VTGaussian-SLAM: RGBD SLAM for Large Scale Scenes with Splatting View-Tied 3D Gaussians

Pengchong Hu [1] **Zhizhong Han** [1]

Abstract

Jointly estimating camera poses and mapping scenes from RGBD images is a fundamental task in simultaneous localization and mapping (SLAM). State-of-the-art methods employ 3D Gaussians to represent a scene, and render these Gaussians through splatting for higher efficiency and better rendering. However, these methods cannot scale up to extremely large scenes, due to the inefficient tracking and mapping strategies that need to optimize all 3D Gaussians in the limited GPU memories throughout the training to maintain the geometry and color consistency to previous RGBD observations. To resolve this issue, we propose novel tracking and mapping strategies to work with a novel 3D representation, dubbed view-tied 3D Gaussians, for RGBD SLAM systems. View-tied 3D Gaussians is a kind of simplified Gaussians, which is tied to depth pixels, without needing to learn locations, rotations, and multi-dimensional variances. Tying Gaussians to views not only significantly saves storage but also allows us to employ many more Gaussians to represent local details in the limited GPU memory. Moreover, our strategies remove the need of maintaining all Gaussians learnable throughout the training, while improving rendering quality, and tracking accuracy. We justify the effectiveness of these designs, and report better performance over the latest methods on the widely used benchmarks in terms of rendering and tracking accuracy and scalability. Please see our project page for code and videos at https://machineperceptionlab.github.io/VTGaussian-SLAM-Project.

[1]Machine Perception Lab, Wayne State University, Detroit, USA. Correspondence to: Zhizhong Han <h312h@wayne.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

SLAM is an important task in computer vision (Keetha et al., 2024; Zhu et al., 2022), 3D reconstruction (Wang et al., 2023; Hu & Han, 2023), and robotics (Adamkiewicz et al., 2022). SLAM methods resolve the computational problem of mapping unknown environments while tracking camera locations. Traditional SLAM methods usually use RGB or RGBD images to sense 3D surroundings and track spatial relationships among different frames. The resulting 3D maps are usually represented by 3D point clouds, which are discrete and not friendly to downstream applications that require continuous geometry representations, such as geometry modeling, editing, and virtual reality.

Due to the continuity and the ability to represent arbitrary topology and appearance, neural representations have been employed with differentiable rendering in SLAM systems (Zhang et al., 2023; Xinyang et al., 2023; Teigen et al., 2023; Sandström et al., 2023b; Sucar et al., 2021; Wang et al., 2023). Previous methods (Zhang et al., 2023; Xinyang et al., 2023; Teigen et al., 2023; Hu & Han, 2023) learn neural radiance fields (NeRF) (Mildenhall et al., 2020), a continuous implicit function modeling scene geometry and appearance, enabling image rendering for scene mapping and camera tracking. More recently, 3DGS (Kerbl et al., 2023) was proposed for high-quality and real-time rendering, which also provides a novel perspective for time-sensitive SLAM problems (Matsuki et al., 2024; Huang et al., 2024b; Yan et al., 2024; Yugay et al., 2023; Sandström et al., 2024). 3DGS can efficiently render 3D Gaussians that model radiance fields explicitly using differentiable rasterization which is faster than the ray tracing in NeRF. However, maintaining numerous 3D Gaussians to cover the whole scene while ensuring color and geometry consistency across previous frames often leads to poor rendering in tracking and mapping. This obstacle makes 3DGS still hard to scale up to extremely large scenes in SLAM, remaining the challenge of improving the rendering quality, tracking accuracy, and scalability of 3DGS in tracking cameras and mapping scenes.

To overcome this challenge, we propose an RGBD SLAM system with splatting view-tied 3D Gaussians. Our method introduces a novel point-based volume representation, dubbed view-tied 3D Gaussians, to represent the color and

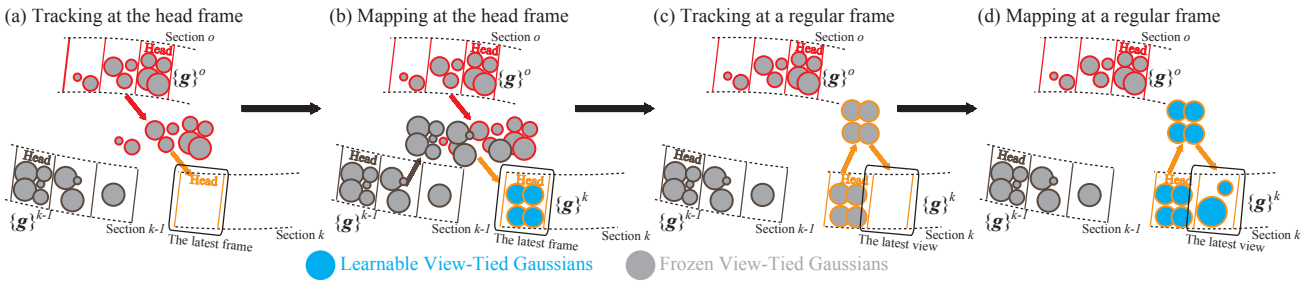

Figure 1. Overview. (a) and (c) are tracking strategies, while (b) and (d) are mapping strategies. Please refer to Sec. 3.1 for more details.

geometry of the scene, dedicated to reducing storage but pursuing better rendering quality. Different from the original 3D Gaussians (Kerbl et al., 2023), we tie a 3D Gaussian to each pixel on the depth, which makes their positions only determined by depth and camera poses, without a need of learning and storing their locations and also the density control. Meanwhile, we simplify an ellipsoid 3D Gaussian as a sphere, without saving rotations and multi-dimensional variances as well, which saves more storage for initiating even more Gaussians for describing more details. With view-tied 3D Gaussians, our novel tracking and mapping strategies can be conducted in rendering and optimizing the Gaussians that are merely related to the most recent views, rather than all Gaussians in the scene. These benefits enable us to use many more Gaussians to represent each frame, just keep the most relevant Gaussians in the limited GPU memories, and remove the need of maintaining the consistency of scene representations to keyframes. This ability scales up the size of scenes that we can handle and also significantly improves the rendering quality, even if just using simplified Gaussians. Our visual and numerical evaluations show our advantages over the latest methods. Our main contributions are listed below.

- We propose view-tied Gaussian splatting that significantly reduces storage but improves rendering quality with 3DGS in SLAM.
- We introduce a novel RGBD SLAM algorithm with view-tied Gaussian splatting. Our tracking and mapping strategies remove the need of holding and optimizing all Gaussians in memory throughout the training, which improves the scalability of 3DGS in SLAM.
- We report the state-of-the-art results on widely used benchmarks over the latest 3DGS-based SLAM.

2. Related Work

Multi-view Reconstruction. Recent multi-view reconstruction methods aim to learn neural implicit representations (Park et al., 2021; Müller et al., 2022; Rückert et al., 2021; Sara Fridovich-Keil and Alex Yu et al., 2022; Zhu et al., 2022; Azinović et al., 2022; Wang et al., 2022a; Bozic et al., 2021; Zou et al., 2022; Sun et al., 2021; Li et al., 2023) from multi-view images through volume rendering. Besides RGB images, we can also leverage depth (Yu et al., 2022; Azinović et al., 2022; Zhu et al., 2022; Zhang et al., 2024c) and normals (Yu et al., 2022; Wang et al., 2022b; Guo et al., 2022; Patel et al., 2024) as rendering supervision to infer more geometry details. With the emergence of 3DGS (Kerbl et al., 2023; Moenne-Loccoz et al., 2024), we can learn implicit representations or accurate depth maps for reconstruction (Zhang et al., 2024b; Huang et al., 2024a; Yu et al., 2024; Wolf et al., 2024; Fan et al., 2024; Zhang et al., 2024a; Charatan et al., 2023; Lin & Li, 2024; Chen et al., 2024). However, these methods require accurate camera poses, which differs from SLAM methods a lot.

SLAM with Volume Rendering. With multi-view consistency, multi-view stereo (MVS) (Schönberger & Frahm, 2016; Schönberger et al., 2016) estimate dense depth maps from multiple RGB images. With the demand for novel view synthesis, recent SLAM methods render depth maps and estimate continuous implicit representations (Zhang et al., 2023; Xinyang et al., 2023; Teigen et al., 2023; Sandström et al., 2023b; Sucar et al., 2021; Wang et al., 2023; Sandström et al., 2023a;b; Hu et al., 2023). They use RGBD images as rendering supervision, and learn implicit representations through learning radiance fields. Some other methods also leverage priors like segmentation priors (Kong et al., 2023; Haghighi et al., 2023), depth fusion priors (Hu & Han, 2023), or object-level priors (Kong et al., 2023), which improve the performance in tracking and mapping.

SLAM with 3D Gaussian Splatting. Because of the rendering efficiency and higher rendering quality, more recent methods used 3DGS to differentially render images (Keetha et al., 2024; Matsuki et al., 2024; Huang et al., 2024b; Yan et al., 2024; Yugay et al., 2023; Sandström et al., 2024). Although these methods adopt different tracking and mapping strategies, they need to maintain all Gaussians covering the scene in the limited GPU memories and optimize all Gaussians throughout the training to keep color and geometry consistency to all previous views. This fact limits the number of Gaussians that they can use and makes it hard to scale up to extremely large scenes. Unlike these methods, our method employed simplified Gaussians, without

storing locations, rotations, or multi-view variances, saving more room for more Gaussians to represent either more details or larger scenes. Our view-tied Gaussians combine the advantages of Gaussian representations in current SLAM systems (Keetha et al., 2024; Yugay et al., 2023).

More complicated systems (Liso et al., 2024; Zhu et al., 2024; Bruns et al., 2024; Sandström et al., 2024) also leverage loop closure in the optimization. However, detecting loop closures among views needs pretrained priors and is sensitive to image quality, which also differs from ours.

Gaussian Alignment. Aligning 3D Gaussians to some entities was also employed in some other works (Yinghao et al., 2024; Zhang et al., 2024a; Gao et al., 2024; Luiten et al., 2024; Seidenschwarz et al., 2024; Zakharov et al., 2024). Gaussians in these methods are either not related to camera positions or with many attributes. Instead, our view-tied Gaussians relate Gaussians' positions with the camera poses and use simplified attributes, dedicated to improving the efficiency and scalability of SLAM systems.

3. Methods

3.1. Overview

Fig. 1 illustrates our tracking and mapping strategies. We organize view-tied 3D Gaussians from several consecutive frames as a section so that we can keep as many Gaussians as the GPU memory allows to represent a local area, access these Gaussians more efficiently, and more importantly, enable more robust completion of missing depth by utilizing neighboring frames' depth. In each section, we mark the first frame as a head to differentiate it from regular frames for different Gaussian initialization strategies in mapping.

With view-tied Gaussians, we manage to keep learnable Gaussians that are the most relevant to the latest frame in the GPU memory. This ability removes the need of maintaining a list of keyframes and optimizing all Gaussians representing the scene to maintain their spatial and appearance consistency to all keyframes, which are widely employed in the latest SLAM systems (Wang et al., 2023; Hu & Han, 2023; Keetha et al., 2024; Zhu et al., 2022), leading to the key of scaling up to much larger scenes.

For tracking the latest frame, we select Gaussians in a section, render them from the camera pose initialized by the constant speed assumption, and optimize the pose by minimizing rendering errors to the latest frame. If the latest frame is the head of a new section in Fig. 1 (a), we select the Gaussians in a certain section in front according to the visibility. These selected Gaussians maintain the spatial consistency in a long view sequence and reduce the error cumulation. If the latest frame is not a head but a regular frame in the current section in Fig. 1 (c), we select the Gaussians in this section for renderings with higher quality.

For mapping the scene using the latest frame, if the latest frame is the head of a new section in Fig. 1 (b), we initialize Gaussians by centering them at all pixels with valid depth values. If the latest frame is not a head but a regular frame in an existing section in Fig. 1 (d), we only initialize Gaussians as a complement in areas where pixels have valid depth values and the existing Gaussians in the current section cannot cover. Fig. 3 visualizes the view-tied Gaussians initialized at both the head frame and the following regular frames in a section. We learn these Gaussians to maintain their color and geometry consistency to all frames in the same section and some previous frames with overlaps.

