Multi-HexPlanes: A Lightweight Map Representation for Rendering and 3D Reconstruction

Jianhao Zheng[1] Gábor Valasek[2] Daniel Barath[3] Iro Armeni[1]
[1]Stanford Univsersity [2]ELTE [3]ETH Zurich
{jianhao,iarmeni}@stanford.edu, oldbrian@gmail.com, barath.daniel@sztaki.mta.hu

Abstract

In the supplementary material, we provide additional implementation details (Section 1), as well as quantitative results on rendering and reconstruction performance with ground truth camera poses, 3DGS initialization with different resolutions, and qualitative results (Section 2).

1. Implementation Details

In the following, we provide more details about our system and specifically on the parameters, the algorithm to encode the surface normal, and the process to filter outlier points before the mesh reconstruction.

System Parameters. We set the threshold of color difference $C_{threshold}$ to be 60. This parameter is used to decide whether a new channel should be added to a pixel. In addition, the maximum weight for each channel W_{max} is set as 5. For the Poisson surface reconstruction [3], we use the default parameters following the Open3D documentation [8].

Surface Normal Encoding. To generate mesh from Multi-HexPlanes, each channel needs to have surface normal information as an extra attribute. In addition to the normal surface, each channel h stores the standard attributes of color, distance of observation to its face, and weight. Each of them requires 4 bytes of memory, resulting in a total of 12 bytes. The surface normal $\mathbf{N_h}$ has three components that have non-integer values. If we simply store $\mathbf{N_h}$ as three `float` values, the storage of a channel will increase to 24 bytes, which doubles the size of the map. This contradicts our aim of saving memory for better texture and reconstruction of the map. Hence, we store $\mathbf{N_h}$ as a `uint32`, which only takes 4 bytes of memory.

We first use octahedral normal vectors (ONV) [5] to encode the 3D surface normal vector $\mathbf{N_h}$ to a 2D vector

(a) Before filter (b) After filter

Figure 1. **Example scene on ScanNet [2] with Multi-HexPlanes.** Our outlier filtering process successfully removes noisy channels which points do not belong to any real-world surface.

(u_h, v_h), as follows:

$$\begin{aligned}
\mathbf{N_h} &\leftarrow \frac{\mathbf{N_h}}{|\mathbf{N_h}(x)| + |\mathbf{N_h}(y)| + |\mathbf{N_h}(z)|},\\
u_h &= \begin{cases} \mathbf{N_h}(x) & \text{if } \mathbf{N_h}(z) \geq 0,\\ (1 - |\mathbf{N_h}(y)|)\text{sign}(\mathbf{N_h}(x)) & \text{otherwise,} \end{cases}\\
v_h &= \begin{cases} \mathbf{N_h}(y) & \text{if } \mathbf{N_h}(z) \geq 0,\\ (1 - |\mathbf{N_h}(x)|)\text{sign}(\mathbf{N_h}(y)) & \text{otherwise,} \end{cases}
\end{aligned} \tag{1}$$

where $u_h, v_h \in [-1, 1]$. This process can be interpreted as mapping the octants of a unit sphere to the faces of an octahedron. Then, the octahedron is unwrapped to a 2D square. This process is efficient to encode and decode with minimal error [1]. We then map u_h and v_h to the range $[0, 2^{16}]$ so as to store them as two `uint16` variables that are further merged as a `uint32`. In our experiments, we found that the reconstruction error has nearly no difference to storing the surface normal as three `float` variables and the encoding time is trivial. However, this representation saves 8 bytes of memory per channel.

