

TMO: Textured Mesh Acquisition of Objects with a Mobile Device by using Differentiable Rendering

Jaehoon Choi^{1,2} Dongki Jung¹ Taejae Lee¹ Sangwook Kim¹ Youngdong Jung¹
 Dinesh Manocha² Donghwan Lee¹
¹NAVER LABS ²University of Maryland

Abstract

We present a new pipeline for acquiring a textured mesh in the wild with a single smartphone which offers access to images, depth maps, and valid poses. Our method first introduces an RGBD-aided structure from motion, which can yield filtered depth maps and refines camera poses guided by corresponding depth. Then, we adopt the neural implicit surface reconstruction method, which allows for high-quality mesh and develops a new training process for applying a regularization provided by classical multi-view stereo methods. Moreover, we apply a differentiable rendering to fine-tune incomplete texture maps and generate textures which are perceptually closer to the original scene. Our pipeline can be applied to any common objects in the real world without the need for either in-the-lab environments or accurate mask images. We demonstrate results of captured objects with complex shapes and validate our method numerically against existing 3D reconstruction and texture mapping methods.

1. Introduction

Recovering the 3D geometry of objects and scenes is a longstanding challenge in computer vision and is essential to a broad range of applications. Depth sensing technologies range from highly specialized and expensive turn-table 3D scanners and structured-light scanners to commodity depth sensors. More recently, advances in mobile devices have developed a new method for 3D capture of real-world environments with high-resolution imaging and miniaturized LiDAR. Specifically, the modern smartphone such as iPhone 13 Pro are equipped with RGB camera, accelerometer, gyroscope, magnetometer, and LiDAR scanner. These various sensors can provide high-resolution images, low-resolution depth from the LiDAR scanner, and associated camera poses offered by off-the-shelf visual-inertial odometry (VIO) systems such as ARKit [1] and ARCore [2].

Today’s smartphones offer low-resolution depth maps

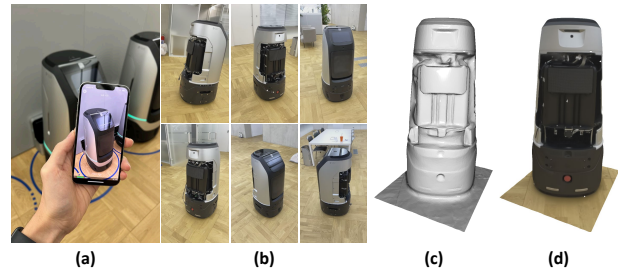


Figure 1. Example reconstruction results collected from a smartphone in the wild. (a) Data acquisition setup. (b) Images captured from a smartphone. (c) A reconstructed mesh. (d) A novel view of textured mesh. Our proposed method can reconstruct the high-quality geometric mesh with a visually realistic texture.

[5] and valid poses. However, depth maps are very noisy and suffer from the limited range of depth sensors. Although this depth sensor can build a simple 3D structure such as a wall or floor, it cannot reconstruct objects with complex and varying shapes. Thus, the RGBD scanning methods [9, 23, 42, 57] are not suitable for these objects. Instead of the depth sensor, multi-view stereo (MVS) algorithms [13, 47, 62] reconstruct high-quality 3D geometry by matching feature correspondences across different RGB images and optimizing photometric consistency. While this pipeline is very robust in real-world environments, it misses the surface of low-textured areas [62]. Additionally, the resulting mesh generated by post-processing like Poisson reconstruction [27] heavily depends on the quality of matching, and the accumulated errors in correspondence matching often cause severe artifacts and missing surfaces. Because of the cumulative error from the above pipeline, the texture mapping process [53, 69] leads to undesirable results. Reconstructing high-fidelity texture and 3D geometry of real-world 3D objects remains an open challenge.

Main Results: In this paper, we present a practical method to capture a high-quality textured mesh of a 3D object in the wild, shown in Fig. 1. We first develop a smartphone app based on ARKit [1] to collect images, LiDAR depths, and poses. Although the smartphone provides quite valid pose estimates, acquiring fine detail geometry and realis-

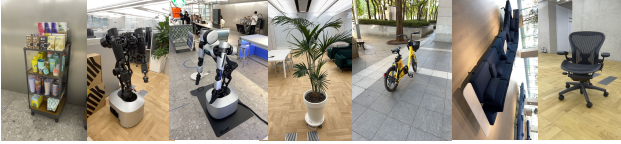


Figure 2. Example objects collected by a smartphone in the wild

tic texture requires highly accurate camera poses. Thus, we present an RGBD-aided Structure from Motion (SfM) which combines the advantages of both VIO [12] and SfM [46]. Since ARKit based on VIO is robust to the degradation of visual information such as low-textured surface, we perform incremental triangulation with initial poses obtained from ARKit. We also propose a depth filtering method to handle noisy depth from the smartphone. These filtered depth points are exploited as an additional depth factor for bundle adjustment. Our RGBD-aided SfM can estimate highly precise camera poses due to the good initial poses and additional constraints from the filtered depth.

Our 3D geometry reconstruction process adopts a neural implicit representation [50, 55] for surface reconstruction with volumetric rendering. We observe that the neural implicit representation can perform more complete and smoother reconstruction than the classical methods which are known to be robust for 3D reconstruction in the wild. Furthermore, we introduce a new training method for neural implicit representations. In the early stage of training, we propose a regularization method that leverages the prior information from the classical MVS method. After obtaining the decent shape, we avoid using the prior information and generate sparse voxel octree to enable efficient sampling for training. Our training method can improve the performance of the neural implicit representations. Consequently, we show the generalization capabilities of our surface reconstruction method in real world objects collected by the smartphone.

