Point-SLAM: Dense Neural Point Cloud-based SLAM — Supplementary Material —

Erik Sandström[1*] Yue Li[1*] Luc Van Gool[1,2] Martin R. Oswald[1,3]
[1]ETH Zürich, Switzerland [2]KU Leuven, Belgium [3]University of Amsterdam, Netherlands

Abstract

This supplementary material accompanies the main paper by providing further information for better reproducibility as well as additional evaluations and qualitative results.

Contents

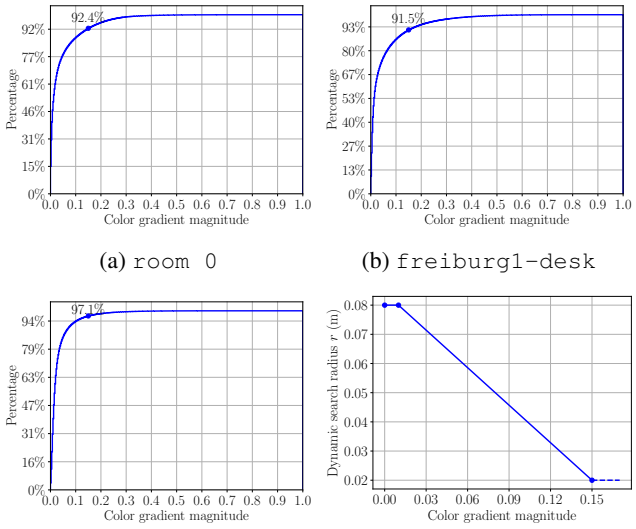
A. Videos

We provide an introductory video to our paper along with this document. The video describes the method and the most important results along with the visualization of the online reconstruction process of our proposed method compared to NICE-SLAM [8] and Vox-Fusion [7].

B. Method

In the following, we provide more details about our method, specifically the hyperparameter choices for the dynamic resolution strategy and architecture of our exposure compensation network.

Design Choices Dynamic Resolution Strategy. We empirically set the upper bound for the color gradient magnitude threshold to $g_u = 0.15$ for all evaluated datasets. Based on the pre-calculated cumulative gradient magnitude histograms shown in Figs. B.1a to B.1c (depicting `room 0`, `freiburg1-desk`, and `scene0000_00`), we observe that approximately less than 10% of all pixels exceed the upper threshold. The threshold $g_u = 0.15$ strikes a good balance between resolving highly textured regions and model compression. The cumulative histograms in



(a) `room 0` (b) `freiburg1-desk`

(c) `scene0000_00` (d) Search radius $r(u, v)$.

Figure B.1: **Color Gradient Magnitude Histograms and Search Radius.** The cumulative histograms (a-c) show the percentage of pixels below a certain gradient magnitude. (d) Search radius $r(u, v)$ as a function of the gradient magnitude at pixel (u, v).

Figs. B.1a to B.1c also reveal that the majority of pixels have close to zero gradient magnitude. We use a lower bound $g_l = 0.01$ for all datasets. The search radius $r(u, v)$ as a function of the color gradient magnitude at pixel (u, v) is shown in Fig. B.1d.

Exposure Network Architecture. For the exposure compensation network G_ϕ, we use one hidden layer with 128 neurons followed by a softplus activation. The input latent vector is 8-dimensional and the output is 12-dimensional, which is reshaped into a 3×3 affine matrix and a 3×1 translation vector. The network G_ϕ and latent vector are jointly optimized both during mapping and tracking. The latent vector is put in shared memory and if the current frame is used for mapping, the latent vector is first optimized during tracking then refined during mapping. We did not explore other optimization strategies which could potentially

*Equal contribution.

Dataset	Map Every	Keyframe Every	Map Window	Track Iter.	Map Iter.
Replica [5]	5	20	12	40	300
TUM-RGBD [6]	2	50	10	200	150
ScanNet [1]	5	10	20	100	300

Table C.1: **Parameter Configurations on Tested Datasets.** Map Every: how often (in frames) mapping is done. Map Window: How many keyframes that are sampled to overlap with the current viewing frustum for mapping. Iter.: Iterations (optimization steps).

improve performance.

