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

Jaehoon Choi [1,2] Dongki Jung [1] Taejae Lee [1] Sangwook Kim [1] Youngdong Jung [1]
Dinesh Manocha [2] Donghwan Lee [1]
[1]NAVER LABS [2]University of Maryland

1. Data Collection

We provide specific properties of our data collection process. As mentioned in the manuscript, ARKit-video collects long video sequences to apply the RGBD SLAM methods [1] to extract poses and fuse a 3D model. However, we fail to apply the state-of-the-art RGBD SLAM algorithm [1] for accurate pose estimation due to noisy depth measurements. Instead, we utilize accurate poses from our RGBD-aided Structure from Motion (SfM) for geometry reconstruction and texture optimization. ARKit-video includes *recliner chair*, *cafe stand*, *delivery robot*, *office chair*, and *camera stand*. We can apply TSDF-fusion [3] with poses from RGBD-adied SfM to reconstruct geometry in Fig. 6. in the manuscript because of the video sequence. AR-capture includes *robot arm 1*, *robot arm 2*, *plant*, *tree*, *sofa*, *bike*, *recliner chair*, *cafe stand*, *delivery robot*, *office chair*, and *camera stand*. Except for *sofa* and *tree*, AR-capture contains less than fifty images. The *sofa* contains 108 images because the mobilephone camera has difficulty capturing this sofa due to the large size (This sofa can seat up to 15 people). We require a number of images to capture the whole shape of *sofa*. The *tree* contains 111 images and is also large size object. In addition to this, the *tree* has a very complex shape (e.g. leaf) and thin structure (e.g. stem).

2. Implementation Details

In preprocessing stage, we apply both single-view and multi-view filtering two times. We regard 10 neighboring frames as source images and set the threshold ϵ to 0.1. We keep depth pixels if the ratio of inliers is greater than 80%. We remove depth points larger than 2.55m. Our RGBD-aided Structure from Motion (SfM) is based on COLMAP [5] implemented in C++. Similar to the existing COLMAP, we use Ceres Solver to perform bundle adjustment. After SfM, we apply MVS algorithm [6, 9] to provide the depth and normal for geometry reconstruction. Here, we also leverage the filtering algorithm to remove the noisy values for the depth and normal maps obtained from the MVS algorithm. For the first stage of trainning process in Sec. 3.2, we set w_d to 1e-5 and w_n to 1e-3. After the first stage, we build sparse voxel volume around surfaces obtained from mesh. We set a depth level of octree to 8 for all classes. During the training for two MLPs (the SDF network and color network), we also model the background by NeRF++ [12]. The sampling strategy is also similar to the NeuS [8]. For geometry reconstruction and texture optimization, we use 41 - 108 images at 1280 x 720 pixels sampled from video sequence of ARKit-video. Since AR-capture has large images at 4032x3096 resolutions, we resize their resolution to 1008x774 resolution for geometry reconstruction and texture optimization. The LiDAR sensor in the iOS device provides depth maps and confidence maps with size 256 x 192. For mesh simplification, we exploit the quadric mesh simplification algorithm provided by [14]. The learning rates for texture optimization start from 1e-3 and decay the learning rate by 0.1 once the epoch reaches 500 and 2000 respectively. As mentioned in manuscript, we use the public github code for the classical MVS [6, 9] and texture mapping [7] which are implemented in C++. We implement all geometry reconstruction and texture optimization in PyTorch [4] and conduct all experiments on V100 GPU.

3. Additional Qualitative Comparisons

In the manuscript, we only show the reconstructed texture of *delivery robot*, *recliner chair*, *robot arm 1*, and *office chair* (refer the Fig. 7 in the manuscript) due to the limited space. Here, we visualize our results of other objects: *cafe stand*, *sofa*, *plant*, *bike*, *robot arm2*, *tree*, and *camera stand*. We compare our method with four different methods following the

manuscript. Compared to these methods, our visual results are visually sharper and perceptually closer to the original scene. Furthermore, we apply our method to 3D indoor scene and show the feasibility of reconstructing the textured mesh for 3D indoor scenes.

