2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

SupGS-SLAM: Gaussian Splatting SLAM with Efficient Keyframe Strategy and Supplementary Mapping

Shuai Liu, Yongcai Wang*, Wenping Chen, Wang Chen, and Deying Li

***Abstract*— Gaussian Splatting SLAM methods have exhibited impressive high-fidelity rendering performance. Existing methods maintain high rendering quality around the current camera viewpoint, but the rendering quality degrades in previously observed regions as the camera moves away, particularly in real-world scenarios. We identify two core factors for high-quality rendering: keyframes should efficiently cover the entire scene while minimizing redundancy, and the mapping strategy should effectively select critical keyframes for full scene optimization. To address these issues, we propose SupGS-SLAM to improve rendering quality across the entire scene. For effective keyframe management, we propose an efficient keyframe strategy, which reduces redundant keyframe selection and prioritizes the optimization of critical keyframes by assigning high weights. For enhanced mapping, we propose a supplementary mapping strategy comprising three components: supplementary densification, supplementary global mapping, and supplementary depth mapping. In supplementary densification, we add supplementary Gaussian primitives to previous regions with insufficient representation. In supplementary global mapping, we select keyframes globally to optimize the full scene. In supplementary depth mapping, we use estimated depth to optimize regions without ground-truth depth. Extensive experiments demonstrate that SupGS-SLAM achieves excellent performance on both synthetic and real-world datasets. The project page is available at https://github.com/rucliushuai/SupGS-SLAM.**

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is a key technology for embodied intelligence, virtual reality (VR), and augmented reality (AR) [1]–[5]. The goal of SLAM is to estimate camera poses while constructing an accurate scene map in unknown environments. Traditional visual SLAM methods [6]–[13] have achieved remarkable progress. However, they struggle to effectively construct and optimize fine-grained maps.

For high-quality reconstruction, several methods [14]–[18] use neural radiance fields (NeRF) [19] for dense scene representation. These methods leverage multi-layer perceptrons (MLPs) as the implicit neural representation of scenes. However, due to the expensive differentiable volume rendering, these methods struggle to achieve fast image rendering, which limits their applications in real-time scenarios.

Recently, 3D Gaussian Splatting [20] has been proposed. Due to its fast and high-quality rendering capabilities, it has become a hot research topic and facilitated significant progress in SLAM tasks. Compared to NeRF-based methods, which leverage implicit neural representations, 3D Gaussian Splatting represents scenes explicitly using a set of Gaussian primitives. This explicit representation enables it to accurately represent scenes and easily adapt to new scenes by adding, modifying, or removing Gaussian primitives.

Fig. 1. With the efficient keyframe strategy and supplementary mapping, SupGS-SLAM maintains high rendering quality across the entire scene.

Significant achievements have been made in Gaussian Splatting SLAM [21]–[26]. However, most of these methods achieve high rendering quality near the current camera position, while failing to maintain high rendering quality for previously observed regions. This phenomenon is more pronounced in real-world scenarios with motion blur and sparse ground-truth depth.

We identify that the key to this phenomenon in these methods lies in two aspects: redundant keyframes and the failure to exploit the most critical keyframes for mapping.

To address these problems, we propose SupGS-SLAM, a Gaussian Splatting SLAM method to improve the rendering quality of the entire scene, as shown in Fig. 1. For efficient keyframe selection, we introduce a keyframe selection strategy based on view overlap between keyframes, which constructs a keyframe list covering the entire scene using a small number of keyframes. We assign each keyframe a weight based on its timestamp of last optimization and rendering quality, and prioritize the optimization of the keyframes with low rendering quality or long unoptimized periods. For enhanced mapping performance, we propose a supplementary mapping strategy, encompassing supplementary densification, supplementary global mapping, and supplementary depth mapping. Specifically, in the supplementary densification process, we add supplementary Gaussian primitives to previously observed regions that are not well represented by existing Gaussian primitives. In the supplementary global mapping process, we randomly sample a set

All authors are with the Department of Computer Science, School of Information, Renmin University of China, Beijing 100872, China. *Corresponding author. Email: {rucliushuai, ycw, chenwenping, chenwang, deyingli}@ruc.edu.cn.

Dr. Li is supported in part by the National Natural Science Foundation of China Grant No. 12071478. Dr. Wang is supported by Public Computing Cloud, Renmin University of China, and the Blockchain Lab, School of Information, Renmin University of China.

of points across the entire scene, select a list of keyframes to cover these points as comprehensively as possible, and then jointly optimize these keyframes to improve the entire scene representation. In the supplementary depth mapping process, we optimize the regions that lack ground-truth depth by leveraging estimated depth. Through the integration of these strategies, the rendering quality of the entire scene is significantly improved.

Extensive experiments on Replica [27], ScanNet [28], and TUM-RGBD [29] datasets demonstrate that SupGS-SLAM performs well on both synthetic and real-world datasets. The method enables accurate tracking and mapping, while significantly reducing the number of redundant keyframes. Furthermore, the rendering quality of the method on real-world datasets is substantially improved.