3.2. View-tied Gaussians

Our view-tied Gaussians aim to achieve memory efficiency in SLAM, which enables us to improve the rendering quality by using many more Gaussians to represent local details. Specifically, we simplify an ellipsoid Gaussian $\boldsymbol{g}$ into a sphere, which only includes a color $\boldsymbol{c} \in \mathbb{R}^{1\times 3}$, a radius as the variance $r \in \mathbb{R}^1$, and an opacity $o \in \mathbb{R}^1$, and removes the 4-dimensional rotation, the 3-dimensional location, the other 2-dimensional variances from the original 3D Gaussians, saving 64.3% (9/14) storage in total.

We remove the need of learning and storing locations by tying a Gaussian $\boldsymbol{g}$ at each pixel with a valid depth value on the depth map. We center $\boldsymbol{g}$ at the 3D location back-projected from the depth value. Thus, the positions of Gaussians are only determined by the depth and the corresponding camera pose, which can be adjusted in the camera tracking procedure. For the i-th frame with a RGB $\boldsymbol{V}_i$ and a depth map $\boldsymbol{D}_i$, we will initialize Gaussians $\{\boldsymbol{g}_j^i\}$ on $\boldsymbol{D}_i$.

Since there may be missing depth values on some depth maps, we organize Gaussians from every N neighboring frames into a section $\boldsymbol{S}_k$, so that we can use Gaussians from neighboring frames to cover the missing area on one specific depth map in the same section. Based on the concept of sections, we only optimize Gaussians in one section containing the latest view, and frozen Gaussians in other sections. This not only enables us to use more Gaussians to represent local details, but also removes the need to maintain the appearance and geometry consistency to all the previous frames or keyframes. This is a key to improving our rendering quality, leading to better performance in tracking and mapping, and scaling up to larger scenes.

3.3. Tracking Cameras

We will estimate the camera pose $\boldsymbol{p}_i$ of the latest frame $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$ first. When tracking cameras, we keep all Gaussians in the scene fixed, and merely optimize the pose $\boldsymbol{p}_i$ by minimizing rendering errors with respect to $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$.

Figure 2. Illustration of selecting overlapping section. We show Gaussian centers and colors in each section.

Figure 3. Initialization of view-tied Gaussians in a section.

If the latest frame $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$ is a head starting a new section $\boldsymbol{S}_k$, we select one section $\boldsymbol{S^o}$ in front of $\boldsymbol{S}_k$, and render Gaussians in $\boldsymbol{S^o}$ into a RGB $\boldsymbol{V}_i'$ and a depth $\boldsymbol{D}_i'$ using the pose $\boldsymbol{p}_i$. We optimize $\boldsymbol{p}_i$ to minimize rendering errors,

$$\min_{\boldsymbol{p}_i} \alpha \boldsymbol{W}_i ||\boldsymbol{V}_i - \boldsymbol{V}_i'||_1 + \beta \boldsymbol{W}_i ||\boldsymbol{D}_i - \boldsymbol{D}_i'||_1, \quad (1)$$

where $\{\boldsymbol{V}_i', \boldsymbol{D}_i'\} = splat(\{\boldsymbol{g}\}^o \in \boldsymbol{S^o}, \boldsymbol{p}_i)$ are rendered images by splatting $\{\boldsymbol{g}\}^o$ in section $\boldsymbol{S^o}$ using the camera pose $\boldsymbol{p}_i$, $\boldsymbol{W}_i$ is a mask which removes pixels either without depth values or uncovered by the rendering of $\{\boldsymbol{g}\}^o$ or invisible to sections $\boldsymbol{S^o}$, and α and β are balance parameters.

Otherwise, if the latest frame $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$ is a regular frame in the current section $\boldsymbol{S}_k$, we will optimize the camera pose $\boldsymbol{p}_i$ using the same equation above but rendering the Gaussians $\{\boldsymbol{g}\}^k$ in the current section $\boldsymbol{S}_k$ into $\{\boldsymbol{V}_i', \boldsymbol{D}_i'\}$.

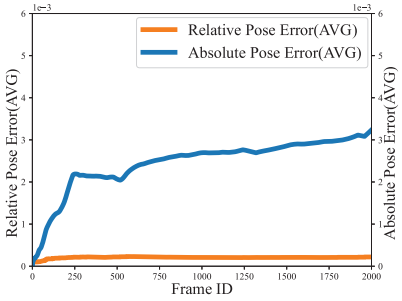

Figure 4. Issue of pose error cumulation.

This design aims to find a balance between the rendering quality and the spatial consistency of the current section to previous frames in a long image sequence. Obviously, rendering Gaussians in the same section will produce better renderings since neighboring frames usually have larger overlaps with the latest frame. Although better renderings are helpful for more accurate camera pose estimations, the higher accuracy is merely meaningful relative to the neighboring frames, resulting in a significant cumulation of pose errors relative to the whole camera trajectory. We illustrate this issue in Fig. 4, where we use Gaussians in the current section and render pretty good images for tracking. At each frame out of a 2000 frame video, the average error of relative pose to the previous frame is pretty small, while the average error relative to the whole trajectory is getting larger and larger. To resolve this issue, we maintain the spatial consistency of the current section by rendering Gaussians $\{\boldsymbol{g}\}^o$ in a certain previous section $\boldsymbol{S^o}$ for tracking the head frame while rendering Gaussians $\{\boldsymbol{g}\}^k$ in the same section $\boldsymbol{S}_k$ for tracking regular frames.

Selecting Overlapping Section $\boldsymbol{S}^o$. In each section, we select a section $\boldsymbol{S}^o$ for the head frame in terms of overlap and visibility. We first set up an overlap candidate view list with an interval of N_1 frames. For a head frame $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$, we project the depth $\boldsymbol{D}_i$ from the initialized pose to each one frame in the overlap candidate view list. We count the number of visible points to each candidate view, and select the candidate views that have more visible points than a threshold γ. We eventually select the most front 3 sections containing at least one of the selected candidate views. We use the most front as a criterion to relieve the impact caused by the error cumulation.

Then, we adopt a pre-tracking strategy to finalize the selection. We use each one of the 3 sections to conduct the tracking for several iterations, respectively. We eventually select a section that produces the minimum rendering error as $\boldsymbol{S^o}$. Here, we do not splat Gaussians from all the 3 sections together for rendering since multiple sections are seldom learned together in the mapping procedure, which may degenerate rendering quality. Fig. 2 illustrates the 3 section candidates selected for a head frame, where all the 3 sections are highly related to the head frame. We finally render Gaussians in section 73 to track the head frame.

Visibility Check. We determine the visibility of a 3D point to a specific view if its projected depth is within 1% of the interpolated depth at its projection on the depth map. For visibility to a section $\boldsymbol{S}^o$ with $\boldsymbol{W}_i$ in Eq. 1, we consider the head frame, the frame in the middle, and the last frame in $\boldsymbol{S}^o$ to calculate three visibility masks. We use the union of these three visibility masks as the visibility mask to a

section. Please see Fig. 10 for more details.

3.4. Mapping Scenes

We initialize Gaussians and learn attributes of Gaussians in the mapping procedure. In each section, we first initialize Gaussians at all pixels on the depth map at the head frame, and then complement Gaussians at pixels uncovered by the current Gaussians' renderings on regular frames. To maintain the appearance and geometry consistency to frames in both the same section and nearby overlapping sections, we adopt different strategies to learn Gaussian attributes.

If the latest frame $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$ is a head frame starting a new section $\boldsymbol{S}_k$, we use the Gaussians $\{\boldsymbol{g}\}^k$ initialized on the latest frame, the Gaussians $\{\boldsymbol{g}\}^{k-1}$ in the previous section $\boldsymbol{S}_{k-1}$, and the Gaussians $\{\boldsymbol{g}\}^o$ in the section $\boldsymbol{S}^o$ having the largest overlaps with the latest frame to render $\{\boldsymbol{V}_i', \boldsymbol{D}_i'\} = splat([\{\boldsymbol{g}\}^k, \{\boldsymbol{g}\}^{k-1}, \{\boldsymbol{g}\}^o], \boldsymbol{p}_i)$. We minimize the rendering errors with respect to observations,

$$\min_{\{\boldsymbol{g}\}^k} \rho||\boldsymbol{V}_i - \boldsymbol{V}_i'||_1 + \tau L_S(\boldsymbol{V}_i, \boldsymbol{V}_i') + \sigma \boldsymbol{U}_i||\boldsymbol{D}_i - \boldsymbol{D}_i'||_1, \quad (2)$$

where L_S is the SSIM loss, $\boldsymbol{U}_i$ is a mask which removes pixels without valid depth values, ρ, τ, σ are balance weights. Only $\{\boldsymbol{g}\}^k$ are learnable, and all other Gaussians are fixed. This rendering can maintain the appearance and geometry consistency to the nearby overlapping sections.

Otherwise, if the latest frame $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$ is a regular frame in the current section $\boldsymbol{S}_k$, we first splat the Gaussians $\{\boldsymbol{g}\}^k$ in the current section into silhouette images to find the uncovered area where we initialize Gaussians. Then, we splat $\{\boldsymbol{g}\}^k$ to render from a random view $\boldsymbol{p}_j$ of $\{\boldsymbol{V}_j, \boldsymbol{D}_j\}$ that is in front of $\{\boldsymbol{V}_i, \boldsymbol{D}_i\}$ in this section, i.e., $\{\boldsymbol{V}_j', \boldsymbol{D}_j'\} = splat(\{\boldsymbol{g}\}^k, \boldsymbol{p}_j)$, where $\{\boldsymbol{V}_j, \boldsymbol{D}_j\} \in \boldsymbol{S}_k$ and $j \leq i$. We still use Eq. 2 to optimize $\{\boldsymbol{g}\}^k$, where we select a random view as a rendering target to maintain the appearance and geometry consistency to frames in the same section. We visualize the rendered images during mapping a head frame in Fig. 5, where the rendering error is progressively minimized.

3.5. Bundle Adjustment

We also conduct bundle adjustment to jointly estimate the camera pose and learn Gaussians when mapping the head frame in each section. We still use the Eq. 2 in the optimization, but also back-propagate gradients to update the camera pose of the head frame. We do not use bundle adjustment on each regular view in a section for stabilizing the optimization in both tracking and mapping.

4. Experiments and Analysis

We only report average results in this section, evaluations on each scene can be found in the supplementary materials. Please also watch our video for more renderings.

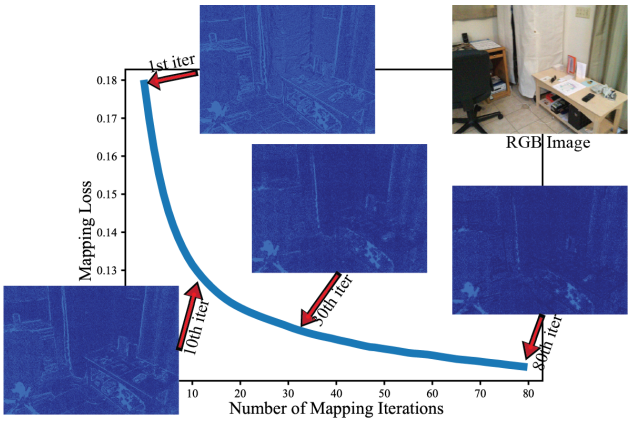

Figure 5. Illustration of optimizing view-tied Gaussians initialized on a head frame. Error maps are shown at different iterations.

Implementation Details. For neighboring views in a section $\boldsymbol{S}_k$, we choose $N = 40$ on Replica (Straub et al., 2019), $N = 30$ on TUM-RGBD (Sturm et al., 2012), $N = 30$ or $N = 50$ on ScanNet (Dai et al., 2017a), and $N = 100$ on ScanNet++ (Yeshwanth et al., 2023), according to image resolutions and mapping iterations. During tracking, we use every 5 frames as a candidate view for overlapping section selection, *i.e.* $N_1 = 5$. Specifically, when mapping a regular frame in an existing section $\boldsymbol{S}_k$, we will choose 0.5 as a mask threshold to determine if current Gaussians in $\boldsymbol{S}_k$ cover a pixel, if they do not, we will initialize a Gaussian as a complement. To work with depth and RGB images with different quality and resolutions, we set $\rho = 0.8$, $\tau = 0.2$, and $\sigma = 1.0$ to balance the mapping loss terms while we set $\{\alpha, \beta\} = \{0.5, 0.025\}$ on Replica, $\{0.5, 1.0\}$ on TUM-RGBD and ScanNet++, $\{0.5, 0.9\}$ on ScanNet to balance the tracking loss terms. More details of hyperparameters are provided in the supplementary materials.