Figure 1. Qualitative comparison between (a) Ours, (b) Waechter et al. [7], (c) CMO [13], (d) ACMP [9], and (e) NeuS [8]. From top to bottom, we show the results of *cafe stand*, *sofa*, *plant*, and *bike*. We collect *bike* at the outdoor environment in uncontrolled settings. The *sofa* has the large size (it can seat up to 15 people). Our method can perform 3D reconstruction and texture mapping for the large object and the object under the outdoor environment.

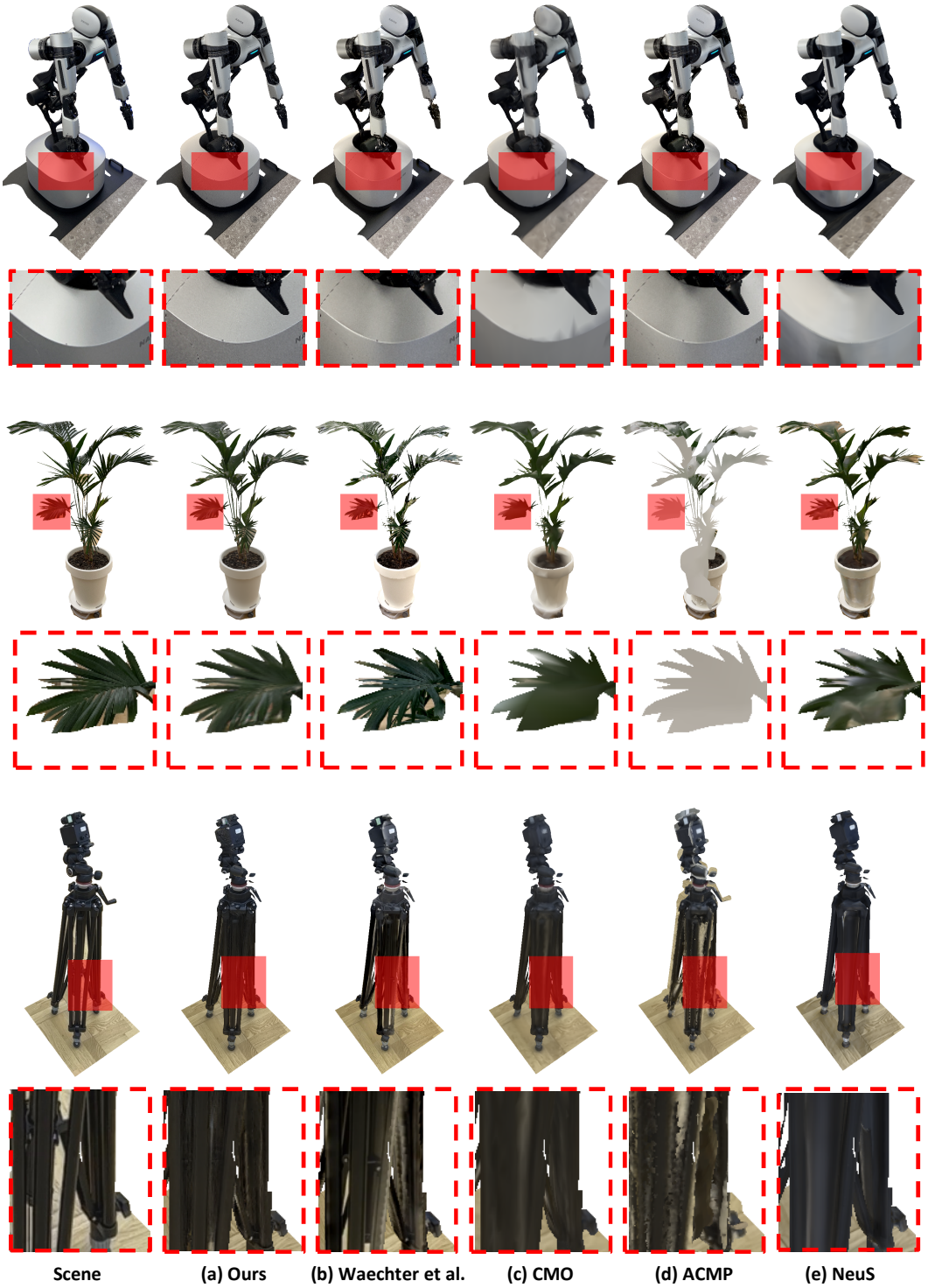

Figure 2. Qualitative comparison between (a) Ours, (b) Waechter et al. [7], (c) CMO [13], (d) ACMP [9], and (e) NeuS [8]. From top to bottom, we show the results of *robot arm2*, *tree*, and *camera stand*. The *tree* has a very complex shape (e.g. leaf) and thin structure (e.g. stem). Also, the *camera stand* has a thin structure. Our method can perform 3D reconstruction and texture mapping for objects with complex shapes and thin structures.