Outlier filtering. Due to inaccurate depth pixels in the input frames of ScanNet dataset [2], the projected 3D points contain noise that does not represent any real-world surface (Figure 1 (a)). This results to channels in Multi-HexPlanes getting falsely updated. To mitigate this, we filter out noise before performing Poisson Reconstruction on the ScanNet

Outlier Filtering	Depth L1 [cm] ↓	Comp. Error [cm] ↓	Acc. Error [cm] ↓
without (1cm)	22.07	4.97	14.16
with (1cm)	**20.85**	**4.80**	**12.10**
without (2cm)	22.13	4.83	14.09
with (2cm)	**21.04**	**4.72**	**12.30**
without (4cm)	22.76	4.55	14.12
with (4cm)	**21.40**	**4.48**	**12.01**
without (8cm)	24.55	4.98	13.54
with (8cm)	**20.89**	**4.61**	**11.82**

Table 1. **Reconstruction evaluation on the ScanNet dataset [2] with/without outlier filtering.** The reported metrics are averaged over 20 scenes. Best results per resolution are in bold.

dataset [2]. Specifically, we record the number of times that a channel is updated by the input points. A channel is considered as an outlier if it is updated less than T_{th} times. Hence, no point will be extracted from that channel. We empirically set T_{th} as the threshold corresponding to the 25^{th} percentile of the update times for all channels. The impact of the outlier filtering is shown in Figure 1 (b). The quantitative results in Table 1 confirm the improvement on mesh reconstruction over all metrics and for all resolutions when outliers are filtered.

2. Additional Results

Rendering with GT Poses. Table 2 shows the rendering performance of our method and Voxblox [7] on the Replica [12] dataset when using ground truth (GT) poses. Same as the performance using poses estimated by ORB-SLAM2 [6], Multi-HexPlanes outperforms Voxblox [7] by a clear margin under the same pixel/voxel resolution. In addition, Multi-HexPlanes still saves 67% of the map size when compared to Voxblox.

We report the results per each scene of Replica [12] in Table 3. The performance of Multi-HexPlanes is consistently better than that of Voxblox [7] on all scenes in Replica, regardless of the resolution and estimated camera poses. When using GT poses, both methods achieve an improvement in the rendering metrics, which further confirms the impact of the camera poses on the rendering quality. The rendering performance is also highly correlated to the pixel/voxel resolution. For each scene, methods using 1 cm resolution achieve better metrics than 8 cm resolution. The improvement is more significant when the GT poses are used. In addition, our method with 1 cm pixel resolution and GT poses has very close metrics to Point-SLAM [9]. In some specific scenes, Multi-HexPlanes even achieves better performance.

Additional Qualitative Results. Figure 2 provides additional qualitative results of rendered images. When using a coarser voxel resolution, Voxblox fuses the colors of different objects if they fall into the same voxel, leading to blurry rendering on the border of two neighboring objects. Thanks to the multiple-plane representation, Multi-HexPlanes still renders distinctive colors on different objects when using the same pixel size. As highlighted in Figure 2, the improvement is especially visible on the edge of the board in `office4`, as well as the pillows and the sofa in `office2` and `room0`. We also highlight additional regions where Voxblox (1 cm) renders invalid pixels due to errors in camera poses. Moreover, failure cases are provided in Figure 2 and are highlighted with red boxes. Multi-HexPlanes renders blue colors on the wall and brown on the ground. The source of the false colors is the texture of the blue chair and the brown desk respectively, which are projected on the same pixels. However, despite the small flaws, the PSNR score of the entire image is still much higher than that rendered by Voxblox [7].

Reconstruction with GT Poses. Figure 3 shows the average reconstruction results when using GT poses to build the map on Replica and ScanNet. For the ScanNet dataset, we still apply the same outlier filtering process since the noisy depth measurements remain despite the GT camera poses. Similarly to the results using poses from [6], our method achieves better performance when the resolution is coarse, while saving around half of the map size. When using GT poses, the accuracy of the 3D location of the input observations is improved and, thus, the mapping is more accurate. As a result, the overall reconstruction performance of both methods is better than when using estimated poses. Moreover, the amount of map size saved by Multi-HexPlanes is much larger on the ScanNet dataset [2] with GT poses, since a more accurate 3D location of input points results to less wrong channel updates and extensions.