Dataset and Metrics. We conduct evaluations on several widely used benchmarks, including Replica (Straub et al., 2019), TUM-RGBD (Sturm et al., 2012), ScanNet (Dai et al., 2017a), and ScanNet++ (Yeshwanth et al., 2023). Here Replica is a synthetic dataset with high-fidelity 3D reconstruction of indoor scenes. We evaluate on the widely used RGBD sequence from eight scenes captured by Sucar (2021) with accurate trajectories. TUM-RGBD, ScanNet, and ScanNet++ are real-world datasets. Note that ScanNet++ is not a dataset designed for SLAM tasks, some sudden large motions are occurring in the DSLR-captured sequences, we follow previous methods (Yugay et al., 2023; Zhu et al., 2024) and only employ the first 250 frames of each scene in evaluations, which present smooth trajectories.

We evaluate the accuracy of estimated cameras at each frame and the rendering quality from either the observed or unobserved view angles. For tracking accuracy, we use the root mean square absolute trajectory error (ATE RMSE) (Sturm et al., 2012) as a metric. Regarding render-

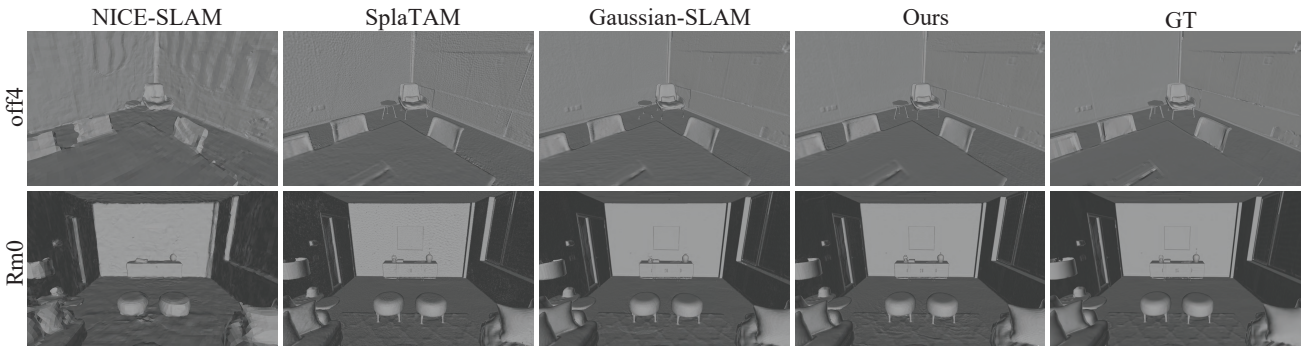

Figure 6. Visual comparisons in reconstruction on Replica.

Table 1. Tracking comparisons in ATE RMSE ↓ [cm] on Replica. ∗ denotes use of pre-trained data-driven priors.

	Neural Implicit Fields						3D Gaussian Splatting					
Methods	NICE-SLAM	DF-Prior	Vox-Fusion	ESLAM	Point-SLAM	Loopy-SLAM∗	SplaTAM	GS-SLAM	Gaussian-SLAM	LoopSplat∗	CG-SLAM∗	Ours
Avg.	1.95	1.81	0.65	0.63	0.52	0.29	0.36	0.50	0.31	0.26	0.27	**0.28**

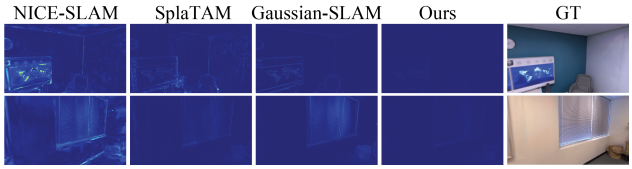

Figure 7. Error map comparisons in rendering on Replica.

ing quality, we measure PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). Similar to (Sandström et al., 2023a; Liso et al., 2024; Zhu et al., 2024; Yugay et al., 2023), all the rendering metrics are computed by rendering the full resolution images along the estimated trajectory every 5 frames. Additionally, we also obtain the meshes of scenes by marching cubes (Lorensen & Cline, 1987) following a similar procedure in (Sandström et al., 2023a). Then we measure the reconstruction performance with F1-score, the harmonic mean of the Precision (P) and Recall (R), using a distance threshold of 1 cm for all evaluations. We also use the depth L1 metric to measure the rendered mesh depth error at sampled novel views as in (Zhu et al., 2022).

Baselines. We compare our method with the latest RGBD SLAM methods, including NeRF-based RGBD SLAM methods: NICE-SLAM (Zhu et al., 2022), Vox-Fusion (Yang et al., 2022), ESLAM (Johari et al., 2023), DF-Prior (Hu & Han, 2023), Co-SLAM (Wang et al., 2023), and Point-SLAM (Sandström et al., 2023a), and 3DGS-based RGBD SLAM methods: SplaTAM (Keetha et al., 2024), MonoGS (Matsuki et al., 2024), GS-SLAM (Yan et al., 2024), and Gaussian-SLAM (Yugay et al., 2023). Note that Point-SLAM (Sandström et al., 2023a) requires ground truth depth images as an input to guide sampling when rendering, which is an unfair advantage compared to other methods. Moreover, relying on data-driven priors, such as pre-trained NetVLAD models (Arandjelović et al., 2016), in loop closure detection and visibility check, SLAM methods with pose graph optimizations Loopy-SLAM (Liso et al., 2024) and LoopSplat (Zhu et al., 2024), and CG-SLAM (Hu et al., 2024) usually reported higher tracking accuracy, which however is not a fair experimental setting to most SLAM methods without using priors.

Table 2. Rendering results in PSNR ↑, SSIM ↑, and LPIPS ↓ on three Datasets. ∗ denotes use of pre-trained data-driven priors.

Dataset	***Replica***			***TUM***			***ScanNet***		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Neural Implicit Fields									
NICE-SLAM	24.42	0.809	0.233	14.86	0.614	0.441	17.54	0.621	0.548
Vox-Fusion	24.41	0.801	0.236	16.46	0.677	0.471	18.17	0.673	0.504
ESLAM	28.06	0.923	0.245	15.26	0.478	0.569	15.29	0.658	0.488
Point-SLAM	35.17	0.975	0.124	16.62	0.696	0.526	19.82	0.751	0.514
Loopy-SLAM∗	35.47	0.981	0.109	12.94	0.489	0.645	15.23	0.629	0.671
3D Gaussian Splatting									
SplaTAM	34.11	0.970	0.100	22.80	0.893	0.178	19.14	0.716	0.358
Gaussian-SLAM	42.08	**0.996**	0.018	25.05	0.929	0.168	27.70	0.923	0.248
LoopSplat∗	36.63	0.985	0.112	22.72	0.873	0.259	24.92	0.845	0.425
Ours	**43.34**	**0.996**	**0.012**	**30.20**	**0.972**	**0.062**	**31.10**	**0.961**	**0.108**

4.1. Comparisons

Results on Replica. We first report our results on 8 scenes in Replica. We compare with the state-of-the-art NeRF-based and GS-based SLAM methods in camera tracking in Tab. 1, mapping scenes with rendered images in Tab. 2, and reconstruction in Tab. 3.

Tab. 1 shows that our method can estimate camera poses more accurately than state-of-the-art NeRF-based methods, such as NICE-SLAM (Zhu et al., 2022), DF-Prior (Hu & Han, 2023), and Point-SLAM (Sandström et al., 2023a), due to higher quality renderings produced by view-tied Gaussians. Our view-tied Gaussians also show advantages over the original Gaussians used in SplaTAM (Keetha et al., 2024), Gaussian-SLAM (Yugay et al., 2023), and GS-SLAM (Yan et al., 2024) in terms of using much more

Table 3. Reconstruction results in D-L1 [cm] ↓ and F1 [%] ↑ on Replica. ∗ denotes use of pre-trained data-driven priors.

	Neural Implicit Fields						3D Gaussian Splatting				
	NICE-SLAM	Vox-Fusion	ESLAM	Co-SLAM	Point-SLAM	Loopy-SLAM∗	SplaTAM	GS-SLAM	Gaussian-SLAM	LoopSplat∗	Ours
D-L1↓	2.97	2.46	1.18	2.59	**0.44**	0.35	0.72	1.16	0.68	0.51	0.53
F1↑	43.9	52.2	79.1	69.7	89.8	90.8	86.1	70.2	88.9	90.4	**90.0**

Table 4. Tracking comparisons in ATE RMSE ↓ [cm] on TUM-RGBD. ∗ denotes use of pre-trained data-driven priors.

	Neural Implicit Fields				3D Gaussian Splatting					
Methods	NICE-SLAM	Vox-Fusion	Point-SLAM	Loopy-SLAM∗	SplaTAM	GS-SLAM	Gaussian-SLAM	LoopSplat∗	CG-SLAM∗	Ours
Avg.	13.3	10.3	3.0	2.9	3.3	3.7	2.9	2.3	2.0	**2.6**

Figure 8. Visual comparisons in rendering on TUM-RGBD.

Gaussians to represent local details in a more efficient manner, leading to better renderings to compare with the observations during tracking. However, relying on data-driven priors, LoopSplat (Zhu et al., 2024) reported more accurate camera tracking in terms of average accuracy, while our method does not need any priors.

Due to the ability of using more Gaussians to describe local details, our method produces the best rendered images, as shown in Tab. 2. Error map comparisons in Fig. 7 highlight our rendering quality, which is much better than others, especially in areas with sudden color changes. Because of better renderings, our methods also produce the most accurate reconstruction in Tab. 3. We follow the previous method (Sandström et al., 2023a) to render depth maps from the estimated camera poses, and fuse these depth maps into a TSDF for reconstruction. Visual comparisons in Fig. 6 show that we can recover more accurate geometry details.

Results on TUM-RGBD. We report our results on the TUM-RGBD dataset in camera tracking in Tab. 4 and rendering in Tab. 2. We follow previous methods (Zhu et al., 2022; Keetha et al., 2024; Yugay et al., 2023; Liso et al., 2024; Zhu et al., 2024; Sandström et al., 2023a) and evaluate our method on the 3 widely used scenes in TUM-RGBD.

Comparisons in camera tracking in Tab. 4 show that GS-based methods estimate camera poses more accurately than NeRF-based methods. Although the input RGBD observations are not in high resolution and with good quality, our method still produces the best tracking accuracy. Regarding the rendering, our view-tied Gaussians show even more advantages over the other methods in Tab. 2. Visual comparisons in Fig. 8 highlight our improvement over the other methods. Compared to previous GS-based SLAM methods, our method can use many more Gaussians tied at each pixel on depth images to fit sudden color change without needing to maintain the consistency of Gaussians over the whole scene to a set of keyframes throughout the training, which enables Gaussians to focus more on local details.

Results on ScanNet. Our evaluations in camera tracking and mapping scenes with rendering views are reported in Tab. 5 and Tab. 2, respectively. We produce the most accurate tracking performance in terms of average performance. Based on the camera poses, our method also significantly improves the rendering quality on ScanNet, as shown in Fig. 9. The rendering improvement also justifies our advantages of using view-tied Gaussians on real-captured scenes. Besides good ability of recovering appearance details, with motion blur and low image quality in real images, our view-tied Gaussians can also limit these negative impact just on several neighboring views but not on all Gaussians in the scene during mapping.

Figure 9. Visual comparisons in rendering on ScanNet.

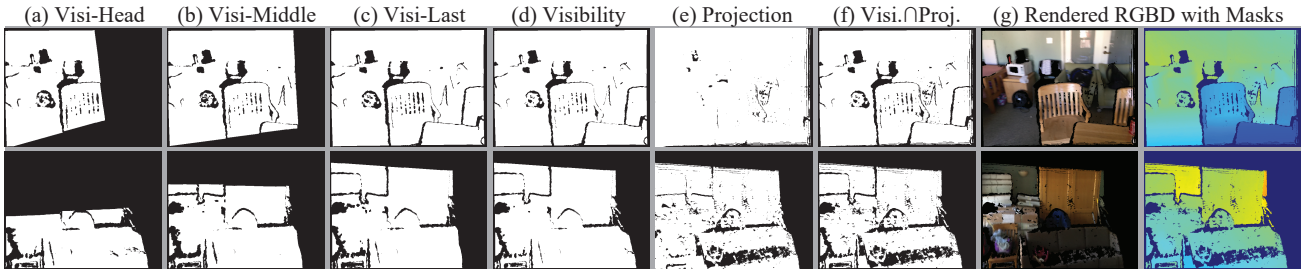

Figure 10. Visualization of visibility and Gaussian projections on a head frame during tracking in a section.