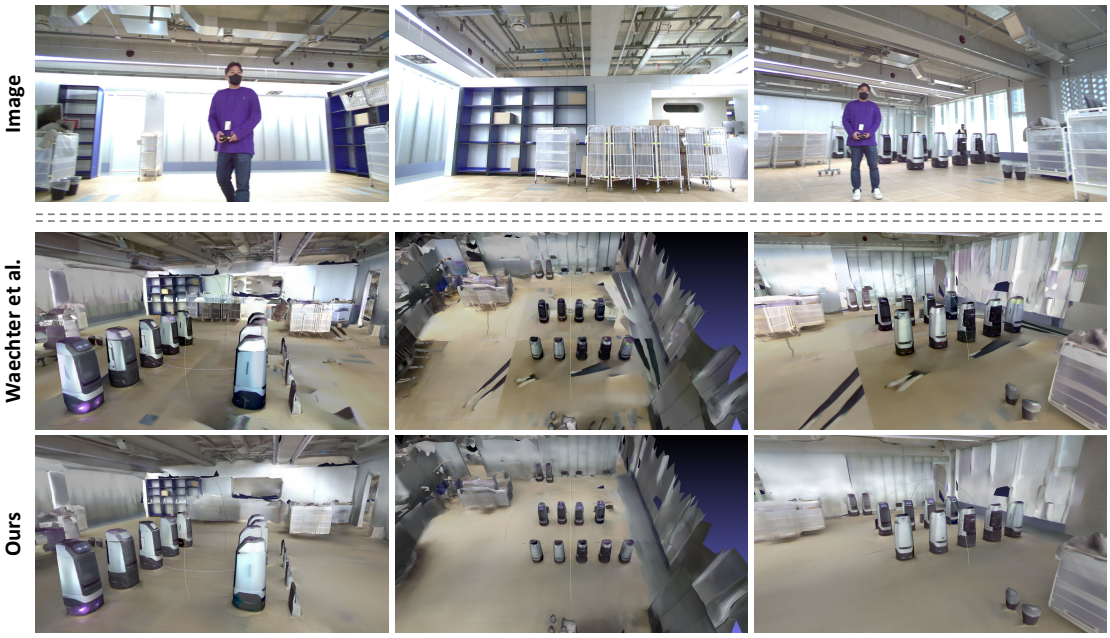

Figure 3. Example reconstruction results of 3D indoor scenes. Although our manuscript focuses on objects, we apply our method to reconstruct 3D indoor scenes from 2D images. We use 507 images for training. Our method can generate quite complete and smooth reconstruction results. In particular, we show the textured mesh reconstructed by classical texture reconstruction [7] (2nd row) and texture fine-tuning (3rd row) in Section 3.3. Our texture optimization method (3rd row) can remove ghosting effects and improve texture misalignment (2nd row) in indoor scenes. For future research, we will be able to extend our method to perform geometry reconstruction and texture optimization for 3D indoor scenes.