Given a mesh extracted from the trained neural implicit representation, we can run classical texture mapping algorithms [53, 69] to generate texture maps that often exhibit blurring artifacts and color misalignment. We propose applying differential rendering [31] to fine-tune these texture maps obtained from classical texture mapping via a photometric loss. Compared to classical 3D reconstruction [23, 47, 62] and texture mapping [53, 69] approaches, our method shows a better ability to reconstruct the 3D model and produce realistic textures. We evaluate our approach on the data collected by our smartphone application. The main contributions of this paper are summarized as follows:

- We present a unified framework to reconstruct the textured mesh using a smartphone.
- We propose a depth filtering scheme for noisy depths and refine initial poses from ARKit by using bundle adjustment with depth factor.

- Our pipeline builds on classical 3D reconstruction and texture mapping. We leverage a neural geometry representation to enable surface reconstruction of complex shapes and a differentiable rendering to generate high-fidelity texture maps.

2. Related Work

Classical 3D Reconstruction Traditional Structure from Motion (SfM) systems [46, 59] have been popular as a vision-based 3D reconstruction due to their robustness to various scenarios [48, 49, 58]. The SfM systems extract and match features [36] of adjacent images of the same scene and generate the camera poses and sparse 3D points via triangulation [20]. Then, Multi-view Stereo (MVS) algorithms [13, 52] compute the dense depth maps of each scene with multiple calibrated images. PatchMatch-based MVS [14, 47, 61, 62] methods are popular because they are proper for depth and normal optimization. After that, surface reconstruction methods [26, 27, 30] can be applied to reconstruct meshes from point clouds. However, all these methods often fail to reconstruct the 3D geometry of low-textured areas. RGB-D Reconstruction techniques fuse many overlapping depth maps into a single 3D model. Starting from the classical TSDF-fusion [8], several methods [9, 23, 42, 57] produce voxel-based TSDF with commodity-level depth cameras such as Kinect. However, these methods have difficulty handling outliers and thin geometry due to noisy depth measurements. Due to the noise and limited range of the LiDAR sensor in smartphone, RGB-D reconstruction methods fail to estimate the camera odometry.

3D Reconstruction with differentiable rendering Recently, NeRF [38] and its variants [37, 39, 65, 68] have used volumetric implicit representations for rendering and learn α -compositing of a radiance field along rays. These methods yield remarkable quality of novel view synthesis results and do not need ground truth masks. However, 3D geometry from volumetric representations is far from satisfactory due to their volumetric properties [68]. Some methods [41, 64, 67] reconstruct the surface by encoding the network as an implicit function such as a signed distance function (SDF) or occupancy function. Many hybrid approaches [4, 10, 43, 55, 63] unify surface and volumetric representations by regarding surface as defining volumes near the surface, enabling accurate surface reconstruction without masks. Their follow-up works [19, 34, 54, 67] import prior information from the learning-based network. We avoid exploiting prior hints obtained from learning-based models because they are expensive to acquire new training datasets and time-consuming to train other neural networks.

Classical Texture Optimization The matching between texture and geometry is a significant step for building realistic 3D models. Lempitsky et al. [32] apply the pairwise Markov Random Field to choose an optimal view for each

face as texture. Waechter et al. [53] present a global color adjustment to mitigate the visual seams between face textures. Zhou et al. [69] optimize the camera poses using a color consistency measurement and geometric error via local image warping to obtain sharp texture. Bi et al. [6] employ a patch-based image synthesis to produce a texture per each face to optimize the camera pose and geometric error.

Texture Learning Recently, learning-based methods [17, 22, 25, 51] have been proposed to generate textures for a 3D model. Huang et al. [22] introduce adversarial learning for texture generation in RGB-D scans. Thies et al. [51] store neural textures for a known mesh with given UV mapping and decode them to color for novel views. Some works [7, 60] learn neural texture with neural volumetric rendering. Differentiable rendering methods [16, 21, 33, 40] usually focus on estimating surface radiometric properties from images. These methods can learn to predict not only texture but also geometry via image loss with ground truth masks. Unfortunately, differentiable rendering methods heavily rely on pixel-level accurate masks. Acquiring accurate ground-truth masks in the real-world environment is a difficult task [29]. Therefore, we apply the differentiable rendering to fine-tune texture maps with a fixed mesh.

3. Our Method

Given images, low-resolution depths and corresponding camera poses from ARKit, our goal is to reconstruct both the texture and geometry of real-world 3D objects. In Fig. 4, our pipeline consists of three stages: RGBD-aided Structure from Motion (SfM), Geometry Reconstruction, and Texture Optimization. Since both depth maps and camera poses provided by ARKit are very noisy, we construct a SfM system to filter out depth maps and refine poses. We then formalize our task as training a neural model for zero-level set of a signed distance function (SDF) guided by classical 3D reconstruction. After training, we extract the mesh with Marching Cubes [35] and apply classical texture reconstruction [53] to generate initial texture maps. Next, we exploit differentiable rendering [31] to finetune the texture maps with the supervision of multi-view images.

3.1. RGBD-aided Structure from Motion

The goal of this section is to refine initial poses from ARKit by performing a bundle adjustment with depth maps from the smartphone. Unlike simple structures like walls or floors, 3D reconstruction for capturing a high level of detail is required to refine inaccurate poses from ARKit.

