

# Online 3D Gaussian Splatting Modeling with Novel View Selection

Byeonggwon Lee<sup>1</sup>, Junkyu Park<sup>1</sup>, Khang Truong Giang<sup>2\*</sup> and Soohwan Song<sup>1\*</sup>

<sup>1</sup>Department of Computer Science and Artificial Intelligence, Dongguk University, Seoul, Korea  
<sup>2</sup>42dot, Seongnam-si, Gyeonggi-do, Korea

{lbg030, dannypk99}@dgu.ac.kr, khangtg.hust.it@gmail.com, songsh@dongguk.edu

## Abstract

This study addresses the challenge of generating online 3D Gaussian Splatting (3DGS) models from RGB-only frames. Previous studies have employed dense SLAM techniques to estimate 3D scenes from keyframes for 3DGS model construction. However, these methods are limited by their reliance solely on keyframes, which are insufficient to capture an entire scene, resulting in incomplete reconstructions. Moreover, building a generalizable model requires incorporating frames from diverse viewpoints to achieve broader scene coverage. However, online processing restricts the use of many frames or extensive training iterations. Therefore, we propose a novel method for high-quality 3DGS modeling that improves model completeness through adaptive view selection. By analyzing reconstruction quality online, our approach selects optimal non-keyframes for additional training. By integrating both keyframes and selected non-keyframes, the method refines incomplete regions from diverse viewpoints, significantly enhancing completeness. We also present a framework that incorporates an online multi-view stereo approach, ensuring consistency in 3D information throughout the 3DGS modeling process. Experimental results demonstrate that our method outperforms state-of-the-art methods, delivering exceptional performance in complex outdoor scenes.

## 1 Introduction

Online 3D modeling from an image stream is a critical challenge in robotics, virtual reality, augmented reality, and related fields. Visual *Simultaneous Localization And Mapping* (SLAM) has emerged as a key solution to this challenge. In particular, dense SLAM techniques [Newcombe *et al.*, 2011] [Teed and Deng, 2021] focus on real-time depth estimation to enable detailed 3D scene reconstruction. Recently, dense SLAM [Rosinol *et al.*, 2023] [Zhang *et al.*, 2023] [Zhu *et al.*, 2024] has been further developed to incorporate neural scene representations [Mildenhall *et al.*, 2021] [Kerbl *et al.*,



Figure 1: The main idea of our novel view selection for online 3DGS modeling. (*upper*) Our method selects non-keyframes that observe the most uncertain Gaussians (red regions in the uncertainty map) for additional training. In the figure, blue views represent keyframes, red views represent selected non-keyframes, and white views represent the remaining non-keyframes. (*lower*) This approach enhances the completeness of 3DGS, yielding high-quality renderings.

2023] [Zhou *et al.*, 2023], enhancing high-quality rendering and realistic scene synthesis. Among these advancements, *3D Gaussian Splatting* (3DGS) [Kerbl *et al.*, 2023] has attracted significant attention for its efficient 3D scene representation and real-time rendering capabilities.

In this study, we tackle the challenge of generating online 3DGS models from RGB-only frames. Previous works [Matsuki *et al.*, 2024] [Huang *et al.*, 2024] [Sandström *et al.*, 2024] have focused mainly on dense SLAM frameworks that utilize 3DGS as a map representation. These methods achieve real-time depth estimation through techniques such as optical flow [Teed and Deng, 2020], depth prediction [Eftekhar *et al.*, 2021], or motion stereo [Song *et al.*, 2021], integrating the estimated depths directly into the 3DGS model. While these approaches have shown promising results, they still face notable limitations that require further attention.

First, many studies rely on sparse or low-resolution depths

\*corresponding authors

to ensure real-time processing, which inherently limits the quality of the final model. Notably, some approaches [Zhu *et al.*, 2024] [Sandström *et al.*, 2024] employ depth prediction networks [Eftekhar *et al.*, 2021] that infer depth directly from images but often generate ambiguous estimates. Second, existing methods [Huang *et al.*, 2024] [Sandström *et al.*, 2024] [Lee *et al.*, 2024] store depth information only for keyframes, restricting 3DGS model training to these frames. This limitation can lead to incomplete reconstruction not only of unobserved areas but also of covered 3D scenes due to insufficient keyframe coverage. Lastly, online 3D modeling must operate within a constrained timeframe, which restricts the number of image frames and training iterations that can be processed. To achieve the best results within these constraints, it is essential to identify and prioritize the frames that most effectively enhance the performance of 3DGS during training.

To address these challenges, we propose a novel method that adaptively improves model completeness through novel view selection. Our approach identifies an optimal set of non-keyframes for additional training by analyzing the online reconstruction quality of the 3DGS model. Specifically, we define the uncertainty of Gaussian primitives based on their shapes and the gradients of their positions. This uncertainty metric is then used to calculate the information gain from a given viewpoint, enabling the selection of optimal views. By incorporating both keyframes and selected non-keyframes, our method refines incomplete regions from diverse viewpoints, resulting in improved completeness (see Fig. 1).

Additionally, we present an efficient framework for online 3DGS mapping that integrates an independent frontend and backend for streamlined processing. The frontend leverages a state-of-the-art *multi-view stereo* (MVS) network [Cao *et al.*, 2022] to generate accurate depths from sequential input images. Operating in parallel, the backend focuses on optimizing 3DGS models, allowing sufficient time for effective parameter refinement. To maintain consistency between the frontend and backend, the framework continuously applies *global bundle adjustment* (GBA) and Gaussian deformation. This approach achieves superior rendering accuracy compared to state-of-the-art methods, demonstrating exceptional performance in handling complex outdoor environments.

The main contributions are summarized as follows:

- We propose an online 3DGS modeling method that leverages novel view selection. This approach selects optimal views from non-keyframes for additional training, significantly improving model completeness.
- We define the uncertainty of Gaussians using their shapes and positional gradients. This approach allows for more effective view selection compared to other uncertainty metrics [Jiang *et al.*, 2025] [Jin *et al.*, 2024].
- We present a 3DGS mapping framework that utilizes an online MVS approach. This framework ensures consistent 3D information throughout the entire process.
- The proposed method was evaluated using two benchmarks for indoor scenes [Sturm *et al.*, 2012] [Straub *et al.*, 2019]. To highlight its generalization capability, we also extended the evaluation to include challenging outdoor scenarios [Song *et al.*, 2021] [Knapitsch *et al.*, 2017].

## 2 Related Works

### 2.1 Monocular Dense SLAM

Monocular SLAM [Mur-Artal *et al.*, 2015] [Forster *et al.*, 2014] traditionally generates sparse feature maps from monocular images and uses them to estimate camera poses. These methods have mainly focused on achieving precise pose estimation and have achieved significant success. The focus has shifted to reconstructing detailed 3D scenes, fueling the development of dense SLAMs [Engel *et al.*, 2017].

With deep learning achieving state-of-the-art performance in optical flow [Teed and Deng, 2020] and depth estimation [Eftekhar *et al.*, 2021] [Yao *et al.*, 2018], many studies also have investigated the incorporation of deep learning modules into dense SLAM systems [Bloesch *et al.*, 2018; Koestler *et al.*, 2022; Teed and Deng, 2021]. DROID-SLAM [Teed and Deng, 2021] combines an optical flow network with GBA to achieve precise trajectory estimation and dense 3D modeling.