C. Implementation Details

We use PyTorch 1.12 and Python 3.10 to implement the pipeline. Training is done with the Adam optimizer and the default hyperparameters *betas* $= (0.9, 0.999)$, *eps* = $1e$-08 and *weight_decay* = 0. The results are gathered using various Nvidia GPUs, all with a maximum memory of 12 GB. The learning rate for tracking is 0.002 on Replica and TUM-RGBD and 0.0005 on ScanNet. We use a learning rate of 0.03 for the initial geometry only optimization stage and 0.005 during color and geometry optimization stage. Table C.1 describes other dataset-specific hyperparameters such as the mapping window size which describes how many frames (current frame and selected keyframes) are used during mapping. We also follow [8] and use a simple keyframe selection strategy which adds frames to the keyframe database at regular intervals (see also Table C.1).

D. Evaluation Metrics

Mapping. We use the following five metrics to quantify the reconstruction performance. We compare the ground truth mesh to the predicted mesh. The F-score is defined as the harmonic mean between Precision (P) and Recall (R), $F = 2\frac{PR}{P+R}$. Precision is defined as the percentage of points on the predicted mesh which lie within some distance τ from a point on the ground truth mesh. Vice versa, Recall is defined as the percentage of points on the ground truth mesh which lie within the same distance τ from a point on the predicted mesh. In all our experiments, we use a distance threshold $\tau = 0.01$ m. Before the Precision and Recall are computed, the input meshes are aligned with the iterative closest point (ICP) algorithm. We use the evaluation script provided by the authors of [4][1]. Finally, we report the depth L1 metric which renders depth maps from randomly sampled view points from the reconstructed and ground truth meshes. The depth maps are then compared and the L1 error is reported and averaged over 1000 sampled views. We use the evaluation code provided by [8].

[1] https://github.com/tfy14esa/evaluate_3d_reconstruction_lib

Metric	off 0	off 1	off 2	off 3	off 4
Mesh Depth L1	0.30	0.61	0.53	0.54	0.45
Rendering Depth L1	**0.037**	**0.025**	**0.054**	**0.082**	**0.061**

Table E.1: **Depth L1 Error [cm] on Replica [5].** The table reports the depth L1 error for the rendered depth map and for the reconstructed mesh (after TSDF fusion and Marching cubes). The results in the main paper only report the depth L1 error for the mesh.

Tracking. We use the absolute trajectory error (ATE) RMSE [6] to compare tracking error across methods. This computes the translation difference between the estimated trajectory and the ground truth. Before evaluating the ATE RMSE, we align the trajectories with Horn's closed form solution [2].