Our contributions are summarized as follows:

(1) We propose SupGS-SLAM, a Gaussian Splatting SLAM method that maintains high rendering quality across the entire scene.

(2) We propose an efficient keyframe strategy to reduce redundant keyframes and assign weights to prioritize the optimization of critical keyframes.

(3) We propose a supplementary mapping strategy to improve the reconstruction quality of the entire scene.

II. RELATED WORK

A. Dense Visual SLAM

Dense Visual SLAM aims to construct dense maps of scenes. BundleFusion [6] uses a single framework to robustly estimate globally optimized poses in real-time. ElasticFusion [7] supports accurate real-time mapping in complex scenes with variable lighting conditions. BAD-SLAM [8] presents a fast direct bundle adjustment formulation, which can be applied in real-time SLAM systems. DTAM [9] does not rely on feature extraction but on every pixel, and tracks camera poses via image alignment.

B. NeRF-Based SLAM

Due to high reconstruction quality, neural radiance fields (NeRF) [19] have been widely used in SLAM tasks. iMAP [14] uses a multilayer perceptron (MLP) as the only scene representation in real-time SLAM. Vox-Fusion [15] encodes and optimizes the scene using a voxel-based neural implicit surface representation. Point-SLAM [16] iteratively generates a point-based neural scene representation to conduct tracking and mapping processes. NICE-SLAM [17] introduces a hierarchical scene representation and incorporates multi-level local information for detailed reconstruction. ESLAM [18] uses multi-scale axis-aligned perpendicular feature planes and shallow decoders for scene representation.

C. Gaussian-Splatting-Based SLAM

Recently, Gaussian Splatting [20] has been proposed for fast and high-quality rendering, and this technology has significantly improved the performance of the methods in SLAM tasks. Gaussian Splatting SLAM [21] utilizes Gaussians as the scene representation for accuracy and efficiency. GS-SLAM [22] utilizes a real-time differentiable splatting rendering pipeline for efficient optimization and rendering. RTG-SLAM [23] proposes a new way for depth rendering to better fit local surface regions. SplaTAM [24] utilizes a silhouette mask to represent the density of the scene. SGS-SLAM [25] and Hier-SLAM [26] introduce semantic information into Gaussian primitives for better rendering.

III. METHOD

The framework of SupGS-SLAM is shown in Fig. 2. We use Gaussian primitives for scene representation. In the tracking process, the camera pose of the current frame is estimated, and in the mapping process, the scene is optimized. With the proposed keyframe strategy, we reduce redundant keyframes and assign weights to keyframes to prioritize the optimization of those with low rendering quality or long unoptimized periods.

A. Scene Representation

For simplicity and convenience, we use $\mathcal{N}$ isotropic Gaussian primitives to represent the scene:

$$\mathcal{G}_i = \{\mu_i, r_i, o_i, c_i\}, i = 1, \ldots, \mathcal{N} \tag{1}$$

where $\mu_i \in \mathbb{R}^3$ is the geometric center, $r_i \in \mathbb{R}^+$ is the radius, $o_i \in [0,1]$ is the opacity, and $c_i \in \mathbb{R}^3$ is the color. The Gaussian primitives change dynamically during the entire process to adapt to the observed scene.

A point in the 3D space $\mathbf{x} \in \mathbb{R}^3$ is influenced by a Gaussian primitive according to the standard Gaussian equation weighted by opacity:

$$f_i(\mathbf{x}) = o_i \exp\left(-\frac{\|\mathbf{x}-\mu_i\|^2}{2r_i^2}\right) \tag{2}$$

Using the rendering method proposed in [20], Gaussian primitives are rendered to RGB images. Specifically, a Gaussian primitive $\mathcal{G}_i$ is splatted into 2D pixel-space:

$$\mu_i^{2\mathrm{D}} = K\frac{E_t\mu_i}{d_i}, \quad r_i^{2\mathrm{D}} = \frac{lr_i}{d_i}, \quad d_i = (E_t\mu_i)_z \tag{3}$$

where $\mu_i^{2\mathrm{D}}$ is the 2D geometric center, $r_i^{2\mathrm{D}}$ is the 2D radius, K is the camera intrinsic matrix, E_t is the camera extrinsic matrix at timestamp t, l is the focal length, d_i is the depth of the Gaussian primitive in camera coordinates.

After sorting the Gaussian primitives from front-to-back in depth order, the RGB map is rendered through alpha-compositing. Specifically, for a pixel $\mathbf{p} = (u, v)$ in 2D pixel-space, the rendered color $C_\mathbf{p}$ is computed as follows:

$$C_\mathbf{p} = \sum_{i=1}^{n} c_i f_i^{2\mathrm{D}}(\mathbf{p}) \prod_{j=1}^{i-1}(1 - f_j^{2\mathrm{D}}(\mathbf{p})) \tag{4}$$

where c_i is the color of the i-th Gaussian primitive $\mathcal{G}_i$, $f_i^{2\mathrm{D}}(\mathbf{p})$ is the standard Gaussian equation weighted by opacity by replacing 3D geometric center μ_i and radius r_i with the 2D geometric center $\mu_i^{2\mathrm{D}}$ and radius $r_i^{2\mathrm{D}}$.