4. Qualitative Results

In Fig. 4, we show the reconstructed mesh for objects (*robot arm 1*, *robot arm2*, *bike*, *sofa*, *camera stand*, *plant*, and *tree*) that are not included in the manuscript due to the limited space. Recently, MonoSDF [11] shows the state-of-the-art performance for the neural implicit surface reconstruction. We avoid using learning-based algorithms because they are expensive to acquire new training datasets and time-consuming to train other neural networks. In contrast, MonoSDF exploits the power of pretrained Omnidata model [2] which is trained by the large and diverse multi-task datasets. This pre-trained model can provide a surface normal and a depth map for a single image. While this pretrained model [2] requires large amounts of data with expensive annotation costs and huge computational costs for training, MonoSDF succeeds to apply this model to boost its performance for surface reconstruction. However, we observe both the advantages and limitations of leveraging the pretrained model in our experiments. Figure 5 visualizes the reconstructed mesh of MonoSDF including our method and VolSDF [10]. Compared to ours and VolSDF, MonoSDF shows advances in reconstructing low-textured areas such as a back rest of the office chair and a paper box in the caffe stand. However, since its performance heavily relies on the quality of the pretrained model, it fails to reconstruct the accurate 3D geometry of plant and tree in Fig. 4.

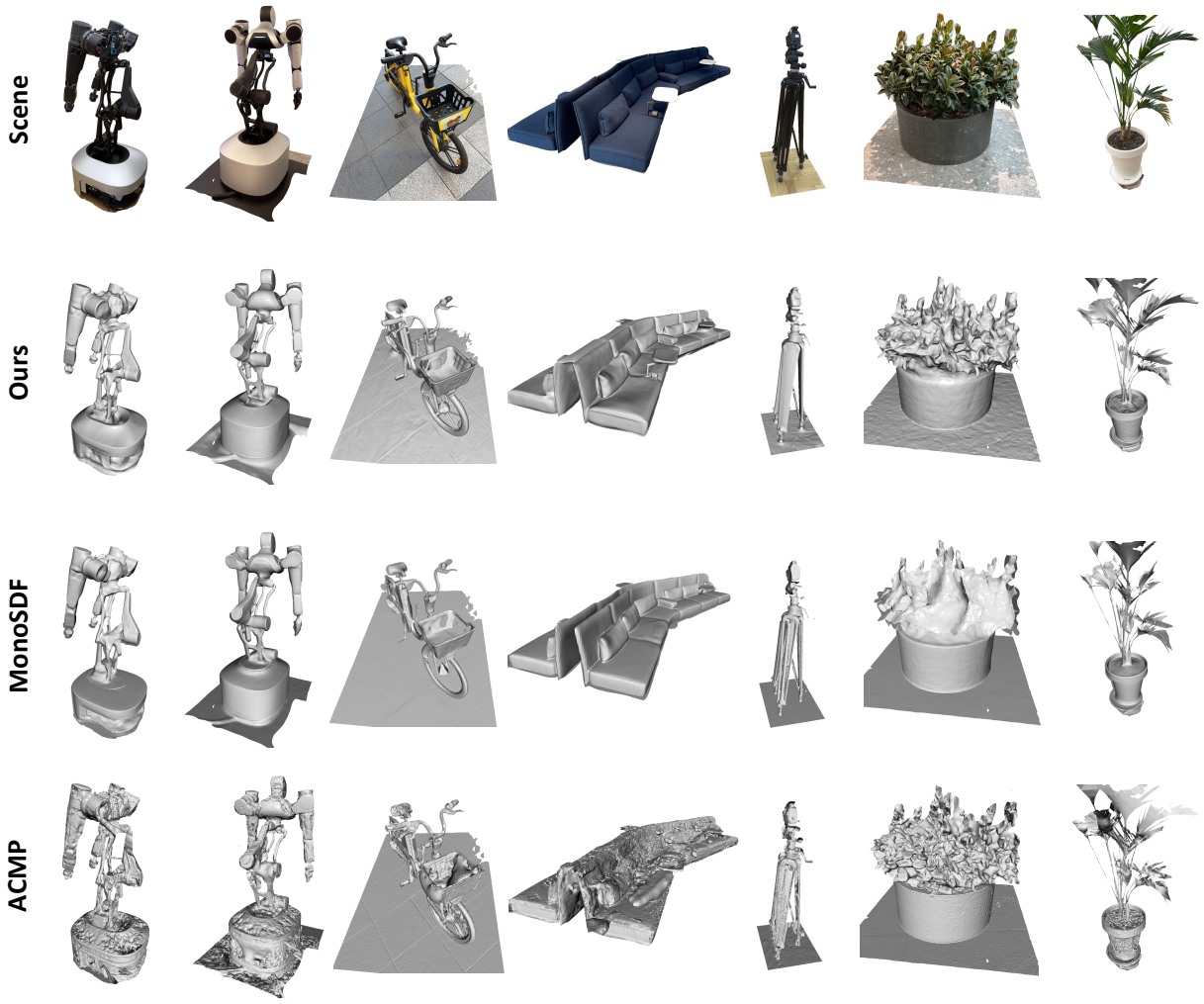

Figure 4. From left to right, we show the reconstruction results of *robot arm 1*, *robot arm2*, *bike*, *sofa*, *camera stand*, *plant*, and *tree*. We notice that MonoSDF shows the high quality mesh for *sofa* and *camera stand* by improving the quality of low-textured regions. However, since the pretrained model provides bad quality of monocular depth and normal maps, Our method performs better than MonoSDF for *plant*, *tree*, and *bike*

Figure 5. Triangular mesh reconstructed by our method, VolSDF [10], and MonoSDF [11]. From top to bottom, we show the results of *delivery robot*, *recliner chair*, *office chair*, and *cafe stand*. Here, MonoSDF can reconstruct the smooth space for low-textured regions compared to our method.