3DGS Initialization. Table 4 shows the results of the initialization of 3DGS [4] with different voxel and pixel sizes. Same as in the main paper, the PSNR scores are averaged over the test set. In general, both Multi-HexPlanes and Voxblox provide better initialization with a higher resolution, which indicates that the denser the feature points the more the training of 3DGS [4] improves. Under the same resolution, Multi-HexPlanes always outperforms Voxblox, showing that our feature points are more representative. When using 8 cm resolution, the performance of Voxblox is even worse than the standard COLMAP [10, 11] initialization, while Multi-HexPlanes still achieves a better performance. In addition, the scores of 3DGS initialized by Multi-HexPlanes is higher after 7k iterations of training than 30k, regardless of the pixel resolution. On the other hand, 3DGS reaches the better performance after 30k of iterations when using the standard COLMAP initialization. This further indicates that our dense feature points lead to faster convergence, making the training of 3DGS more efficient.

Method	PSNR [dB] ↑	SSIM ↑	LPIPS ↓	Map Size [MB] ↓	Mapping Time/Frame [s] ↓
Voxblox-1cm [7]	33.00	0.902	0.127	284.13	0.11
Multi-HexPlanes-1cm	**33.40**	**0.914**	**0.118**	96.33	0.27
Voxblox-2cm [7]	29.84	0.850	0.203	68.34	0.07
Multi-HexPlanes-2cm	30.92	0.868	0.189	25.60	0.19
Voxblox-4cm [7]	27.01	0.809	0.271	17.76	**0.05**
Multi-HexPlanes-4cm	28.30	0.829	0.247	6.89	0.14
Voxblox-8cm [7]	24.35	0.780	0.360	5.01	**0.05**
Multi-HexPlanes-8cm	25.84	0.801	0.322	**1.86**	0.12

Table 2. **Rendering Performance with GT Poses.** The reported results are averaged over the 8 scenes on Replica [12]. Multi-HexPlanes consistently outperforms Voxblox [7] with the same voxel/pixel resolution, while significantly saving storage. Best results per metric are highlighted as **1st**, 2nd, and 3rd.