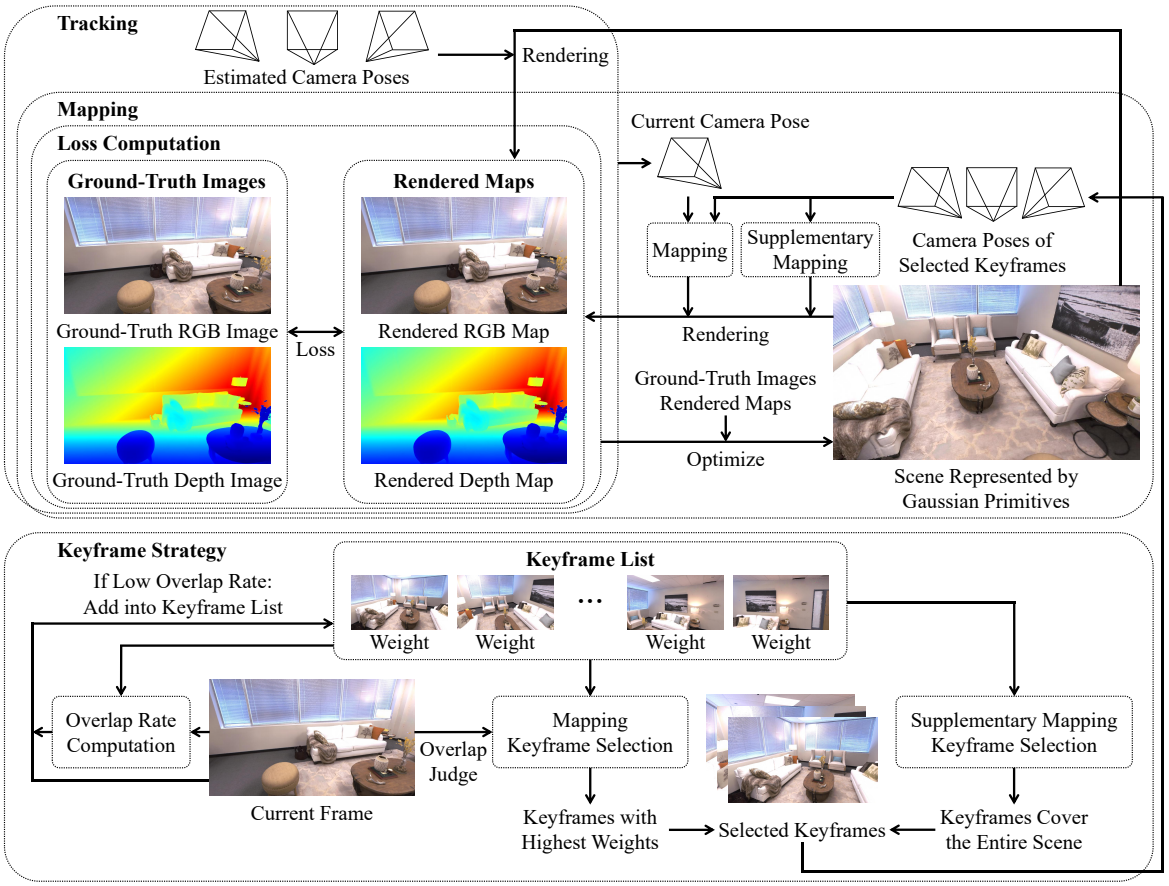


Fig. 2. The overview of SupGS-SLAM. The scene is represented by Gaussian primitives. In the tracking process, the current camera pose is estimated by minimizing the tracking loss. If there is a low overlap rate between the current frame and the already selected keyframes, we select the current frame as a keyframe and assign a weight to it. Keyframes with low rendering quality or long unoptimized periods are assigned high weights. In the mapping process, we select the top-K keyframes with the highest weights and overlap with the current frame, and in the supplementary global mapping process, we select keyframes to cover the entire scene. The selected keyframes are used to jointly optimize the scene.

Following [24], the depth map D and silhouette map S are rendered similarly to the color map:

$$D_{\mathbf{p}} = \sum_{i=1}^{n} d_i f_i^{2\mathrm{D}}(\mathbf{p}) \prod_{j=1}^{i-1} (1 - f_j^{2\mathrm{D}}(\mathbf{p})) \tag{5}$$

$$S_{\mathbf{p}} = \sum_{i=1}^{n} f_i^{2\mathrm{D}}(\mathbf{p}) \prod_{j=1}^{i-1} (1 - f_j^{2\mathrm{D}}(\mathbf{p})) \tag{6}$$

B. Keyframe Strategy

Keyframe Selection. Keyframe selection aims to select a small number of critical keyframes for optimization, instead of processing every frame. Some methods [24]–[26] select keyframes at a constant time interval, but this strategy can lead to redundant keyframes when the camera lingers over the same scene or revisits previously observed areas.