### 2.2 Differentiable Rendering SLAM

Differentiable volumetric rendering, popularized by NeRF [Mildenhall *et al.*, 2021], has recently emerged as a key technique for achieving photorealistic scenes and has been applied to dense SLAM. GO-SLAM [Zhang *et al.*, 2023] and NeRF-SLAM [Rosinol *et al.*, 2023] integrate low-resolution depth maps estimated from DROID-SLAM [Teed and Deng, 2021] into NeRFs. NICER-SLAM [Zhu *et al.*, 2024] and GLORIE-SLAM [Zhang *et al.*, 2024] leverage depth prediction networks to generate high-resolution depth maps, which are further utilized to improve NeRFs.

Recently, 3DGS [Kerbl *et al.*, 2023; Ma *et al.*, 2024] has gained attention as a promising solution for 3D modeling, offering faster rendering speeds compared to NeRF. By representing 3D space with Gaussian primitives, 3DGS achieves efficient rendering through rasterization and spatial optimization, making it highly suitable for dense SLAM. Photo-SLAM [Huang *et al.*, 2024] and MGSO [Hu *et al.*, 2024] generate Gaussian points from sparse feature points extracted using ORB-SLAM [Mur-Artal *et al.*, 2015] and Direct Sparse Odometry [Engel *et al.*, 2017], respectively. Splat-SLAM [Sandström *et al.*, 2024] further enhances 3DGS mapping by combining proxy depth maps with depth prediction results. It employs deformable Gaussians and globally optimized optical flow tracking, resulting in improved accuracy.

Existing methods [Kerbl *et al.*, 2023] [Matsuki *et al.*, 2024] [Huang *et al.*, 2024] [Hu *et al.*, 2024] [Sandström *et al.*, 2024] suffer from reduced quality due to their reliance on sparse or imprecise depth data, with depth prediction performing poorly in untrained environments like outdoor scenes. To address this limitation, MVS-GS [Lee *et al.*, 2024] leverages an online MVS approach [Song *et al.*, 2021] to estimate high-quality depth maps, enabling accurate Gaussian initialization. Building on the MVS-GS framework, our method further improves 3DGS mapping by incorporating iterative GBA with Gaussian deformation. Additionally, we propose a novel view selection strategy that prioritizes non-keyframes with high information gain, integrating them into the training process. These enhancements significantly improve the completeness and quality of the resulting 3DGS model.

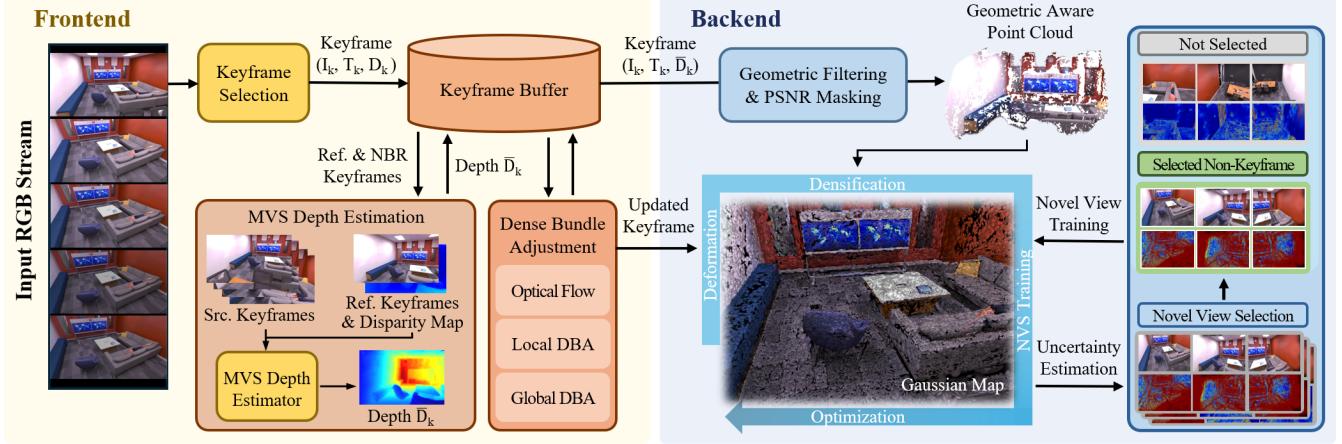


Figure 2: System overview. The frontend tracks camera poses and estimates depth maps for keyframes using MVS networks, with the disparity map derived from tracking module used to initialize the network. The backend generates new Gaussians from keyframes and continuously optimizes the 3DGS model. To further improve performance, our system periodically performs GBA and conducts additional training on non-keyframes selected through NVS.

### 2.3 Next-Best-View Planning

The next-best-view (NBV) problem [Scott *et al.*, 2003] focuses on identifying optimal viewpoints to gather specific information. This process aims to maximize information gain, typically defined based on uncertainty, which varies depending on the type of 3D modeling.

With advancements in NeRF, many NBV-related studies have emerged. These approaches estimate uncertainty by analyzing factors such as occupancy probability [Yan *et al.*, 2023], color similarity [Ran *et al.*, 2023], or implicit neural representations [Feng *et al.*, 2024] derived from NeRF models. However, NBV research specifically tailored to 3DGS remains limited. FisherRF [Jiang *et al.*, 2025] quantifies information gain using Fisher information in Gaussian parameters, but it relies on computationally intensive gradient calculations. GS-Planner [Jin *et al.*, 2024] evaluates completeness by comparing rendered color and depth values with their captured ones, using this measure to estimate information gain.

These methods [Jiang *et al.*, 2025; Jin *et al.*, 2024] measure uncertainty using rendered images; however, they provide inaccurate results when the original image contains noise and are heavily affected by less significant areas, such as the sky. In contrast, the proposed method effectively represents the quality of each Gaussian by leveraging its shape and gradients, allowing for efficient and precise uncertainty estimation.

## 3 Proposed Method

We present a system for online 3DGS modeling using RGB images, designed for high-quality rendering. This system is built on the MVS-GS framework [Lee *et al.*, 2024], featuring parallel operation of frontend and backend modules. For the frontend module, we directly adopt the tracking method of [Teed and Deng, 2021] and the MVS networks from [Cao *et al.*, 2022] to simultaneously estimate camera poses and depth maps from an image stream. Based on the outputs of the frontend, we propose novel techniques to learn a high-fidelity 3DGS scene representation in the backend module.

### 3.1 System Overview

Fig. 2 depicts an overview of the proposed system. Given an input image stream, the frontend dynamically constructs a frame-graph  $G$ , where each node  $I_k$  represents a selected keyframe, and each edge  $(I_k, I_l)$  denotes a co-visibility connection between two keyframes. During the construction of the graph  $G$ , the frontend tracks the camera pose and estimates the depth map for each inserted keyframe. To mitigate the effects of pose and scale drift and enhance global geometric consistency, we perform GBA online on the entire graph  $G$  every 30 keyframes. The original tracking system [Teed and Deng, 2021] performs GBA only at the end of the mapping process. In contrast, we continuously perform GBA to consistently optimize the global consistency of 3D model.