Figure 11. Visualization of Gaussian centers with colors in the selected overlapping section for tracking (a) head frames and (b) regular frames. The Gaussian centers nearby are shown without color.

Table 5. Tracking comparisons in ATE RMSE ↓ [cm] on ScanNet. ∗ denotes use of pre-trained data-driven priors.

	Neural Implicit Fields				3D Gaussian Splatting				
Methods	NICE-SLAM	Vox-Fusion	Point-SLAM	Loopy-SLAM∗	SplaTAM	Gaussian-SLAM	LoopSplat∗	CG-SLAM∗	Ours
Avg.	**10.7**	26.9	12.2	7.7	11.9	15.4	7.7	8.1	11.3

Table 6. Tracking comparisons in ATE RMSE↓ [cm] on ScanNet++. ∗ denotes use of pre-trained data-driven priors.

	Neural Implicit Fields			3D Gaussian Splatting			
Methods	Point-SLAM	ESLAM	Loopy-SLAM∗	SplaTAM	Gaussian-SLAM	LoopSplat∗	Ours
Avg.	511.24	22.14	113.63	89.41	2.68	2.05	**1.55**

Results on ScanNet++. We report tracking results on the widely used 5 scenes in ScanNet++ in Tab. 6. Compared to GS-based methods, our methods can estimate more accurate camera poses thanks to the more accurate renderings. Regarding novel view synthesis, please refer to our supplementary materials for more details.

4.2. Ablation Studies and Analysis

We justify the effectiveness of each design on synthetic and real scenes in Replica (Straub et al., 2019) and TUM-RGBD (Sturm et al., 2012).

3D Gaussians. We conduct experiments to highlight the effect of view-tied Gaussians in Tab. 7. We replace our Gaussians with ellipsoid Gaussians with either fixed ("aniso + w/ VT") or learnable locations ("aniso + w/o VT"). We also show the effect of learnable locations with our simplified Gaussians ("iso + w/o VT"). Without tying Gaussians to depth maps, we need to store Gaussian locations, which limits the number of Gaussians we can use, degenerating the rendering quality. The comparisons show that our view-tied Gaussians not only significantly reduce the size of each Gaussian (number of parameters) but also achieve good rendering quality with our tracking and mapping strategies.

Section Length. The number of frames in a section is also a factor impacting the performance. Tab. 8 shows that too few or too many frames in one section will degenerate the performance if we do not adjust other parameters like optimization iterations during tracking or mapping. Too few frames will increase the possibility of cumulating camera pose errors while changing into the next section. Instead, too many frames will need more iterations during mapping to learn Gaussians well. We cannot use a large number of Gaussians

Table 7. Ablation study on attributes of 3D Gaussians (aniso: anisotropic Gaussians, iso: isotropic Gaussians, VT: view-tied Gaussians).

Metric	aniso + w/o VT	aniso+ w/ VT	iso + w/o VT	iso + w/ VT (Ours)
PSNR↑ [dB]	37.62	38.46	36.73	**39.95**
ATE RMSE ↓ [cm]	19.60	25.65	21.45	**0.22**
Param./Gaussian	14	11	8	**5**

Table 8. Ablation study on the length of section $\boldsymbol{S}$, overlap selecting strategy, and visible mask.

	Length of section $\boldsymbol{S}$					Overlap section selecting strategy				Visibility mask	
Metric	20	40 (Ours)	60	80	100	nearest	largest	multiple	Ours	w/o	w/ (Ours)
PSNR↑ [dB]	39.92	**39.95**	39.34	38.87	38.56	38.52	38.74	38.03	**39.95**	30.40	**30.51**
ATE RMSE ↓ [cm]	0.25	**0.22**	0.23	0.26	0.24	0.30	0.34	0.28	**0.22**	6.8	**4.4**

Table 9. Runtime and Memory Usage on Replica.

Method	NICE-SLAM	Point-SLAM	SplaTAM	Gaussian-SLAM	Ours
Tracking/Frame(s)	1.06	1.11	2.70	0.83	1.92
Mapping/Frame(s)	1.15	3.52	4.89	0.93	2.43
Total Num of Gaussians	-	-	$5832K$	$32592K$	$97823K$
Max Num of Gaussians	-	-	$5832K$	$1983K$	$2664K$

if setting the length of the section as 1, which degenerates the rendering and tracking performance. We visualize some overlapping sections during tracking in Fig. 11.

Overlapping Section Selection Strategy. Our strategy of selecting overlapping sections is also important for good renderings in tracking the head frame. Our selection based on visibility and preference to the most front sections significantly reduces the impact of pose error cumulation. We compare this strategy with selecting the nearest section ("nearest"), the section with the largest overlaps ("largest"), or multiple sections ("multiple") that add the nearest section to the selection we selected. The comparisons in Tab. 8 show that our selection strategy achieves the best performance.

Visibility. We consider visibility in both tracking and mapping to reduce the error brought by unseen or occluded areas. Using no visibility in the loss function will degenerate the performance, as shown in Tab. 8, since the error in the unseen area will not be minimized by adjusting camera poses or optimizing Gaussian attributes. We visualize the visibility masks and projection masks in Fig. 10. The visibility masks of a head frame to the first, the middle, and the last frame in the selected overlapping section are shown in Fig. 10 (a)-(c), respectively. The overall visibility to the overlapping section is the union of these 3 visibility masks in Fig. 10 (d). The silhouette produced by projections of Gaussians in the same section is shown in Fig. 10 (e). We use the intersection of visibility mask and silhouette mask in Fig. 10 (f) to weight rendered RGBD images in Fig. 10 (g).

Runtime, Storage, and Scalability. We report the average time for tracking and mapping one frame in Tab. 9 on a single NVIDIA RTX4090. Our runtime is comparable to other GS-based methods. Regarding the number of Gaussians, we show a great advantage over other methods. We can initiate many more Gaussians over the whole scene than other GS-based methods for better rendering. Meanwhile, around each frame, we still have good control of the number of Gaussians so that we can maximize the usage of the limited GPU memory, which finds a good balance between scalability and limited hardware resources. This enables us to handle much larger scenes in more frames with more Gaussians than other SLAM methods.

5. Conclusion

We propose VTGaussian-SLAM to improve the performance of SLAM with 3D Gaussian splatting in terms of rendering quality, tracking accuracy, and scalability. We showed that our view-tied Gaussians can significantly save storage so that we can maintain a large amount of these Gaussians in the limited GPU memory for either higher rendering quality or larger areas. Our tracking and mapping strategies take good advantage of these benefits, which allows us to merely maintain and optimize Gaussians that contribute to the most recent views a lot. This ensures the high rendering quality for the latest views, and leads to more accurate camera tracking and mapping. We justified the effectiveness of each design and reported visual and numerical evaluations to illustrate our advantages over the latest SLAM methods.

Acknowledgements

This project was partially supported by an NVIDIA academic award and a Richard Barber research award.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Adamkiewicz, M., Chen, T., Caccavale, A., Gardner, R., Culbertson, P., Bohg, J., and Schwager, M. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2):4606–4613, 2022.

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Azinović, D., Martin-Brualla, R., Goldman, D. B., Nießner, M., and Thies, J. Neural rgb-d surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6290–6301, 2022.

Bozic, A., Palafox, P., Thies, J., Dai, A., and Niessner, M. Transformerfusion: Monocular rgb scene reconstruction using transformers. *Advances in Neural Information Processing Systems*, 2021.

Bruns, L., Zhang, J., and Jensfelt, P. Neural graph mapping for dense slam with efficient loop closure. *arXiv preprint arXiv:2405.03633*, 2024.

Charatan, D., Li, S., Tagliasacchi, A., and Sitzmann, V. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *arXiv*, 2023.

Chen, D., Li, H., Ye, W., Wang, Y., Xie, W., Zhai, S., Wang, N., Liu, H., Bao, H., and Zhang, G. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction, 2024. URL https://arxiv.org/abs/2406.06521.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. A., and Nießner, M. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017a.

Dai, A., Nießner, M., Zollöfer, M., Izadi, S., and Theobalt, C. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics*, 2017b.

Fan, L., Yang, Y., Li, M., Li, H., and Zhang, Z. Trim 3d gaussian splatting for accurate geometry representation. *arXiv preprint arXiv:2406.07499*, 2024.

Gao, L., Yang, J., Zhang, B.-T., Sun, J.-M., Yuan, Y.-J., Fu, H., and Lai, Y.-K. Mesh-based gaussian splatting for real-time large-scale deformation, 2024. URL https://arxiv.org/abs/2402.04796.

Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition*, 2012.

Guo, H., Peng, S., Lin, H., Wang, Q., Zhang, G., Bao, H., and Zhou, X. Neural 3d scene reconstruction with the manhattan-world assumption. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

Haghighi, Y., Kumar, S., Thiran, J.-P., and Gool, L. V. Neural implicit dense semantic slam, 2023.

Hu, J., Mao, M., Bao, H., Zhang, G., and Cui, Z. Cp-slam: Collaborative neural point-based slam system. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Hu, J., Chen, X., Feng, B., Li, G., Yang, L., Bao, H., Zhang, G., and Cui, Z. Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field. *arXiv preprint arXiv:2403.16095*, 2024.

Hu, P. and Han, Z. Learning neural implicit through volume rendering with attentive depth fusion priors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Huang, B., Yu, Z., Chen, A., Geiger, A., and Gao, S. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024a. doi: 10.1145/3641519.3657428.

Huang, H., Li, L., Hui, C., and Yeung, S.-K. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024b.

Johari, M. M., Carta, C., and Fleuret, F. ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Keetha, N., Karhade, J., Jatavallabhula, K. M., Yang, G., Scherer, S., Ramanan, D., and Luiten, J. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

Kong, X., Liu, S., Taher, M., and Davison, A. J. vmap: Vectorised object mapping for neural field slam. *arXiv preprint arXiv:2302.01838*, 2023.

Li, M., Liu, S., Zhou, H., Zhu, G., Cheng, N., Deng, T., and Wang, H. Sgs-slam: Semantic gaussian splatting for neural dense slam, 2024. URL https://arxiv.org/abs/2402.03246.

Li, Z., Lyu, X., Ding, Y., Wang, M., Liao, Y., and Liu, Y. Rico: Regularizing the unobservable for indoor compositional reconstruction, 2023.

Lin, A. and Li, J. Direct learning of mesh and appearance via 3d gaussian splatting, 2024. URL https://arxiv.org/abs/2405.06945.

Liso, L., Sandström, E., Yugay, V., Van Gool, L., and Oswald, M. R. Loopy-slam: Dense neural slam with loop closures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20363–20373, 2024.

Lorensen, W. E. and Cline, H. E. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

Luiten, J., Kopanas, G., Leibe, B., and Ramanan, D. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024.

Matsuki, H., Murai, R., Kelly, P. H. J., and Davison, A. J. Gaussian Splatting SLAM. 2024.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 2020.

Moenne-Loccoz, N., Mirzaei, A., Perel, O., de Lutio, R., Esturo, J. M., State, G., Fidler, S., Sharp, N., and Gojcic, Z. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics and SIGGRAPH Asia*, 2024.

Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv:2201.05989*, 2022.

Park, J., Zhou, Q.-Y., and Koltun, V. Colored point cloud registration revisited. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 143–152, 2017. doi: 10.1109/ICCV.2017.25.

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R. Nerfies: Deformable neural radiance fields. *IEEE International Conference on Computer Vision*, 2021.

Patel, A., Laga, H., and Sharma, O. Normal-guided detail-preserving neural implicit functions for high-fidelity 3d surface reconstruction, 2024. URL https://arxiv.org/abs/2406.04861.

Rückert, D., Franke, L., and Stamminger, M. Adop: Approximate differentiable one-pixel point rendering. *arXiv:2110.06635*, 2021.

Sandström, E., Tateno, K., Oechsle, M., Niemeyer, M., Van Gool, L., Oswald, M. R., and Tombari, F. Splat-slam: Globally optimized rgb-only slam with 3d gaussians. *arXiv preprint arXiv:2405.16544*, 2024.

Sandström, E., Li, Y., Van Gool, L., and R. Oswald, M. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023a.

Sandström, E., Ta, K., Gool, L. V., and Oswald, M. R. Uncle-slam: Uncertainty learning for dense neural slam, 2023b.

