consists of an implicit surface representation (SDF) and a volumetric representation for the radiance. As illustrated in Fig. 2, we use differentiable volumetric integration of the radiance values [46] to compute color images from this representation. Besides the scene representation network, we optimize for the camera poses and intrinsics. We initialize the camera poses $T_i$ using BundleFusion [14]. At evaluation time, we use Marching Cubes [43] to extract a triangle mesh from the optimized implicit scene representation.

3. Method

We propose an optimization-based approach for geometry reconstruction from an RGB-D sequence of a consumer-level camera (e.g., a Microsoft Kinect). We leverage both the $N$ color frames $I_i$ as well as the corresponding aligned depth frames $D_i$ to optimize a coordinate-based scene representation network. Specifically, our hybrid scene representation consists of an implicit surface representation based on a truncated signed distance function (TSDF) and a volumetric representation for the radiance. As illustrated in Fig. 2, we use differentiable volumetric integration of the radiance values [46] to compute color images from this representation. Besides the scene representation network, we optimize for the camera poses and intrinsics. We initialize the camera poses $T_i$ using BundleFusion [14]. At evaluation time, we use Marching Cubes [43] to extract a triangle mesh from the optimized implicit scene representation.

3.1. Hybrid Scene Representation

Our method is built upon a hybrid scene representation which combines an implicit surface representation with a
volumetric appearance representation. Specifically, we implement this representation using a multi-layer perceptron (MLP) which can be evaluated at arbitrary positions \( \mathbf{p} \) in space to compute a truncated signed distance value \( D_i \) and view-dependent radiance value \( \mathbf{c}_i \). As a conditioning to the MLP, we use a sinusoidal positional encoding \( \gamma(\cdot) \) \cite{10.1145/3381360.3398574} to encode the 3D query point \( \mathbf{p} \) and the viewing direction \( \mathbf{d} \).

Inspired by the recent success of volumetric integration in neural rendering \cite{10.1145/3381360.3398574}, we render color as a weighted sum of radiance values along a ray. Instead of computing the weights as probabilities of light reflecting at a given sample point based on the density of the medium \cite{10.1145/3381360.3398574}, we compute weights directly from signed distance values as the product of two sigmoid functions:

\[
    w_i = \sigma \left( \frac{D_i}{tr} \right) \cdot \sigma \left( -\frac{D_i}{tr} \right),
\]

where \( tr \) is the truncation distance. This bell-shaped function has its peak at the surface, i.e., at the zero-crossing of the signed distance values. A similar formulation is used in concurrent work \cite{8285245}, since this function produces unbiased estimates of the signed distance field. The truncation distance \( tr \) directly controls how quickly the weights fall to zero as the distance from the surface increases. To account for the possibility of multiple intersections, weights of samples beyond the first truncation region are set to zero. The color along a specific ray is approximated as a weighted sum of the \( K \) sampled colors:

\[
    \mathbf{C} = \frac{1}{\sum_{i=0}^{K-1} w_i} \sum_{i=0}^{K-1} w_i \cdot \mathbf{c}_i.
\]

This scheme gives the highest integration weight to the point on the surface, while points farther away from the surface have lower weights. Although such an approach is not derived from a physically-based rendering model, as is the case with volumetric integration over density values, it represents an elegant way to render color in a signed distance field in a differentiable manner, and we show that it helps deduce depth values through a photometric loss (see Sec. 4). In particular, this approach allows us to predict hard boundaries between occupied and free space which results in high-quality 3D reconstructions of the surface. In contrast, density-based models \cite{10.1145/3381360.3398574} can introduce semi-transparent matter in front of the actual surface to represent view-dependent effects when integrated along a ray. This leads to noisy reconstructions and artifacts in free space, as can be seen in Sec. 4.

**Network Architecture** Our hybrid scene representation network is composed of two MLPs which represent the shape and radiance, as depicted in Fig. 2. The shape MLP takes the encoding of a queried 3D point \( \gamma(\mathbf{p}) \) as input and outputs the truncated signed distance \( D_i \) to the nearest surface. The task of the second MLP is to produce the surface radiance for a given encoded view direction \( \gamma(\mathbf{d}) \) and an intermediate feature output of the shape MLP. The view vector conditioning allows our method to deal with view-dependent effects like specular highlights, which would otherwise have to be modeled by deforming the geometry. Since color data is often subject to varying exposure or white-balance, we learn a per-frame latent corrective code vector as additional input to the radiance MLP \cite{891502}.

**Pose and Camera Refinement** The camera poses \( T_i \), represented with Euler angles and a translation vector for every frame, are initialized with BundleFusion \cite{222816} and refined during the optimization. Inspired by \cite{10.1145/118048.118051}, an additional image-plane deformation field in form of a 6-layer ReLU MLP is added as a residual to the pixel location before unprojecting into a 3D ray to account for possible distortions in the input images or inaccuracies of the intrinsic camera parameters. Note that this correction field is the same for every frame. During optimization, camera rays are first shifted with the 2D vector retrieved from the deformation field, before being transformed to world space using the camera pose \( T_i \).