Preprocessing We introduce two processes to obtain reliable depth estimates: single-view filtering and multi-view filtering. Single-view filtering utilizes a confidence map provided by ARKit, which has the same resolution as the depth map. Each pixel in the confidence map has 0, 1, or 2 with the value 0 as least confident and 2 serving as most

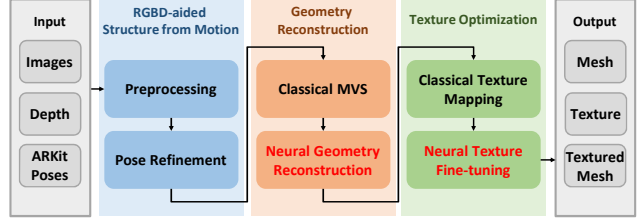


Figure 3. Overview of our pipeline. It is comprised of three steps: RGBD-aided Structure from Motion (Sec. 3.1), Geometry Reconstruction (Sec. 3.2), and Texture Optimization (Sec. 3.3). Our method builds on classical 3D reconstruction and texture mapping. To overcome the limitations of classical methods, we apply a neural geometry representation and neural texture fine-tuning to update 3D geometry and texture maps.

confident with respect to depth-value accuracy. We choose depth pixels corresponding to 1 or 2 in the confidence map. A main concept of the multi-view filtering is to remove outliers in terms of the reprojection error. Given the reference image I_{ref} with depth D_{ref} , the source image I_{src} with depth D_{src} , the depth map D'_{src} can be transformed from the source view to the reference view with accurate pixel-level correspondence. Then, the depth of pixel p in the reference image will be considered as an inlier if the difference δ_{depth} between D_{src} and D'_{src} is smaller than a small constant value ϵ . Then, assuming a set of neighboring N source images, we compute the number of inliers which fulfills $\delta_{depth} < \epsilon$ for each source image, and keep depth pixels if the number of inliers is greater than a certain threshold. Finally, we apply both single-view and multi-view filtering to remove all depth outliers.

Pose Refinement We consider camera poses obtained from ARKit as initial poses and exploit matching correspondences to perform incremental triangulation [46]. After combining filtered depth points from LiDAR and 3D points from incremental triangulation, we propose a depth factor and run bundle adjustment with the depth factor and reprojection error. Let P_i denote the camera parameter of the i -th camera and X_j denote point parameters seen by the i -th camera. We minimize the proposed joint objective function E , which is formulated as follows:

$$E = \sum_i \sum_j \rho_i(\|\phi(P_i, X_j) - x_{ij}\|^2) + \frac{1}{\eta} |D_i - z_i|, \quad (1)$$

where Φ is the function that projects points to the image plane, and ρ denotes the Cauchy function as the robust loss function. x_{ij} is the feature point corresponding to the 3D point X_j seen by the i -th camera. z_i is the depth value of points projected from the 3D point X_j and D_i is the filtered depth of pixels corresponding to z_i . We define η as sensor measurement noise and use it to down-weight noisy depth. We notice that the depth factor has two benefits. First, additional information from the depth sensor is a useful con-

straint for accurate pose estimation and reliable triangulation. Furthermore, this depth factor guides the scale of the SfM to follow the metric scale of the depth sensor.

3.2. Geometry Reconstruction

Neural Geometry Representations Instead of classical 3D reconstruction, we adopt NeuS [55], which combines the advantages of both volume and surface rendering. In other words, it can extract accurate surfaces even without pixel-level masks. Our scanned object is represented by two Multi-layer Perceptrons (MLPs): geometry network f_{sdf} , which maps a 3D point $x \in \mathbb{R}^3$ to a signed distance, and color network f_{color} which converts a 3D point x and a viewing direction $v \in \mathbb{R}^3$ to a color c . The 3D geometry of an object is represented implicitly as an SDF with zero-level set $\{x \in \mathbb{R}^3 | f_{sdf}(x) = 0\}$. These two parameters of geometry and color are learned by the volume rendering technique. For a ray $\{x(t) = o + tv | t \geq 0\}$ with camera center o and viewing direction d , the color is rendered with n sampled points along the ray as follows:

$$\hat{C} = \sum_{i=1}^n T_i \alpha_i c(x(t), v), \quad (2)$$

where $T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$ is the accumulated transmittance, and α_i is the discrete opacity. Here, $\alpha_i = 1 - \exp(-\int_{t_i}^{t_{i+1}} \rho(t) dt)$, and $\rho(t)$ is opaque density function borrowed from the definition in NeuS [55].

Training Process RGBD-aided SfM already estimates both poses and sparse depths by matching feature correspondences. Given this prior information, we leverage an MVS algorithm [62] to estimate the depth and normal for all images with less cost. It is worth noting that we avoid using learning-based MVS algorithms because they are not robust to diverse real-world scenes and are hard to evaluate reliability. In this section, we modify the training process of NeuS to avoid losing fine details of complex shape. Our training process is divided into two stages. In the first stage, we train a neural radiance field with a prior depth and normal obtained from the MVS [62] algorithm to supervise the training process. Specifically, we sample K pixels p_k and their corresponding color values $I(p_k)$ in each iteration. Here, we propose a regularization method that relies on prior depth D and normal N from MVS algorithm. We unproject depth under the given pose to obtain 3D point $X(p_k)$. Then, we use this 3D point and its corresponding normal $N(p_k)$ from MVS to regularize the geometry network by minimizing

$$L_{reg} = \sum_k w_d |f_{sdf}(X(p_k))| + w_n (1 - \langle \nabla_p f_{sdf}(X(p_k)), N(p_k) \rangle), \quad (3)$$

where w_d and w_n are weights for each term. This regular-

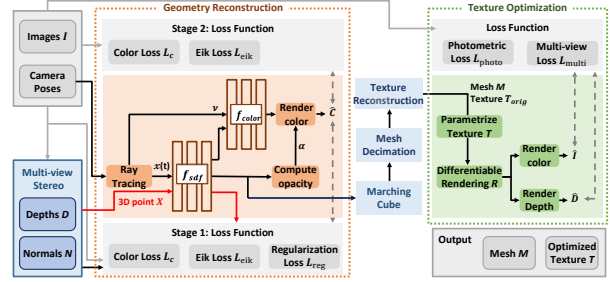


Figure 4. Overview of Geometry Reconstruction (Section 3.2) and Texture Optimization (Section 3.3). Our goal in geometry reconstruction is to train geometry network f_{sdf} and color network f_{color} via volumetric rendering with two training steps. In step 1, we leverage prior geometric information from the MVS algorithm [62]. Then, our method utilizes a sparse voxel octree to efficiently sample points near the surface. In texture optimization, we first utilize a classical texture reconstruction to generate texture maps for a given mesh. Then, our method applies a differentiable rendering to fine-tune texture maps.

