TAMBRIDGE: Bridging Frame-Centered Tracking and 3D Gaussian Splatting for Enhanced SLAM

Peifeng Jiang[1], Hong Liu[1,✉], Xia Li[2,✉], Ti Wang[1], Fabian Zhang[2], Joachim Buhmann[2]

[1]National Key Laboratory of General Artificial Intelligence,
Peking University, Shenzhen Graduate School
[2]Institute for Machine Learning, Department of Computer Science, ETH Zurich
jpf@stu.pku.edu.cn, hongliu@pku.edu.cn, xia.li@inf.ethz.ch
tiwang@stu.pku.edu.cn, fabzhang@ethz.ch, jbuhmann@inf.ethz.ch

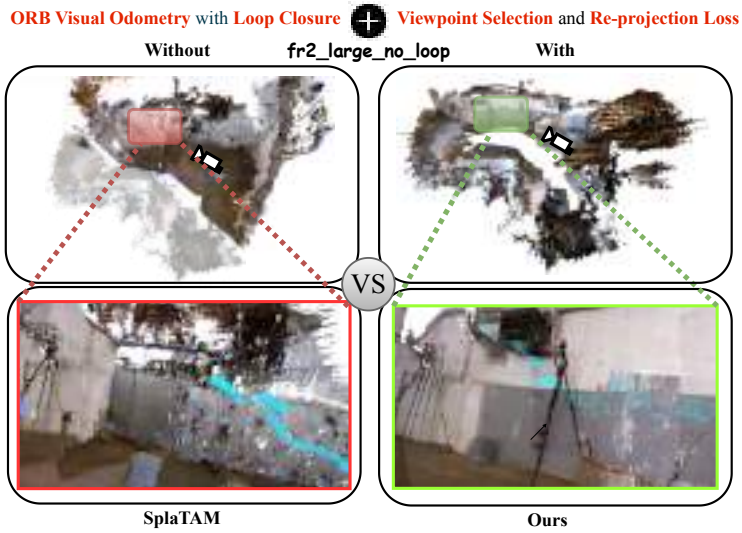

Figure 1: By seamlessly integrating ORB Visual Odometry with viewpoint selection and re-projection loss, our method significantly improves the robustness towards sensor noise and motion blur especially in long-session robotic tasks.

Abstract

The limited robustness of 3D Gaussian Splatting (3DGS) to motion blur and camera noise, along with its poor real-time performance, restricts its application in robotic SLAM tasks. Upon analysis, the primary causes of these issues are the density of views with motion blur and the cumulative errors in dense pose estimation from calculating losses based on noisy original images and rendering results, which increase the difficulty of 3DGS rendering convergence. Thus, a cutting-edge 3DGS-based SLAM system is introduced, leveraging the efficiency and flexibility of 3DGS to achieve real-time performance while remaining robust against sensor noise, motion blur, and the challenges posed by long-session SLAM. Central to this approach is the Fusion Bridge module, which seamlessly integrates tracking-centered ORB Visual Odometry with mapping-centered online 3DGS. Precise pose initialization is enabled by this module through joint optimization of re-projection and rendering loss, as well as strategic view selection, enhancing rendering convergence in large-scale scenes. Extensive experiments demonstrate state-of-the-art rendering quality and localization accuracy, positioning this system

Preprint. Under review.

as a promising solution for real-world robotics applications that require stable, near-real-time performance. Our project is available at `https://ZeldaFromHeaven.github.io/TAMBRIDGE/`

1 Introduction

Without prior knowledge, Real-time Simultaneous Localization and Mapping (SLAM) is essential for intelligent robots to form a comprehensive environmental understanding of the environment. Over the last two decades, the primary challenge in SLAM has been to balance computational demands between tracking accuracy and mapping fidelity. Frame-centered, sparse, and effective information can improve localization accuracy but lacks a high-level understanding of the entire scene. Conversely, mapping-centered, dense, and global information contributes to building a globally consistent map. Thus, the essence of the challenge lies in the contradiction between these two aspects: efficiency but sparse, and dense but noisy.

Methods prioritizing localization accuracy are known as "tracking-centered" methods, while those focusing on high-fidelity mapping are called "mapping-centered" methods. Initially, due to computational limitations, tracking-centered methods became a focal area of research since their high real-time performance, accuracy, and robustness in localization, including the use of feature point clouds [3, 21, 22], surfels [27, 34], depth maps [24, 28] and implicit representations [2, 23, 25]. However, these methods often lack the capacity for consistent scene understanding.

As computing capabilities evolve, the convergence issues in mapping-centered methods are gradually addressed through judicious selection and processing of scene representations. Concurrently, the operational efficiency of these paradigms is also steadily improving. A significant advancement is adopting neural radiance fields (NeRF) [18], which utilizes implicit neural networks for globally consistent representations and incorporates ray-cast triangulation for high-fidelity rendering and novel view synthesis. Nevertheless, the extensive computational demands of neural network inference and dense, per-pixel optimization calculations challenge the robustness of NeRF-based SLAM methods against sensor noise and motion blur in real-world robotic scenarios.

Recently, the 3D Gaussian Splatting (3DGS) [13] utilizes Gaussian primitives that are both interpretable and editable, serving as a globally consistent representation of the scene. This method facilitates even more rapid, differentiable rasterization rendering and generates higher-fidelity novel views through Gaussian splatting. SLAM systems incorporating 3DGS now achieve localization accuracy comparable to NeRF-based SLAM while significantly outperforming in terms of rendering quality and speed. However, these systems still face persistent challenges typical of mapping-centered SLAM, including non-real-time convergence rates and high sensitivity to random noise, which have difficulties in handling long-term, high-noise robotic tasks.

To solve this problem, our focus is to enhance the convergence capabilities of 3DGS towards online rendering. Random noise of real sensors and increasing perspective overlap are two main barriers to the rendering convergence speed, especially in long-session robotic tasks. Therefore, locally optimal poses from tracking-centered SLAM are leveraged to provide a robust initial estimate. By jointly optimizing sparse re-projection and dense rendering errors, the impact of image noise is effectively reduced. Additionally, a viewpoint selection strategy is adapted to effectively pick viewpoints suitable for online reconstruction, ensuring viewpoint sparsity and reducing overlap. The above two processes are integrated into a plug-and-play Fusion Bridge module, which acts as an intermediary between the tracking-centered frontend (ORB [26] Visual Odometry) and the mapping-centered backend (Online 3DGS). As shown in Figure 1, our method obtains better tracking accuracy for long-session tasks by using the ORB Visual Odometry's loop closure to reduce the accumulated trajectory error. The introduction of viewpoint selection and re-projection loss in joint optimization helps us to eliminate the view overlapping problem and enhance the robustness towards sensor noise, resulting in better rendering performance. Extensive experiments on real-world datasets show that our method achieves state-of-the-art (SOTA) rendering quality and localization accuracy in long-session robotic tasks. It is also the first 3DGS-based SLAM system capable of consistently achieving (5+ FPS) real-time performance. Our contributions can be summarized as follows:

- The significance of precise pose initialization and strategic viewpoint selection in facilitating the convergence of online 3DGS, especially in large-session tasks, is emphasized.