Next, the frontend estimates accurate and reliable depth maps based on the MVS network, MVSFormer [Cao *et al.*, 2022]. Depth prediction methods commonly employed in dense SLAM [Ranftl *et al.*, 2021] [Bhat *et al.*, 2023] suffer from scale ambiguity issues due to the inherent limitations of single-image-based estimation, which can result in reduced accuracy and consistency in depth estimation. In contrast, MVS leverages information from surrounding frames, effectively overcoming the scale ambiguity inherent in single-frame estimation and generating more accurate and consistent depth maps. We compute the coarse depths based on the dense optical flow results estimated during the tracking step and use it to initialize the depth map in the first layer of MVSFormer. This approach improves consistency between the tracking and MVS stages, enhancing the robustness of depth estimation while reducing computational costs.

The backend module consistently trains a 3DGS model for scene representation using the estimated camera pose  $T_k$  and depth map  $\bar{D}_k$  of each keyframe  $I_k$ . The module begins by using the depths of some initial keyframes to initialize a 3DGS model. Subsequently, it incrementally integrates new 3D Gaussians from incoming keyframes into the existing model (Section 3.2). Additionally, we perform a *novel view*

**selection** (NVS) strategy to identify a set of non-keyframes that significantly enhance 3DGS quality, using them for further training of the 3DGS model (Section 3.3). The NVS quantifies the uncertainty of Gaussians and determines the non-keyframes that cover the largest number of uncertain Gaussians. Our method uses the selected frames for additional training, significantly improving the completeness of the 3DGS model and the generalizability of its rendering.

### 3.2 Online 3DGS Modeling

**Map Construction.** For each keyframe, we first transform the depth map  $\bar{D}_k$  into a point cloud based on the camera pose  $T_k$ . We remove noisy points and outliers from the point cloud to obtain reliable and consistent 3D points. This step was implemented using geometric and photometric consistency criteria commonly employed in MVS [Cao *et al.*, 2022].

The initial point cloud, generated from the first few keyframes, is used to create a set of Gaussian primitives for the 3DGS model. As new keyframes are extracted, the 3DGS model is incrementally updated by incorporating additional Gaussians. Following the approach of [Lee *et al.*, 2024], we generate new Gaussians exclusively from unexplored regions with low rendering quality to improve computational efficiency. The unexplored regions are identified by comparing the rendered image  $\hat{I}_k$  with the original images  $I_k$  using the peak signal-to-noise ratio (PSNR), marking those below a threshold as such regions. Throughout the map construction process, we regularly perform GBA and Gaussian optimization to refine camera poses and update the model parameters.

**Gaussian Representation.** We utilize the generalized exponential splatting (GES) [Hamdi *et al.*, 2024] to parameterize each Gaussian in a 3DGS model. Specifically, a Gaussian point  $g_i$  is defined by several attributes: a covariance matrix  $\sum_i \in R^{3 \times 3}$ , a mean  $\mu_i \in R^3$ , opacity  $o_i \in [0, 1]$ , color  $c_i \in R^3$ , and a shape parameter  $\beta_i \in (0, \infty)$ . The shape parameter  $\beta_i$ , which is learnable, controls the sharpness of the Gaussian. The GES framework efficiently captures high-frequency details of the scene using fewer primitives, making it well-suited for real-time mapping and rendering. The Gaussian density function in GES is defined as follows:

$$g_i(x) = \exp \left\{ -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right\}^2 \beta_i \quad (1)$$

To render an image  $\hat{I}_k$  from an input pose  $T_k$ , the 3D Gaussians are projected onto the image plane. The color  $\hat{c}(p)$  of each pixel  $p$  is determined by sorting the Gaussian points by depth and performing alpha blending from front to back:

$$\hat{c}(p) = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (2)$$

where  $\alpha_i = o_i g_i(x)$  ensures that closer Gaussian points contribute more to the color of the pixels. Similarly, the depth  $\hat{D}_k$  can be rendered using the same projection approach:

$$\hat{D}(p) = \sum_{i \in N} z_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (3)$$

where  $z_i$  is the distance to the Gaussian mean along the ray.

**Map Deformation.** During incrementally building the 3D Gaussian map, the GBA module updates camera poses and disparity map of keyframes in entire history. Therefore, the 3D Gaussian map should be updated accordingly after each GBA update. However, our map is only rigidly deformed based on the updated camera poses instead of using non-rigid deformation as in [Sandström *et al.*, 2024]. The 3D data generated by MVS is highly accurate, allowing for the generation of a consistent model with rigid deformation. This approach helps improve the efficiency of the deformation process. The map deformation process is executed after the GBA and before the Gaussian optimization step. This helps refine the entire Gaussian map to adapt to the updated camera poses.

### 3.3 3DGS Optimization with Novel View Selection

Existing online methods [Sandström *et al.*, 2024; Huang *et al.*, 2024; Lee *et al.*, 2024] learn Gaussian parameters using only tracked keyframes. The keyframes are usually selected by thresholding mean of optical flow between image frames [Teed and Deng, 2020]. Those keyframes sometimes cannot fully cover the entire scene due to fast motion changes or occlusions. Additionally, it is infeasible to use all frames, including both keyframes and non-keyframes, for optimization. To address this, our NVS selects top-k non-keyframes that cover poorly reconstructed regions, based on estimated information gain.

**Uncertainty Estimation.** For each non-keyframe  $I_n$ , we extract visible Gaussians and compute an uncertainty score based on their shapes and positional gradients. For each Gaussian  $g_{n,i}$  visible in  $I_n$ , we extract the largest eigenvalue of its covariance  $\Sigma_{n,i}$ . Here,  $\Sigma_{n,i} = R_{n,i} S_{n,i} S_{n,i}^T R_{n,i}^T$ , where  $R_{n,i}$  and  $S_{n,i}$  represent the orientation and scale components of Gaussian  $g_{n,i}$ , respectively. Since  $S_{n,i} = \text{diag}(s_{n,i})$  where  $s_{n,i} \in R^3$ , it follows that  $S_{n,i} S_{n,i}^T = \text{diag}(s_{n,i}^2)$ . The largest eigenvalue  $\lambda_{n,i}$  is then calculated as:

$$\lambda_{n,i} = \max(s_{n,i}^2) \quad (6)$$

This eigenvalue  $\lambda_{n,i}$  represents the largest size of  $g_{n,i}$  in the direction corresponding to the rotation  $R_{n,i}$ . Incorporating the eigenvalue into the uncertainty formulation helps address the over-reconstruction indicated by large Gaussians [Hu *et al.*, 2025; Matsuki *et al.*, 2024].

The second component of uncertainty is derived from the adaptive density control mechanism in existing 3DGS models [Matsuki *et al.*, 2024][Sandström *et al.*, 2024]. A large positional gradient indicates that the 3DGS model is still undergoing optimization. Therefore, high gradients suggest that the parameters have not yet been fully optimized [Rota Bulò *et al.*, 2024] [Kerbl *et al.*, 2023]. Therefore, we adopted the gradient as a factor for measuring the uncertainty of Gaussians. Let  $d\mu_{n,i} \in R^3$  be the positional gradient of Gaussian  $g_{n,i}$ . We calculate the magnitude of the gradient  $A_{n,i} = \|d\mu_{n,i}\|$  as a measure of uncertainty estimation. This approach helps prioritize regions requiring further optimization.