### 3.2. Optimization

We optimize our scene representation network by randomly sampling a batch of \( P_b \) pixels from the input dataset of color and depth images. For each pixel \( p \) in the batch, a ray is generated using its corresponding camera pose and \( S_p \) sample points are generated on the ray. Our global objective function \( \mathcal{L}(\mathbb{P}) \) is minimized w.r.t. the unknown parameters \( \mathbb{P} \) (the network parameters \( \Theta \) and the camera poses \( T_i \)) over all \( B \) input batches and is defined as:

\[
    \mathcal{L}(\mathbb{P}) = \sum_{b=0}^{B-1} \lambda_1 \mathcal{L}_{rgb}^b(\mathbb{P}) + \lambda_2 \mathcal{L}_{fs}^b(\mathbb{P}) + \lambda_3 \mathcal{L}_{tr}^b(\mathbb{P}).
\]

\( \mathcal{L}_{rgb}^b(\mathbb{P}) \) measures the squared difference between the observed pixel colors \( \hat{C}_p \) and predicted pixel colors \( C_p \) of the \( b \)-th batch of rays:

\[
    \mathcal{L}_{rgb}^b(\mathbb{P}) = \frac{1}{|P_b|} \sum_{p \in P_b} (C_p - \hat{C}_p)^2.
\]

\( \mathcal{L}_{fs}^b(\mathbb{P}) \) is a ‘free-space’ objective, which forces the MLP to predict a value of \( tr \) for samples \( s \in S_p^{fs} \), which lie between the camera origin and the truncation region of a surface:

\[
    \mathcal{L}_{fs}^b(\mathbb{P}) = \frac{1}{|P_b|} \sum_{p \in P_b} \frac{1}{|S_p^{fs}|} \sum_{s \in S_p^{fs}} (D_s - tr)^2.
\]

For samples within the truncation region \( (s \in S_p^{tr}) \), we apply \( \mathcal{L}_{tr}^b(\mathbb{P}) \), the signed distance objective of samples close
ADAM optimizer [36] with a learning rate of

to the surface:

\[ L^b_{tr} (P) = \frac{1}{P_b} \sum_{p \in P_b} \frac{1}{|S(c)|} \sum_{s \in S(c)} (D_s - \hat{D}_s)^2, \]  

where \( D_s \) is the predicted signed distance of sample \( s \), and \( \hat{D}_s \) the signed distance observed by the depth sensor, along the optical axis. In our experiments, we use a truncation distance \( tr = 5 \) cm, and scale the scene so that the truncation region maps to \([-1, 1]\) (positive in front of the surface, negative behind).

The \( S_p \) sample points on the ray are generated in two steps. In the first step \( S'_c \) sample points are generated on the ray using stratified sampling. Evaluating the MLP on these \( S'_c \) sample points allows us to get a coarse estimate for the ray depth by explicitly searching for the zero-crossing in the predicted signed distance values. In the second step, another \( S'_f \) sample points are generated around the zero-crossing and a second forward pass of the MLP is performed with these additional samples. The output of the MLP is concatenated to the output from the first step and color is integrated using all \( S'_c + S'_f \) samples, before computing the objective loss. It is important that the sampling rate in the first step is high enough to produce samples within the truncation region of the signed distance field, otherwise the zero-crossing may be missed.

We implement our method in Tensorflow using the ADAM optimizer [36] with a learning rate of \( 5 \times 10^{-4} \) and set the loss weights to \( \lambda_1 = 0.1, \lambda_2 = 10 \) and \( \lambda_3 = 6 \times 10^3 \). We run all of our experiments for \( 2 \times 10^5 \) iterations, where in each iteration we compute the gradient w.r.t. \( |P_b| = 1024 \) randomly chosen rays. We set the number of \( S'_f \) samples to 16. \( S'_c \) is chosen such that there is on average one sample for every 1.5 cm of the ray length. The ray length itself needs to be greater than the largest distance in the scene that is to be reconstructed and ranges from 4 to 8 meters in our scenes.

4. Results

In the following, we evaluate our method on real, as well as on synthetic data. For the shown results, we use Marching Cubes [43] with a spatial resolution of 1 cm to extract a mesh from the reconstructed signed distance function.

Results on real data. We test our method on the ScanNet dataset [12] which provides RGB-D sequences of room-scale scenes. The data has been captured with a StructureIO camera which provides quality similar to that of a Kinect v1. The depth measurements are noisy and often miss structures like chair legs or other thin geometry. To this end our method proposes the additional usage of a dense color reconstruction loss, since regions that are missed by the range sensor are often captured by the color camera. To compensate for the exposure and white balancing of the used camera, our approach learns a per-frame latent code as proposed in [45]. In Fig. 3, we compare our method to the original ScanNet BundleFusion reconstructions which often suffer from severe camera pose misalignment. Our approach jointly optimizes for the scene representation network as well as the camera poses, leading to substantially reduced misalignment artifacts in the reconstructed geometry.

Quantitative evaluation. We perform a quantitative evaluation of our method on a dataset of 10 synthetic scenes for which the ground truth geometry and camera trajectory are known. Note that the ground truth camera trajectory is only used for the rendering and evaluation, and not for the reconstruction. For each frame, we render a photo-realistic image using Blender [8, 17]. We apply noise and artifacts, similar to those of a real depth sensor [1, 3, 30, 31]. On this data, we compare our technique to several state-of-the-art methods that use either depth input only, or both color and depth data to reconstruct geometry (see Tab. 1).

BundleFusion. BundleFusion [14] uses the color and depth input to reconstruct the scene. It is a classical depth fusion approach [9] which compensates misalignments using a global bundle adjustment approach.