Method	Metric	Room 0	Room 1	Room 2	Office 0	Office 1	Office 2	Office 3	Office 4	Avg.
Neural Implicit Fields										
NICE-SLAM [15]	PSNR [dB] ↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42
	SSIM ↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809
	LPIPS ↓	0.330	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233
Vox-Fusion* [13]	PSNR [dB] ↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41
	SSIM ↑	0.683	0.751	0.798	0.857	0.876	0.794	0.803	0.847	0.801
	LPIPS ↓	0.303	0.269	0.234	0.241	0.184	0.243	0.213	0.199	0.236
Point-SLAM [9]	PSNR [dB] ↑	**32.40**	**34.08**	**35.50**	**38.26**	**39.16**	**33.99**	**33.48**	33.49	**35.17**
	SSIM ↑	**0.974**	**0.977**	**0.982**	**0.983**	**0.986**	**0.960**	**0.960**	**0.979**	**0.975**
	LPIPS ↓	**0.113**	**0.116**	**0.111**	**0.100**	**0.118**	0.156	0.132	0.142	0.124
CPU-only methods										
Voxblox-1cm [7]	PSNR [dB] ↑	24.94	26.72	27.57	32.20	34.01	27.82	27.05	30.13	28.91
	SSIM ↑	0.731	0.767	0.846	0.893	0.916	0.829	0.839	0.892	0.844
	LPIPS ↓	0.219	0.252	0.175	0.157	0.129	0.202	0.174	0.150	0.181
Multi-HexPlanes 1cm	PSNR [dB] ↑	26.22	28.73	29.28	33.28	34.69	30.03	29.33	31.33	30.49
	SSIM ↑	0.775	0.836	0.867	0.903	0.919	0.866	0.879	0.908	0.876
	LPIPS ↓	0.183	0.198	0.160	0.156	0.138	0.141	0.136	0.137	0.154
Voxblox [7] 1cm-GT-pose	PSNR [dB] ↑	29.69	30.80	31.78	36.49	36.29	32.44	32.65	33.86	33.00
	SSIM ↑	0.851	0.861	0.895	0.937	0.932	0.902	0.913	0.929	0.902
	LPIPS ↓	0.136	0.176	0.141	0.115	0.122	0.132	0.096	0.098	0.127
Multi-HexPlanes 1cm-GT-pose	PSNR [dB] ↑	30.08	30.89	32.33	37.30	36.61	33.20	32.76	**34.31**	33.40
	SSIM ↑	0.870	0.880	0.910	0.943	0.933	0.917	0.926	0.936	0.914
	LPIPS ↓	0.120	0.159	0.127	0.109	0.128	0.120	**0.085**	**0.094**	**0.118**
Voxblox-8cm [7]	PSNR [dB] ↑	21.28	23.09	24.08	26.93	28.28	22.13	22.50	23.51	24.02
	SSIM ↑	0.645	0.736	0.784	0.830	0.845	0.790	0.772	0.827	0.783
	LPIPS ↓	0.429	0.446	0.374	0.343	0.332	0.352	0.327	0.313	0.364
Multi-HexPlanes 8cm	PSNR [dB] ↑	22.07	24.31	24.67	27.93	29.62	24.46	24.27	25.11	25.35
	SSIM ↑	0.664	0.753	0.798	0.842	0.864	0.818	0.800	0.839	0.801
	LPIPS ↓	0.398	0.403	0.343	0.315	0.289	0.307	0.297	0.289	0.329
Voxblox [7] 8cm-GT-pose	PSNR [dB] ↑	21.96	23.45	24.05	27.30	28.15	22.25	23.29	24.39	24.35
	SSIM ↑	0.648	0.739	0.783	0.833	0.845	0.790	0.775	0.829	0.780
	LPIPS ↓	0.419	0.443	0.374	0.337	0.330	0.349	0.318	0.310	0.360
Multi-HexPlanes 8cm-GT-pose	PSNR [dB] ↑	23.22	24.81	25.17	28.48	29.35	34.55	25.27	25.90	25.84
	SSIM ↑	0.671	0.756	0.801	0.846	0.863	0.818	0.809	0.843	0.801
	LPIPS ↓	0.379	0.398	0.339	0.306	0.291	0.308	0.277	0.282	0.322

Table 3. **Rendering Performance on each scene of Replica [12].** Results of NICE-SLAM [15], Vox-Fusion [13], and Point-SLAM [9] are from Point-SLAM [9]. The reported results are based on estimated poses unless explicitly stated as 'GT-pose'. Best results per GPU/CPU methods are highlighted as **1st**, 2nd, and 3rd.

References

[1] Zina H Cigolle, Sam Donow, Daniel Evangelakos, Michael Mara, Morgan McGuire, and Quirin Meyer. A survey of efficient representations for independent unit vectors. *Journal of Computer Graphics Techniques*, 3(2), 2014. 1