- A plug-and-play Fusion Bridge module is introduced, seamlessly integrating tracking-centered SLAM frontend with online 3DGS-centered backend.
- A SLAM system is developed that consistently delivers stable, near real-time performance (>5 FPS) for long-session robotic tasks.

2 Related Works

2.1 Tracking-centered and Mapping-centered SLAM

Tracking-centered SLAM has evolved into three primary approaches: feature-based, deep learning-based, and hybrid. Feature-based approaches like MonoSLAM [3], PTAM [14], and ORB-SLAM [21, 22] utilize traditional strategies such as extended Kalman Filters [8] and place recognition to estimate camera motion real-time, with ORB-SLAM standing out among feature-based methods for its effective integration of tracking, local mapping, and loop closing, ensuring real-time performance. Deep learning-based methods, such as DeepVO [6, 33] and UnDeepVO [15], employ neural networks to directly estimate camera poses and environmental depth, offering innovative solutions but facing accuracy challenges. Hybrid approaches like DF-SLAM [11, 22] and SuperPointVO [4, 5] merge deep learning for feature extraction with traditional frameworks, aiming to bridge the gap between conventional methods and modern deep learning innovations. These hybrid systems show promise in improving the accuracy and robustness of SLAM, making them a significant development in the field.

The mapping-centered SLAM paradigm emphasizes the util of 3D representation within the framework and leverages global information through consistently reconstructed 3D maps from tracking data. A prominent technology in this domain is the NeRF, which allows for high-fidelity view-synthesis through the implicit representation based on neural networks. A growing body of dense neural SLAM methods [10, 30, 32, 35, 38, 39] has been developed, which build a high-fidelity map and achieve the same level of tracking accuracy compared with tracking-centered works.

2.2 NeRF based SLAM

The pursuit of photorealistic scene capture has popularized differentiable volumetric rendering, notably through techniques like NeRF. These methods use a single multilayer perceptron (MLP) to derive opacity and color along pixel rays, optimized via multiview information. Due to the high computational costs of training these models, alternatives like multi-resolution voxel grids [7, 16, 39] and hashing functions [20] have been developed. Techniques such as Point-NeRF [35] expedite 3D reconstruction using neural point clouds and feature interpolation, maintaining high rendering quality without relying on MLPs. The evolution of differentiable volumetric rendering highlights the significance of developing efficient rendering techniques to facilitate high-fidelity 3D reconstructions in fields like robotics and computer vision, by moving beyond traditional MLP-based methods.

2.3 3DGS based SLAM

Utilizing 3D Gaussian Splatting (3DGS) for scene representation, initiatives like SplaTAM [12], MonoGS [17], GS-SLAM [36], and Photo-SLAM [9] achieve efficient, deformable, and modifiable geometry. These methods opt for Gaussian splatting over ray marching as in NeRF, resulting in faster, high-fidelity rendering. SplaTAM employs isotropic 3D Gaussians for scene reconstruction, facilitating rapid differentiable rasterization. It segments the SLAM process into localization, densification, and mapping, achieving high localization accuracy and superior reconstruction capabilities. However, its localization phase is notably sensitive to motion blur, depth noise, and aggressive rotations, which along with significant time overheads, limits its application in robotics. Similarly, MonoGS uses a comparable framework with enhanced features like analytic Jacobian determinants and Gaussian shape regularization for camera pose estimation, extending input capabilities from RGBD to monocular systems. Although it surpasses NeRF-based SLAM in reconstruction performance, it shares similar sensitivities to motion conditions and exhibits increased time overhead due to per-pixel photometric and geometric residuals, impacting robustness to noise.

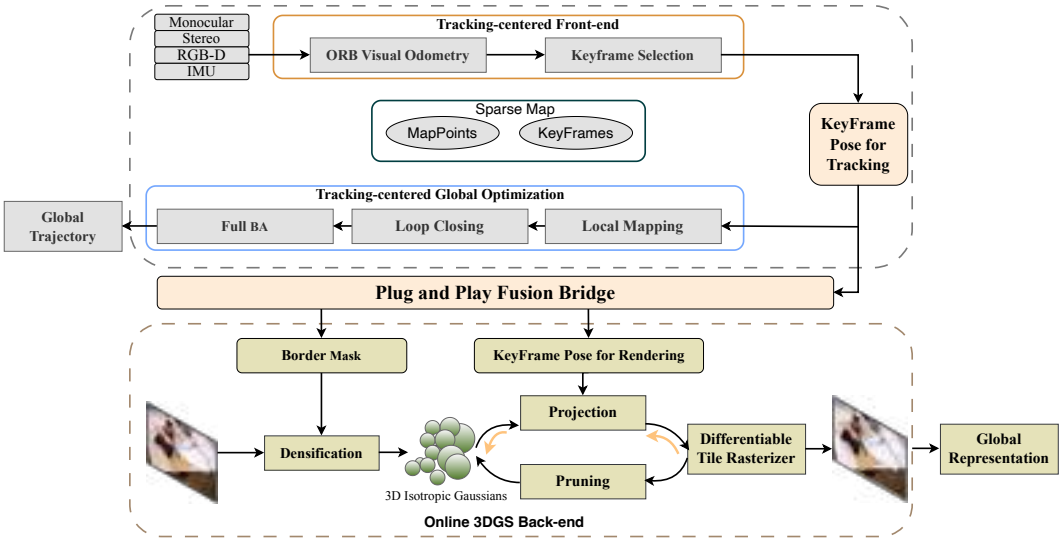

Figure 2: Overview. ORB-based visual odometry analyzes RGB-D data to estimate poses and select keyframes for local mapping. These keyframes feed into the Global Optimization featuring Bundle Adjustment (BA) to refine the path. Simultaneously, the Fusion Bridge module selects reconstruction frames and calculates rendering poses and Border Masks. An online 3DGS backend processes the selected frames to create a globally consistent, high-fidelity scene representation.