Using these two components, an uncertainty score  $U_{n,i}$  is computed for each Gaussian  $g_{n,i}$  as follows:

$$U_{n,i} = \alpha_1 \lambda_{n,i} + \alpha_2 A_{n,i} \quad (7)$$

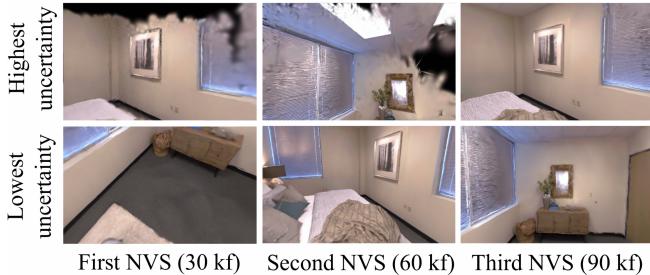


Figure 3: Rendered images from views with the highest and lowest information gain in the candidate set at every 30th keyframe.

where  $\alpha_1 = 0.7$ ,  $\alpha_2 = 0.3$  are weighted hyperparameters.

Finally, the information gain of the view  $I_n$  is defined as the total uncertainty of all Gaussians visible from  $I_n$ .

$$U_n = \sum_{i=1}^{|g_{n,i}|} \frac{1}{z_{n,i}^2} U_{n,i} \quad (8)$$

where  $z_{n,i}$  represents the depth of the Gaussian  $g_{n,i}$  in the view  $I_n$ . This equation highlights that Gaussians closer to the view  $I_n$  have a greater impact on the uncertainty score.

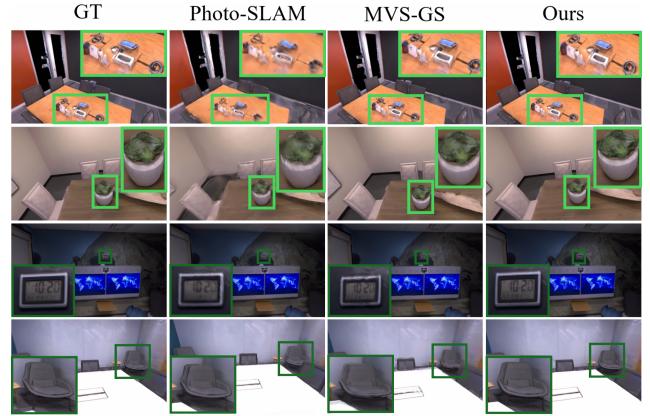
After calculating an information gain for each non-keyframe, we sort all candidate frames in descending order based on their scores. Since consecutive frames often have similar scores, we apply non-maximum suppression (NMS) [Redmon, 2016] within a local window to the sorted list. This process avoids selecting frames within the local window of previously chosen frames, promoting the selection of frames from diverse viewpoints. The top-k frames with the highest information gain are then selected and combined with the 30 most recently tracked keyframes to form the training frame set. The frame set is used for additional training, enabling the 3DGS model to enhance under-reconstructed areas and refine recently observed regions that require further training.

We construct a candidate set consisting of non-keyframes captured within the span of the 30 most recent keyframes, and apply NVS to this set. We identify 20 high-information-gain frames selected from the current set and include them in the subsequent candidate set. This approach allows for an efficient evaluation of information gain by focusing on the latest non-keyframes while incorporating selected older non-keyframes. Fig. 3 depicts the non-keyframes with the highest and lowest gain within the set constructed at each iteration.

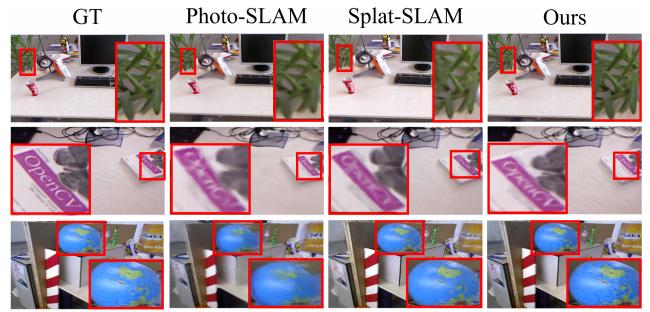
**Gaussian Training** This optimization step is iteratively performed to refine Gaussian parameters. These Gaussians are trained using both keyframes and selected non-keyframes. The loss function for keyframes is defined as follows:

$$\begin{aligned} L_{\text{KF}} = & \lambda_{L1} L_1 + \lambda_{\text{SSIM}} L_{\text{SSIM}} \\ & + \lambda_{\text{depth}} L_{\text{depth}} + \lambda_{\text{smooth}} L_{\text{smooth}} \end{aligned} \quad (9)$$

where  $L_1$  represents the L1 loss between the rendered images (Eq. 2) and the original images.  $L_{\text{SSIM}}$  is the structural similarity index measure (SSIM) loss, which preserves image quality [Wang *et al.*, 2004].  $L_{\text{depth}}$  is calculated between the rendered depths (Eq. 3) and the MVS depths, helping to maintain the geometric structure. Lastly  $L_{\text{smooth}}$  is the



(a) Results on the Replica dataset



(b) Results on the TUM-RGBD dataset

Figure 4: Qualitative evaluation on (a) the Replica and (b) TUM-RGBD datasets. The boxes highlight areas where our method produces fine details with higher accuracy than the competing methods.

smoothness loss, designed to minimize depth differences between neighboring pixels, which is expressed as:

$$\begin{aligned} L_{\text{smooth}} = & \frac{1}{N_x} \sum_{i,j} |d(i,j) - d(i,j+1)| \\ & + \frac{1}{N_y} \sum_{i,j} |d(i,j) - d(i+1,j)| \end{aligned} \quad (10)$$

This smooths abrupt depth changes between pixels, yielding natural transitions and fewer discontinuities.

Non-keyframes contain only image information and do not include depth information. Therefore, for the non-keyframes, the loss function simplifies image rendering losses:

$$L_{\text{NKF}} = \lambda_{L1} L_1 + \lambda_{\text{SSIM}} L_{\text{SSIM}} + \lambda_{\text{smooth}} L_{\text{smooth}} \quad (11)$$

This approach ensures that both keyframes and non-keyframes contribute effectively to improving the 3DGS model, balancing geometric accuracy and image quality.