Sara Fridovich-Keil and Alex Yu, Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. Plenoxels: Radiance fields without neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

Schönberger, J. L. and Frahm, J.-M. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, 2016.

Seidenschwarz, J., Zhou, Q., Duisterhof, B., Ramanan, D., and Leal-Taixé, L. Dynomo: Online point tracking by dynamic online monocular gaussian reconstruction, 2024. URL https://arxiv.org/abs/2409.02104.

Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H. M., Nardi, R. D., Goesele, M., Lovegrove, S., and Newcombe, R. A. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. A benchmark for the evaluation of rgb-d slam

systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, 2012. doi: 10.1109/IROS.2012.6385773.

Sucar, E., Liu, S., Ortiz, J., and Davison, A. J. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6229–6238, 2021.

Sun, J., Xie, Y., Chen, L., Zhou, X., and Bao, H. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Teigen, A. L., Park, Y., Stahl, A., and Mester, R. Rgb-d mapping and tracking in a plenoxel radiance field, 2023.

Wang, H., Wang, J., and Agapito, L. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam, 2023.

Wang, J., Bleja, T., and Agapito, L. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. In *International Conference on 3D Vision*, 2022a.

Wang, J., Wang, P., Long, X., Theobalt, C., Komura, T., Liu, L., and Wang, W. NeuRIS: Neural reconstruction of indoor scenes using normal priors. In *European Conference on Computer Vision*, 2022b.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Wei, J. and Leutenegger, S. Gsfusion: Online rgb-d mapping where gaussian splatting meets tsdf fusion, 2024. URL https://arxiv.org/abs/2408.12677.

Wolf, Y., Bracha, A., and Kimmel, R. Gs2mesh: Surface reconstruction from gaussian splatting via novel stereo views. *arXiv preprint arXiv:2404.01810*, 2024.

Xinyang, L., Yijin, L., Yanbin, T., Hujun, B., Guofeng, Z., Yinda, Z., and Zhaopeng, C. Multi-modal neural radiance field for monocular dense slam with a light-weight tof sensor. In *International Conference on Computer Vision (ICCV)*, 2023.

Yan, C., Qu, D., Xu, D., Zhao, B., Wang, Z., Wang, D., and Li, X. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, 2024.

Yang, X., Li, H., Zhai, H., Ming, Y., Liu, Y., and Zhang, G. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Dec 2022. doi: 10.1109/ISMAR55827.2022.00066. URL http://dx.doi.org/10.1109/ismar55827.2022.00066.

Yeshwanth, C., Liu, Y.-C., Nießner, M., and Dai, A. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023.

Yinghao, X., Zifan, S., Wang, Y., Hansheng, C., Ceyuan, Y., Sida, P., Yujun, S., and Gordon, W. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation, 2024.

Yu, Z., Peng, S., Niemeyer, M., Sattler, T., and Geiger, A. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. *ArXiv*, abs/2022.00665, 2022.

Yu, Z., Sattler, T., and Geiger, A. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024.

Yugay, V., Li, Y., Gevers, T., and Oswald, M. R. Gaussian-slam: Photo-realistic dense slam with gaussian splatting, 2023.

Zakharov, E., Sklyarova, V., Black, M. J., Nam, G., Thies, J., and Hilliges, O. Human hair reconstruction with strand-aligned 3d gaussians. *ArXiv*, Sep 2024.

Zhang, K., Bi, S., Tan, H., Xiangli, Y., Zhao, N., Sunkavalli, K., and Xu, Z. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *European Conference on Computer Vision*, 2024a.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

Zhang, W., Liu, Y.-S., and Han, Z. Neural signed distance function inference through splatting 3d gaussians pulled on zero-level set. In *Advances in Neural Information Processing Systems*, 2024b.

Zhang, W., Shi, K., Liu, Y.-S., and Han, Z. Learning unsigned distance functions from multi-view images with volume rendering priors. *European Conference on Computer Vision*, 2024c.

Zhang, Y., Tosi, F., Mattoccia, S., and Poggi, M. Go-slam: Global optimization for consistent 3d instant reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

Zhu, L., Li, Y., Sandström, E., Huang, S., Schindler, K., and Armeni, I. Loopsplat: Loop closure by registering 3d gaussian splats, 2024.

Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M. R., and Pollefeys, M. Nice-slam: Neural implicit scalable encoding for slam. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

Zou, Z., Huang, S., Cao, Y., Mu, T., Shan, Y., and Fu, H. Mononeuralfusion: Online monocular neural 3d reconstruction with geometric priors. *CoRR*, abs/2209.15153, 2022.

Supplementary Material

In this supplementary material, we will cover more details about the implementation and results on each scene. Additionally, we also show more results regarding Novel View Synthesis on ScanNet++ (Yeshwanth et al., 2023).

A. Implementation Details

We implemented VTGaussian-SLAM in Python using the PyTorch framework, and ran all experiments on NVIDIA RTX4090 GPUs. During mapping, we will initialize view-tied Gaussians from input RGBD images. Following (Keetha et al., 2024), we initialized the radius of each Gaussian r using the following equation:

$$r = \frac{\boldsymbol{D_{GT}}}{f}, \tag{3}$$

Here $\boldsymbol{D_{GT}}$ is the ground-truth depth, and f is the focal length. As for the learning rate of 3D Gaussians, we set $lr_{color} = 0.0025$ for the color, $lr_{radius} = 0.005$ for the radius, and $lr_{opacity} = 0.05$ for the opacity separately. For camera tracking, we initialize the pose using the constant speed assumption on Replica (Straub et al., 2019), TUM-RGBD (Sturm et al., 2012), and ScanNet (Dai et al., 2017a). Specifically, on ScanNet++ (Yeshwanth et al., 2023), since it is not a dataset originally designed for SLAM tasks, some sudden large motion changes are occurring in the DSLR-captured sequences, we follow previous methods (Zhu et al., 2024; Yugay et al., 2023) to utilizing multi-scale RGBD odometry (Park et al., 2017) to help the pose initialization if the rendering error with the pose initialized by constant speed assumption is 50 times larger than the average of the rendering loss for previous frames after the tracking optimization. We set the learning rate of pose to $lr_{rot} = 0.0004$ and $lr_{trans} = 0.002$ on Replica, $lr_{rot} = 0.002$ and $lr_{trans} = 0.002$ on TUM-RGBD and ScanNet, and $lr_{rot} = 0.001$ and $lr_{trans} = 0.01$ on ScanNet++ separately. Additionally, we set the overlapping threshold $\gamma = 0.26$ on TUM-RGBD, and $\gamma = 0.24$ on ScanNet and ScanNet++ separately.

B. More Results

Per-scene Results. We present more detailed results on each scene in Replica (Straub et al., 2019), TUM-RGBD (Sturm et al., 2012), ScanNet (Dai et al., 2017a), and ScanNet++ (Yeshwanth et al., 2023). Replica is a synthetic dataset, whereas TUM-RGBD, ScanNet, and ScanNet++ are real-world datasets. TUM-RGBD were captured using an external motion capture system, while ScanNet uses poses from BundleFusion (Dai et al., 2017b), and ScanNet++ utilizes a laser scan to register the images for acquiring corresponding camera poses. We follow previous methods (Keetha et al., 2024; Yugay et al., 2023; Matsuki et al., 2024; Zhu et al., 2024; Wei & Leutenegger, 2024; Li et al., 2024) and conduct experiments on three scenes of TUM-RGBD, six scenes of ScanNet, and five scenes of ScanNet++ ((a) `b20a261fdf`, (b) `8b5caf3398`, (c) `fb05e13ad1`, (d) `2e74812d00`, (e) `281bc17764`) to evaluate our performance.

We report numerical comparisons in camera tracking in each scene in Replica (Straub et al., 2019) in Tab. 10, in TUM-RGBD (Sturm et al., 2012) in Tab. 11, in ScanNet (Dai et al., 2017a) in Tab. 13, and in ScanNet++ (Yeshwanth et al., 2023) in Tab. 15. The comparisons show that our methods can estimate more accurate camera poses in most scenes.

Moreover, we report reconstruction comparisons in each scene in Replica (Straub et al., 2019) in Tab. 19. We utilize depth L1 and F1-score as metrics to evaluate the mesh obtained by marching cubes (Lorensen & Cline, 1987) following a similar procedure in (Sandström et al., 2023a). Compared to the latest methods, our methods can recover more accurate geometry, although Point-SLAM (Sandström et al., 2023a) requires ground truth depth images as input to guide sampling when rendering, which is an unfair advantage compared to other methods. Specifically, we also show visual comparisons regarding camera tracking and reconstruction in Fig. 12. Please refer to our supplementary video for more details of this comparison.

We also report comparisons in rendering in each scene from the training views in Replica (Straub et al., 2019), TUM-RGBD (Sturm et al., 2012), ScanNet (Dai et al., 2017a), and ScanNet++ (Yeshwanth et al., 2023) separately. Due to our view-tied strategy, there will be more 3D Gaussians used for rendering, which leads to better rendering quality compared to the latest methods as shown in Tab. 18, Tab. 12, Tab. 14, and Tab. 16.

Large-scale scenes Results. We additionally report our performance on extremely large scenes, such as city-level scenes in KITTI (Geiger et al., 2012). Since many moving objects exist in KITTI sequences, we only select part of the sequences to evaluate our tracking and rendering performance in Tab. 20 and Tab. 21. The comparisons show that our methods can estimate more accurate camera poses with high quality rendering performance. Meanwhile, we report memory consumption on KITTI in Tab. 21. Each method uses the most Gaussians until no improvement can be made. We use a little bit more memory, but we manage to use more Gaussians to produce much better rendering.

C. Novel View Synthesis

We evaluate the performance in novel view synthesis on ScanNet++ (Yeshwanth et al., 2023). The testing views in ScanNet++ are from held-out views, which are much dif-

ferent from the training views. To evaluate PSNR on all test views, we use a post-processing step after training is finished. In post-processing, we use all training views to refine all trained sections $\{\boldsymbol{S}_k\}$. Specifically, for each training view, we find all its overlapping sections and refine all these sections using this view. Compared to the latest methods (Zhu et al., 2024; Yugay et al., 2023), which take $10K$ iterations to refine their global map, our refinement only needs $1K$ iterations for novel view synthesis. In testing, given a novel view, we will find all overlapping sections $\boldsymbol{S}_k$, and concatenate all these sections to render the novel view image. Tab. 17 shows that our methods can obtain comparable novel view synthesis results to the latest methods. We also show qualitative rendering results of the training views in Fig. 13 and novel view synthesis in Fig. 14.

D. More analysis and visualization

D.1. Noise depth

Our view-tied Gaussians can also resist the impact brought by noise in depth. Although Gaussians are fixed at depth with noise, Gaussian splatting is flexible enough to overfit the current frame and neighboring frames by tuning other attributes like color, opacity, and shape. Our results show that depth noises do not significantly impact the rendering performance. Meanwhile, we try to optimize the position of Gaussians along the ray direction, but we do not find an obvious improvement in rendering performance. We report additional results in Tab. 22. We also report a visual comparison using either fixed Gaussians or movable Gaussians (along the ray) in Fig. 15.

D.2. Issue of pose error cumulation

Here we present the effectiveness of our tracking strategy. As shown by average pose accuracy (ATE RMSE) in Fig. 16 (a) and (b), using overlapping selection for rendering in tracking can prevent the estimated camera pose from drifting away from the trajectory caused by the error cumulation. The absolute pose error at each frame is shown in Fig. 16 (c) and (d). Since our method can render pretty good images, we produce small pose errors relative to the previous view in Fig. 16 (e) and (f).

D.3. Visualization of 3D Gaussians

Fig. 18 visualizes optimized 3D Gaussians. Our methods can initialize much denser 3D Gaussians to represent the whole scene due to the view-tied strategy. With more 3D Gaussians, our methods can render more realistic images.

We also report a comparison of different kinds of Gaussians in Fig. 17. We employ the same number of Gaussians, but highlight the performance of our Gaussians in rendering with different specifications, such as using ellipsoid Gaussian ("Aniso.") and learnable Gaussian locations ("w/o VT"). The comparison shows that our Gaussians can produce the minimum rendering errors.

Many more advantages can be shown with our estimated camera poses during mapping in Fig. 19. Our method can recover more details of the scene and produce more accurate renderings.

E. Code

Please see our project page for code at https://machineperceptionlab.github.io/VTGaussian-SLAM-Project.

F. Video

We present more visualization in our video, such as visual comparisons and visualization of the optimization. Please watch our video for more details.