3 Method

To address convergence issues in 3DGS-based SLAM systems caused by sensor noise, motion blur, and overlapping views as shown in Figure 2, we propose the following modifications:

- **The Tracking-centered Frontend Module** obtains initial pose estimates and calculates keyframe sequences using an ORB visual odometry module, similar to ORB-SLAM3.
- **The Tracking-centered Global Optimization Module** optimizes the final trajectory through global Bundle Adjustment (BA) after local mapping and loop closing.
- **The Plug and Play Fusion Bridge Module** utilizes covisibility to select reconstruction frames from the keyframe sequence further and jointly optimizes rendering poses and border masks by minimizing re-projection errors and color-depth rendering errors.
- **The Online 3DGS Backend Module** uses the poses of reconstruction frames and border masks to construct a high-fidelity, globally consistent Gaussian scene representation nearly in real-time.

The coarse-to-fine pose estimation during the Tracking-centered frontend and Global Optimization is described in Section 3.1. Section 3.2 will elaborate on the Plug and Play Fusion Bridge module, and Section 3.3 will discuss the online 3DGS reconstruction.

3.1 Pose Estimation

In the Tracking-centered Frontend and Global Optimization process, pose is refined from coarse to fine based on a sparse feature point cloud based on Bundle Adjustment (BA) [31].

In the Tracking-centered Frontend, re-projection error, used for short-term data associations, is calculated from the matched ORB feature points extracted from consecutive frames as follows:

$$L_{rpj} = \min_{\{[R|t]_j\}} \sum_{ij} \left\| u_{ij} - \pi \left(\mathcal{C}_j, P_i \right) \right\|_2 . \tag{1}$$

The projection $\pi(\mathcal{C}_j, P_i)$ converts a 3D point $P_i = [X, Y, Z]^T$ to pixel position u_{ij} using intrinsic parameters K_j and extrinsic parameters $[R|t]_j$, which include rotation R_j and translation t_j of camera $\mathcal{C}_j$. The process optimizes camera poses $\{[R|t]_j\}$ by minimizing the reprojection error L_{rpj}.

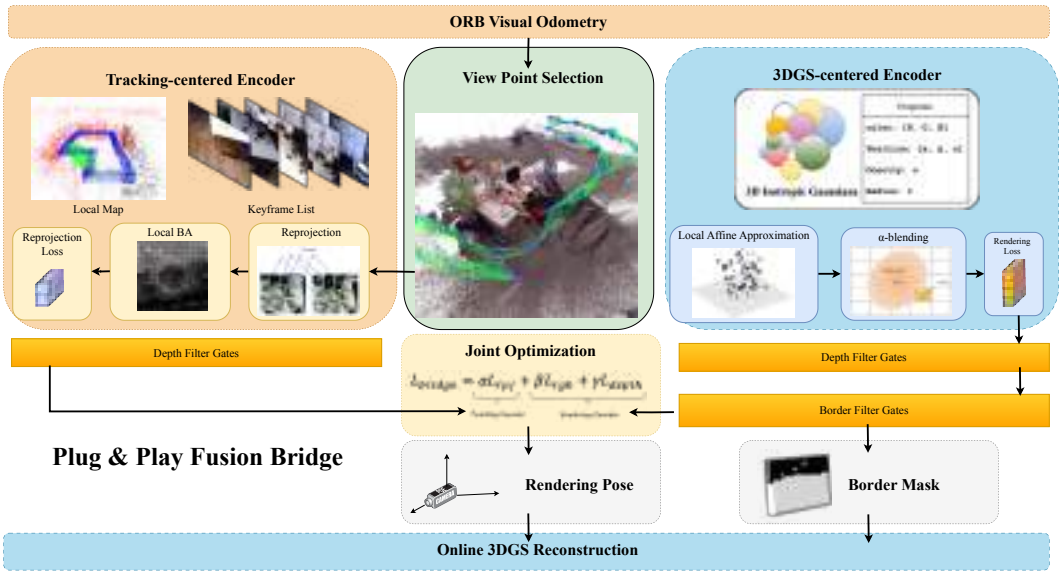

Figure 3: The Fusion Bridge module selects reconstruction keyframes from a local map based on viewpoint covisibility. It projects the 3DGS and local map point cloud onto the reconstruction frame, filtering projections through pixel gates. The module then optimizes the pose by jointly minimizing rendering and point cloud reprojection losses, setting the initial rendering pose for the Online 3DGS.

During global optimization, new map points are added to the local map by triangulation. The reprojection error for both camera poses of keyframes (described by K and $[R|t]$) and the observed map points is minimized through full BA, detailed in equation 2. This process constitutes a nonlinear least squares problem, tackled using the Levenberg-Marquardt method according to [19].

$$L_{rpj} = \min_{\{[R|t]_j\},P_i} \sum_{ij} \left\| u_{ij} - \pi\left(\mathcal{C}_j, P_i\right) \right\|_2 . \tag{2}$$

3.2 Plug and Play Fusion Bridge

The fusion bridge process shown in Figure 3 consists of five components: **viewpoint selection**, a **Tracking-centered encoder**, a **3DGS-centered encoder**, **filter gates**, and a **joint optimization module**. This process aims to minimize the gap between the sparse point cloud generated by visual odometry and the backend used for Gaussian Splatting, specifically 3DGS with selected viewpoints to get proper and sparse poses for reconstruction.

The View Point Selection involves selecting advantageous viewpoints for rendering by filtering through the covisibility relationships among keyframes. The covisibility between two frames should fall within a closed interval, balancing sufficient matching points for calculating reprojection errors and maintaining adequate diversity between viewpoints:

$$R_i = \begin{cases} 1 & \text{if } \beta \leq M_{i-1,i} < \alpha \cdot T_{i-1,i} \\ 0 & \text{otherwise} \end{cases} , \tag{3}$$

where R_i indicates if the current keyframe i is selected for reconstruction (1 for yes, 0 for no). $M_{i-1,i}$ counts the matched feature points. $T_{i-1,i}$ is the total number of matches in the previous keyframe, which, alongside α, sets the percentage threshold for maximum matches, while β specifies the minimum required matches for selection.

Tracking-centered Encoder encodes the sparse point cloud generated by visual odometry, enhancing the anchoring effect of robust feature-based tracking poses. By re-projecting the visible map points from the local map of the rendering frame, the reprojection loss can be optimized as Equation 1.