## 4 Experiment Results

To assess the quality of our method, we compared to state-of-the-art methods on various datasets, including indoor and outdoor scenes. For indoor scenes, we conducted experiments using the Replica [Straub *et al.*, 2019] and TUM-RGBD [Sturm *et al.*, 2012] datasets, following the same experimental setups as other methods ([Matsuki *et al.*, 2024],

Method	Off0	Off1	Off2	Off3	Off4	Rm0	Rm1	Rm2	Avg.
PSNR $\uparrow$									
Q-SLAM	36.31	37.22	30.68	30.21	31.96	29.58	32.74	31.25	32.49
MonoGS	32.00	31.21	23.26	25.77	23.85	23.53	25.00	22.42	25.88
Splat-SLAM	40.81	40.64	35.19	35.03	37.40	32.25	34.31	35.95	36.45
Photo-SLAM	36.99	37.52	31.79	31.62	34.17	29.77	31.30	33.18	33.29
MGS-SLAM	35.51	34.25	30.83	31.86	34.38	29.91	31.06	31.49	32.41
MVS-GS	41.02	42.04	34.00	34.65	33.33	32.20	31.54	35.84	35.58
Ours	<b>43.93</b>	<b>43.98</b>	<b>37.98</b>	<b>36.31</b>	<b>39.59</b>	<b>34.88</b>	<b>37.99</b>	<b>39.60</b>	<b>39.28</b>
SSIM $\uparrow$									
Q-SLAM	0.94	0.94	0.90	0.88	0.89	0.83	0.91	0.87	0.89
MonoGS	0.90	0.88	0.82	0.84	0.86	0.75	0.79	0.81	0.83
Splat-SLAM	0.97	<b>0.99</b>	<b>0.97</b>	<b>0.97</b>	<b>0.97</b>	<b>0.96</b>	<b>0.97</b>	<b>0.96</b>	<b>0.97</b>
Photo-SLAM	0.96	0.95	0.93	0.92	0.94	0.87	0.91	0.93	0.93
MGS-SLAM	0.94	0.93	0.90	0.92	0.95	0.89	0.90	0.91	0.92
MVS-GS	<b>0.98</b>	<b>0.98</b>	0.95	0.96	0.95	<b>0.95</b>	<b>0.92</b>	<b>0.96</b>	0.96
Ours	<b>0.99</b>	<b>0.99</b>	<b>0.98</b>	<b>0.97</b>	<b>0.98</b>	<b>0.96</b>	<b>0.97</b>	<b>0.98</b>	<b>0.98</b>
LPIPS $\downarrow$									
Q-SLAM	0.13	0.15	0.20	0.19	0.18	0.18	0.16	0.15	0.17
MonoGS	0.23	0.22	0.30	0.24	0.34	0.33	0.35	0.39	0.30
Splat-SLAM	<b>0.05</b>	0.07	<b>0.06</b>	<b>0.04</b>	0.10	0.09	<b>0.06</b>	<b>0.05</b>	<b>0.06</b>
Photo-SLAM	0.06	0.06	0.09	0.09	<b>0.07</b>	0.10	0.08	0.07	0.08
MGS-SLAM	0.07	0.11	0.12	0.07	0.08	<b>0.08</b>	0.09	0.09	0.09
MVS-GS	<b>0.05</b>	<b>0.05</b>	0.09	0.07	0.10	0.10	0.13	0.07	0.08
Ours	<b>0.02</b>	<b>0.02</b>	<b>0.03</b>	<b>0.03</b>	<b>0.04</b>	<b>0.04</b>	<b>0.03</b>	<b>0.03</b>	<b>0.03</b>

Table 1: Quantitative rendering performance on the Replica dataset.

[Sandström *et al.*, 2024], [Zhang *et al.*, 2024]). For outdoor scenarios, we used the Aerial [Song *et al.*, 2021] and Tanks&Temples datasets [Knapitsch *et al.*, 2017]).

**Implementation Details.** All experiments were carried out on a desktop equipped with an AMD Ryzen9 7900X 12-core processor and an NVIDIA GeForce RTX 4090 GPU. Training and evaluation were performed in PyTorch with CUDA to accelerate rasterization and gradient computations. While most hyperparameters follow the original 3DGS setting [Kerbl *et al.*, 2023], we empirically set  $\lambda_{L1}$ ,  $\lambda_{SSIM}$ ,  $\lambda_{depth}$  and  $\lambda_{smooth}$  to 0.95, 0.2, 0.2, and 0.1, respectively, for the loss function of Gaussian training. We used 100 non-keyframes for NVS, and set the NMS gap between frames to 3 to exclude neighboring frames.

**Baseline Methods.** We evaluated the performance of our method by comparing it with state-of-the-art monocular dense SLAM methods, including NeRF-based (Q-SLAM [Peng *et al.*, 2024] and GLORIE-SLAM [Zhang *et al.*, 2024]) and 3DGS-based (MonoGS [Matsuki *et al.*, 2024], Splat-SLAM [Sandström *et al.*, 2024], Photo-SLAM [Huang *et al.*, 2024], and MVS-GS [Lee *et al.*, 2024]) methods.

**Evaluation Metrics.** We assessed the modeling quality using rendered images and depth maps. For image rendering quality, we report PSNR, SSIM, and LPIPS by comparing the rendered images with the original images.

#### 4.1 Evaluation in Indoor Scenes

Table 1 shows the rendering performance results on the Replica dataset, highlighting that our method significantly outperforms other methods. Notably, it achieves superior results in both PSNR and SSIM, with a substantial improvement compared to the second-best method, Splat-SLAM.

Fig. 4a shows a qualitative comparison between our method and the other 3DGS-based methods. Our method effectively captures intricate scene details, resulting in high-quality renderings with fewer artifacts. Our texture details are significantly better than those produced by Photo-SLAM. Notably, MVS-GS fails to handle loop closing, causing the

Method	Metrics	f1/desk	f2/xyz	f3/off	Avg.
MonoGS	PSNR $\uparrow$	19.67	16.17	20.63	18.82
Photo-SLAM		20.97	21.07	19.59	20.54
GLORIE-SLAM		20.26	25.62	21.21	22.36
Splat-SLAM		<b>25.61</b>	<b>29.53</b>	<b>26.05</b>	<b>27.06</b>
MVS-GS		20.67	24.53	22.37	22.52
Ours		<b>26.34</b>	<b>28.72</b>	<b>28.10</b>	<b>27.72</b>
MonoGS	SSIM $\uparrow$	0.73	0.72	0.77	0.74
Photo-SLAM		0.74	0.73	0.69	0.72
GLORIE-SLAM		<b>0.87</b>	<b>0.96</b>	<b>0.84</b>	<b>0.89</b>
Splat-SLAM		0.84	0.90	<b>0.84</b>	0.86
MVS-GS		0.77	0.86	0.80	0.81
Ours		<b>0.89</b>	<b>0.91</b>	<b>0.91</b>	<b>0.90</b>
MonoGS	LPIPS $\downarrow$	0.33	0.31	0.34	0.33
Photo-SLAM		0.23	0.17	0.24	0.21
GLORIE-SLAM		0.31	0.09	0.32	0.24
Splat-SLAM		<b>0.18</b>	<b>0.08</b>	<b>0.20</b>	<b>0.15</b>
MVS-GS		0.25	0.15	0.24	0.21
Ours		<b>0.12</b>	<b>0.07</b>	<b>0.10</b>	<b>0.10</b>

Table 2: Quantitative rendering performance on the TUM-RGBD.

chair to appear doubled. In contrast, our method accurately reconstructs a single, complete representation of the chair.

Table 2 presents the evaluation results on real-world scenes from the TUM-RGBD dataset. Despite the inherent challenges of real-world datasets, including noisy images and complex scene variations, our approach maintains stable and reliable rendering quality. The qualitative results are shown in Fig. 4b. This demonstrates the robustness of our method in handling real-world data, ensuring consistent performance across diverse indoor scenes.

#### 4.2 Evaluation in Outdoor Scenes

In this section, we validated the generalization capability of our method by evaluating its performance on outdoor scenes. Depth prediction-based methods, including Splat-SLAM and GLORIE-SLAM, failed to generate 3DGS models in these environments. Fig. 5a presents a performance evaluation of aerial scenes, demonstrating that our method outperforms both Photo-SLAM and MVS-GS, particularly in capturing small structural details. Leveraging the strengths of MVS in accurately reconstructing large-scale scenes, our method significantly outperformed Photo-SLAM in aerial scenes.