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and*

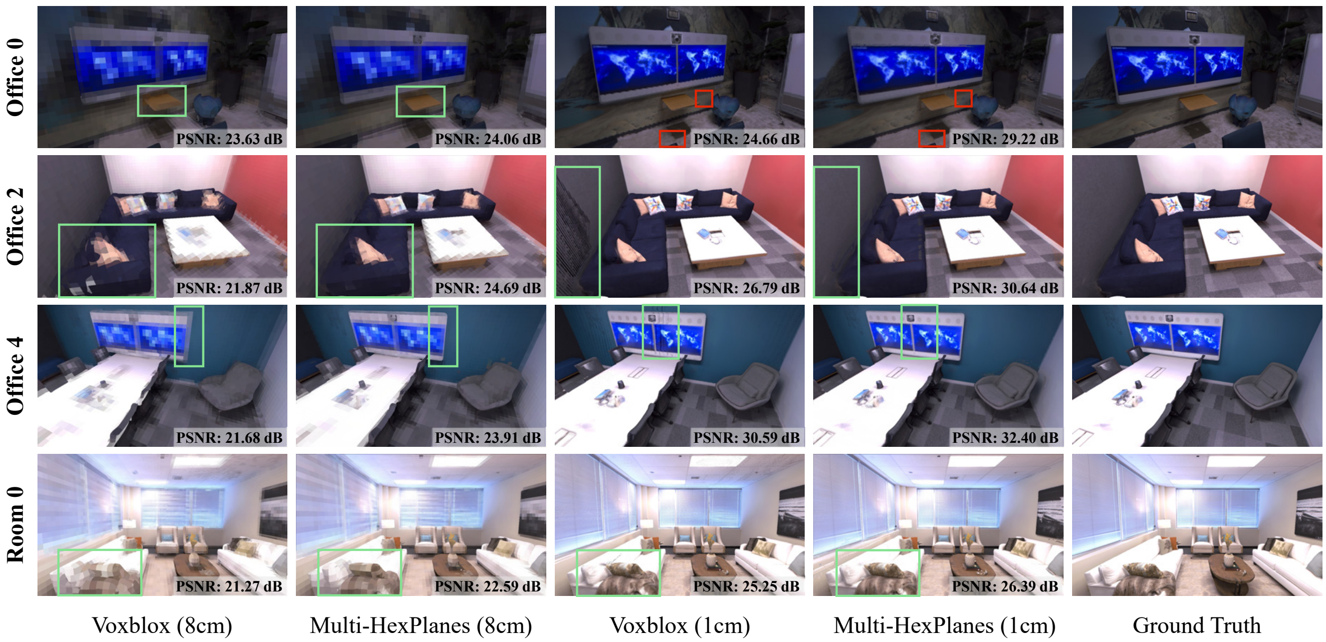

Figure 2. **Additional rendering results on Replica [12].** We highlight the regions where Multi-HexPlanes has noticeable improvement with green boxes. We also show failure cases where our method generates worse rendering with red boxes.