ization will guide the SDF to learn explicit supervision from the MVS algorithm. Although this regularization is helpful for learning prior geometry, it is susceptible to the noisy results derived from the MVS algorithm which tends to lose fine details and texture-less regions. Thus, we will only use this regularization in the first stage. The color of each pixel can be computed by E.q. 4. The color loss L_c is defined as

$$L_c = \sum_k |\hat{C}(p_k) - I(p_k)|. \quad (4)$$

The Eikonal loss [18] to regularize the SDF is formulated as

$$L_{eik} = \sum_{k,i} (\|\nabla f_{sdf}(p_{k,i})\|_2 - 1)^2. \quad (5)$$

The overall loss function L in the first stage is defined as

$$L = L_c + L_{eik} + L_{reg}. \quad (6)$$

In the second stage, inspired by [50], we adopt a sparse voxel octree to guide the sampling process and capture fine details. After the first stage, we can extract a triangular mesh from the SDF via the Marching Cube [35] and define a sparse voxel volume based on this mesh. Then, based on the sparse voxel volume, we can avoid redundant point samples outside the sparse voxel and focus on point samples near the surface samples. Here, our total loss is $L = L_c + L_{eik}$, which is the same as E.q. 6 except for L_{reg} . To capture fine details, we will not use L_{reg} in E.q. 6.

Mesh Simplification A highly detailed 3D mesh from the SDF network with marching cube algorithms [35] leads to enormous memory requirements. To obtain a lightweight mesh, we explore popular quadric mesh simplification [15]. Although the mesh simplification generated small holes, the lightweight mesh enables us to reduce the computation

complexity of the texture mapping process. We set the decimation ratio to 0.4%.

3.3. Texture Optimization

Classical Texture Reconstruction For a triangular mesh M provided by the previous section, we first utilize Waechter et al. [53] with poses and images across different views to construct a texture map T_{orig} . Compared to the color network from Section 3.2, this classical method [53] can acquire visually realistic texture images. However, it cannot avoid texture bleeding on the boundary of different views and blurring artifacts.

Texture Fine-tuning To tackle blurring and texture bleeding artifacts, we propose a fine-tuning step by optimizing the texture map. Here, we parameterize a texture map T obtained from classical texture reconstruction. Given a 3D Mesh M with the 2D texture map T and multi-view images I_i with the corresponding camera poses P_i , we leverage a differentiable renderer R to generate an image \hat{I}_i and depth \hat{D}_i , i.e. $\hat{I}_i, \hat{D}_i = R(M, P_i)$. A differentiable renderer R [31] performs mesh rasterization and then texture sampling to render an image \hat{I}_i . In other words, we project texture onto the image plane and compare it to the corresponding view of the image. Furthermore, we compare the rendered image with a different view image by minimizing multi-view photometric loss. We use the combination of the L1 and SSIM [56] as the photometric loss L_{photo} in image space between the rendered image \hat{I}_i and the reference image I_i as follows:

$$L_{photo} = \sum_i (1 - \alpha) \|\hat{I}_i - I_i\| + \frac{\alpha}{2} (1 - SSIM(\hat{I}_i, I_i)), \quad (7)$$

where α is set to 0.85. Additionally, the differentiable render estimates depth \hat{D}_i by interpolating z coordinates of each vertex. With the estimated depth \hat{D}_i and intrinsic parameter K , we can generate a synthesized frame I'_i by reprojecting adjacent frames $I_{i'}$ to the current frame I_i with the camera intrinsic matrix K , the estimated depth \hat{D}_i , and the relative pose $P_{i \rightarrow i'}$. The multi-view photometric loss L_{multi} similar to Eq. 7 is formulated as:

$$I'_i(p) = I_{i'} \langle \pi(K P_{i \rightarrow i'} \hat{D}_i(p) K^{-1} \tilde{p}) \rangle, \\ L_{multi} = (1 - \alpha) \|I'_i - I_i\| + \frac{\alpha}{2} (1 - SSIM(I'_i, I_i)), \quad (8)$$

where \tilde{p} is the homogeneous coordinate of p , π means projection from homogeneous to image coordinates, and $\langle \cdot \rangle$ indicates the bilinear sampling function. We define our total loss L to be

$$L = L_{photo} + L_{multi}. \quad (9)$$

Finally, the differentiable rendering enables the loss gradients to back-propagate to update the textures parameters.

Since our method does not require ground truth masks, we avoid considering optimization for the geometry. We observe that joint optimization with geometry and texture fails to converge without the ground truth mask.