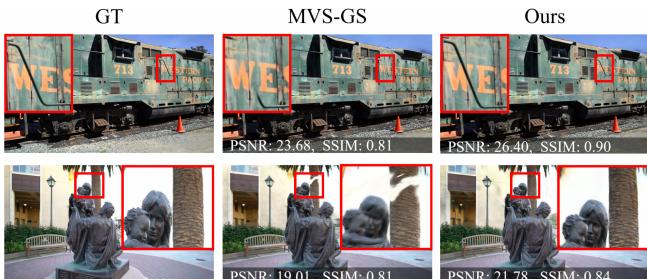
On the Tanks and Temples dataset, Splat-SLAM, MonoGS, and even Photo-SLAM failed to generate 3DGS models due to the lack of sufficient continuous motion in the image frames. Fig. 5b presents the rendering results for this dataset. While MVS-GS successfully produced a 3DGS model, its quality was compromised by limited training views and the absence of loop closure. In contrast, our method delivered high-quality rendering results that closely resembled real photographs. These findings highlight the effectiveness of the proposed NVS method and consistent geometric alignment, even in challenging outdoor scenes.

#### 4.3 Ablation Study

To evaluate the impact of each key component in our method, we performed an ablation study on the “Office0” scene from the Replica dataset. Each component was progressively assessed as follows:



(a) Results on the Aerial dataset



(b) Results on the Tanks and Temples dataset

Figure 5: Qualitative evaluation on (a) the aerial scenes and (b) Tanks and Temples datasets. The red boxes highlight areas where our method achieves better accuracy in capturing fine details.

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Depth L1 $\downarrow$	FPS $\uparrow$	#Gaussians (K) $\downarrow$	Map Size [GiB] $\downarrow$
A	40.92	0.979	0.023	0.042	19.39	1365	0.305
B	42.37	0.984	0.019	0.046	15.66	1253	0.280
C	42.71	0.985	0.018	0.044	15.28	1357	0.303
D	42.73	0.985	0.019	0.038	15.11	1377	0.308
E	43.93	0.988	0.016	0.034	9.18	1078	0.241

Table 3: Comparison results of our method with different model variants on the Replica “office 0” scene.

- **Method-A:** The baseline method, MVS-GS.
- **Method-B:** Incorporates the global GBA module.
- **Method-C:** Builds on Method-B by applying depth initialization using the disparity map from DROID-SLAM.
- **Method-D:** Extends Method-C by introducing a smoothness loss for training Gaussians.
- **Method-E:** Represents our full model, including the NVS.

We evaluated both image and depth rendering performance using metrics such as PSNR, SSIM, and LPIPS for images, and L1 error for depths. We also reported the number of Gaussians, map size, and computational speed (measured in frames per second, FPS) for each method. Table 3 summarizes the results for the variant methods.

Compared to Method-A, both Method-B and Method-C showed progressive improvements in rendering performance. This indicates that the application of the GBA and the depth initialization approach enhances the completeness of 3DGS modeling. Furthermore, Method-D incorporated the smooth depth loss, which showed no noticeable improvement in rendering quality but led to substantial enhancements in reconstruction performance. Fig. 6 shows the comparison results

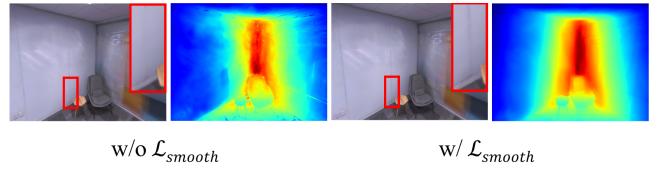


Figure 6: The effect of depth smooth regularization loss on the rendered image and depth map.

Metrics	$\alpha_1=0.0$ $\alpha_2=1.0$	$\alpha_1=0.1$ $\alpha_2=0.9$	$\alpha_1=0.3$ $\alpha_2=0.7$	$\alpha_1=0.5$ $\alpha_2=0.5$	$\alpha_1=0.7$ $\alpha_2=0.3$	$\alpha_1=0.9$ $\alpha_2=0.1$	$\alpha_1=1.0$ $\alpha_2=0.0$
PSNR $\uparrow$	43.802	43.820	43.859	43.877	<b>43.932</b>	43.841	43.812
SSIM $\uparrow$	0.986	0.986	0.987	0.987	<b>0.988</b>	0.987	0.985
LPIPS $\downarrow$	0.018	0.017	0.017	0.016	<b>0.016</b>	0.018	0.019

Table 4: Performance variations with respect to changes in the weights  $\alpha_1$  and  $\alpha_2$  of the uncertainty on the Replica “office 0”.

between Method-C and Method-D, where the improvements in reconstruction performance are visually apparent. Finally, when the NVS technique was applied, Method-E achieved superior performance in both rendering quality and depth reconstruction. Furthermore, in terms of model compactness, Method-E significantly reduces the number of Gaussians compared to Method-D, resulting in approximately a 60 MB reduction in map size. The NVS optimization helps eliminate or merge redundant keyframe Gaussians, thereby minimizing unnecessary overhead.

We further analyzed the impact of the uncertainty score parameters in Eq. (7) on the effectiveness of view selection. Table 4 shows the effect of the  $\alpha_1$  and  $\alpha_2$ . The largest eigenvalue primarily captures uncertainty in high-frequency regions, while the positional gradient focuses on low-frequency areas. Thus,  $\alpha_1$  and  $\alpha_2$  serve to balance these two complementary measures. If either value is too large, uncertainty estimation becomes biased and less effective. Therefore, we empirically determined suitable  $\alpha_1$  and  $\alpha_2$  values for balanced and accurate estimation.

## 5 Conclusion

We present a novel online mapping approach for high-quality 3DGS modeling that introduces uncertainty estimation directly within Gaussians. Our method utilizes an uncertainty-aware view selection strategy to extract the most informative views, improving the completeness of 3DGS models. Additionally, we develop a framework that integrates an online MVS technique, maintaining consistency in 3D information throughout the modeling process. Extensive experiments show that the proposed approach surpasses state-of-the-art dense SLAM methods, demonstrating outstanding performance, especially in complex outdoor environments. To the best of our knowledge, this is the first work to introduce non-keyframe selection in a 3DGS-based SLAM framework. The proposed formulation is general and applicable to various SLAM systems.

## **Acknowledgments**

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2025-RS-2020-II201789), and the Artificial Intelligence Convergence Innovation Human Resources Development(IITP-2025-RS-2023-00254592) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

# Appendix

## A Detailed Method

### A.1 Camera Tracking

We utilized DROID-SLAM [Teed and Deng, 2021] to track camera poses of input image stream. The core component of DROID-SLAM is an update operator that iteratively refines camera pose and disparity map of each keyframe. Specifically, for each keyframe pair  $(I_k, I_l)$ , where  $I_k$  is the incoming keyframe to be tracked and  $I_l$  is a neighboring keyframe selected from the existing frame graph  $G$ , we first extract feature maps and construct a correlation cost volume  $C_{k,l}$ . This volume is then used as input for the update operator. During each iteration, the update operator employs ConvGRU recurrent layers and dense bundle adjustment (DBA) layers to refine the pose and disparity estimates. After several iterations, the outputs for keyframe  $I_k$  converge to a stable solution, allowing the keyframe to be added to the frame graph.