Therefore, we propose an efficient keyframe selection strategy. We select the current frame as a keyframe only when there is a low overlap rate between it and the already selected keyframes. For the current frame with timestamp t, we randomly sample k pixels in the frame, and convert these 2D pixels into 3D points in the camera coordinate system using their ground-truth depth. Then these sampled points are transformed into the global coordinate system via the camera pose of the current frame. Next, we check whether these points are covered by the previously selected keyframes.

For sampled points $\mathcal{P} = \{P_i \in \mathbb{R}^3 \mid i = 1, \ldots, k\}$ and the existing keyframe list, we first exclude keyframes that have large rotations or translations relative to the current frame to obtain a filtered keyframe list $\mathcal{K}$ (where $|\mathcal{K}|$ denotes the number of keyframes in $\mathcal{K}$). Then we project each point P_i onto the image plane of each keyframe $K_j \in \mathcal{K}, j = 1, \ldots, |\mathcal{K}|$ to obtain pixel coordinates $\mathbf{p}_{i,j} = (u_{i,j}, v_{i,j})$ and depth $d_{i,j}$ in the keyframe coordinates. If $u_{i,j} \in [0, W)$, $v_{i,j} \in [0, H)$, and $d_{i,j} \in (0, D^{\mathrm{GT}}_{\mathbf{p}_{i,j}}]$ (where W and H are the width and height of the image, and $D^{\mathrm{GT}}_{\mathbf{p}_{i,j}}$ is the ground-truth depth at pixel $\mathbf{p}_{i,j}$), we consider that the point P_i is covered by the keyframe K_j. If the ratio of points that are not covered by any keyframe is higher than a threshold T_{keyframe}, we select the current frame as a keyframe since it captures significant new regions.

Keyframe Weighting. Keyframes with low rendering quality or long unoptimized periods should be prioritized for optimization. We assign weights to keyframes to prioritize the optimization of these keyframes. For each keyframe K_i, we compute the weight W_i as follows:

$$W_i = (T_{\mathrm{P}} - PSNR_i) \cdot (1 - e^{-\tau(t - Time_i)}) \tag{7}$$

where T_{P} is a large predefined threshold, $PSNR_i$ is the

PSNR value of this keyframe after the last optimization, t is the current timestamp, $Time_i$ is the timestamp of the last optimization, and τ is a hyperparameter.

C. SLAM System

Tracking. The tracking process aims to estimate camera poses, including rotations and translations. The camera poses are estimated by minimizing the differences between the ground-truth images and rendered maps, while the parameters of Gaussian primitives are fixed.

For the first frame, the camera pose is set to a pose with zero translation and identity rotation, and this camera pose defines the reference coordinate system.

For a new frame with timestamp t, the initial camera pose E_t is computed assuming the transform between the current frame and the last frame is equal to the transform between the last frame and the second-to-last frame. Specifically, the camera pose of the second frame is initialized to be identical to that of the first frame.

With the initial camera pose, based on the rendered RGB map C, depth map D, and silhouette map S, the tracking loss $\mathcal{L}_{\mathrm{t}}$ is computed as:

$$M_{\mathrm{t}} = \{(u,v),\ u \in [\lambda_W, W-\lambda_W),\ v \in [\lambda_H, H-\lambda_H)\} \tag{8}$$

$$\mathcal{L}_{\mathrm{t}} = \sum_{\mathbf{p}\in M_{\mathrm{t}}} [S_{\mathbf{p}} > T_{\mathrm{sil}}]\left(\lambda_C |C_{\mathbf{p}} - C_{\mathbf{p}}^{\mathrm{GT}}| + \lambda_D |D_{\mathbf{p}} - D_{\mathbf{p}}^{\mathrm{GT}}|\right) \tag{9}$$

where M_{t} is a mask used to exclude image margins where rendering quality tends to be low. These margin pixels are not used for tracking loss computation. W and H are the width and height of the image, and λ_H and λ_W are predefined thresholds. T_{sil} is a threshold that filters in pixels with high silhouette values for tracking loss computation. $C_{\mathbf{p}}^{\mathrm{GT}}$ and $D_{\mathbf{p}}^{\mathrm{GT}}$ are the ground-truth RGB and depth values at pixel $\mathbf{p}$. Additionally, only pixels with ground-truth depth are used for tracking loss computation. λ_C and λ_D are hyperparameters to scale the contributions of RGB loss and depth loss.

Densification. The densification process aims to add Gaussian primitives for scene representation. We add Gaussian primitives in regions that are inadequately represented.