RoutedFusion. RoutedFusion [79] uses a routing network which filters sensor-specific noise and outliers from depth maps and computes pixel-wise confidence values, which are used by a fusion network to produce the final SDF. It takes the depth maps and camera poses as input.

COLMAP with screened Poisson surface reconstruction. We obtain camera poses using COLMAP [63–65] and use these to back-project depth maps into world space. We obtain a mesh by applying screened Poisson surface reconstruction [35] on the resulting point cloud.

Convolutional Occupancy Networks. We accumulate the point clouds from the depth maps using BundleFusion poses and evaluate the pre-trained convolutional occupancy networks model [58] provided by the authors (which has been used on similar data [6]).

<table>
<thead>
<tr>
<th>Method</th>
<th>C-( \ell_1 )</th>
<th>IoU ↑</th>
<th>NC ↑</th>
<th>F-score ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleFusion</td>
<td>0.062</td>
<td>0.594</td>
<td>0.892</td>
<td>0.805</td>
</tr>
<tr>
<td>RoutedFusion</td>
<td>0.057</td>
<td>0.615</td>
<td>0.864</td>
<td>0.838</td>
</tr>
<tr>
<td>COLMAP + Poisson</td>
<td>0.057</td>
<td>0.619</td>
<td>0.901</td>
<td>0.839</td>
</tr>
<tr>
<td>Conv. Occ. Nets</td>
<td>0.077</td>
<td>0.461</td>
<td>0.849</td>
<td>0.643</td>
</tr>
<tr>
<td>SIREN</td>
<td>0.060</td>
<td>0.603</td>
<td>0.893</td>
<td>0.816</td>
</tr>
<tr>
<td>NeRF + Depth</td>
<td>0.065</td>
<td>0.550</td>
<td>0.768</td>
<td>0.782</td>
</tr>
<tr>
<td>Ours (w/o pose)</td>
<td>0.049</td>
<td>0.655</td>
<td>0.908</td>
<td>0.868</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.044</strong></td>
<td><strong>0.747</strong></td>
<td><strong>0.918</strong></td>
<td><strong>0.924</strong></td>
</tr>
</tbody>
</table>

Table 1. Reconstruction results on a dataset of 10 synthetic scenes. The Chamfer \( \ell_1 \) distance, normal consistency and the F-score [37] are computed between point clouds sampled with a density of 1 point per cm\(^2\), using a threshold of 5 cm for the F-score. We voxelize the mesh to compute the intersection-over-union (IoU) between the predictions and ground truth.
Figure 3. We compare our model without pose optimization and our full model with both the pose optimization and image-plane deformation field to BundleFusion, RoutedFusion, SIREN and a NeRF optimized with depth supervision in scenes 2, 5, 12, and 50 of the ScanNet dataset. Our model without pose optimization recovers smoother meshes than the density-based NeRF model, but still suffers from misalignment artifacts. These are solved by our full model to recover a clean reconstruction.
Figure 4. Accuracy shows how close ground truth points are to predicted points, while completeness shows how close predicted points are to ground truth points. Geometry reconstructed purely through the photometric loss has slightly lower accuracy than geometry for which depth observations were also available. Furthermore, the accuracy and completeness drop in distant areas, which had less multi-view constraints and more noise in the depth measurements.

**SIREN.** We optimize a SIREN [67] per scene using the back-projected point cloud data. The ICL-NUIM [31] scene on which the method was originally tested, is also included in our synthetic dataset.

**NeRF with an additional depth loss.** NeRF [48] proposes using the expected ray termination distance as a way to visualize the depth of the scene. In our baseline, we add an additional loss to NeRF, where this depth value is compared to the input depth using an L2 loss. Note that this baseline still uses NeRF’s density field to represent geometry.

As can be seen in Tab. 1, our approach with camera refinement results in the lowest Chamfer distance, and the highest IoU, normal consistency (mean of the dot product of the ground truth and predicted normals), and F-score [37]. Especially, the comparison to the density-based NeRF with an additional depth constraint shows the benefit of our proposed hybrid scene representation.

**Ablation studies.** We conduct ablation studies to justify our choice of network architecture and training parameters. In Fig. 3, we show the difference between a volumetric representation (density field, ‘NeRF with Depth’) to an implicit surface representation (signed distance field, ‘Ours-Full’) on real data from ScanNet [12]. While representing scenes with a density field works great for color integration, extracting the geometry itself is a challenging problem. Although small variations in density may not affect the integrated color much, they cause visible noise in the extracted geometry and produce floating artifacts in free space. These artifacts can be reduced by choosing a different iso-level for geometry extraction with Marching Cubes, but this leads to less complete reconstructions. In contrast, a signed distance field models a smooth boundary between occupied and free space, and we show that it can be faithfully represented by an MLP. However, the reconstruction quality is still limited by the provided camera poses, as can be seen in Fig. 3 (e.g., the cabinet in the left column). Optimizing for pose corrections further improves the quality of our reconstructions.

**Effect of the photometric term.** A fundamental component of our method is the use of a photometric term to infer depth values which are missing from camera measurements. We analyze the effect of this term on the synthetic scene in Fig. 4, where we simulate missing geometry of the table legs and the meshed basket. In the figure, we visualize the completeness and accuracy. In contrast to a model without the photometric term, our method is still able to reconstruct the missing geometry leveraging the RGB observations.