### A.2 MVS Depth Estimation

When keyframe  $I_k$  was tracked, we subsequently perform multi-view depth estimation to obtain an accurate depth map for this keyframe  $I_k$ . To do so, we utilize a learning-based MVS method, MVSFormer [Cao *et al.*, 2022]. This is a coarse-to-fine architecture, which first estimates a depth map at a low image resolution and then gradually refines the depth in finer resolution. In our implementation, we find that the stability of initial depth map is crucial in the above process. Therefore, we leverage the disparity map estimated by the DROID-SLAM to initialize a depth map for MVSFormer. This also improves the model efficiency at the same time. However, it is observed that the disparity map usually exhibits noisy artifacts at boundary regions. To address this, we replace pixel values at the boundaries to average values of their neighboring pixels. This approach enhances the quality of initial depth map and provides a robust foundation for generating more accurate depth maps in subsequent stages.

### A.3 Global Dense Bundle Adjustment

Our global DBA module is executed every 30 keyframes to optimize camera poses and disparities across the entire frame graph  $G$ . To ensure numerical stability, the disparities and poses are normalized before each optimization step. Specifically, the average disparity  $\bar{d}$  is calculated across all keyframes. Then, each disparity of each keyframe is normalized by  $d_{norm} = d/\bar{d}$ . For camera pose normalization, the translation vector is computed as  $t_{norm} = \bar{t}$ .

Assume that a camera pose  $T_k$  is updated to a new pose  $T'_k$  after GBA. To facilitate deformation, a relative transformation is computed as  $T'_k^{rel} = T'_k T_k^{-1}$ . After that, the mean  $\mu_i$  and covariance  $\sum_i$  of each Gaussian  $g_i$  is transformed accordingly as follows:

$$\mu'_i = T_k^{rel} \mu_i, \quad R'_i = T_k^{rel} R_i \quad (4)$$

$$\Sigma'_i = R'_i \Sigma_i (R'_i)^{\top} \quad (5)$$

where  $R_k^{rel}$  is the rotation component of camera pose  $T_k^{rel}$ . Note that the means  $\mu_i, \mu'_i$  are written in homogeneous coordinate.

### A.4 Pruning and Densification

During the 3DGS mapping, we adopt the pruning and densification strategies outlined in [Hamdi *et al.*, 2024]. For pruning, Gaussians with an opacity below 0.005 are globally removed every 150 mapping iterations. Additionally, Gaussians with excessively large scales in both 2D and 3D projections are also eliminated. Densification is performed every 150 mapping iterations based on accumulated gradients. The gradient threshold for densification is set to 0.0002. Furthermore, during the Novel View Training process, densification is selectively applied to Gaussians with high uncertainty. All Gaussians are periodically reset following the method described in [Hamdi *et al.*, 2024], with opacity being reinitialized every 3000 mapping iterations and shape parameters being reinitialized every 100 iterations.

### A.5 Note on Rendering Evaluation

To ensure a fair comparison across all methods, we conducted at least five experiments and averaged the results for evaluation. Additionally, for methods that do not render depth maps as described in [Kerbl *et al.*, 2023], we applied a consistent implementation to ensure uniform evaluation.

## B More Experiments

### B.1 Implementation Details

In all frames, we apply a voxel downsampling factor of 0.005. For the Replica dataset [Straub *et al.*, 2019], the final refinement process during Gaussian training uses  $\beta = 2000$  iterations. For the TUM-RGBD [Sturm *et al.*, 2012] and ScanNet[Dai *et al.*, 2017] datasets,  $\beta$  is adjusted to 26000, following the configurations used in other methods ([Matsuki *et al.*, 2024], [Sandström *et al.*, 2024], [Zhang *et al.*, 2024]). We adopted the hyperparameter settings specified in [Hamdi *et al.*, 2024] for Gaussian mapping and [Teed and Deng, 2021] for tracking.

### B.2 Quantitative Performance on the ScanNet

To further assess rendering performance, we conducted additional evaluations using the ScanNet dataset [Dai *et al.*, 2017]. The results, summarized in Table S1, highlight the comparative performance of our method.

Method	0000	0059	0106	0169	0181	0207	Avg.
PSNR ↑	MonoGS	16.91	19.15	18.57	20.21	19.51	18.37
	GLORIE-SLAM	23.42	20.66	20.41	25.23	21.28	23.68
	Splat-SLAM	28.68	27.69	27.70	31.14	<b>31.15</b>	30.49
	MVS-GS	23.91	22.60	21.65	25.56	22.18	24.22
	Ours	<b>29.06</b>	<b>29.22</b>	<b>27.88</b>	<b>32.42</b>	29.48	<b>30.68</b>
SSIM ↑	MonoGS	0.62	0.69	0.74	0.74	0.75	0.70
LPIPS →	GLORIE-SLAM	0.87	0.87	0.83	0.84	<b>0.91</b>	0.76
	Splat-SLAM	0.83	0.87	0.86	0.87	0.84	0.84
	MVS-GS	0.82	0.88	0.84	0.85	0.80	0.77
	Ours	<b>0.88</b>	<b>0.91</b>	<b>0.90</b>	<b>0.91</b>	0.90	<b>0.88</b>
	MonoGS	0.70	0.51	0.55	0.54	0.63	0.58
LPIPS ←	GLORIE-SLAM	0.26	0.31	0.31	0.21	0.44	0.29
	Splat-SLAM	0.19	<b>0.15</b>	<b>0.18</b>	<b>0.15</b>	<b>0.23</b>	<b>0.19</b>
	MVS-GS	0.24	0.22	0.27	0.21	0.37	0.27
	Ours	<b>0.14</b>	<b>0.15</b>	<b>0.22</b>	<b>0.16</b>	0.27	<b>0.21</b>
							0.19

Table S1: Quantitative rendering performance on the ScanNet.

		Metrics	O-0	O-1	O-2	O-3	O-4	R-0	R-1	R-2	Avg.	
Keyframes		Depth L1 ↓	0.034	0.055	0.049	0.053	0.046	0.060	0.048	0.054	<b>0.050</b>	
Ours	Every 5 Frames	PSNR ↑	42.47	42.39	36.49	35.72	37.84	33.89	36.32	37.67	<b>37.85</b>	
		SSIM ↑	0.98	0.98	0.97	0.97	0.97	0.95	0.97	0.97	<b>0.97</b>	
		LPIPS ↓	0.01	0.02	0.03	0.03	0.03	0.04	0.04	0.03	<b>0.03</b>	
		PSNR ↑	27.15	28.85	28.56	33.95	36.27	26.78	26.95	28.11	<b>29.58</b>	
GLORIE-SLAM	Every 5 Frames	SSIM ↑	0.95	0.96	0.95	0.97	0.99	0.96	0.96	0.95	<b>0.96</b>	
		LPIPS ↓	0.15	0.12	0.16	0.11	0.08	0.17	0.13	0.17	<b>0.14</b>	
		Avg. FPS ↑	8.70	11.94	8.66	7.96	8.67	10.77	9.14	7.61	<b>9.18</b>	
GPU Usage [GiB] ↓			16.52	16.78	16.12	16.89	18.74	16.34	17.03	18.25	<b>17.24</b>	
Number of Gaussians (1000x)			1035	921	1062	1624	970	1472	1067	1290	<b>1183</b>	

Table S2: Full Evaluation on Replica [Straub *et al.*, 2019]. For keyframes, performance is measured only on the frames used for mapping, following the same approach as existing methods. Additionally, rendering results are provided every 5 frames, regardless of whether they are keyframes or not. These results are labeled as "Every 5 Frames" and evaluated under the same conditions as other method [Zhang *et al.*, 2024]. We report not only the depth evaluation results for keyframes and every 5 frames but also the maximum GPU usage and FPS.