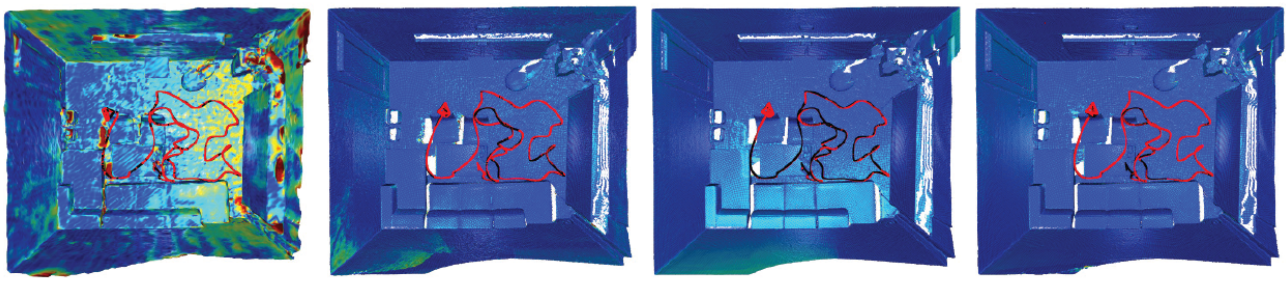

Figure 12. Visual comparisons in camera tracking on Replica. We also show error maps on reconstructions. Please refer to our video for a more complete comparison during scanning.

Table 10. Tracking performance comparisons in ATE RMSE ↓ [cm] on Replica (Straub et al., 2019). ∗ indicates methods relying on pre-trained data-driven priors.

Method	Rm0	Rm1	Rm2	Off0	Off1	Off2	Off3	Off4	Avg.
Neural Implicit Fields									
NICE-SLAM (Zhu et al., 2022)	1.69	2.04	1.55	0.99	0.90	1.39	3.97	3.08	1.95
DF-Prior (Hu & Han, 2023)	1.39	1.55	2.60	1.09	1.23	1.61	3.61	1.42	1.81
Vox-Fusion (Yang et al., 2022)	**0.27**	1.33	0.47	0.70	1.11	**0.46**	**0.26**	**0.58**	0.65
ESLAM (Johari et al., 2023)	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63	0.63
Point-SLAM (Sandström et al., 2023a)	0.61	**0.41**	**0.37**	**0.38**	**0.48**	0.54	0.72	0.63	**0.52**
Loopy-SLAM∗ (Liso et al., 2024)	0.24	0.24	0.28	0.26	0.40	0.29	0.22	0.35	0.29
3D Gaussian Splatting									
SplaTAM (Keetha et al., 2024)	0.31	0.40	0.29	0.47	0.27	**0.29**	0.32	0.55	0.36
GS-SLAM (Yan et al., 2024)	0.48	0.53	0.33	0.52	0.41	0.59	0.46	0.70	0.50
Gaussian-SLAM (Yugay et al., 2023)	0.29	0.29	0.22	0.37	**0.23**	0.41	0.30	**0.35**	0.31
LoopSplat∗ (Zhu et al., 2024)	0.28	0.22	0.17	0.22	0.16	0.49	0.20	0.30	0.26
CG-SLAM∗ (Hu et al., 2024)	0.29	0.27	0.25	0.33	0.14	0.28	0.31	0.29	0.27
Ours	**0.22**	**0.26**	**0.19**	**0.28**	0.26	0.34	**0.25**	0.43	**0.28**

Table 11. Tracking performance comparisons in ATE RMSE ↓ [cm] on TUM-RGBD (Sturm et al., 2012). ∗ indicates methods relying on pre-trained data-driven priors.

Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Neural Implicit Fields				
NICE-SLAM (Zhu et al., 2022)	4.3	31.7	3.9	13.3
Vox-Fusion (Yang et al., 2022)	**3.5**	1.5	26.0	10.3
Point-SLAM (Sandström et al., 2023a)	4.3	**1.3**	**3.5**	**3.0**
Loopy-SLAM∗ (Liso et al., 2024)	3.8	1.6	3.4	2.9
3D Gaussian Splatting				
SplaTAM (Keetha et al., 2024)	3.4	1.2	5.2	3.3
GS-SLAM (Yan et al., 2024)	3.3	1.3	6.6	3.7
Gaussian-SLAM (Yugay et al., 2023)	2.6	1.3	4.6	2.9
LoopSplat∗ (Zhu et al., 2024)	2.1	1.6	3.2	2.3
CG-SLAM∗ (Hu et al., 2024)	2.4	1.2	2.5	2.0
Ours	**2.4**	**1.1**	**4.4**	**2.6**

Table 12. Rendering performance comparison in PSNR ↑, SSIM ↑, and LPIPS ↓ on TUM-RGBD (Sturm et al., 2012). ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	fr1/desk	fr2/xyz	fr3/office	Avg.
Neural Implicit Fields					
NICE-SLAM (Zhu et al., 2022)	PSNR↑	13.83	**17.87**	12.89	14.86
	SSIM↑	0.569	**0.718**	0.554	0.614
	LPIPS↓	0.482	**0.344**	0.498	**0.441**
Vox-Fusion (Yang et al., 2022)	PSNR↑	**15.79**	16.32	17.27	16.46
	SSIM↑	0.647	0.706	0.677	0.677
	LPIPS↓	0.523	0.433	0.456	0.471
ESLAM (Johari et al., 2023)	PSNR↑	11.29	17.46	17.02	15.26
	SSIM↑	**0.666**	0.310	0.457	0.478
	LPIPS↓	**0.358**	0.698	0.652	0.569
Point-SLAM (Sandström et al., 2023a)	PSNR↑	13.87	17.56	**18.43**	**16.62**
	SSIM↑	0.627	0.708	**0.754**	**0.696**
	LPIPS↓	0.544	0.585	**0.448**	0.526
Loopy-SLAM∗ (Liso et al., 2024)	PSNR↑	-	-	-	12.94
	SSIM↑	-	-	-	0.489
	LPIPS↓	-	-	-	0.645
3D Gaussian Splatting					
SplaTAM (Keetha et al., 2024)	PSNR↑	22.00	24.50	21.90	22.80
	SSIM↑	0.857	0.947	0.876	0.893
	LPIPS↓	0.232	0.100	0.202	0.178
Gaussian-SLAM (Yugay et al., 2023)	PSNR↑	24.01	25.02	26.13	25.05
	SSIM↑	0.924	0.924	0.939	0.929
	LPIPS↓	0.178	0.186	0.141	0.168
LoopSplat∗ (Zhu et al., 2024)	PSNR↑	22.03	22.68	23.47	22.72
	SSIM↑	0.849	0.892	0.879	0.873
	LPIPS↓	0.307	0.217	0.253	0.259
Ours	PSNR↑	**27.09**	**33.01**	**30.50**	**30.20**
	SSIM↑	**0.959**	**0.982**	**0.974**	**0.972**
	LPIPS↓	**0.085**	**0.038**	**0.063**	**0.062**

Table 13. Tracking performance comparisons in ATE RMSE ↓ [cm] on ScanNet (Dai et al., 2017a). ∗ indicates methods relying on pre-trained data-driven priors.

Method	0000	0059	0106	0169	0181	0207	Avg.
Neural Implicit Fields							
NICE-SLAM (Zhu et al., 2022)	12.0	14.0	**7.9**	**10.9**	**13.4**	**6.2**	**10.7**
Vox-Fusion (Yang et al., 2022)	68.8	24.2	8.4	27.3	23.3	9.4	26.9
Point-SLAM (Sandström et al., 2023a)	**10.2**	**7.8**	8.7	22.2	14.8	9.5	12.2
Loopy-SLAM∗ (Liso et al., 2024)	4.2	7.5	8.3	7.5	10.6	7.9	7.7
3D Gaussian Splatting							
SplaTAM (Keetha et al., 2024)	**12.8**	10.1	17.7	12.1	11.1	**7.5**	11.9
Gaussian-SLAM (Yugay et al., 2023)	24.8	**8.6**	**11.3**	14.6	18.7	14.4	15.4
LoopSplat∗ (Zhu et al., 2024)	6.2	7.1	7.4	10.6	8.5	6.6	7.7
CG-SLAM∗ (Hu et al., 2024)	7.1	7.5	8.9	8.2	11.6	5.3	8.1
Ours	17.8	8.7	11.8	**10.5**	**10.6**	8.6	**11.3**

Table 14. Rendering performance comparison in PSNR ↑, SSIM ↑, and LPIPS ↓ on ScanNet (Dai et al., 2017a). ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	0000	0059	0106	0169	0181	0207	Avg.
Neural Implicit Fields								
NICE-SLAM (Zhu et al., 2022)	PSNR↑	18.71	16.55	17.29	**18.75**	15.56	18.38	17.54
	SSIM↑	0.641	0.605	0.646	0.629	0.562	0.646	0.621
	LPIPS↓	0.561	0.534	0.510	0.534	0.602	0.552	0.548
Vox-Fusion (Yang et al., 2022)	PSNR↑	19.06	16.38	**18.46**	18.69	16.75	19.66	18.17
	SSIM↑	0.662	0.615	**0.753**	0.650	0.666	0.696	0.673
	LPIPS↓	0.515	0.528	**0.439**	0.513	0.532	**0.500**	0.504
ESLAM (Johari et al., 2023)	PSNR↑	15.70	14.48	15.44	14.56	14.22	17.32	15.29
	SSIM↑	0.687	0.632	0.628	0.656	0.696	0.653	0.658
	LPIPS↓	0.449	**0.450**	0.529	**0.486**	0.482	0.534	**0.488**
Point-SLAM (Sandström et al., 2023a)	PSNR↑	**21.30**	**19.48**	16.80	18.53	**22.27**	**20.56**	**19.82**
	SSIM↑	**0.806**	**0.765**	0.676	**0.686**	**0.823**	**0.750**	**0.751**
	LPIPS↓	**0.485**	0.499	0.544	0.542	**0.471**	0.544	0.514
Loopy-SLAM∗ (Liso et al., 2024)	PSNR↑	-	-	-	-	-	-	15.23
	SSIM↑	-	-	-	-	-	-	0.629
	LPIPS↓	-	-	-	-	-	-	0.671
3D Gaussian Splatting								
SplaTAM (Keetha et al., 2024)	PSNR↑	19.33	19.27	17.73	21.97	16.76	19.8	19.14
	SSIM↑	0.660	0.792	0.690	0.776	0.683	0.696	0.716
	LPIPS↓	0.438	0.289	0.376	0.281	0.420	0.341	0.358
Gaussian-SLAM (Yugay et al., 2023)	PSNR↑	28.54	26.21	26.26	28.60	27.79	28.63	27.70
	SSIM↑	0.926	0.934	0.926	0.917	0.922	0.914	0.923
	LPIPS↓	0.271	0.211	0.217	0.226	0.277	0.288	0.248
LoopSplat∗ (Zhu et al., 2024)	PSNR↑	24.99	23.23	23.35	26.80	24.82	26.33	24.92
	SSIM↑	0.840	0.831	0.846	0.877	0.824	0.854	0.845
	LPIPS↓	0.450	0.400	0.409	0.346	0.514	0.430	0.425
Ours	PSNR↑	**31.51**	**30.60**	**31.27**	**32.02**	**29.60**	**31.58**	**31.10**
	SSIM↑	**0.957**	**0.974**	**0.975**	**0.962**	**0.954**	**0.946**	**0.961**
	LPIPS↓	**0.131**	**0.080**	**0.074**	**0.091**	**0.145**	**0.124**	**0.108**

Table 15. Tracking performance comparisons in ATE RMSE ↓ [cm] on ScanNet++ (Yeshwanth et al., 2023). ∗ indicates methods relying on pre-trained data-driven priors.

Method	a	b	c	d	e	Avg.
Neural Implicit Fields						
Point-SLAM (Sandström et al., 2023a)	246.16	632.99	830.79	271.42	574.86	511.24
ESLAM (Johari et al., 2023)	**25.15**	**2.15**	**27.02**	**20.89**	**35.47**	**22.14**
Loopy-SLAM∗ (Liso et al., 2024)	-	-	25.16	234.25	81.48	113.63
3D Gaussian Splatting						
SplaTAM (Keetha et al., 2024)	1.50	**0.57**	**0.31**	443.10	1.58	89.41
Gaussian-SLAM (Yugay et al., 2023)	**1.37**	5.97	2.70	2.35	**1.02**	2.68
LoopSplat∗ (Zhu et al., 2024)	1.14	3.16	3.16	1.68	0.91	2.05
Ours	2.79	1.50	0.96	**1.18**	1.31	**1.55**

Table 16. Rendering performance comparison in PSNR ↑ on ScanNet++ (Yeshwanth et al., 2023). ∗ indicates methods relying on pre-trained data-driven priors.