For our full approach, we also separately evaluate the reconstruction quality of geometry where depth measurements were available and where they were missing. Regions that relied only on color have a somewhat worse average accuracy of 11 mm, compared to 8 mm for regions that had access to depth measurements. We refer the reader to the supplemental material for more details and a qualitative comparison on real data.

**Effect of pose refinement.** We show that initial camera pose estimates can be further improved by jointly optimizing for the rotation and translation parameters of the cameras which are initialized with BundleFusion [14]. We quantitatively evaluate this on all scenes in our synthetic dataset. An aggregate of the positional and rotational errors of different methods is presented in Tab. 2. A detailed per-
Figure 5. Our method improves the camera alignment over the baseline, as visible in the tiles of the floor. The additional image-plane distortion correction results in straight and aligned edges in the reconstruction.

Table 2. Based on our synthetic dataset, we evaluate the average positional and rotational errors of the estimated camera poses. Our method is able to further increase the pose estimation accuracy compared to its BundleFusion initialization.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pos. error (meters) ↓</th>
<th>Rot. error (degrees) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleFusion</td>
<td>0.033</td>
<td>0.571</td>
</tr>
<tr>
<td>COLMAP</td>
<td>0.038</td>
<td>0.692</td>
</tr>
<tr>
<td>Ours</td>
<td>0.021</td>
<td>0.144</td>
</tr>
</tbody>
</table>

Table 3. Ablation of the image-plane deformation field (IPDF) which compensates image space distortions and incorrect intrinsic parameters. The experiment is based on a synthetic scene, where we assume an incorrect focal length of 570 instead of 554.26 (GT).

<table>
<thead>
<tr>
<th>Method</th>
<th>C-ℓ ↓</th>
<th>IoU ↑</th>
<th>NC ↑</th>
<th>F-score ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (w/o IPDF)</td>
<td>0.061</td>
<td>0.266</td>
<td>0.886</td>
<td>0.406</td>
</tr>
<tr>
<td>Ours (w/ IPDF)</td>
<td>0.031</td>
<td>0.609</td>
<td>0.911</td>
<td>0.904</td>
</tr>
</tbody>
</table>

Effect of the image-plane deformation field. To evaluate the effect of the pixel-space deformation field, we initialize the camera with an incorrect focal length and optimize our model with and without the deformation field. Tab. 3 shows that the deformation field mitigates this inaccuracy in the camera’s intrinsic parameters which leads to significantly better reconstruction results compared to the model that does not use the deformation field. Fig. 5 showcases the effects of our camera pose and image-plane deformation field [14]. Blurry frames and sparse features lead to systematic camera pose errors in BundleFusion. Our method improves these camera poses and the camera distortion model, and, thus, is able to better align scene features, resulting in higher reconstruction quality.

Limitations and future work. Similar to other methods that are based on a scene representation which uses a scene-specific MLP, our method runs offline (around 9 hours for $2 \times 10^3$ iterations using an NVIDIA RTX 3090). Recent methods that utilize voxel grids to optimize a radiance field [70, 86] have shown significantly faster convergence compared to earlier MLP-based methods and we believe that they would also be good candidates for improving our method. Nonetheless, our proposed method offers a high-quality scene reconstruction which outperforms online fusion approaches. Another limitation is the global MLP which stores the entire scene information which comes at the cost of missing high-frequency local detail in very large scenes. Approaches like IF-Nets [7] or Convolutional Occupancy Networks [58] benefit from locally-conditioned MLPs and can be integrated in future work. Finally, our method was designed to handle only opaque surfaces.

5. Conclusion

We have presented a new method for 3D surface reconstruction from RGB-D sequences by introducing a hybrid scene representation that is based on an implicit surface function and a volumetric representation of radiance. This allows us to efficiently incorporate depth observations, while still benefiting from the differentiable volumetric rendering of the original neural radiance field formulation. As a result, we obtain high-quality surface reconstructions, outperforming traditional and learned RGB-D fusion methods. Overall, we believe our work is a stepping stone towards leveraging the success of implicit, differentiable representations for 3D surface reconstruction.

Acknowledgements

This work was supported by a Google Gift Grant, a TUM-IAS Rudolf Mößbauer Fellowship, an NVIDIA Professorship Award, the ERC Starting Grant Scan2CAD (804724), and the German Research Foundation (DFG) Grant Making Machine Learning on Static and Dynamic 3D Data Practical. We would also like to thank Angela Dai for the video voice-over.
References


[20] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In European Conference on Computer Vision (ECCV), September 2014. 2


[87] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 3


Our method obtains a high-quality 3D reconstruction from an RGB-D input sequence by training a multi-layer perceptron. In comparison to state-of-the-art methods like BundleFusion [14] or the theoretical NeRF [48] with additional depth constraints, our approach results in cleaner and more complete reconstructions. As can be seen, the pose optimization of our approach is key to resolving misalignment artifacts.

APPENDIX

In this appendix we show a per-scene breakdown of the quantitative evaluation from Tab. 1, an ablation study on additional scenes from the ScanNet dataset (see Fig. 6), as well as further ablation studies on synthetic data. For the purpose of reproducibility, we also provide further details on the parameters that were used for optimization in each of the scenes.