3DGS-centered Encoder encodes the 3DGS used for Gaussian Splatting, where local affine transformations project these isotropic 3D Gaussians onto the camera plane, and the alpha-blending algorithm is then used to render the color and depth image, obtaining the rendering loss L_{rgb} and

L_{depth}. Additionally, the border masks $M_b(\mathbf{p})$ that reflect the uncertainty of rendered pixels $\mathbf{p}$ are computed based on the opacity information of the Gaussians as follows:

$$M_b(\mathbf{p}) = \sum_{i=1}^{n} f_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - f_j(\mathbf{p})). \tag{4}$$

Filter Gates eliminate noise during the optimization. The depth gates filter out depth anomalies from point clouds and Gaussians. The border gate filters out rendered pixels with high uncertainty according to the border mask:

$$G(\mathbf{p}) = \begin{cases} 1 & \text{if } D_c(\mathbf{p}) > 0, D_g(\mathbf{p}) > 0 \text{ and } M_b(\mathbf{p}) > \gamma \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where $G(\mathbf{p})$ indicates if the current pixel $\mathbf{p}$ is selected for optimization (1 for yes, 0 for no). $D_c(\mathbf{p})$ and $D_g(\mathbf{p})$ is the depth of cloud point and Gaussian at pixel $\mathbf{p}$. γ is the threshold of the minimize border mask value of $\mathbf{p}$.

Joint Optimization employs alternating optimization based on the rendering error of depth and color, as well as the reprojection error based on feature coordinates:

$$\mathbf{L} = w_1 \cdot \mathbf{L}_{rpj} + w_2 \cdot \mathbf{L}_c + w_3 \cdot \mathbf{L}_d. \tag{6}$$

Here, $\mathbf{L}$ denotes the aggregate loss, incorporating the reprojection loss of the point cloud features, $\mathbf{L}_{rpj}$, the rendering loss of the Gaussian color $\mathbf{L}_c$ and depth $\mathbf{L}_d$. The coefficients w_1, w_2 and w_3 respectively represent the weight hyper-parameters for these three types of loss.

3.3 Online 3DGS Reconstruction

The online 3DGS reconstruction method enables near real-time reconstruction using sparse frames and border masks filtered through the Fusion Bridge module. Differing from primary 3DGS works, SLAM operates without global pose and point cloud priors, requiring incremental reconstruction based on the current and preceding frames. Presently, 3DGS-based SLAM typically utilizes incremental densification, initializing under-reconstructed pixels in the current frame as new 3DGS, marking the key distinction between online and offline 3DGS.

Gaussian Representation. Similar to SplaTAM, we simplified the model from [13] by using view-independent colors and isotropic Gaussians. Each Gaussian is defined by eight parameters: three for RGB color $\mathbf{c}$, three for center position $\boldsymbol{\mu} \in \mathbb{R}^3$, one for radius r, and one for opacity $o \in [0, 1]$, as illustrated in Figure 3. Each Gaussian affects a point $\mathbf{x}$ in 3D space, belonging to $\mathbb{R}^3$, based on the standard (unnormalized) Gaussian equation, with the influence modulated by its opacity.

$$f(\mathbf{x}) = o \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\mu}\|^2}{2r^2}\right). \tag{7}$$

Densification. Pixels outside the Border Mask are under-reconstructed, lacking a robust Gaussian representation, necessitating densification. During the first frame, all pixels from that frame are selected for reconstruction. During the reconstruction, the new Gaussians are added with a radius of one pixel length on the pixel plane as shown in equation 8, where f represents the focal length.

$$d = rf. \tag{8}$$

Projection. Global Gaussians are projected onto the rendering pose plane via local affine transformations, resulting in a splatted 2D projection of each Gaussian.

Differentiable Rasterization. Gaussian Splatting involves rendering an RGB image by first arranging a set of 3D Gaussians in a front-to-back order relative to the camera pose. The RGB image is then efficiently generated by alpha-blending the 2D projections of these splatted Gaussians in sequential order within the pixel space. The color of a rendered pixel $\mathbf{p} = (u, v)$ is defined by the equation:

$$C(\mathbf{p}) = \sum_{i=1}^{n} \mathbf{c}_i f_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - f_j(\mathbf{p})), \tag{9}$$

where each $f_i(\mathbf{p})$ is calculated using Equation 7. The specific equations for transforming the Gaussians to 2D pixel-space are:

$$\boldsymbol{\mu}^{\text{2D}} = K\frac{E_t\boldsymbol{\mu}}{d}, \quad r^{\text{2D}} = \frac{fr}{d}, \tag{10}$$

Here, K represents the camera's intrinsic matrix, E_t is the extrinsic matrix indicating the camera's rotation and translation at frame t, f is the known focal length, and d is the depth of each Gaussian in camera coordinates which is defined as follows:

$$D(\mathbf{p}) = \sum_{i=1}^{n} d_i f_i(\mathbf{p}) \prod_{j=1}^{i-1}(1 - f_j(\mathbf{p})), \tag{11}$$

The Gaussian parameters are iteratively refined using gradient-based optimization by minimizing the specified loss, while the pose of the camera is kept constant:

$$\mathbf{L}_m = (1-\zeta)(w_4\mathbf{L}_1(D(\mathbf{p}) + w_5\mathbf{L}_1(C(\mathbf{p})) + \zeta\mathbf{L}_{SSIM}. \tag{12}$$

This is an L1 loss on both the depth and color renders, with different weights w_4 and w_5. Furthermore, we add an SSIM loss to RGB rendering.

Gaussian Pruning. After updating the Gaussian parameters, Gaussians with opacity below zero or excessively high are removed during rendering, as they are deemed redundant or non-contributory to the reconstruction:

$$P(\mathbf{p}) = \begin{cases} 0 & \text{if } 0 \leq o < \tau \\ 1 & \text{otherwise} \end{cases}, \tag{13}$$

where $P(\mathbf{p})$ indicates if the pixel $\mathbf{p}$ is pruned (1 for yes, 0 for no) and τ is the threshold of opacity.

4 Experiments

4.1 Experimental Setup

Implementation Details. Experiments were performed with an Intel i7-9700 CPU and an NVIDIA RTX 3090 GPU. We developed the TAMBRIDGE code in Python, C++, and CUDA, computed reprojection errors using the G2O framework with the Levenberg-Marquardt algorithm and used the differentiable rasterization modules from the paper [13].

Dataset and Baselines. The TUM RGB-D dataset [29], collected by Pioneer robots and handheld devices, is characterized by sensor noise, motion blur, and aggressive rotations. These 18 sequences were categorized into four types: common, long-session mobile, handheld, and tiny-motion. For the common type including 3 sequences, we conducted comparisons with baseline methods of NeRF-based SLAM, SplaTAM, the SOTA of 3DGS-based SLAM, and ORB-SLAM3. Comparisons were also made with SplaTAM in the other three categories including 15 sequences, since most NeRF-based methods didn't share settings on these sequences.