5. Run-time Analysis

Our pipeline consists of three modules. First, **RGBD-aided SfM** (Sec. 3.1) is imperative to refine initial poses due to noisy sensor data in Fig. 5 (a). Without RGBD-aided SfM, our geometric reconstruction with initial poses shows poor quality in Fig. 5 (b). This process takes around 5 minutes on i9 CPU and Nvidia 2080 GPU to estimate accurate poses. In **Geometry Reconstruction** (Sec. 3.2), both classical 3D reconstruction methods like MVS and depth Fusion (ACMP and TSDF-Fusion in Fig. 6) fail to generate a decent mesh. Thus, we emphasize that applying neural geometry reconstruction after MVS is necessary to generate a high-quality mesh (our method in Fig. 6). Overall, most of the time in our pipeline is spent on geometry reconstruction (around 10 hours on Nvidia V100 GPU). Then, the mesh simplification which only takes a few seconds. Lastly, in **Texture optimization** (Sec. 3.3), the classical texture reconstruction [7] takes only a few seconds. Fig. 7 and Table 1. show that our proposed texture fine-tuning solves the seams and texture misalignment issues often seen with this classical method. Our proposed module, which takes less than 15 minutes on i9 CPU and Nvidia 2080 GPU, can generate visually realistic textures.

References

[1] 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] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021. 4

[3] 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

[4] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 1

[5] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 1

[6] 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

[7] 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, 4, 7

[8] 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. 1, 2, 3

[9] Qingshan Xu and Wenbing Tao. Planar prior assisted patchmatch multi-view stereo. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12516–12523, 2020. 1, 2, 3

[10] 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. 4, 6

[11] 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. 4, 6

[12] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1

[13] 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. 2, 3

[14] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 1