A. Implementation Details

We implement our method in TensorFlow v2.4.1 using the ADAM [36] optimizer with a learning rate of $5 \times 10^{-4}$ and an exponential learning rate decay of $10^{-1}$ over $2.5 \times 10^5$ iterations. In each iteration, we compute a gradient w.r.t. $|P_b| = 1024$ randomly chosen rays. We set the number of $S'_f$ samples to 16. $S'_c$ is chosen so that there is on average one sample for every 1.5 cm of the ray length. Tab. 4 gives an overview of ray length and number of samples for each of the experiments. Internally, we translate and scale each scene so that it lies within a $[-1, 1]^3$ cube. Depending on scene size, our method takes between 9 and 13 hours to converge on a single NVIDIA RTX 3090 (see Sec. F). We set the loss weights to $\lambda_1 = 0.1$, $\lambda_2 = 10$ and $\lambda_3 = 6 \times 10^3$. We use 8 bands for the positional encoding of the point coordinates and 4 bands to encode the view direction vector.

To account for distortions or inaccuracies of the intrinsic parameters, a 2D deformation field of the camera pixel space in form of a 6-layer MLP, with a width of 128, is used.

B. Per-scene Quantitative Evaluations

In Tab. 8 and Tab. 9 we present a per-scene breakdown of the quantitative analysis from the main paper (see Sec. 4, Tab. 1 and Tab. 2 in the main paper). The corresponding qualitative results are shown in Fig. 14 and Fig. 15.

Reconstruction Evaluation. The goal of our method is to reconstruct a scene from color and depth data, i.e., we do not aim for scene completion. To evaluate the reconstruction quality, we evaluate the quality of reconstructions w.r.t. Chamfer distance (C-$\ell_1$), intersection-over-union (IoU), normal consistency (NC) based on cosine similarity, and F-score. These metrics are computed on surfaces...
Scene | $S'_c$ | ray length (m) | #frames
---|---|---|---
Scene 0 | 512 | 8 | 1394
Scene 2 | 256 | 4 | 1299
Scene 5 | 256 | 4 | 1159
Scene 12 | 320 | 5 | 1335
Scene 24 | 512 | 8 | 849
Scene 50 | 256 | 4 | 1163
Scene 54 | 256 | 4 | 1250
Breakfast room | 320 | 5 | 1167
Green room | 512 | 8 | 1442
Grey-white room | 512 | 8 | 1493
ICL living room | 320 | 5 | 1510
Kitchen 1 | 512 | 8 | 1517
Kitchen 2 | 640 | 10 | 1221
Morning apartment | 256 | 4 | 920
Staircase | 512 | 8 | 1149
Thin geometry | 256 | 4 | 395
White room | 512 | 8 | 1676

Table 4. We list the number of samples $S'_c$ and the ray length in meters that were used to reconstruct each of the ScanNet scenes and the synthetic scenes. Note that these settings are dependent on the scene size.

which were visible in the color and depth streams (geometry within the viewing frusta of the input images). Specifically, we subdivide all meshes to have a maximum edge length of below 1.5 cm and use the ground truth trajectory to detect vertices which are visible in at least one camera. Triangles which have no visible vertices, either due to not being in any of the viewing frusta or due to being occluded by other geometry, are culled. This is necessary to avoid computing the error in regions such as occluded geometry in the synthetic ground truth mesh or in regions where the network output is unpredictable because the region was never seen at training time. The culled geometry is sampled with a density of 1 point per cm$^2$ and the error metrics are evaluated on the sampled point clouds. To evaluate the IoU, we voxelize the reconstruction using voxels with an edge length of 5 cm. The F-score is also computed using a 5 cm threshold.

**Synthetic Dataset.** Our synthetic dataset which we use for numeric evaluation purposes consists of 10 scenes published under either the CC-BY or CC-0 license (see Tab. 5). We define a trajectory by a Catmull-Rom spline interpolation [88] on several manually chosen control points. We use BlenderProc [17] to render color and depth images for each camera pose in the interpolated trajectory. Noise is applied to the depth maps to simulate sensor noise of a real depth sensor [1, 3, 30, 31]. For the ICL scene [31], we use the color and noisy depth provided by the authors and do not render our own images. The scenes in the dataset have various sizes, complexity and materials like highly specular surfaces or mirrors. BundleFusion [14] is used to get an initial estimate of the camera trajectory. This estimated trajectory is used by all methods other than COLMAP to allow a fair comparison.

**C. Ablation Studies**

In this section, we present additional details for the ablation studies described in the main paper, and show further studies to test the robustness and the limitations of our method. In Fig. 6, the additional results on real data demonstrate the advantages of the signed distance field and our camera refinement.

**C.1. Effect of the Photometric Energy Term**

In Tab. 6, we list the quantitative evaluation of the experiment on the effectiveness of the photometric energy term from Fig. 4 in the main paper. Fig. 7 shows the effect of the term on a real scene from the ScanNet dataset. The
Table 6. Detailed reconstruction results for Fig. 4 from the main paper. Our method reconstructs geometry visible only in color images, leading to significantly better reconstruction results in scenes with geometry which is not captured by the depth sensor.

<table>
<thead>
<tr>
<th>Method</th>
<th>C-ℓ₁↓</th>
<th>IoU↑</th>
<th>NC↑</th>
<th>F-score↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (depth-only)</td>
<td>0.017</td>
<td>0.791</td>
<td>0.910</td>
<td>0.944</td>
</tr>
<tr>
<td>Ours (full)</td>
<td>0.009</td>
<td>0.865</td>
<td>0.910</td>
<td>0.995</td>
</tr>
</tbody>
</table>

legs of the piano stool were not visible in any of the depth maps. Nevertheless, our method is able to reconstruct them by making use of the corresponding color data.