Evaluation Metrics. For the precision of the localization, we calculated the Root Mean Square Error (RMSE) between the ground truth and the estimated trajectories. For online 3DGS reconstruction, we further adapt the peak signal-to-noise ratio (PSNR), SSIM, and LPIPS by following [12] to evaluate the rendering performance of each frame. For real-time performance, we measure the frame rate (FPS) of the whole system.

4.2 Evaluation

Evaluation on Common Sequences. On these three sequences commonly used in SLAM domain, we compare two versions of our method with different FPS, as shown in Table 1. Towards the version of steadily achieve **5 FPS**, our localization performance significantly surpasses, **at least 3 times**, than other methods, matching the level of tracking-based SLAM like ORB-SLAM3. In terms of rendering quality, our method **greatly outperforms** other NeRF-based SLAM methods and is **on par** with SplaTAM. Regarding real-time performance, our method achieves a **near real-time** performance (5 FPS) for optimal localization and rendering that is more than **6 times** the average frame rate of SplaTAM and on average **50 times** over other NeRF-based methods. Towards the version with an

Table 1: **Tracking and Rendering Performance on TUM-RGBD [29].**

Common Sequences	fr1-desk				fr2-xyz				fr3-office			
Method	RMSE (cm)↓	PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓
ORB-SLAM3 [1]	2.06	-	-	-	0.33	-	-	-	1.04	-	-	-
Nice-SLAM [39]	19.32	12.00	0.42	0.51	36.10	18.20	0.60	0.31	25.31	16.34	0.55	0.39
Vox-Fusion [37]	3.52	15.79	0.65	0.52	1.49	16.32	0.71	0.43	26.01	17.27	0.68	0.46
Point-SLAM[35]	4.34	13.87	0.63	0.54	31.73	17.56	0.71	0.59	3.87	18.43	0.75	0.45
ESLAM [10]	3.36	17.49	0.56	0.48	31.44	22.22	0.72	0.23	25.80	19.11	0.61	0.35
Co-SLAM [32]	3.09	16.41	0.48	0.59	31.34	19.17	0.59	0.37	25.37	17.86	0.54	0.45
Go-SLAM [38]	2.11	15.79	0.53	0.53	31.78	16.11	0.53	0.41	26.80	16.49	0.56	0.56
SplaTAM [12]	3.38	**22.00**	0.86	0.23	1.34	**24.50**	**0.95**	**0.10**	5.03	**21.90**	**0.88**	**0.20**
Ours (5 FPS)	**1.75**	21.22	**0.88**	**0.19**	**0.32**	23.44	0.90	**0.10**	**1.42**	20.15	0.82	0.25
Ours (10 FPS+)	**1.75**	17.98	0.75	0.27	**0.32**	20.74	0.85	0.15	**1.42**	16.59	0.60	0.40

Table 2: **Tracking and Rendering Performance on TUM-RGBD [29] Mobile Slam Sequence**. Convergence failure is represented by "✗" .

Mobile SLAM	fr2-pio-360				fr2-pio-slam				fr2-pio-slam2			
Method	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓
SplaTAM [12]	101.37	**24.33**	**0.82**	**0.22**	✗	✗	✗	✗	627.31	13.62	0.42	0.54
Ours	**8.43**	24.32	0.79	0.24	**10.06**	**23.54**	**0.80**	**0.24**	**6.7**	**24.52**	**0.82**	**0.19**

average **10 FPS+** frame rate with the same tracking accuracy, the rendering quality of our method is still **on par with** the SOTA NeRF-based method with on average over **100 times** faster than these methods, achieving **real-time** performance.

Evaluation on Long-Session SLAM Sequences. Towards robotic long-session SLAM tasks that typically involve larger scenes and longer task durations, our method **consistently outperforms** SplaTAM in terms of localization accuracy and rendering quality. As shown in Table 2, SplaTAM exhibits significant localization errors and inferior reconstruction quality over long distances. Our methods, which steadily achieve **5 FPS** near real-time performance, by employing viewpoint filtering and robust initial estimates from joint optimization of reprojection errors, enables the algorithm to converge to a better rendering performance **20 times faster** than SplaTAM in long-duration and long-distance rendering tasks.

Evaluation on Handheld Sequences. Handheld sequences, taken by a handheld camera for extended periods, underscore the other methods vulnerability to motion blur. From Table 3, SplaTAM fails at **fr2/hemi** and **fr2/kidn** because of its extended periods and great motion blur, while our method can achieve consistently accurate tracking and high-fidelity rendering with an average PSNR over than **20** in a near real-time performance (5 FPS).

Table 3: **Tracking and Rendering Performance on TUM-RGBD [29] Handheld Slam Sequence**. Convergence failure is represented by "✗" .

Method	Metric	fr1/360	fr1/floor	fr2/hemi	fr2/kidn	fr2/desk	fr2/nloop	fr2/wloop	Avg.
SplaTAM* [12]	RMSE↓	**8.64**	78.9	✗	✗	99.74	211.9	206.27	✗
	PSNR↑	**18.41**	18.77	✗	✗	19.26	17.89	**21.19**	✗
	SSIM↑	**0.74**	0.65	✗	✗	0.77	**0.72**	0.81	✗
	LPIPS↓	**0.27**	0.38	✗	✗	0.34	**0.27**	**0.21**	✗
Ours	RMSE ↓	10.67	**5.98**	**13.93**	**8.22**	**1.02**	**18.48**	**1.42**	**8.53**
	PSNR ↑	16.93	**20.55**	**20.63**	**22.85**	**20.16**	**19.49**	20.15	**20.10**
	SSIM ↑	0.67	**0.71**	**0.77**	**0.87**	**0.8**	**0.72**	**0.82**	**0.77**
	LPIPS↓	0.32	**0.27**	**0.24**	**0.15**	**0.21**	0.29	0.25	**0.25**

4.3 Qualitive Results

As illustrated in Figure 4, our method achieves rendering quality that surpasses NeRF-based SLAM and is comparable to SplaTAM, all while maintaining a near-real-time frame rate significantly higher than other approaches, at 5 FPS.