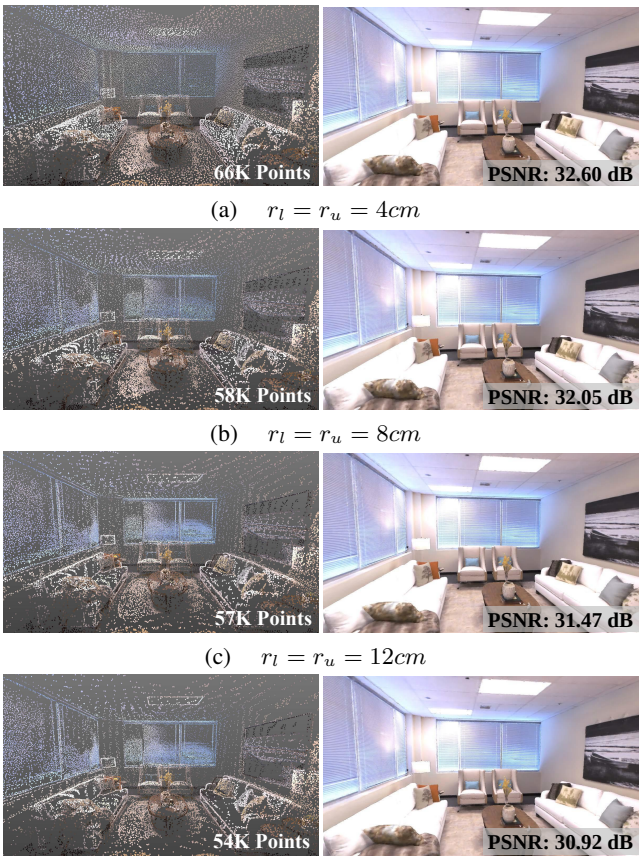
E. More Experiments

Dynamic Search Radius Visualization. In the main paper, we ablate how the tracking, reconstruction and rendering performance metrics vary as the upper bound r_u of the search radius is changed. In Fig. E.1 we show qualitative examples of the surface point cloud for some selected values r_u on `room 0`. Fig. E.1a shows the point cloud without dynamic resolution using fixed $r_l = r_u = 4cm$ for all points. In Figs. E.1b to E.1d we enable dynamic resolution and show how the point density varies across the scene for different values of r_u. We use $r_l = 2cm$ for all these experiments. The total number of points decreases from 66K to 54K. This is due to the sparsification of the point density in regions with little texture information. It is clear that using a dynamic search radius preserves rich textures, while effectively sparsifying the point density in textureless regions such as the sofas and walls.

Additional Qualitative Reconstructions. In Fig. E.4 we show additional reconstructions from the Replica dataset where our method is compared to NICE-SLAM [8] and Vox-Fusion [7].

Additional Qualitative Renderings. In Fig. E.5 we show additional renderings from the Replica dataset where our method is compared to NICE-SLAM [8] and Vox-Fusion [7].

Evaluating Depth Error on Rendered Depth Maps. Table E.1 shows additional results when the depth L1 error is evaluated directly on the rendered depth maps from the neural point cloud. This is in contrast to the main paper where we report the depth L1 on the predicted mesh from randomly sampled views. Compared to the mesh depth L1 metric, we report one order of magnitude smaller error from our rendered depth maps along the estimated trajectory.



(a) $r_l = r_u = 4cm$

(b) $r_l = r_u = 8cm$

(c) $r_l = r_u = 12cm$

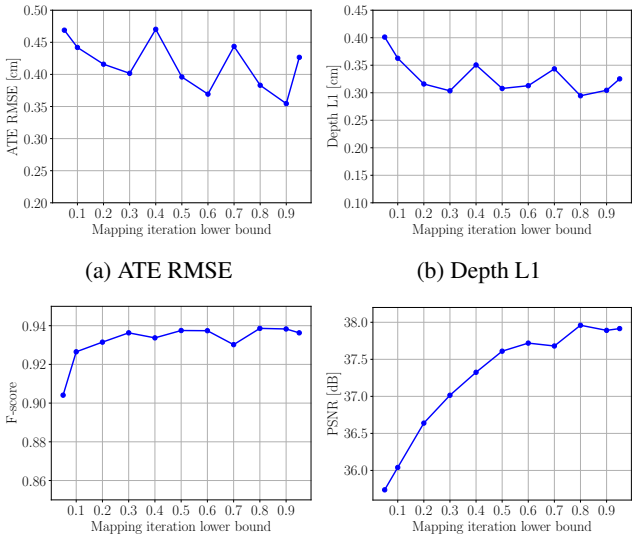
(d) $r_l = r_u = 16cm$

Figure E.1: **Dynamic Search Radius Visualization.** With the dynamic point density enabled (Figs. E.1b to E.1d), we use less points than without the dynamic point density (Fig. E.1a) while preserving high point densities in texture-rich regions, such as the window blinds.

Adaptive Mapping Ablation. As mentioned in the implementation details in the main paper, the number of mapping iterations is computed as $m_i = m_i^d n/300$, where m_i^d is the default mapping iterations and n is the number of added points for the frame at hand. By default, we clip m_i to lie within $[0.95m_i^d, 2m_i^d]$. We further decrease the lower bound from 0.95 to [0.9, 0.05] on the `office 0` scene. The resulting average mapping iterations per frame and associated per frame mapping runtimes are presented in Table E.2. We find that we can speed up the mapping phase by a factor of four compared to the results reported in the main paper.

Lower Bound	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.05
Avg. Iter./Frame	281	253	225	199	172	147	123	102	78	72
Mapping/Frame [s]	9.22	8.30	7.39	6.53	5.65	4.83	4.04	3.35	2.56	2.36

Table E.2: **Adaptive Mapping Ablation.** For various choices of the lower bound, Average Mapping Iterations, and Per-frame Mapping Runtime using adaptive iterations.



(a) ATE RMSE

(b) Depth L1

(c) F-score

(d) PSNR

Figure E.2: **Adaptive Mapping Ablation.** We report the rendering, tracking and reconstruction accuracy for different lower bound values and find that only the rendering quality is marginally worse as fewer mapping iterations are used.