C.2. Number of Input Frames

The reconstruction quality of any reconstruction method is dependent on the number of input frames. We evaluate our method on the ‘whiteroom’ synthetic scene through multiple experiments in which we remove different numbers of frames in the dataset used for optimization. Reconstruction results are presented in Fig. 8. Note that for these experiments we use the camera poses initialized with BundleFusion which uses all 1676 depth frames.

C.3. Robustness to Noisy Pose Initialization

To analyze the robustness of our method w.r.t. presence of inaccuracies in camera alignment, we apply Gaussian noise to every camera’s position and direction in the ‘white-room’ scene. In Fig. 9 we present reconstruction results for poses of increasing inaccuracy. We separately show the pose errors of the refined cameras in Fig. 10. On the reconstruction metrics, our method is robust to camera position and orientation errors of up to 5 cm and 5° respectively. The pose refinement is robust up to a noise level of 3 cm and 3°. At noise levels with a standard deviation of 10 cm and higher, some cameras are initially positioned inside geometry, preventing our method from refining their position and leading to large errors in geometry reconstruction.

Figure 8. We test the robustness of our method by removing frames from the dataset used for optimization. Our method achieves good reconstruction results using as few as 13 frames.

Figure 9. We test the robustness of our reconstructions to noise in the initial camera position and direction. Our method achieves good results even in the presence of significant noise. At σ = 10 cm, some of the cameras intersect geometry, degrading the reconstruction quality.

Figure 10. We test the robustness of our pose refinement to noise in the initial camera position and direction. The rotation error has been scaled by a factor of 10 for better visibility. Our method is able to correct poses even in the presence of significant noise. At σ = 10 cm, some of the cameras start intersecting geometry, making refinement impossible.

Table 7. Impact of the truncation region width on reconstruction quality.

<table>
<thead>
<tr>
<th>Truncation (cm)</th>
<th>C-ℓ₁↓</th>
<th>IoU↑</th>
<th>NC↑</th>
<th>F-score↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.053</td>
<td>0.671</td>
<td>0.855</td>
<td>0.862</td>
</tr>
<tr>
<td>3</td>
<td>0.023</td>
<td>0.766</td>
<td>0.901</td>
<td>0.930</td>
</tr>
<tr>
<td>5</td>
<td>0.021</td>
<td>0.786</td>
<td>0.912</td>
<td>0.933</td>
</tr>
<tr>
<td>10</td>
<td>0.024</td>
<td>0.742</td>
<td>0.908</td>
<td>0.912</td>
</tr>
</tbody>
</table>
C.4. Batch Size

Optimization with a lower batch size leads to more noise and might miss areas without depth supervision due to a lower number of multi-view constraints within the batch. A batch size that is too large will slow down the optimization and consume more GPU memory, while not offering improvements in reconstruction quality (see Fig. 11).

C.5. Truncation Size

The reconstruction quality is dependent on the width of the truncation region, as shown in Tab. 7. The truncation region needs to account for the noise in the input (i.e., needs to be greater than the noise of the depth camera). In our experiments a truncation radius of $tr = 5$ cm gives the best results (evaluated based on the mean across multiple scenes).

D. Comparison to RGB-based methods

NeuS [76] and VolSDF [83] are concurrent works that propose learning a signed distance field of an object from a set of RGB images. In contrast to these methods, our focus lies on reconstructing indoor scenes which often have large textureless regions (e.g., a white wall). Methods which use only color input will not have enough multi-view constraints to properly reconstruct these regions. In Fig. 12, we show a case where methods that rely only on color input struggle to reconstruct high-quality geometry.

E. Color Reproduction of Classic and NeRF-style Methods

While our focus lies on geometry reconstruction and not accurate view synthesis, we conducted a brief analysis of the advantages and drawbacks of classic reconstruction methods [14, 92] and MLP-based radiance fields [48] when synthesizing unseen views. Classic reconstruction methods usually do not try to decouple intrinsic material parameters [14, 92] and instead optimize a texture that represents the average observation of all the input views. The resulting texture is usually high-resolution (bounded by the resolution of the input images), but does not allow for correct synthesis of view-dependent effects. Furthermore, inaccuracies in camera calibration may lead to visible seams in the optimized texture. Methods like NeRF that focus purely on high-quality novel view synthesis do not explicitly reconstruct geometry and may thus produce images riddled with artifacts for views that are too far from the input views.

We believe that it is possible to combine both of these approaches to improve novel view synthesis on views far away from the ones used during the optimization and would like to encourage research in this direction. Fig. 13 shows an example view synthesis result on the ScanNet dataset, for an out-of-trajectory camera position and orientation.
F. Runtime and Memory Requirements

Our method. The runtime and memory requirements of our method are dependent on the scene size. For smaller scenes where it is enough to have $S_c' = 256$ samples, our method completes $2 \times 10^5$ iterations in 9 hours on an NVIDIA RTX 3090 and requires 8.5 GB of GPU memory. When $S_c'$ is set to 512, the runtime increases to 13 hours and the memory requirement to 10.5 GB. The memory consumption can be reduced by using smaller batches.