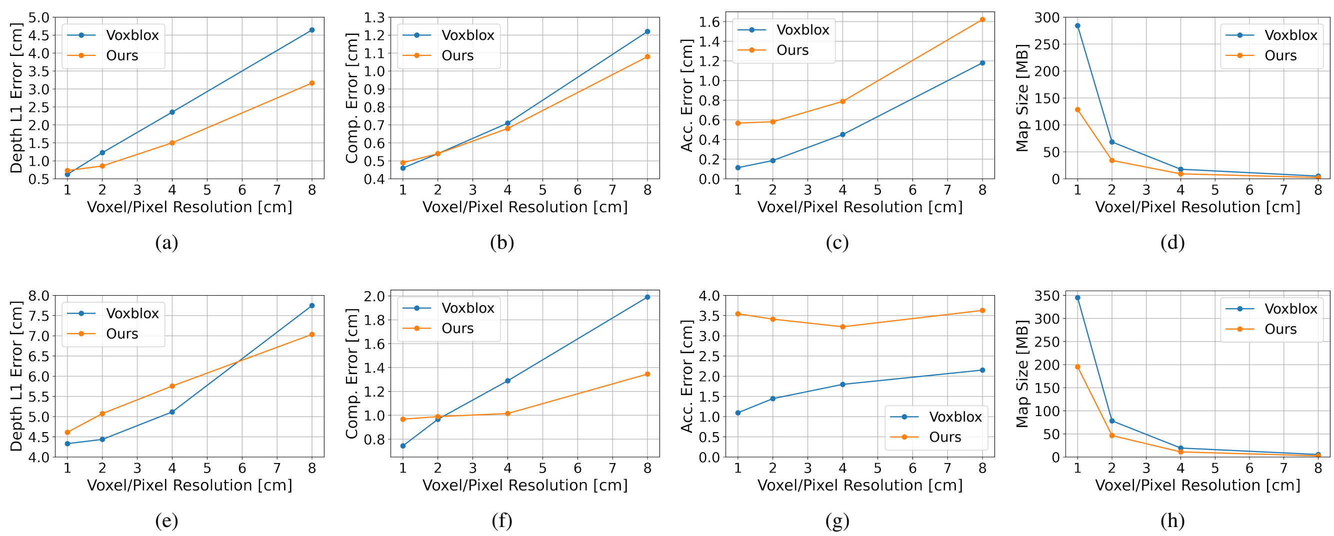

Figure 3. **Reconstruction evaluation with GT poses** We show the average results on the Replica [12] (a-d) and ScanNet [2] (e-h) datasets. For all metrics, lower values are better. Similar to the results with estimated poses, Multi-HexPlanes performs better in coarse resolution levels, while taking less map size in all resolution levels.

pattern recognition, pages 5828–5839, 2017. 1, 2, 4

[3] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 1

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2, 5

[5] Quirin Meyer, Jochen Süßmuth, Gerd Sußner, Marc Stamminger, and Günther Greiner. On floating-point normal vectors. In *Computer Graphics Forum*, volume 29, pages 1405–1409. Wiley Online Library, 2010. 1

[6] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. 2

[7] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017*

Initialization Method	S1	S2	S3	S4	S5	Avg.
COLMAP-7k [10, 11]	26.95	27.84	25.12	25.10	**20.25**	25.05
COLMAP-30k [10, 11]	27.03	27.96	25.79	25.07	20.00	25.17
Voxblox-1cm-7k [7]	27.03	27.96	25.79	25.07	19.99	25.16
Voxblox-1cm-30k [7]	27.20	28.43	25.76	**26.03**	20.18	25.50
Multi-HexPlanes-1cm-7k	26.83	28.29	**25.98**	25.85	19.51	25.29
Multi-HexPlanes-1cm-30k	**27.22**	**28.48**	25.70	25.99	20.20	**25.52**
Voxblox-2cm-7k [7]	27.15	28.23	25.62	25.86	20.12	25.40
Voxblox-2cm-30k [7]	26.78	28.08	25.91	25.79	19.63	25.24
Multi-HexPlanes-2cm-7k	27.16	**28.48**	25.68	25.89	20.08	25.46
Multi-HexPlanes-2cm-30k	26.80	28.38	25.90	25.76	19.72	25.31
Voxblox-4cm-7k [7]	26.98	28.11	25.33	25.60	20.00	25.20
Voxblox-4cm-30k [7]	26.86	28.04	25.78	25.57	19.37	25.13
Multi-HexPlanes-4cm-7k	26.98	28.32	25.57	25.68	20.02	25.31
Multi-HexPlanes-4cm-30k	26.87	28.29	25.81	25.57	19.63	25.23
Voxblox-8cm-7k [7]	27.03	27.68	24.99	25.31	19.99	25.00
Voxblox-8cm-30k [7]	26.89	27.96	25.57	25.40	19.45	25.05
Multi-HexPlanes-8cm-7k	27.01	28.13	25.21	25.46	20.13	25.19
Multi-HexPlanes-8cm-30k	26.91	28.07	25.62	25.55	19.56	25.14

Table 4. **Novel view synthesis performance using 3DGS [4] with different initialization on ScanNet++ [14].** We report the average PSNR scores for Multi-HexPlanes and Voxblox [7] with different pixel and voxel resolutions respectively. Best results per scene are highlighted as **1st**, 2nd, and 3rd.

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1366–1373. IEEE, 2017. 2, 3, 5

[8] Open3D. Surface reconstruction — open3d 0.17.0 documentation, 2024. Accessed: 2024-09-10. 1

[9] Erik Sandström, Yue Li, Luc Van Gool, and Martin R Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18433–18444, 2023. 2, 3

[10] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5

[11] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 5

[12] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 2, 3, 4

[13] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507. IEEE, 2022. 3

[14] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023. 5

[15] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. 3