Table 4: **Experimental Results without Fusion Bridge.**

Common Sequences	fr1-desk			fr2-xyz			fr3-office		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Ours (with bridge)	**21.22**	**0.88**	**0.19**	**23.44**	**0.90**	**0.10**	**20.15**	**0.82**	**0.25**
Ours (no bridge)	13.15	0.45	0.48	14.59	0.59	0.35	15.59	0.53	0.46

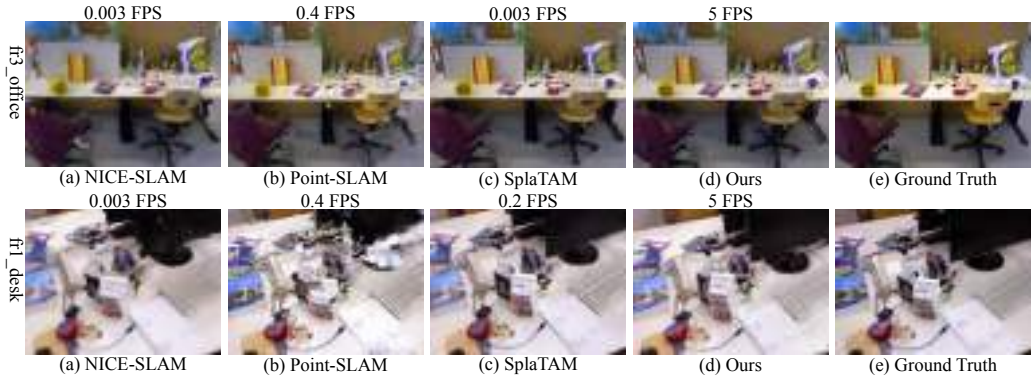

Figure 4: Qualitative results on TUM RGBD fr1_desk and fr3_office sequences.

4.4 Ablations

Plug and Play Fusion Bridge. It is observable that without integrating the modality fusion module, there is a general decline in the reconstruction quality metrics, as shown in Table 4. This deterioration is attributable to the larger gap between the ORB frontend poses and the Gaussian Splatting, which tends to accumulate over the course of the task. The specific reconstruction results can be seen in the figure below. This highlights one of the most crucial aspects of this work: identifying and demonstrating the significance of the fusion process in bridging traditional SLAM frontend with 3DGS backend. This integration enhances the accuracy and quality of mapping and underscores the necessity of effective pose and data integration techniques in SLAM systems.

FPS-Induced Rendering Degradation. In robotics perception tasks where real-time performance is crucial, we investigate how rendering quality evolves with increasing frame rates by adjusting the number of optimization iterations t and the perspective selection threshold k in the Fusion Bridge module. We offer a detailed set of quality-frame rate curves, as shown in Figure 5, to facilitate the selection of suitable hyperparameters for specific applications to achieve a balance between real-time performance and rendering quality. Additionally, our approach not only significantly outperforms SplaTAM and NICE-SLAM in terms of frame rate but also maintains a consistent near-real-time performance, consistently exceeding **5 FPS with best rendering quality**, consistently exceeding **9 FPS with competent rendering quality** with SOTA NeRF-based methods, and even achieve **a maximum 30 FPS+ performance** on the scenes with great viewpoint overlappings like fr2/xyz.

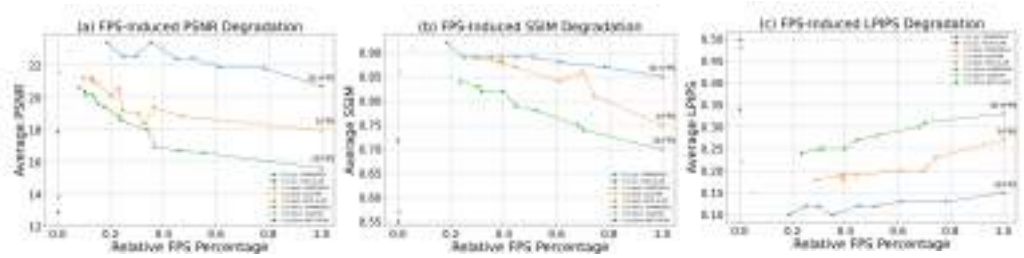

Figure 5: (a) The change in TAMBRIDGE PSNR with FPS. (b) The change in TAMBRIDGE SSIM with FPS. (c) The change in TAMBRIDGE LPIPS with FPS.

5 Conclusion

Our goal was to develop a 3DGS-based SLAM system suitable for intelligent robotic perception tasks. The system is designed to meet the real-time requirements of robotic perception, and is robust against random sensor noise, motion blur and challenges posed by long-session SLAM. To achieve this, we have implemented a plug-and-play Fusion Bridge module that filters redundant views and leverages the anchoring effect of reprojection errors to estimate poses for rendering initially. This approach marks the first integration of a tracking-centered paradigm with an ORB-based visual odometry frontend together with a 3DGS-centered backend. Our system consistently achieves real-time localization and near-real-time rendering at over 5 FPS on the real-world TUM RGB-D dataset. Furthermore, it significantly surpasses SplaTAM in localization accuracy, rendering quality, and robustness in scenarios involving motion blur and long-distance SLAM tasks.

Acknowledgments and Disclosure of Funding

This project is supported by the National Natural Science Foundation of China (No.62373009) and the interdisciplinary doctoral grant (iDoc 2021-360) from the Personalized Health and Related Technologies (PHRT) of the ETH domain, Switzerland.

References

[1] Carlos Campos, Richard Elvira, Juan J. G´omez, Jos'e M. M. Montiel, and Juan D. Tard'os. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics (TRO)*, 37(6):1874–1890, 2021.

[2] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017.

[3] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 29(6):1052–1067, 2007.

[4] Daniel Detone, Tomasz Malisiewicz, and Andrew Rabinovich. Self-improving visual odometry. *arXiv preprint arXiv:1812.03245*, 2018.

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, pages 224–236, 2018.

[6] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015.

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022.

[8] Masaru Hoshiya and Etsuro Saito. Structural identification by extended kalman filter. *Journal of engineering mechanics (J ENG MECH)*, 110(12):1757–1770, 1984.

[9] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. *arXiv preprint arXiv:2311.16728*, 2023.

[10] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. In *CVPR*, pages 17408–17419, 2023.

[11] Rong Kang, Jieqi Shi, Xueming Li, Yang Liu, and Xiao Liu. Df-slam: A deep-learning enhanced visual slam system based on deep local features. *arXiv preprint arXiv:1901.07223*, 2019.

[12] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. *arXiv preprint arXiv:2312.02126*, 2023.

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[14] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, pages 225–234. IEEE, 2007.

[15] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. *CoRR*, abs/1709.06841, 2017.

[16] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[17] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *arXiv preprint arXiv:2312.06741*, 2023.

[18] 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. *Communications of the ACM (COMMUN ACM)*, 65(1):99–106, 2021.

[19] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*, pages 105–116. Springer, 2006.

[20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

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

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

[23] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136. Ieee, 2011.

[24] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, pages 2320–2327. IEEE, 2011.

[25] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013.

[26] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, pages 2564–2571, 2011.

[27] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *CVPR*, pages 134–144, 2019.