Fig. E.2 summarizes the rendering, tracking and reconstruction metrics for different lower bound values. All metrics are virtually unchanged until the lower bound drops to 0.8. When the lower bound 0.05 is used, the per frame mapping speed is increased by 406% compared to the default case, while the tracking accuracy only degrades by 10%, depth L1 by 23%, F-score by 3% and PSNR by 6%. This suggests the effectiveness of our adaptive mapping iteration strategy.

Additional ScanNet Results. In Table E.3, we provide additional evaluation on four ScanNet scenes over the main paper and show competitive performance compared to the baseline methods. When taking the average over all scenes (the scenes in the main paper and the additional four scenes we select), we find that our method outperforms NICE-SLAM [8] and Vox-Fusion [7].

Qualitative Results on TUM-RGBD and ScanNet. We compare our method to NICE-SLAM [8] and ESLAM [3] in Fig. E.3. In the cases where ground truth is available, we also compare to that. We showcase, from top to bottom, the rendering performance on TUM-RGBD, the colored mesh and the phong shaded mesh. The results suggest that our method can produce high quality renderings, textured and untextured meshes.

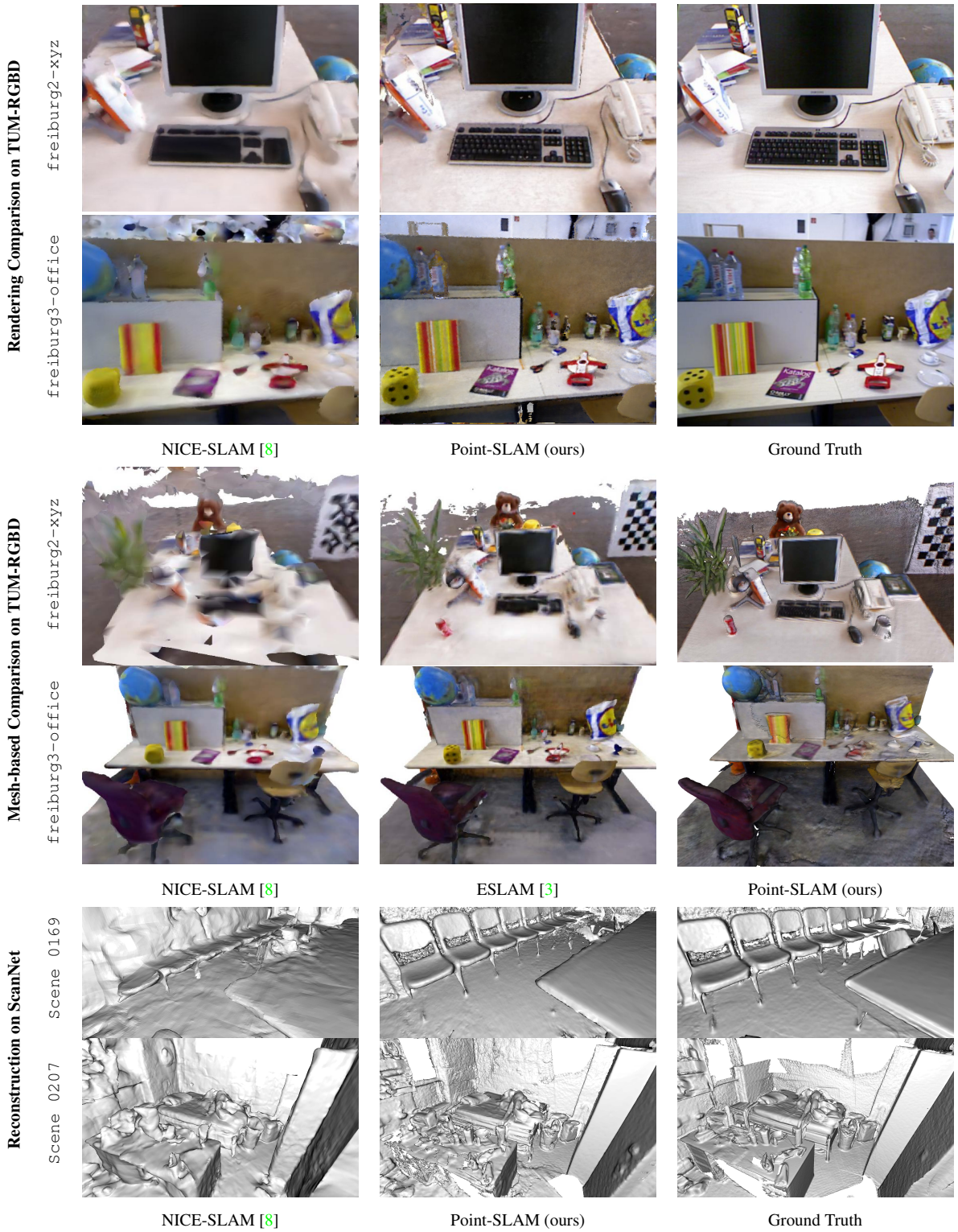


Figure E.3: **Rendering and Reconstruction Comparisons**. We showcase, from top to bottom, the rendering performance on TUM-RGBD, the colored mesh and the phong shaded mesh. The results suggest that our method can produce high quality renderings, textured and untextured meshes.

Method \ Scene	0000_00	0025_02	0059_00	0062_00	0103_00	0106_00	0126_00	0169_00	0181_00	0207_00	Avg.
NICE-SLAM [8]	12.00	10.11	14.00	4.59	**4.94**	**7.90**	21.80	**10.90**	**13.40**	**6.20**	10.58
Vox-Fusion* [7]	68.84 (16.55)	8.54	24.18	7.96	5.26	8.41	**5.77**	27.28	23.30	9.41	18.90 (13.67)
Point-SLAM (Ours)	**10.24**	**8.05**	**7.81**	**3.75**	7.79	8.65	8.10	22.16	14.77	9.54	**10.08**

Table E.3: **ScanNet Tracking** We report the ATE RMSE (↓ [cm]) as the average over three runs. For failed runs we report the average of only successful runs in parentheses. All methods work differently well on various scenes, but our method performs better on average. Best results are highlighted as **first**, second, and third.