BundleFusion. We run BundleFusion at a voxel resolution of 1 cm for all scenes. On an NVIDIA GTX TITAN Black, depending on the size of the scene and number of frames in the camera trajectory, it takes 10 to 40 minutes to integrate the depth frames into a truncated signed distance field and extract a mesh using Marching Cubes. The memory usage is around 5.8 GB.

RoutedFusion. To train and test RoutedFusion, we used an NVIDIA RTX 3090. The routing network was trained for 24 hours on images with a resolution of $320 \times 240$ pixels. As per suggestion of the authors, we train the fusion network for 20 epochs which takes about 1.5 hours. We reconstruct all scenes at a voxel resolution of 1 cm for a fair comparison to other methods. The runtime ranges from 40 minutes to 6 hours depending on scene size and number of frames. The memory usage also heavily depends on scene size and ranges from 5.5 GB to 23 GB.

COLMAP + Poisson. In the COLMAP + Poisson baseline, the bottleneck is the global bundle adjustment process performed by COLMAP. The total runtime depends on the number of frames in the trajectory. Using all 8 cores of an Intel i7-7700K CPU, it took us about 4 hours to align all 1167 cameras in the ‘breakfast room’. The couple of minutes needed to backproject all depth maps at full resolution and run the screened Poisson surface reconstruction are negligible in comparison.

Convolutional Occupancy Networks. We reconstruct each scene using the pre-trained model provided by the authors. This takes about 2 minutes per scene and requires about 10 GB of memory.

SIREN. We train SIREN for $10^4$ epochs on each scene. SIREN is trained over the complete point cloud in each epoch, so the runtime depends on the number of points in the point cloud. In our experiments on an NVIDIA RTX 3090, this ranged from 6 to 12 hours with 12 GB of memory being in use.