Following [24], for the first frame, we initialize a Gaussian primitive for every pixel based on the ground-truth RGB and depth, and for the subsequent frames, we add Gaussian primitives using two types of pixels: (1) The silhouette value of the pixel is lower than a threshold T_{dens}, indicating insufficient representation by existing Gaussian primitives. (2) The rendered depth of the pixel is much higher than its ground-truth depth, indicating the presence of new geometry.

Mapping. The mapping process aims to optimize the scene. We select k keyframes to optimize the Gaussian primitives while keeping the camera poses fixed. For each selected keyframe, based on the rendered RGB map C and depth map D, the mapping loss $\mathcal{L}_{\mathrm{m}}$ is computed as:

$$M_{\mathrm{m}} = \{(u,v),\ u \in [0,W),\ v \in [0,H)\} \tag{10}$$

$$\mathcal{L}_{\mathrm{m}} = \sum_{\mathbf{p}\in M_{\mathrm{m}}} \left(\lambda_C \mathcal{L}_C(\mathbf{p}) + \lambda_D |D_{\mathbf{p}} - D_{\mathbf{p}}^{\mathrm{GT}}|\right) \tag{11}$$

where M_{m} denotes the set of pixels used for mapping loss computation. All pixels contribute to the computation of the mapping loss. λ_C and λ_D are hyperparameters that scale the contributions of RGB loss and depth loss, and $\mathcal{L}_C(\mathbf{p})$ is the RGB loss combining L1 loss and SSIM loss:

$$\mathcal{L}_C(\mathbf{p}) = \alpha |C_{\mathbf{p}} - C_{\mathbf{p}}^{\mathrm{GT}}| + (1-\alpha)(1 - \mathrm{ssim}(C_{\mathbf{p}}, C_{\mathbf{p}}^{\mathrm{GT}})) \tag{12}$$

where α is a hyperparameter that balances the contributions of L1 loss and SSIM loss.

We select the current frame, the latest keyframe, and the $k-2$ most critical keyframes for mapping. To select the critical keyframes, we first filter keyframes that overlap with the current frame, and then select the top $k-2$ keyframes with the highest weights. This ensures that keyframes most in need of optimization are prioritized.

D. Supplementary Mapping

Supplementary Densification. After the mapping process, the rendering quality can be maintained at a high level in the short term, but it degrades as the camera moves away. The most obvious phenomenon is that some pixels exhibit low silhouette values, indicating that these pixels cannot be well represented by Gaussian primitives. As a result, the rendered RGB and depth values of these pixels differ significantly from the ground-truth values. This phenomenon is more pronounced in real-world scenarios due to motion blur and sparse ground-truth depth.

To solve this problem, we propose a supplementary densification strategy to add supplementary Gaussian primitives to previous regions that are inadequately represented by Gaussian primitives. In the mapping process, after selecting k keyframes and before optimizing them, we add Gaussian primitives for the pixels whose silhouette values are lower than a threshold T_{sup} in these keyframes. Instead of using ground-truth RGB values, we calculate a supplementary color c for the Gaussian primitive as follows:

$$c = (C_{\mathbf{p}}^{\mathrm{GT}} - (1-o)\cdot C_{\mathbf{p}})/o \tag{13}$$

where $C_{\mathbf{p}}^{\mathrm{GT}}$ is the ground-truth color at pixel $\mathbf{p}$, $C_{\mathbf{p}}$ is the rendered color at pixel $\mathbf{p}$ before adding the supplementary Gaussian primitives, and o is the predefined initial opacity of the added Gaussian primitive. The supplementary color is clipped to the valid range. This makes the rendered color of the pixel match its ground-truth color after addition.

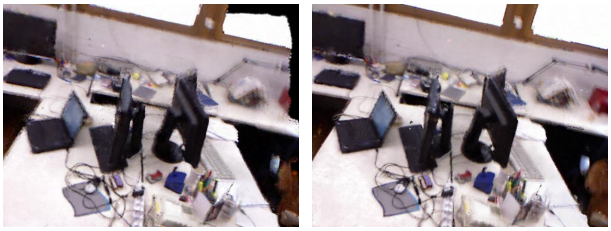
Supplementary Global Mapping. The mapping process focuses on optimizing the scene near the current frame and selects keyframes that overlap with it. However, keyframes corresponding to regions no longer observed by the camera remain unoptimized, resulting in degraded rendering quality.

To solve this problem, we propose a supplementary global mapping strategy to globally select keyframes for optimizing the entire scene. We randomly select $2k$ keyframes from the keyframe list and sample n_p 3D points from these keyframes. For each keyframe, we randomly sample $n_p/(2k)$ pixels and transform them into 3D points in global coordinates. Next, we use a greedy strategy to select k keyframes to cover these points as comprehensively as possible.

Similar to the method used in keyframe selection, for each keyframe, we transform the sampled points from global coordinates into its camera coordinates and count the number of points that the keyframe can cover. We select the keyframe that covers the most points. Then we remove the selected keyframe and the points covered by it, and select the next keyframe from the remaining keyframe list based on the remaining points. This process is repeated iteratively until we select k keyframes. Then we perform supplementary densification and use these globally selected keyframes for optimization. This approach optimizes the global scene.