4. Experiments

4.1. Experiment Settings

Data Collection: Our smartphone application, running on an iPhone 13 Pro using ARKit 6, can record various sensor outputs: synchronized images, poses, confidence maps, dense depth maps from the LiDAR scanner, and IMU sensor information. We built two types of datasets: ARKit-video and AR-capture. ARKit-video collects long video sequences similar to datasets constructed from RGBD-SLAM [70]. The second type is collected by our smartphone application based on Object Capture¹. AR-capture is a multi-view dataset of objects consisting of high-quality 4K images with depth and gravity data. The ARKit-video collects thousands of frames sampled from the video with resolution of 1280x720. In contrast, AR-capture mostly contains less than fifty images at 4032x3096 pixels. We collect 11 objects: *robot arm 1*, *robot arm 2*, *plant*, *tree*, *sofa*, *bike*, *recliner chair*, *cafe stand*, *delivery robot*, *office chair*, and *camera stand*. We randomly select 70 % as the training images and utilize the 30 % as test images for evaluating the novel view synthesis.

DTU Dataset: The DTU dataset [24] is a MVS dataset that provides reference point clouds acquired with laser sensor in the controlled lab. Since AR-capture and ARKit-video do not include 3D point clouds acquired with specialized hardware, we use the DTU dataset to evaluate the performance of geometry reconstruction. We follow the evaluation of NeuS [55] and exploit the chamfer distance as metric.

Implementation Details: We build the RGBD-aided Structure from Motion based on COLMAP [46], an existing SfM tool. Instead of SIFT [36], our system uses learning-based features [11, 44, 45] for feature extraction and matching to avoid failure in a featureless scene. ARKit offers an interface for accessing LiDAR data as a depth image. Following the work [55], the SDF network f_{sdf} is implemented as an 8-layer MLP a hidden dimension of 256, and geometric initialization [3] for the network parameters. We construct the color network f_{color} as MLPs with 4-layer MLP with hidden dimension of 256. We optimize networks in geometry reconstruction with Adam [28] and an initial learning rate of 5e-4. We train the first step for 60K iterations and the second step for 140K iterations. In terms of objects with complex shape, it takes 240K iterations for second step. In Texture Optimization, we utilize the public github code for classical method [53]. Then, for texture fine-tuning, the Adam optimizer is used with the learning rate for the tex-

¹<https://developer.apple.com/videos/play/wwdc2021/10076/>

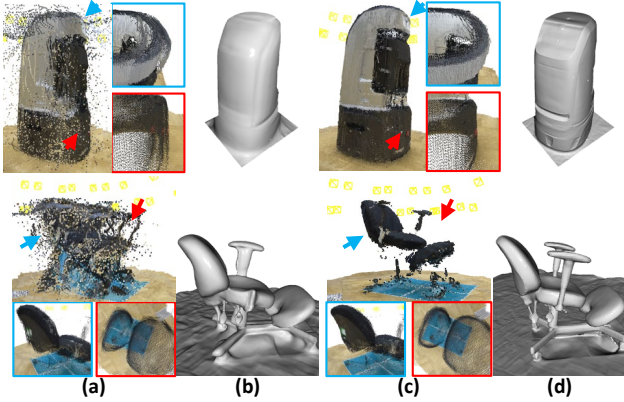


Figure 5. We show the benefits of RGBD-aided Structure from Motion. (a) illustrates the 3D point clouds obtained from initial depths with initial poses. In (c), we accumulate the 3D point clouds (by using filtered depths and refined poses) from RGBD-aided SfM. For a detailed description, we visualize the zoomed-in blue and red regions using filtered depth. Initial depth maps and poses from ARKit are input to the geometry reconstruction in Section 3.2 and its results are shown in (b). The results of our pipeline in (d) lead to higher quality reconstructions than (b) which does not apply RGBD-aided SfM.

ture parameter. This step takes 3000 epochs to fine-tune the texture parameter.

4.2. Experimental Results

In Fig. 5, we visualize the effectiveness of RGBD-aided Structure from Motion in Section 3.1 on our dataset. Figure 5 (a) represents the 3D point clouds by initial depth maps and poses obtained from ARKit. Unfortunately, initial sensor data is too noisy to perform an accurate 3D reconstruction. After multi-view depth filtering as preprocessing, zoomed-in regions in (a) show the point clouds by reprojecting filtered depth maps with initial poses. Compared to initial data, multi-view depth filtering can handle noisy depth measurements. Figure 5 (c) provides the results of preprocessing and pose refinement. Pose refinement enables us to globally align 3D reconstruction without noticeable camera drift and with better local quality. For qualitative comparison, we take as input initial sensor data from ARKit and perform geometry reconstruction in Section 3.2 in Fig. 5 (b). Compared to our experimental results in Fig. 5 (d), it is difficult to achieve comparable quality and better reconstruction. Consequently, our proposed RGBD-aided SfM significantly improves geometry reconstruction performance.

Figure 6 qualitatively compares our method to VolSDF [63], TSDF-fusion [23], and ACMP [62]. We apply screened Poisson surface reconstruction [27] to generate a mesh from point clouds of ACMP. Among all four algorithms, we apply the same pose estimation algorithm from Section 3.1 for fair comparison. TSDF-fusion, which requires sufficient scene overlap, exploits more frames than our method and ACMP. Due to the low-textured regions,



Figure 6. Reconstruction of our method, VolSDF [63], TSDF-Fusion [23], and a mesh from point clouds of ACMP [62] processed with screened Poisson surface reconstruction [27]. Note that the neural surface reconstruction is more perceptually convincing in the ARKit-video dataset.

the surface of ACMP is very noisy. TSDF-fusion utilizes depth maps obtained from low-resolution LiDAR. Here, since this low-resolution depth sensor is very noisy, we only use filtered depth maps after preprocessing in Section 3.1. However, this model still cannot reconstruct either thin or challenging structures accurately (e.g., a chair armrest). Both implicit neural surface reconstruction approaches show comparable performance, and both are superior to classical 3D reconstruction. We observe that the difference between neural geometric representations (Ours and VolSDF) is negligible unlike pose quality. It is worth noting that using accurate poses as input significantly improves the 3D reconstruction quality as shown in Fig. 5.