Method	a	b	c	d	e	Avg.
3D Gaussian Splatting						
SplaTAM (Keetha et al., 2024)	28.02	27.93	29.48	19.65	28.48	26.71
Gaussian-SLAM (Yugay et al., 2023)	30.06	30.02	31.15	28.75	31.94	30.38
LoopSplat∗ (Zhu et al., 2024)	30.15	30.08	30.04	28.94	31.78	30.20
Ours	**32.84**	**31.02**	**32.44**	**31.43**	**33.38**	**32.22**

Table 17. Novel View Synsthesis performance comparison in PSNR ↑ on ScanNet++ (Yeshwanth et al., 2023). ∗ indicates methods relying on pre-trained data-driven priors.

Method	a	b	c	d	e	Avg.
3D Gaussian Splatting						
SplaTAM (Keetha et al., 2024)	23.95	22.66	13.95	8.47	20.06	17.82
Gaussian-SLAM (Yugay et al., 2023)	**26.66**	**24.42**	15.01	18.35	21.91	21.27
LoopSplat∗ (Zhu et al., 2024)	25.60	23.65	15.87	18.86	22.51	21.30
Ours	25.55	24.25	**16.94**	**18.59**	**21.95**	**21.46**

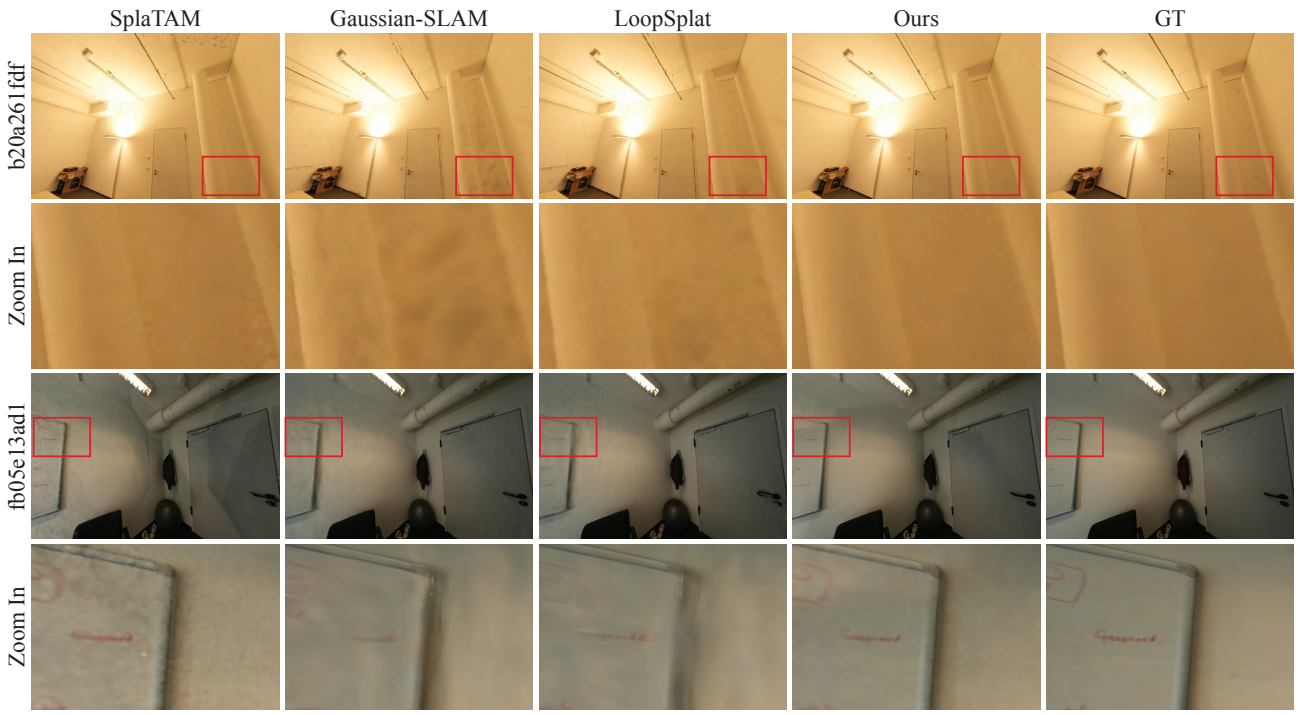

Figure 13. Visual comparisons in training view rendering on ScanNet++ (Yeshwanth et al., 2023).

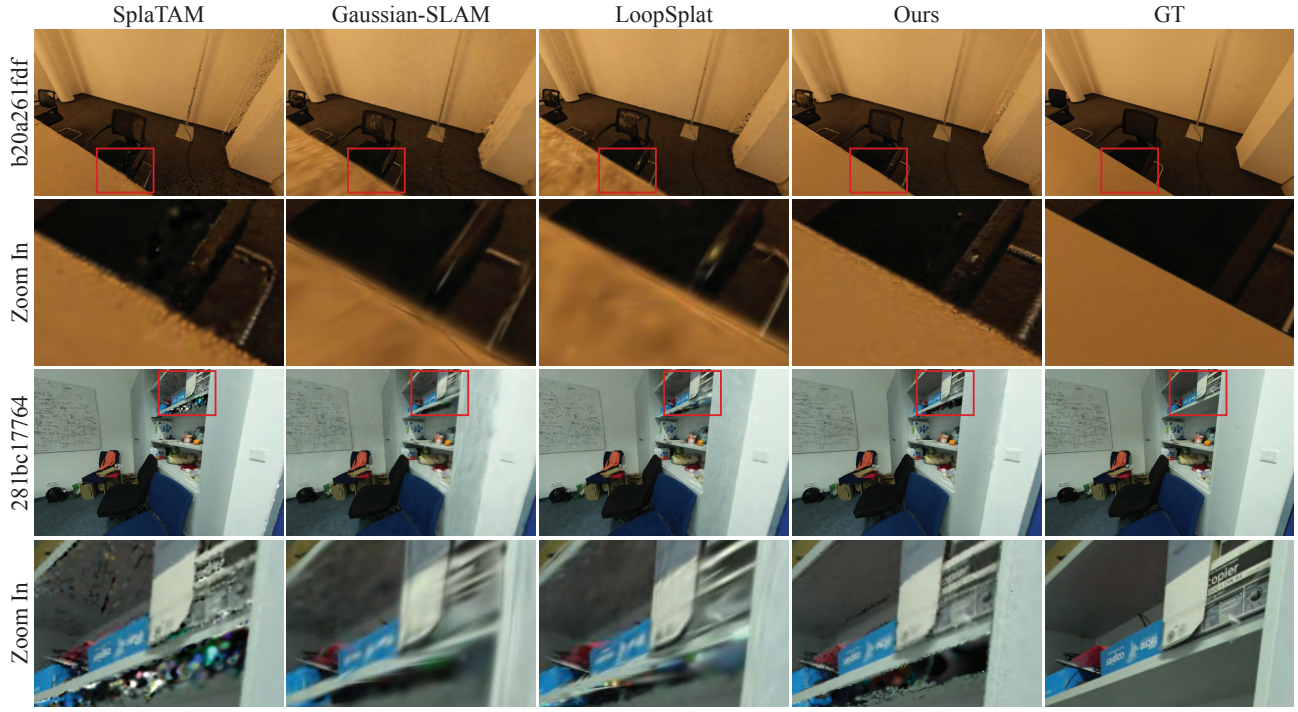

Figure 14. Visual comparisons in novel view rendering on ScanNet++ (Yeshwanth et al., 2023).

Table 18. Rendering performance comparisons in PSNR ↑, SSIM ↑, and LPIPS ↓ on Replica (Straub et al., 2019). ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	Rm0	Rm1	Rm2	Off0	Off1	Off2	Off3	Off4	Avg.
Neural Implicit Fields										
NICE-SLAM (Zhu et al., 2022)	PSNR↑	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 (Yang et al., 2022)	PSNR↑	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
ESLAM (Johari et al., 2023)	PSNR↑	25.25	27.39	28.09	30.33	27.04	27.99	29.27	29.15	28.06
	SSIM↑	0.874	0.89	0.935	0.934	0.910	0.942	0.953	0.948	0.923
	LPIPS↓	0.315	0.296	0.245	0.213	0.254	0.238	0.186	0.210	0.245
Point-SLAM (Sandström et al., 2023a)	PSNR↑	**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**
Loopy-SLAM∗ (Liso et al., 2024)	PSNR↑	-	-	-	-	-	-	-	-	35.47
	SSIM↑	-	-	-	-	-	-	-	-	0.981
	LPIPS↓	-	-	-	-	-	-	-	-	0.109
3D Gaussian Splatting										
SplaTAM (Keetha et al., 2024)	PSNR↑	32.86	33.89	35.25	38.26	39.17	31.97	29.70	31.81	34.11
	SSIM↑	0.98	0.97	0.98	0.98	0.98	0.97	0.95	0.95	0.97
	LPIPS↓	0.07	0.10	0.08	0.09	0.09	0.10	0.12	0.15	0.10
SGS-SLAM (Li et al., 2024)	PSNR↑	32.50	34.25	35.10	38.54	39.20	32.90	32.05	32.75	34.66
	SSIM↑	0.976	0.978	0.981	0.984	0.980	0.967	0.966	0.949	0.973
	LPIPS↓	0.070	0.094	0.070	0.086	0.087	0.101	0.115	0.148	0.096
GS-SLAM (Yan et al., 2024)	PSNR↑	31.56	32.86	32.59	38.70	41.17	32.36	32.03	32.92	34.27
	SSIM↑	0.968	0.973	0.971	0.986	0.993	0.978	0.970	0.968	0.975
	LPIPS↓	0.094	0.075	0.093	0.050	0.033	0.094	0.110	0.112	0.082
MonoGS (Matsuki et al., 2024)	PSNR↑	34.83	36.43	37.49	39.95	42.09	36.24	36.70	36.07	37.50
	SSIM↑	0.954	0.959	0.965	0.971	0.977	0.964	0.963	0.957	0.960
	LPIPS↓	0.068	0.076	0.075	0.072	0.055	0.078	0.065	0.099	0.070
Gaussian-SLAM (Yugay et al., 2023)	PSNR↑	38.88	41.80	42.44	46.40	45.29	40.10	39.06	42.65	42.08
	SSIM↑	**0.993**	**0.996**	**0.996**	**0.998**	**0.997**	**0.997**	**0.997**	**0.997**	**0.996**
	LPIPS↓	0.017	0.018	0.019	0.015	0.016	0.020	0.020	0.020	0.018
LoopSplat∗ (Zhu et al., 2024)	PSNR↑	33.07	35.32	36.16	40.82	40.21	34.67	35.67	37.10	36.63
	SSIM↑	0.973	0.978	0.985	0.992	0.990	0.985	0.990	0.989	0.985
	LPIPS↓	0.116	0.122	0.111	0.085	0.123	0.140	0.096	0.106	0.112
CG-SLAM∗ (Hu et al., 2024)	PSNR↑	33.27	-	-	-	-	-	34.60	-	-
	SSIM↑	-	-	-	-	-	-	-	-	-
	LPIPS↓	-	-	-	-	-	-	-	-	-
Ours	PSNR↑	**39.95**	**43.06**	**43.13**	**46.88**	**47.20**	**42.14**	**40.99**	**43.35**	**43.34**
	SSIM↑	0.992	**0.996**	**0.996**	**0.998**	**0.997**	0.996	0.996	0.996	**0.996**
	LPIPS↓	**0.014**	**0.013**	**0.014**	**0.009**	**0.009**	**0.012**	**0.013**	**0.015**	**0.012**