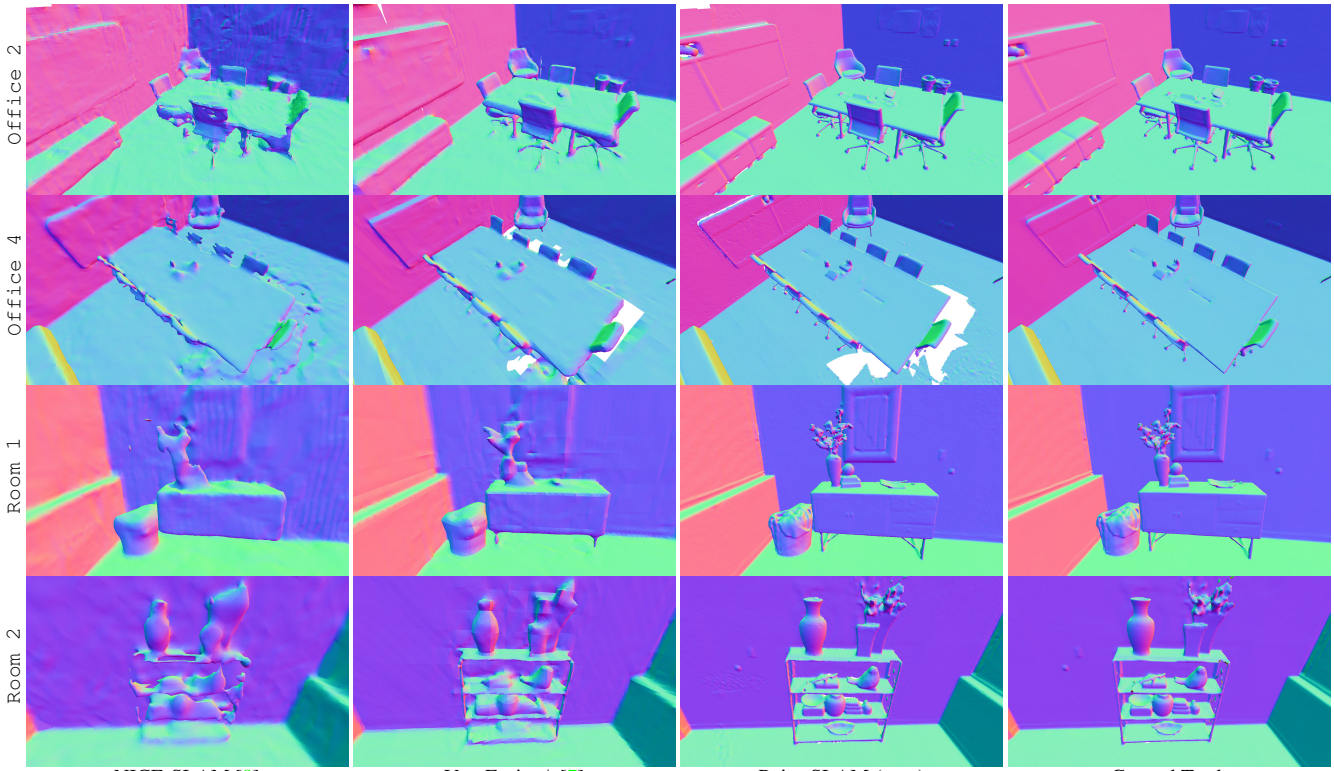


Figure E.4: **Reconstruction Performance on Replica [5]**. Point-SLAM yields on average more precise reconstructions than existing methods. We use normal shading to highlight geometric changes better.

References

[1] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE/CVF, 2017. 2

[2] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988. 2

[3] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. *arXiv e-prints*, pages arXiv–2211, 2022. 3, 4

[4] Erik Sandström, Martin R. Oswald, Suryansh Kumar, Silvan Weder, Fisher Yu, Cristian Sminchisescu, and Luc Van Gool. Learning Online Multi-Sensor Depth Fusion. In *European Conference Computer Vision (ECCV)*. CVF, 2022. 2