Supplementary Depth Mapping. In the densification process, only pixels with ground-truth depth are used to add Gaussian primitives. However, in some scenarios, especially real-world scenarios, it is common that some regions such as windows lack ground-truth depth. If these regions persistently lack ground-truth depth, no Gaussian primitives are used to represent them.

To solve this problem, we propose a supplementary depth mapping strategy to add Gaussian primitives to positions lacking ground-truth depth using estimated depth. For keyframes with a considerable time interval from the current timestamp, if there are pixels without ground-truth depth and with silhouette values lower than a threshold T_{sup}, we apply the Depth Anything model [30] to estimate the depth of the entire image, then align the estimated depth with the existing ground-truth depth using the least squares method. Using the estimated depth, we perform supplementary densification and mapping processes to ensure these regions can be represented by Gaussian primitives. As shown in Fig. 3, with the supplementary depth mapping strategy, some regions such as windows in real-world scenarios can be represented significantly better.



Without Supplementary Depth Mapping With Supplementary Depth Mapping

Fig. 3. The Visualization of Supplementary Depth Mapping on TUM-RGBD Dataset [29].

IV. EXPERIMENTS

In the tracking process, the accuracy of camera poses highly relies on the rendering quality of the current frame. The supplementary global mapping strategy, which improves global rendering quality, may alter the scene near the current frame and degrade tracking accuracy. Additionally, the supplementary depth mapping strategy will add Gaussian primitives with relatively high error. Therefore, we propose three variants of SupGS-SLAM. For better tracking, we only apply supplementary densification. For better mapping, we apply supplementary densification and supplementary global mapping. For better scene representation, we apply the full supplementary mapping.

A. Experimental Setup

Datasets. We evaluate SupGS-SLAM on three widely-used datasets: Replica [27], ScanNet [28], and TUM-RGBD [29]. Replica is a synthetic dataset, while ScanNet and TUM-RGBD are real-world datasets. Following SplaTAM [24], we conduct the experiments on 8 scenes of Replica, 6 scenes of ScanNet, and 5 scenes of TUM-RGBD.

Metrics. Following SplaTAM [24], we use PSNR, SSIM, and LPIPS to evaluate image rendering quality, Depth L1 loss to evaluate depth rendering quality, and average absolute trajectory error (ATE RMSE) to evaluate the accuracy of camera poses.

Baselines. We compare the experimental results with state-of-the-art methods: Vox-Fusion [15], Point-SLAM [16], NICE-SLAM [17], ESLAM [18], SplaTAM [24], SGS-SLAM [25], and Hier-SLAM [26].

Implementation Details. The experiments are conducted on an NVIDIA RTX 3090 GPU. In the keyframe strategy, we set $T_{\text{keyframe}} = 0.2$, $T_{\text{P}} = 40$ for the ScanNet and TUM-RGBD datasets, and $T_{\text{P}} = 60$ for the Replica dataset, and $\tau = 0.01$. In tracking, we set $\lambda_H = 30$, $\lambda_W = 30$, $T_{\text{sil}} = 0.99$, $\lambda_C = 0.5$, and $\lambda_D = 1.0$. In densification, we set $T_{\text{dens}} = 0.5$. In mapping, we set $\lambda_C = 0.5$, $\lambda_D = 1.0$, and $\alpha = 0.8$. In supplementary mapping, we set $T_{\text{sup}} = 0.5$ and $o = 0.5$.

B. Evaluation of Tracking

The comparisons of the camera pose estimation results on the Replica [27], ScanNet [28], and TUM-RGBD [29] datasets are listed in Tab. I, Tab. II, and Tab. III. On the Replica dataset, the tracking is accurate. On the ScanNet and TUM-RGBD datasets, due to motion blur and sparse ground-truth depth, none of the methods can perform as well as they do on the Replica dataset. SupGS-SLAM achieves excellent results, and in some scenes, the performance is much better than that of other methods.

TABLE I

CAMERA POSE ESTIMATION RESULTS ON THE REPLICA DATASET [27] (ATE RMSE ↓[CM]). BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, AND THIRD.

Methods	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
Vox-Fusion [15]	3.09	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94
NICE-SLAM [17]	1.06	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13
ESLAM [18]	0.63	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63
Point-SLAM [16]	0.52	0.61	**0.41**	0.37	**0.38**	0.48	0.54	0.69	0.72
SGS-SLAM [25]	**0.41**	**0.46**	0.45	**0.29**	**0.46**	**0.23**	**0.45**	**0.42**	**0.55**
SplaTAM [24]	**0.36**	**0.31**	**0.40**	**0.29**	**0.47**	**0.27**	**0.29**	**0.32**	**0.55**
Ours	**0.35**	**0.24**	**0.31**	**0.27**	0.51	**0.22**	**0.39**	**0.31**	**0.55**

TABLE II

CAMERA POSE ESTIMATION RESULTS ON THE SCANNET DATASET [28] (ATE RMSE ↓[CM]). BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, AND THIRD.