[28] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. Real-time dense geometry from a handheld camera. In *Joint Pattern Recognition Symposium*, pages 11–20. Springer, 2010.

[29] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, pages 573–580, 2012.

[30] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *ICCV*, pages 6229–6238, 2021.

[31] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, pages 298–372. Springer, 2000.

[32] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In *CVPR*, pages 13293–13302, 2023.

[33] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *ICRA*, pages 2043–2050, 2017.

[34] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. In *RSS*, volume 11, page 3, 2015.

[35] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, pages 5438–5448, 2022.

[36] Chi Yan, Delin Qu, Dong Wang, Dan Xu, Zhigang Wang, Bin Zhao, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. *ArXiv*, abs/2311.11700, 2023.

[37] 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 *ISMAR*, pages 499–507, 2022.

[38] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *ICCV*, pages 3727–3737, 2023.

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

A Supplementary Material

A.1 Implementation Details of TAMBridge

The detailed implementation of hyper-parameters are shown in Table 5. α sets the percentage threshold for maximum matching, and β sets the minimum numbers of matching points in Viewpoint Selection; γ is the threshold of the minimize border mask value of each pixel; ζ is the weight for SSIM loss and L1 loss in Online 3DGS Mapping; τ is the maximum threshold of opacity used in Gaussian Pruning; w_1, w_2, w_3 are the weights of reprojection loss, color rendering loss and depth rendering loss, respectively, in Fusion Bridge; w_4 and w_5 are the weights for depth rendering loss and color rendering loss in Online 3DGS Mapping.

Table 5: Hyper-parameter Values of Implementation

Hyper-parameters	α	β	γ	ζ	τ	w_1	w_2	w_3	w_4	w_5
Value	0.75	20	0.99	0.3	0.005	1.5	0.5	1	0.5	1

A.2 Evaluation on tiny motion sequences

The table clearly shows that in sequences with minimal movement, our method outperforms SplaTAM in localization accuracy and matches its reconstruction quality. Due to the presence of many redundant viewpoints in these sequences, where the camera movement is confined to a small area, our selection mechanism achieves superior localization precision and facilitates easier convergence to a comparable level of reconstruction quality. Particularly, in the fr2_rpy sequence, our method has successfully addressed the convergence issues that previously affected SplaTAM as shown in Table 6.

Table 6: **Tracking and Rendering Performance on TUM-RGBD [29] Tiny Motion Sequence**. Convergence failure is represented by "✗" .

Method	Metric	fr1/xyz	fr1/rpy	fr2/rpy	Avg.
ORB-SLAM3 [1]	RMSE↓	**1.02**	1.9	0.46	1.13
	PSNR↑	–	–	–	–
	SSIM↑	–	–	–	–
	LPIPS↓	–	–	–	–
SplaTAM* [12]	RMSE↓	3.38	✗	5.03	✗
	PSNR↑	**22.89**	✗	**21.29**	✗
	SSIM↑	**0.91**	✗	**0.86**	✗
	LPIPS↓	**0.15**	✗	**0.22**	✗
Ours	RMSE↓	**1.29**	**0.32**	**1.42**	**1.01**
	PSNR↑	21.08	**23.16**	20.15	**22.33**
	SSIM↑	0.88	**0.91**	0.82	**0.87**
	LPIPS↓	0.19	**0.10**	0.25	**0.19**

A.3 Reconstruction Quality on TUM RGB-D

Figure 6 shows the visualization results on 8 sequences of the TUM RGB-D dataset, where all the perspectives in the images are novel viewpoints, and the sequences contain rich sensor noise and motion blur.

A.4 FPS Induced Rendering Degradation with Hyper-paramters Values

Based on the FPS-Quality 5 curves presented in the experimental section 4, we have supplemented the data points with corresponding hyperparameter information, as shown in Figure 7, Figure 8, and Figure 9. Specifically, It represents the number of iterations for the joint optimization cycle, and k represents the value of α in the Fusion Bridge module.

Figure 6: Reconstruction results on TUM RGBD sequences.

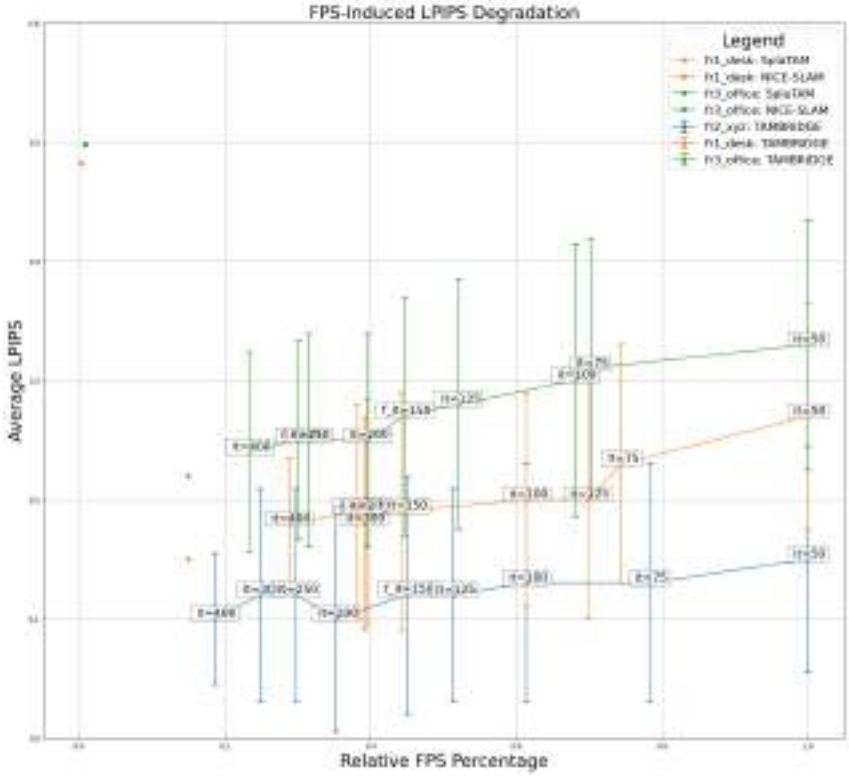

Figure 7: Reconstruction results on TUM RGBD sequences.