[5] 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, 5, 6

[6] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2012. 2

[7] 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 *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507. IEEE, 2022. 1, 2, 3, 5, 6

[8] 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 *IEEE/CVF Conference on Computer Vision and Pattern*

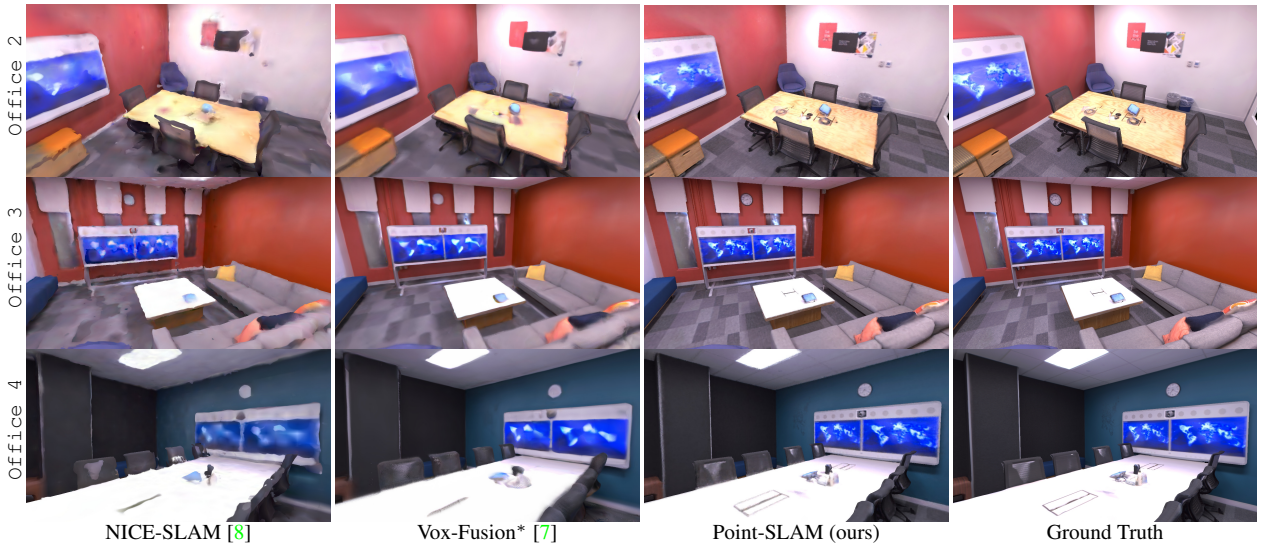


Figure E.5: **Rendering Performance on Replica [5]**. Thanks to the adaptive density of the neural point cloud, Point-SLAM is able to encode more high-frequency details and to substantially increase the fidelity of the renderings.

Recognition, pages 12786–12796, 2022. 1, 2, 3, 4, 5, 6