Methods	Avg.	0000	0059	0106	0169	0181	0207
Vox-Fusion [15]	26.90	68.84	24.18	**8.41**	27.28	23.30	9.41
Point-SLAM [16]	12.19	**10.24**	**7.81**	**8.65**	22.16	14.77	9.54
NICE-SLAM [17]	**10.70**	**12.00**	14.00	**7.90**	**10.90**	**13.40**	**6.20**
SplaTAM [24]	**11.88**	12.83	**10.10**	17.72	**12.08**	**11.10**	**7.46**
Ours	**11.62**	**12.55**	**11.82**	17.59	**10.78**	**9.19**	**7.79**

Fig. 4. The Qualitative Comparison on ScanNet Dataset [28].

TABLE III

CAMERA POSE ESTIMATION RESULTS ON THE TUM-RGBD DATASET [29] (ATE RMSE ↓[CM]). BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, AND THIRD.

Methods	Avg.	fr1/ desk	fr1/ desk2	fr1/ room	fr2/ xyz	fr3/ off.
NICE-SLAM [17]	15.87	4.26	**4.99**	34.49	31.73	**3.87**
Vox-Fusion [15]	11.31	**3.52**	6.00	**19.53**	1.49	26.01
Point-SLAM [16]	**8.92**	4.34	**4.54**	30.92	**1.31**	**3.48**
SplaTAM [24]	**5.48**	**3.35**	6.54	**11.13**	**1.24**	**5.16**
Ours	**5.45**	**2.89**	**4.39**	**12.65**	**1.31**	6.00

TABLE IV

QUANTITATIVE TRAINING VIEW RENDERING QUALITY ON THE REPLICA DATASET [27]. BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, AND THIRD.

Methods	Metrics	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
NICE-SLAM [17]	PSNR ↑	24.42	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94
	SSIM ↑	0.81	0.69	0.76	0.81	0.87	0.89	0.80	0.80	0.86
	LPIPS ↓	0.23	0.33	0.27	0.21	0.23	0.18	0.24	0.21	0.20
Point-SLAM [16]	PSNR ↑	**35.17**	**32.40**	**34.08**	**35.50**	**38.26**	**39.16**	**33.99**	**33.48**	**33.49**
	SSIM ↑	**0.98**	**0.97**	**0.98**	**0.98**	**0.98**	**0.99**	**0.96**	**0.96**	**0.98**
	LPIPS ↓	**0.12**	**0.11**	**0.12**	**0.11**	**0.10**	**0.12**	**0.16**	**0.13**	**0.14**
SplaTAM [24]	PSNR ↑	**34.11**	**32.86**	**33.89**	**35.25**	**38.26**	**39.17**	**31.97**	**29.70**	**31.81**
	SSIM ↑	**0.97**	**0.98**	**0.97**	**0.98**	**0.98**	**0.98**	**0.97**	**0.95**	**0.95**
	LPIPS ↓	**0.10**	**0.07**	**0.10**	**0.08**	**0.09**	**0.09**	**0.10**	**0.12**	**0.15**
Ours	PSNR ↑	**34.38**	**32.15**	**34.31**	**35.01**	**37.01**	**37.52**	**33.72**	**32.15**	**33.16**
	SSIM ↑	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.97**
	LPIPS ↓	**0.09**	**0.08**	**0.08**	**0.08**	**0.09**	**0.12**	**0.08**	**0.08**	**0.12**

C. Evaluation of Mapping and Rendering

In Tab. IV, we list the quantitative training view rendering results of SupGS-SLAM compared with other state-of-the-art methods. The results indicate that SupGS-SLAM achieves excellent performance similar to that of other methods. Fig. 4 shows the qualitative comparison, which demonstrates that the rendering quality has been significantly improved.

D. Evaluation of Keyframe Selection Strategy

We select a small number of keyframes from all frames. The proportions are 3.4%, 6.1%, and 4.2% on the Replica, ScanNet, and TUM-RGBD datasets, respectively. Compared to SplaTAM [24] with 20% keyframe proportion using the constant time interval strategy, SupGS-SLAM significantly reduces the number of redundant keyframes while maintaining excellent performance.

E. Ablation Study

In Tab. V, we list the ablation study on the fr1_desk scene of the TUM-RGBD dataset [29]. The results demonstrate that with the keyframe strategy and supplementary densification strategy, the tracking results are much more accurate. Furthermore, with the supplementary global mapping strategy, the ATE RMSE decreases slightly, but the results of other metrics are significantly improved.

TABLE V

ABLATION STUDY ON THE SCENE FR1_DESK OF THE TUM-RGBD DATASET [29].