For qualitative comparison of textured meshes, we compare our proposed method with four different methods: 1) ACMP [62] is a state-of-the-art multi-view stereo algorithm; 2) Color Map Optimization (CMO) [69] is a texture mapping method for RGB-D scanning; 3) Waechter et al. [53] is a large-scale texture reconstruction based on global color adjustment; 4) NeuS [55] is the recent neural surface reconstruction based on volumetric rendering. We reconstruct a 3D mesh from the point cloud of ACMP with Screened Poisson Surface reconstruction [27]. In the case of CMO, since explicit depth maps from low-resolution LiDAR are too noisy, we utilize depth maps rendered by a triangular mesh from our geometry reconstruction step for a fair comparison. With these rendered depth maps and corresponding images, we run a color map optimization for mapping color images onto our reconstructed mesh. To train



Figure 7. Qualitative comparison between (a) Ours, (b) Waechter et al. [53], (c) CMO [69], (d) ACMP [62], and (e) NeuS [55]. Compared to textured mesh (3-6 columns) obtained from different algorithms, the result of our method is perceptually closer to the original scene. Additional qualitative results are added in the supplementary material due to the limited space.

NeuS, we follow a training process similar to that in Sec. 3.2. We then extract a mesh from the SDF network in NeuS and apply the Blender Smart UV Project tool to obtain a per-vertex UV mapping. To fill a texture for this mesh, points densely sampled from triangle mesh with interpolated UV coordinates are fed to the color network, and eventually we can obtain texture maps for the given mesh.

In Fig. 7, we show the reconstructed texture for novel views. The visual results of our approach are visually sharper and perceptually closer to the ground truth scene. Results of NeuS and CMO exhibit blurring artifacts and

color misalignment. The reconstructed mesh from ACMP is very noisy due to incorrect geometry. Here, Waechter et al. (Fig. 7 (b)) is the result of classical texture reconstruction in Section 3.3. This method is more photorealistic and sharper than NeuS, CMO, and ACMP. Then, after texture fine-tuning, our method avoids the seams and texture misalignment often seen with Waechter et al. Table 1 reports the quantitative results on two datasets. Our method outperforms other texture optimization methods on all evaluation metrics. It indicates that our method satisfies both the high fidelity of the texture and its conformity to the ground truth.

| Category | Data | NeuS [55] | | | CMO [69] | | | Waechter et al. [53] | | | Ours | | |
|----------------|------|-----------------|-----------------|--------------------|-----------------|-----------------|--------------------|----------------------|-----------------|--------------------|-----------------|-----------------|--------------------|
| | | PSNR \uparrow | SSIM \uparrow | LPIPS \downarrow | PSNR \uparrow | SSIM \uparrow | LPIPS \downarrow | PSNR \uparrow | SSIM \uparrow | LPIPS \downarrow | PSNR \uparrow | SSIM \uparrow | LPIPS \downarrow |
| robot arm 1 | Ac | 20.292 | 0.889 | 0.083 | 18.473 | 0.891 | 0.083 | 19.909 | 0.872 | 0.060 | 20.049 | 0.871 | 0.057 |
| robot arm 2 | Ac | 20.741 | 0.882 | 0.086 | 21.107 | 0.883 | 0.092 | 20.190 | 0.862 | 0.067 | 20.652 | 0.858 | 0.062 |
| plant | Ac | 16.748 | 0.642 | 0.295 | 18.462 | 0.627 | 0.316 | 17.921 | 0.635 | 0.186 | 19.970 | 0.691 | 0.146 |
| tree | Ac | 18.409 | 0.869 | 0.117 | 18.448 | 0.870 | 0.127 | 17.518 | 0.858 | 0.096 | 18.331 | 0.858 | 0.102 |
| sofa | Ac | 19.910 | 0.872 | 0.139 | 21.170 | 0.840 | 0.136 | 19.759 | 0.829 | 0.113 | 21.608 | 0.894 | 0.109 |
| bike | Ac | 20.971 | 0.778 | 0.183 | 21.311 | 0.778 | 0.206 | 19.822 | 0.740 | 0.088 | 19.943 | 0.737 | 0.078 |
| Mean | Ac | 19.512 | 0.822 | 0.151 | 19.825 | 0.815 | 0.160 | 19.186 | 0.799 | 0.101 | 20.095 | 0.818 | 0.092 |
| recliner chair | Av | 24.920 | 0.893 | 0.099 | 24.761 | 0.876 | 0.112 | 23.692 | 0.864 | 0.054 | 25.045 | 0.896 | 0.056 |
| cafe stand | Av | 20.498 | 0.779 | 0.173 | 22.934 | 0.828 | 0.154 | 17.915 | 0.798 | 0.112 | 22.634 | 0.818 | 0.095 |
| delivery robot | Av | 22.412 | 0.897 | 0.126 | 23.549 | 0.908 | 0.128 | 25.246 | 0.929 | 0.073 | 23.451 | 0.906 | 0.086 |
| office chair | Av | 20.722 | 0.892 | 0.096 | 21.120 | 0.899 | 0.103 | 20.315 | 0.865 | 0.094 | 20.982 | 0.892 | 0.083 |
| camera stand | Av | 21.901 | 0.914 | 0.069 | 21.886 | 0.907 | 0.074 | 21.121 | 0.899 | 0.059 | 22.208 | 0.919 | 0.060 |
| Mean | Av | 22.092 | 0.875 | 0.113 | 22.850 | 0.884 | 0.115 | 21.658 | 0.871 | 0.079 | 22.868 | 0.886 | 0.076 |