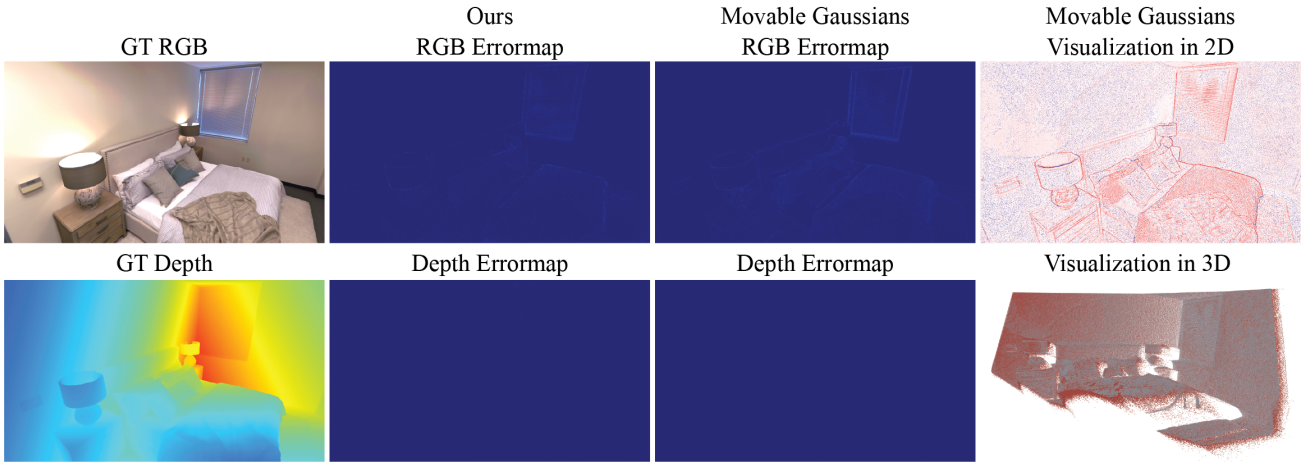

Figure 15. Visual comparisons on fixed Gaussians or movable Gaussians (along the ray) in rendering on Replica. We provide rendering results, including RGB error maps and depth error maps rendered by using either our view-tied Gaussians or movable Gaussians along the ray. Additionally, we provide a 2D visualization of the optimized movable Gaussians (blue: movement backward along the ray, red: movement forward along the ray. Furthermore, we provide a 3D visualization of the optimized movable Gaussians (gray: central points of Gaussians fixed at depth, red: central points of Gaussians optimized along the ray).

Table 19. Reconstruction performance comparison in Depth L1 [cm]↓ and F1 [%] ↑ on Replica (Straub et al., 2019). ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	Rm0	Rm1	Rm2	Off0	Off1	Off2	Off3	Off4	Avg.
Neural Implicit Fields										
NICE-SLAM (Zhu et al., 2022)	Depth L1 [cm]↓	1.81	1.44	2.04	1.39	1.76	8.33	4.99	2.01	2.97
	F1 [%] ↑	45.0	44.8	43.6	50.0	51.9	39.2	39.9	36.5	43.9
Vox-Fusion (Yang et al., 2022)	Depth L1 [cm]↓	1.09	1.90	2.21	2.32	3.40	4.19	2.96	1.61	2.46
	F1 [%] ↑	69.9	34.4	59.7	46.5	40.8	51.0	64.6	50.7	52.2
ESLAM (Johari et al., 2023)	Depth L1 [cm]↓	0.97	1.07	1.28	0.86	1.26	1.71	1.43	1.06	1.18
	F1 [%] ↑	81.0	82.2	83.9	78.4	75.5	77.1	75.5	79.1	79.1
Co-SLAM	Depth L1 [cm]↓	0.99	0.82	2.28	1.24	1.61	7.70	4.65	1.43	2.59
	F1 [%] ↑	77.7	74.2	69.3	75.2	75.2	54.3	56.8	75.3	69.7
Point-SLAM (Sandström et al., 2023a)	Depth L1 [cm]↓	**0.53**	**0.22**	**0.46**	**0.30**	**0.57**	**0.49**	**0.51**	**0.46**	**0.44**
	F1 [%] ↑	**86.9**	**92.3**	**90.8**	**93.8**	**91.6**	**89.0**	**88.2**	**85.6**	**89.8**
Loopy-SLAM∗ (Liso et al., 2024)	Depth L1 [cm]↓	0.30	0.20	0.42	0.23	0.46	0.60	0.37	0.24	0.35
	F1 [%] ↑	91.6	92.4	90.6	93.9	91.6	88.5	89.0	88.7	90.8
3D Gaussian Splatting										
SplaTAM (Keetha et al., 2024)	Depth L1 [cm]↓	**0.43**	0.38	**0.54**	0.44	0.66	1.05	1.60	0.68	0.72
	F1 [%] ↑	89.3	88.2	88.0	91.7	90.0	85.1	77.1	80.1	86.1
GS-SLAM (Yan et al., 2024)	Depth L1 [cm]↓	1.31	0.82	1.26	0.81	0.96	1.41	1.53	1.08	1.16
	F1 [%] ↑	62.9	79.9	66.8	80.0	81.6	66.0	59.2	65.0	70.2
Gaussian-SLAM (Yugay et al., 2023)	Depth L1 [cm]↓	0.61	**0.25**	**0.54**	0.50	0.52	0.98	1.63	**0.42**	0.68
	F1 [%] ↑	88.8	91.4	90.5	91.7	90.1	87.3	84.2	**87.4**	88.9
LoopSplat∗ (Zhu et al., 2024)	Depth L1 [cm]↓	0.39	0.23	0.52	0.32	0.51	0.63	1.09	0.40	0.51
	F1 [%] ↑	90.6	91.9	91.1	93.3	90.4	88.9	88.7	88.3	90.4
Ours	Depth L1 [cm]↓	0.48	0.28	0.61	**0.41**	**0.48**	**0.62**	**0.86**	0.53	**0.53**
	F1 [%] ↑	**90.7**	**91.7**	**90.7**	**93.0**	**90.8**	**88.3**	**87.5**	87.0	**90.0**

Table 20. Tracking performance comparisons in ATE RMSE ↓ [m] on KITTI (Geiger et al., 2012).

Sequence	Gaussian-SLAM	SplaTAM	LoopSplat	Ours
00	3.02	58.83	2.22	**2.06**
01	77.51	84.45	74.47	**29.01**
05	128.88	80.39	117.43	**7.74**
10	10.60	43.82	11.39	**4.54**

Table 21. Rendering performance comparisons in PSNR ↑ on KITTI (Geiger et al., 2012).

Sequence	Gaussian-SLAM	SplaTAM	LoopSplat	Ours
00	15.51	9.82	15.82	**28.54**
01	15.95	12.89	14.69	**30.33**
05	16.22	26.48	15.98	**28.19**
10	15.58	25.58	14.58	**27.59**
Peak GPU Use (GiB)	2.74	22.37	3.56	4.79

Table 22. Impact of depth noise and movability of Gaussians on the rendering performance in PSNR ↑, SSIM ↑, and LPIPS ↓ on Replica (Straub et al., 2019).

Metric	10% pixels w/ noises	20% pixels w/ noises	30% pixels w/ noises	Gaussians movable along ray	Ours(w/o additional noises & fix)
PSNR↑	43.41	43.40	43.29	42.89	43.06
SSIM↑	0.996	0.996	0.996	0.995	0.996
LPIPS↓	0.015	0.015	0.015	0.020	0.013

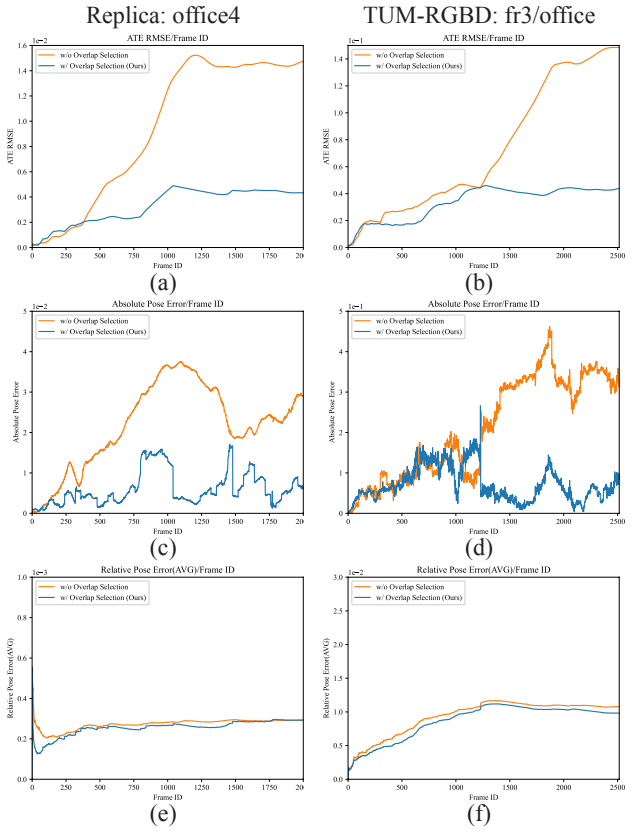

Figure 16. Comparison on w/ or w/o overlap selection when tracking.

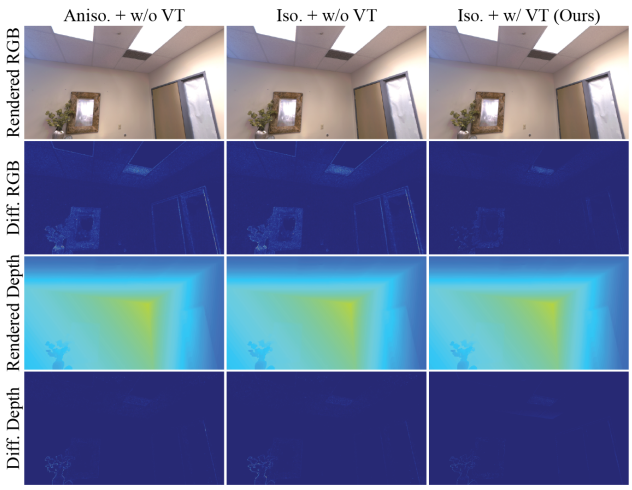

Figure 17. Comparisons of different kinds of Gaussians (with the same number). Our view-tied Gaussians in the 3rd column can recover more accurate RGB color and depth in renderings, while the ellipsoid Gaussians in the 1st column and sphere Gaussians in the 2nd column produce worse rendering quality with adjustable Gaussians. Please refer to our video for a complete comparison of optimization.

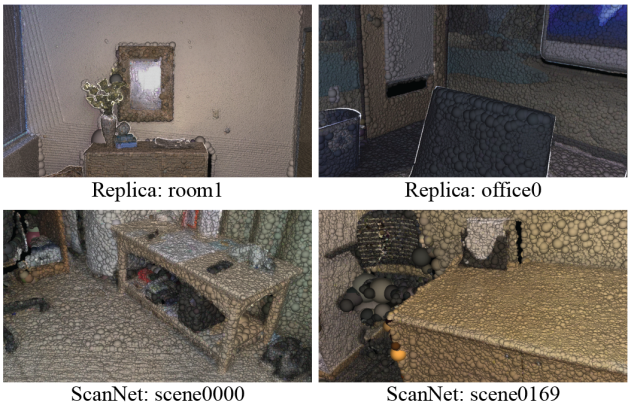

Figure 18. Visualization of optimized 3D Gaussians.

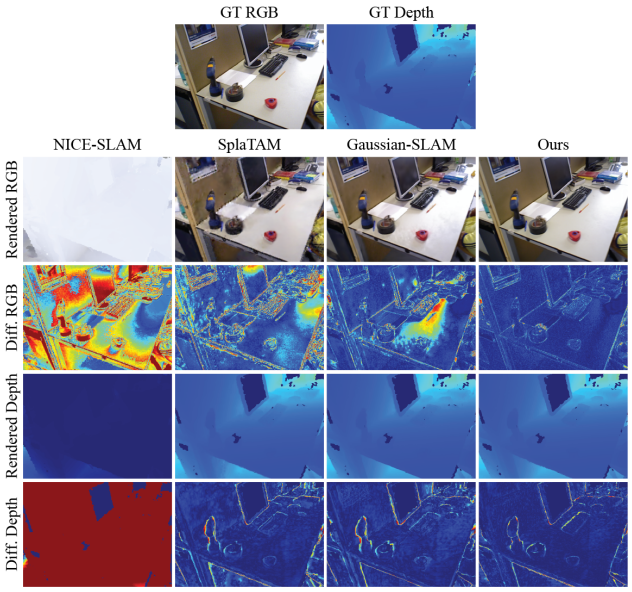

Figure 19. Visual comparisons of rendered images and depths. We also show error maps (large rendering errors are shown in red). Please refer to our video for more visual comparisons of rendered images.