Keyframe	Densify	Global	ATE RMSE↓	PSNR ↑	Depth L1↓	SSIM ↑	LPIPS ↓
×	×	×	3.15	22.84	3.01	0.913	0.146
✓	×	×	2.98	22.21	3.74	0.846	0.223
✓	✓	×	2.89	21.72	3.90	0.884	0.187
✓	✓	✓	3.00	24.85	1.84	0.914	0.170

Keyframe ×: With constant time interval strategy.
Keyframe ✓: With keyframe strategy.
Densify: With/Without supplementary densification.
Global: With/Without supplementary global mapping.

V. CONCLUSION

We propose SupGS-SLAM, a Gaussian Splatting SLAM method. For better keyframe efficiency, we propose an efficient keyframe strategy to cover the entire scene with a small number of representative keyframes and prioritize the optimization of critical keyframes. For better mapping, we propose a supplementary mapping strategy to fully optimize the scene. Experiments demonstrate that SupGS-SLAM achieves low keyframe redundancy while maintaining excellent performance in tracking, mapping, and rendering.

REFERENCES

[1] Xudong Jiang, Lifeng Zhu, Jia Liu, and Aiguo Song. A slam-based 6dof controller with smooth auto-calibration for virtual reality. *Vis. Comput.*, 39(9):3873–3886, 2023.

[2] Denis Chekhlov, Andrew P. Gee, Andrew Calway, and Walterio W. Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual SLAM. In *Sixth IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, 13-16 November 2007, Nara, Japan*, pages 153–156. IEEE Computer Society, 2007.

[3] Charalambos Theodorou, Vladan Velisavljevic, Vladimir Dyo, and Fredi Nonyelu. Visual SLAM algorithms and their application for ar, mapping, localization and wayfinding. *Array*, 15:100222, 2022.

[4] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, 31(5):1147–1163, 2015.

[5] Zachary Teed and Jia Deng. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 16558–16569, 2021.

[6] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph.*, 36(3):24:1–24:18, 2017.

[7] Thomas Whelan, Renato F. Salas-Moreno, Ben Glocker, Andrew J. Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense SLAM and light source estimation. *Int. J. Robotics Res.*, 35(14):1697–1716, 2016.

[8] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: bundle adjusted direct RGB-D SLAM. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 134–144. Computer Vision Foundation / IEEE, 2019.

[9] Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison. DTAM: dense tracking and mapping in real-time. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2320–2327. IEEE Computer Society, 2011.

[10] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision, 3DV 2013, Seattle, Washington, USA, June 29 - July 1, 2013*, pages 1–8. IEEE Computer Society, 2013.

[11] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice F. Fallon, John J. Leonard, and John McDonald. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *Int. J. Robotics Res.*, 34(4-5):598–626, 2015.

[12] Fabio Ruetz, Emili Hernández, Mark Pfeiffer, Helen Oleynikova, Mark Cox, Thomas Lowe, and Paulo V. K. Borges. OVPC mesh: 3d free-space representation for local ground vehicle navigation. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 8648–8654. IEEE, 2019.

[13] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. Surfelmeshing: Online surfel-based mesh reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(10):2494–2507, 2020.

[14] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 6209–6218. IEEE, 2021.

[15] 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 Henry B. L. Duh, Ian Williams, Jens Grubert, J. Adam Jones, and Jianmin Zheng, editors, *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2022, Singapore, October 17-21, 2022*, pages 499–507. IEEE, 2022.

[16] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based SLAM. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 18387–18398. IEEE, 2023.

[17] 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 Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 12776–12786. IEEE, 2022.

[18] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. ESLAM: efficient dense SLAM system based on hybrid representation of signed distance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 17408–17419. IEEE, 2023.

[19] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, 2022.

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023.

[21] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 18039–18048. IEEE, 2024.

[22] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. GS-SLAM: dense visual SLAM with 3d gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 19595–19604. IEEE, 2024.

[23] Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. RTG-SLAM: real-time 3d reconstruction at scale using gaussian splatting. In Andres Burbano, Denis Zorin, and Wojciech Jarosz, editors, *ACM SIGGRAPH 2024 Conference Papers, SIGGRAPH 2024, Denver, CO, USA, 27 July 2024- 1 August 2024*, page 30. ACM, 2024.

[24] Nikhil Varma Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian A. Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense RGB-D SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21357–21366. IEEE, 2024.

[25] Mingrui Li, Shuhong Liu, Heng Zhou, Guohao Zhu, Na Cheng, Tianchen Deng, and Hongyu Wang. SGS-SLAM: semantic gaussian splatting for neural dense SLAM. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXXI*, volume 15089 of *Lecture Notes in Computer Science*, pages 163–179. Springer, 2024.

[26] Boying Li, Zhixi Cai, Yuan-Fang Li, Ian Reid, and Hamid Rezatofighi. Hier-slam: Scaling-up semantics in slam with a hierarchically categorical gaussian splatting. *CoRR*, abs/2409.12518, 2024.

[27] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Yuheng Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019.

[28] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2432–2443. IEEE Computer Society, 2017.

[29] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 573–580. IEEE, 2012.

[30] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 10371–10381. IEEE, 2024.