Table 1. Quantitative comparison on our dataset. Ac and Av denote the AR-capture dataset and ARKit-video dataset, respectively. We improve the LPIPS metric by 8.9% in AR-capture and by 3.8% in ARKit-video.

| Method | scan24 | scan37 | scan40 | scan55 | scan63 | scan65 | scan69 | scan83 | scan97 | scan105 | scan106 | scan110 | scan114 | scan118 | scan122 | Mean |
|-------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| COLMAP [47] with trim=7 | 0.45 | 0.91 | 0.37 | 0.37 | 0.90 | 1.00 | 0.54 | 1.22 | 1.08 | 0.64 | 0.48 | 0.59 | 0.32 | 0.43 | 0.45 | 0.65 |
| UNISURF [43] | 1.32 | 1.36 | 1.72 | 0.44 | 1.35 | 0.79 | 0.80 | 1.49 | 1.37 | 0.89 | 0.59 | 1.47 | 0.46 | 0.59 | 0.62 | 1.02 |
| NeuS [55] | 1.37 | 1.21 | 0.73 | 0.40 | 1.20 | 0.70 | 0.72 | 1.01 | 1.16 | 0.82 | 0.66 | 1.69 | 0.39 | 0.61 | 0.51 | 0.87 |
| VoISDF [63] | 1.14 | 1.26 | 0.81 | 0.49 | 1.25 | 0.70 | 0.72 | 1.29 | 1.18 | 0.70 | 0.66 | 1.08 | 0.42 | 0.61 | 0.55 | 0.86 |
| Ours without 2nd step | 1.04 | 1.61 | 0.67 | 0.64 | 0.88 | 0.70 | 0.62 | 1.14 | 1.10 | 0.89 | 0.66 | 1.02 | 0.50 | 0.58 | 0.51 | 0.84 |
| Ours | 0.96 | 1.37 | 0.67 | 0.60 | 0.88 | 0.70 | 0.45 | 1.14 | 1.10 | 0.84 | 0.66 | 1.11 | 0.50 | 0.58 | 0.50 | 0.80 |

Table 2. Quantitative comparison on DTU [24]. Chamfer distance is employed as the evaluation metric. COLMAP is superior to other neural geometry reconstructions. Our training process can improve the NeuS by 8 %.

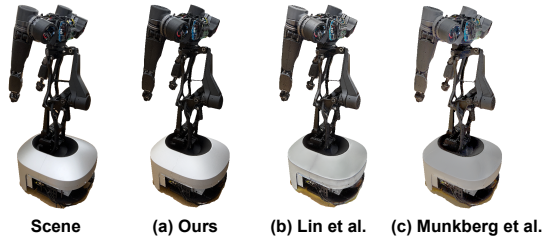


Figure 8. Qualitative comparisons of texture optimization algorithms based on differentiable rendering: (a) our method, (b) Lin et al. [33], and (c) Munkberg et al. [40].

In Fig. 8, we show the results of other texture optimization methods [33, 40] based on differentiable rendering [31]. Both methods aim to learn geometry, textures, and lighting parameters. However, they heavily rely on ground truth masks which require additional annotation costs. Without ground truth masks, we note that joint optimization over geometry and texture is unstable and often has negative effects on geometry. For a fair comparison, we therefore provide the fixed mesh topology and train only texture and lighting parameters through differentiable rendering [31]. Since we capture objects in the wild under unknown environmental lighting conditions, it is difficult to disentangle materials and lighting via differentiable rendering. Neither method [33, 40] is able to generate realistic texture maps compared to either our method or classical texture reconstruction approaches.

Experimental Results on DTU dataset: Unlike our dataset, DTU [24] is captured by specialized hardware in the controlled lab environment. Thus, this dataset provides the ground truth poses and the RGBD-aided SfM is not necessary. As shown in Table 2, the surfaces obtained by

COLMAP [47] with trimming value 7 achieve high-quality reconstruction. Our method exploits MVS [47] to super-vise the training NeuS and outperforms NeuS. For the ablation study, when we only train our network with the first step (ours without 2nd step), its result is not better than our method in some scans such as scan37. In our experiment, sparse voxel octree sampling does not have a significant effect on geometry performance. Since our training process can be applied to any other 3D neural surface reconstruction methods. In the future, we will apply this training process to other recent 3D neural surface reconstruction algorithms [10, 63, 66], which shows better results on DTU.

5. Conclusion, Limitations and Future Work

In this paper, we have developed a practical framework to reconstruct high-quality textured meshes of 3D objects by using a smartphone. Given initial poses from ARKit, multi-view images, and low-resolution depth maps, our approach can refine initial poses and depth maps from ARKit. We adopt neural geometry representation [55] for surface reconstruction and improve its performance by modifying the training process. Finally, a differentiable rendering is designed for fine-tuning the texture map acquired from the classical texture mapping method. In the future, our goal is to reduce the training time which can be comparable with classical methods. Unfortunately, most neural geometry reconstruction [10, 55, 63, 66] requires a long training time. We would like to reduce the training time in our framework for practical purposes.

Acknowledgements This research was partly supported by Army Cooperative Agreement W911NF2120076.