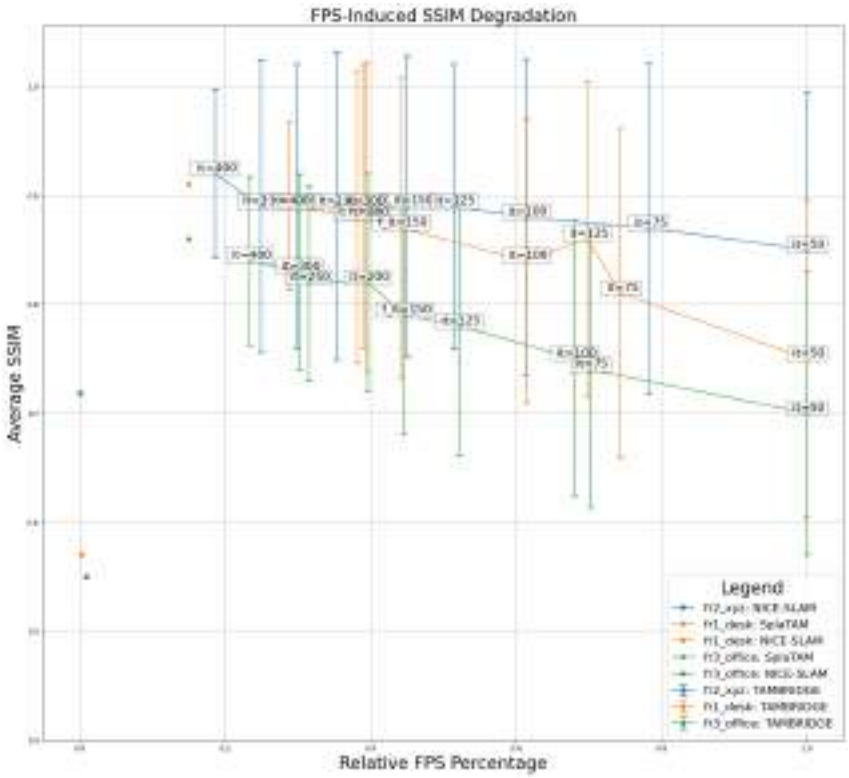

Figure 8: Reconstruction results on TUM RGBD sequences.

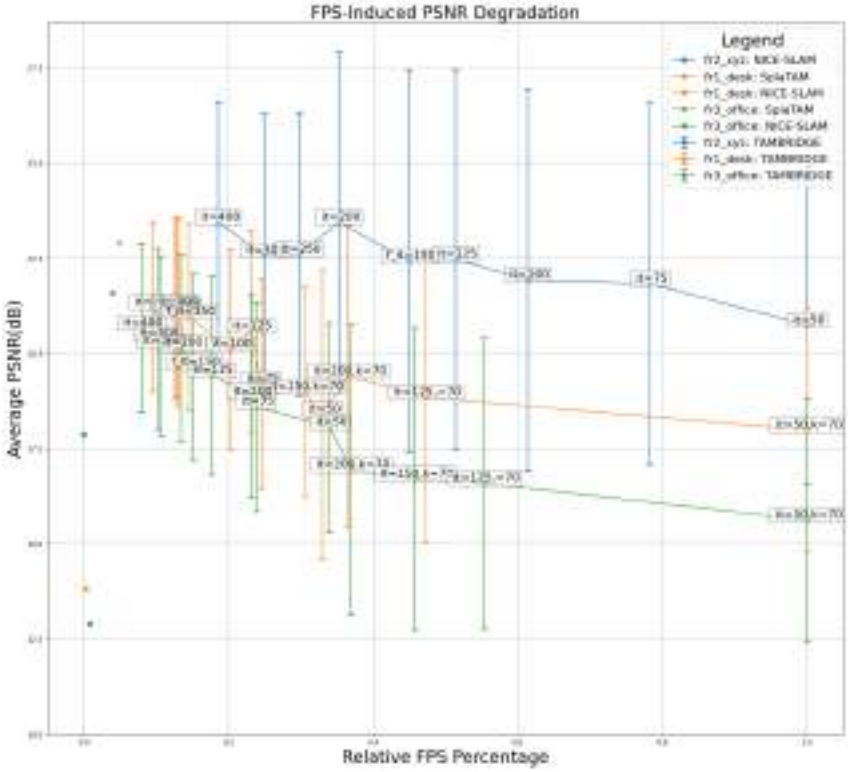

Figure 9: Reconstruction results on TUM RGBD sequences.

B Limitations and Future Work

B.1 Limitations of Method

The limitations of our method are listed as follow:

1. The Viewpoint Selection module of the Fusion Bridge, which selects based on the number of keypoint matches and manual thresholds, lacks self-learning capabilities. It can be considered to utilize neural networks such as CNN or Transformer to learn autonomously and determine the degree of viewpoint overlap. Additionally, other factors like sensor noise can also be introduced to filter the original image quality.

2. After the viewpoint selection, the reconstructed map may contain some voids, which, although having no practical impact on scene understanding and usage, need to be filled with effective methods or avoided during the selection process.

3. The online 3DGS module builds upon existing 3DGS-based SLAM methods. However, many lightweight improvements to 3DGS have been proposed in the field of 3D reconstruction. Adapting these new 3DGS methods for online reconstruction could further enhance the real-time performance and effectiveness of the current approach.

B.2 Limitations of Experiment

1. **Metrics.** SplaTAM and other 3DGS-based SLAM are commonly evaluated using metrics such as PSNR, LPIPS, and SSIM. However, most of these metrics calculate their values by averaging each frame in the sequence, which does not accurately assess the overall quality of the environmental reconstruction. For instance, in experiments, SplaTAM may exhibit good single-frame indicators in some long-distance sequences, but due to the lack of a global evaluation metric, the reconstructed scene may contain significant noise and viewpoint-overlapping, even to the point of being unintelligible, as shown in Figure 10.

2. **Frontends.** Additional experiments can be conducted with other traditional SLAM front-ends such as VINS-MONO. The plug-and-play nature of the Fusion Bridge allows it to connect with any sparse feature-based front-end and an online 3DGS back-end.

(a) fr2_desk (PSNR = 19.26)

(b) fr2_floor (PSNR = 18.77)

Figure 10: Noisy scenes rendered by SplaTAM.

Table 7: **Experimental Results without Fusion Bridge.**

Method	HRI_MEETING		
	PSNR ↑	SSIM ↑	LPIPS ↓
SplaTAM	16.93	**0.62**	0.41
Ours	**17.91**	**0.62**	**0.38**

C Appendix Experiment on HRI_MEETING Sequence

We conducted real-world experiments in our lab's conference room using a Dashgo-E1 mobile robot as shown in Figure 11 equipped with a Realsense-d435i RGB-D camera. The experimental results, as shown in the table 7, indicate that our method outperforms SplaTAM in both reconstruction quality and localization accuracy. More sequences will be added and published as a new dataset in the future.

Figure 11: The Dashgo-E1 mobile robot.