NeRF + Depth. We optimize NeRF using 64 samples for the coarse network and 128 samples for the fine network. On an NVIDIA RTX 3090 it takes 6 hours for $2 \times 10^5$ iterations to run. The memory usage is 4.7 GB.
Table 8. We compare the quality of our reconstruction on several synthetic scenes for which ground truth data is available. The Chamfer \( \ell_1 \) distance, normal consistency and the F-score [37] are computed between point clouds sampled with a density of 1 point per cm\(^2\). We use a threshold of 5 cm for the F-score. We further voxelize each mesh to compute the intersection-over-union (IoU) between the predictions and ground truth.
<table>
<thead>
<tr>
<th>Scene</th>
<th>Method</th>
<th>$C_{\ell_1}$</th>
<th>IoU</th>
<th>NC</th>
<th>F-score</th>
<th>Pos. error</th>
<th>Rot. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kitchen 2</td>
<td>BundleFusion</td>
<td>0.089</td>
<td>0.441</td>
<td>0.856</td>
<td>0.687</td>
<td>0.050</td>
<td>0.566</td>
</tr>
<tr>
<td></td>
<td>RoutedFusion</td>
<td>0.059</td>
<td>0.572</td>
<td>0.842</td>
<td>0.787</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>COLMAP + Poisson</td>
<td>0.037</td>
<td><strong>0.675</strong></td>
<td><strong>0.919</strong></td>
<td>0.818</td>
<td><strong>0.043</strong></td>
<td>1.154</td>
</tr>
<tr>
<td></td>
<td>Conv. Occ. Nets</td>
<td>0.052</td>
<td>0.484</td>
<td>0.861</td>
<td>0.653</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SIREN</td>
<td>0.055</td>
<td>0.453</td>
<td>0.898</td>
<td>0.735</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NeRF + Depth</td>
<td>0.051</td>
<td>0.435</td>
<td>0.708</td>
<td>0.630</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours (w/o pose)</td>
<td>0.034</td>
<td>0.488</td>
<td>0.908</td>
<td>0.796</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td><strong>0.032</strong></td>
<td><strong>0.637</strong></td>
<td><strong>0.903</strong></td>
<td><strong>0.890</strong></td>
<td><strong>0.083</strong></td>
<td><strong>0.450</strong></td>
</tr>
<tr>
<td>Morning apartment</td>
<td>BundleFusion</td>
<td>0.012</td>
<td>0.767</td>
<td>0.885</td>
<td>0.968</td>
<td>0.008</td>
<td>0.165</td>
</tr>
<tr>
<td></td>
<td>RoutedFusion</td>
<td>0.013</td>
<td><strong>0.815</strong></td>
<td>0.870</td>
<td>0.976</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>COLMAP + Poisson</td>
<td>0.017</td>
<td>0.668</td>
<td>0.877</td>
<td>0.959</td>
<td>0.017</td>
<td>0.380</td>
</tr>
<tr>
<td></td>
<td>Conv. Occ. Nets</td>
<td>0.045</td>
<td>0.450</td>
<td>0.802</td>
<td>0.784</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SIREN</td>
<td>0.013</td>
<td>0.727</td>
<td>0.873</td>
<td>0.966</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NeRF + Depth</td>
<td>0.022</td>
<td>0.587</td>
<td>0.838</td>
<td>0.975</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours (w/o pose)</td>
<td><strong>0.011</strong></td>
<td>0.787</td>
<td>0.887</td>
<td><strong>0.983</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td><strong>0.011</strong></td>
<td><strong>0.716</strong></td>
<td><strong>0.888</strong></td>
<td><strong>0.982</strong></td>
<td><strong>0.005</strong></td>
<td><strong>0.093</strong></td>
</tr>
<tr>
<td>Staircase</td>
<td>BundleFusion</td>
<td>0.091</td>
<td>0.373</td>
<td>0.860</td>
<td>0.623</td>
<td>0.039</td>
<td>0.643</td>
</tr>
<tr>
<td></td>
<td>RoutedFusion</td>
<td>0.069</td>
<td>0.340</td>
<td>0.864</td>
<td>0.622</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>COLMAP + Poisson</td>
<td>0.074</td>
<td>0.322</td>
<td>0.895</td>
<td>0.628</td>
<td>0.043</td>
<td>0.305</td>
</tr>
<tr>
<td></td>
<td>Conv. Occ. Nets</td>
<td>0.069</td>
<td>0.315</td>
<td>0.838</td>
<td>0.508</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SIREN</td>
<td>0.067</td>
<td>0.432</td>
<td>0.885</td>
<td>0.676</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NeRF + Depth</td>
<td>0.087</td>
<td>0.396</td>
<td>0.644</td>
<td>0.624</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours (w/o pose)</td>
<td>0.057</td>
<td>0.457</td>
<td>0.899</td>
<td>0.704</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td><strong>0.045</strong></td>
<td><strong>0.565</strong></td>
<td><strong>0.920</strong></td>
<td><strong>0.853</strong></td>
<td><strong>0.016</strong></td>
<td><strong>0.123</strong></td>
</tr>
<tr>
<td>Thin geometry</td>
<td>BundleFusion</td>
<td>0.019</td>
<td>0.764</td>
<td>0.909</td>
<td>0.922</td>
<td><strong>0.009</strong></td>
<td>0.126</td>
</tr>
<tr>
<td></td>
<td>RoutedFusion</td>
<td>0.023</td>
<td>0.708</td>
<td>0.829</td>
<td>0.881</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>COLMAP + Poisson</td>
<td>0.047</td>
<td>0.440</td>
<td>0.820</td>
<td>0.721</td>
<td>0.079</td>
<td>2.400</td>
</tr>
<tr>
<td></td>
<td>Conv. Occ. Nets</td>
<td>0.022</td>
<td>0.723</td>
<td>0.882</td>
<td>0.910</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SIREN</td>
<td>0.021</td>
<td>0.733</td>
<td>0.887</td>
<td>0.913</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NeRF + Depth</td>
<td>0.014</td>
<td>0.825</td>
<td>0.847</td>
<td>0.989</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours (w/o pose)</td>
<td><strong>0.009</strong></td>
<td><strong>0.857</strong></td>
<td><strong>0.911</strong></td>
<td><strong>0.995</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td><strong>0.009</strong></td>
<td><strong>0.865</strong></td>
<td><strong>0.910</strong></td>
<td><strong>0.995</strong></td>
<td><strong>0.010</strong></td>
<td><strong>0.037</strong></td>
</tr>
<tr>
<td>White room</td>
<td>BundleFusion</td>
<td>0.062</td>
<td>0.528</td>
<td>0.869</td>
<td>0.701</td>
<td>0.045</td>
<td>0.375</td>
</tr>
<tr>
<td></td>
<td>RoutedFusion</td>
<td>0.038</td>
<td>0.545</td>
<td>0.817</td>
<td>0.799</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>COLMAP + Poisson</td>
<td>0.036</td>
<td>0.652</td>
<td>0.904</td>
<td>0.796</td>
<td><strong>0.018</strong></td>
<td>0.167</td>
</tr>
<tr>
<td></td>
<td>Conv. Occ. Nets</td>
<td>0.061</td>
<td>0.424</td>
<td>0.853</td>
<td>0.470</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SIREN</td>
<td>0.046</td>
<td>0.617</td>
<td>0.888</td>
<td>0.752</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NeRF + Depth</td>
<td>0.073</td>
<td>0.385</td>
<td>0.716</td>
<td>0.619</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours (w/o pose)</td>
<td>0.034</td>
<td>0.631</td>
<td>0.902</td>
<td>0.813</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td><strong>0.028</strong></td>
<td><strong>0.738</strong></td>
<td><strong>0.911</strong></td>
<td><strong>0.915</strong></td>
<td><strong>0.028</strong></td>
<td><strong>0.133</strong></td>
</tr>
</tbody>
</table>

Table 9. We compare the quality of our reconstruction on several synthetic scenes for which ground truth data is available. The Chamfer $\ell_1$ distance, normal consistency and the F-score [37] are computed between point clouds sampled with a density of 1 point per cm$^2$. We use a threshold of 5 cm for the F-score. We further voxelize each mesh to compute the intersection-over-union (IoU) between the predictions and ground truth.
Figure 14. We show a qualitative comparison of synthetic scene reconstructions obtained using our method and several baseline methods. The BundleFusion reconstruction is incomplete in some regions, screened Poisson and SIREN attempt to fit noise in the depth data, while the NeRF reconstruction suffers from noise in the density field. Our method manages to fill in gaps in geometry, while maintaining the smoothness of classic fusion approaches.
Figure 15. We show a qualitative comparison of synthetic scene reconstructions obtained using our method and several baseline methods. The BundleFusion reconstruction is incomplete in some regions, screened Poisson and SIREN attempt to fit noise in the depth data, while the NeRF reconstruction suffers from noise in the density field. Our method manages to fill in gaps in geometry, while maintaining the smoothness of classic fusion approaches.