References

- [1] Apple arkit. <https://developer.apple.com/documentation/arkit/>. Accessed: 2022-11-09. **1**
- [2] Google arcort. <https://developers.google.com/ar>. Accessed: 2022-11-09. **1**
- [3] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. **5**
- [4] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6290–6301, 2022. **2**
- [5] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Yuri Feigin, Peter Fu, Thomas Gebauer, Daniel Kurz, Tal Dimry, Brandon Joffe, Arik Schwartz, and Elad Shulman. ARKitScenes: A diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. **1**
- [6] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph.*, 36(4):106–1, 2017. **3**
- [7] Chong Bao and Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision (ECCV)*, 2022. **3**
- [8] 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. **2**
- [9] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017. **1, 2**
- [10] François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces geometry with patch warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6260–6269, 2022. **2, 8**
- [11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Deep Learning for Visual SLAM Workshop*, 2018. **5**
- [12] Alex Flint, Oleg Naroditsky, Christopher P Broaddus, Andriy Grygorenko, Stergios Roumeliotis, and Oriol Bergig. Visual-based inertial navigation, Dec. 11 2018. US Patent 10,152,795. **2**
- [13] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. **1, 2**
- [14] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. June 2015. **2**
- [15] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. **4**
- [16] Shubham Goel, Georgia Gkioxari, and Jitendra Malik. Differentiable stereopsis: Meshes from multiple views using differentiable rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8635–8644, June 2022. **3**
- [17] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and viewpoint without keypoints. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV*, page 88–104, Berlin, Heidelberg, 2020. Springer-Verlag. **3**
- [18] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of the 37th International Conference on Machine Learning, ICML’20*. JMLR.org, 2020. **4**
- [19] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3d scene reconstruction with the manhattan-world assumption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5511–5520, 2022. **2**
- [20] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. **2**
- [21] Paul Henderson, Vagia Tsiminaki, and Christoph H Lampert. Leveraging 2d data to learn textured 3d mesh generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7498–7507, 2020. **3**
- [22] Jingwei Huang, Justus Thies, Angela Dai, Abhijit Kundu, Chiyu Jiang, Leonidas J Guibas, Matthias Nießner, Thomas Funkhouser, et al. Adversarial texture optimization from rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1559–1568, 2020. **3**
- [23] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. **1, 2, 6**
- [24] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. **5, 8**
- [25] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. **3**
- [26] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. **2**
- [27] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. **1, 2, 6**

- [28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. [5](#)
- [29] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. [3](#)
- [30] Patrick Labatut, J-P Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. In *Computer graphics forum*, volume 28, pages 2275–2290. Wiley Online Library, 2009. [2](#)
- [31] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. [2](#), [3](#), [5](#), [8](#)
- [32] Victor Lempitsky and Denis Ivanov. Seamless mosaicing of image-based texture maps. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–6. IEEE, 2007. [2](#)
- [33] Lixiang Lin, Jianke Zhu, and Yisu Zhang. Multiview textured mesh recovery by differentiable rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2022. [3](#), [8](#)
- [34] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. *arXiv preprint arXiv:2206.05737*, 2022. [2](#)
- [35] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. [3](#), [4](#)
- [36] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. [2](#), [5](#)
- [37] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. [2](#)
- [38] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. [2](#)
- [39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multi-resolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. [2](#)
- [40] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. *arXiv:2111.12503*, 2021. [3](#), [8](#)
- [41] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. [2](#)
- [42] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013. [1](#), [2](#)
- [43] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021. [2](#), [8](#)
- [44] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: repeatable and reliable detector and descriptor. *arXiv preprint arXiv:1906.06195*, 2019. [5](#)
- [45] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. [5](#)
- [46] Johannes L Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. [2](#), [3](#), [5](#)
- [47] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016. [1](#), [2](#), [8](#)
- [48] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. [2](#)
- [49] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International journal of computer vision*, 80(2):189–210, 2008. [2](#)
- [50] Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural 3d reconstruction in the wild. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. [2](#), [4](#)
- [51] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), jul 2019. [3](#)
- [52] George Vogiatzis, Philip HS Torr, and Roberto Cipolla. Multi-view stereo via volumetric graph-cuts. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 391–398. IEEE, 2005. [2](#)
- [53] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *European conference on computer vision*, pages 836–850. Springer, 2014. [1](#), [2](#), [3](#), [5](#), [6](#), [7](#), [8](#)
- [54] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. Neuris: Neural reconstruction of indoor scenes using normal priors. *arXiv preprint arXiv:2206.13597*, 2022. [2](#)
- [55] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. [2](#), [4](#), [5](#), [6](#), [7](#), [8](#)

- [56] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. [5](#)
- [57] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. *Robotics: Science and Systems*, 2015. [1](#), [2](#)
- [58] Aji Resindra Widya, Yusuke Monno, Masatoshi Okutomi, Sho Suzuki, Takuji Gotoda, and Kenji Miki. Stomach 3d reconstruction based on virtual chromoendoscopic image generation. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1848–1852. IEEE, 2020. [2](#)
- [59] Changchang Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013. [2](#)
- [60] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7119–7128, June 2021. [3](#)
- [61] Qingshan Xu, Weihang Kong, Wenbing Tao, and Marc Pollefeys. Multi-scale geometric consistency guided and planar prior assisted multi-view stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. [2](#)
- [62] Qingshan Xu and Wenbing Tao. Planar prior assisted patch-match multi-view stereo. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12516–12523, 2020. [1](#), [2](#), [4](#), [6](#), [7](#)
- [63] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. [2](#), [6](#), [8](#)
- [64] 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*, 33:2492–2502, 2020. [2](#)
- [65] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenotrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. [2](#)
- [66] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *arXiv preprint arXiv:2206.00665*, 2022. [8](#)
- [67] Jingyang Zhang, Yao Yao, and Long Quan. Learning signed distance field for multi-view surface reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6525–6534, 2021. [2](#)
- [68] Kai Zhang, Gernot Riegler, Noah Snaveley, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. [2](#)
- [69] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (ToG)*, 33(4):1–10, 2014. [1](#), [2](#), [3](#), [6](#), [7](#), [8](#)
- [70] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3d reconstruction with rgb-d cameras. In *Computer graphics forum*, volume 37, pages 625–652. Wiley Online Library, 2018. [5](#)