		Metrics	f1_desk	f2_xyz	f3_office	Avg.	
Keyframes		Depth L1 ↓	0.121	0.313	0.241	<b>0.225</b>	
Ours	Every 5 Frames	PSNR ↑	23.59	22.79	25.44	<b>23.94</b>	
		SSIM ↑	0.81	0.77	0.86	<b>0.81</b>	
		LPIPS ↓	0.17	0.15	0.13	<b>0.15</b>	
		PSNR ↑	17.16	21.72	17.25	<b>17.65</b>	
GLORIE-SLAM	Every 5 Frames	SSIM ↑	0.79	0.92	0.73	<b>0.77</b>	
		LPIPS ↓	0.40	0.15	0.44	<b>0.37</b>	
		Avg. FPS ↑	1.41	7.77	4.76	<b>4.65</b>	
GPU Usage [GiB] ↓			11.31	12.81	13.25	<b>12.46</b>	
Number of Gaussians (1000x)			927	565	633	<b>708</b>	

Table S3: Full Evaluation on TUM-RGBD [Sturm *et al.*, 2012]. The setups and evaluated metrics are similar to Table S1.

		Metrics	scene_0000	scene_0059	scene_0106	scene_0169	scene_0181	scene_0207	Avg.	
Ours	Every 5 Frames	Keyframes	Depth L1 ↓	0.169	0.278	0.439	0.216	0.208	0.183	<b>0.248</b>
		PSNR ↑	26.23	26.23	21.84	29.90	26.92	28.76	<b>26.64</b>	
		SSIM ↑	0.80	0.85	0.78	0.87	0.87	0.84	<b>0.84</b>	
		LPIPS ↓	0.21	0.12	0.38	0.19	0.30	0.25	<b>0.25</b>	
GLORIE-SLAM	Every 5 Frames	PSNR ↑	21.80	18.66	17.54	22.85	19.14	21.52	<b>20.25</b>	
		SSIM ↑	0.83	0.77	0.77	0.88	0.81	0.81	<b>0.80</b>	
		LPIPS ↓	0.30	0.36	0.42	0.42	0.47	0.32	<b>0.38</b>	
Avg. FPS ↑			1.982	1.84	3.09	3.65	3.29	1.90	<b>2.63</b>	
GPU Usage [GiB] ↓			18.978	15.37	15.90	13.22	16.10	15.69	<b>15.97</b>	
Number of Gaussians (1000x)			2668	1695	1280	787	642	1690	<b>1460</b>	

Table S4: Full Evaluation on ScanNet [Dai *et al.*, 2017].

### B.3 Full Evaluation Results

In Tables S2, S3, and S4, we present comprehensive scene-by-scene results across all commonly used evaluation metrics for the Replica, TUM-RGBD, and ScanNet datasets. The results for keyframes in the image rendering evaluation can be found in the experiment section, and Tables S2, S3, and S4 provide results for every 5 frames. Every-5-frame rendering reflects the generalization performance of the 3D model, showing superior performance compared to [Zhang *et al.*, 2024]. It is also worth noting that there is not a significant performance difference compared to keyframes. This highlights that our approach demonstrates excellent generalization capabilities even for unseen views. Additionally, even with a large number of Gaussians, our method ensures real-time performance and can be effectively executed on standard GPUs, not just high-end server-grade hardware.

### B.4 Impact of the Number of Novel Views

Method	Metric	0	100(Ours)	150
Keyframes	PSNR $\uparrow$	38.871	39.283	39.310
	SSIM $\uparrow$	0.974	0.976	0.976
	LPIPS $\downarrow$	0.032	0.029	0.028
Every 5 Frames	Depth L1 $\downarrow$	0.056	0.050	0.058
	PSNR $\uparrow$	36.475	37.847	38.002
	SSIM $\uparrow$	0.968	0.971	0.964
	LPIPS $\downarrow$	0.033	0.031	0.034
	Depth L1 $\downarrow$	0.058	0.059	0.059
	FPS	12.07	9.180	8.54

Table S5: Impact of the number of novel views

We evaluated the effect of varying the number of frames used in the NVS training process following global DBA (see Table S5). Increasing the number of novel views consistently improved rendering quality but led to a decline in reconstruction quality. The enhancement in rendering quality can be attributed to the increased diversity of viewpoints, which contributes to better scene reconstruction. However, the absence of depth information in novel views resulted in a reduction in reconstruction quality. Moreover, a significant decrease in FPS was observed as the number of views increased, which presents a limitation. To achieve an optimal trade-off between real-time performance and accuracy, we selected 100 frames, as it provided the best balance for SLAM applications.

### B.5 Impact of Novel View Selection (NVS)

To evaluate the effectiveness of Novel View Selection (NVS), we conducted additional experiments on the TartanAir dataset [Wang *et al.*, 2020], which contains many challenging scenes such as occlusions, motion blur, and low-texture regions. As shown in Fig. S1, applying NVS significantly improves modeling completeness and performance in these difficult cases. This demonstrates that NVS effectively handles challenging viewing conditions and enhances robustness in complex environments.

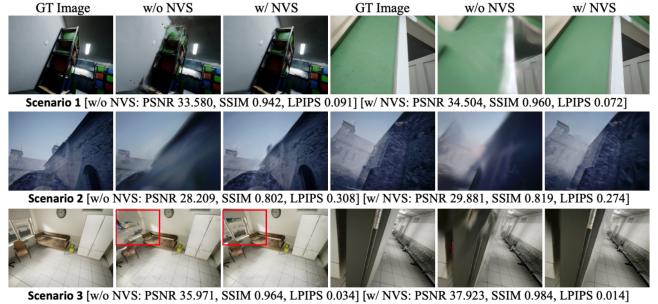


Figure S1: Comparison of results with and without NVS on the TartanAir dataset

### B.6 Final Refinement Iterations

Method	Metric	2K(Ours)	5K	10K	26K
Ours	PSNR $\uparrow$	39.28	40.19	41.18	42.39
	SSIM $\uparrow$	0.98	0.98	0.98	0.99
	LPIPS $\downarrow$	0.03	0.02	0.02	0.01
Splat-SLAM	PSNR $\uparrow$	36.77	37.80	38.41	38.95

Table S6: Impact of the number of final refinement iterations. Interestingly, our method, using only 2k iterations for refinement, outperforms Splat-SLAM [Sandström *et al.*, 2024], which uses 26k refinement iterations

After processing all sequences, we perform a final refinement inspired by the approaches in [Matsuki *et al.*, 2024], [Sandström *et al.*, 2024], and [Zhang *et al.*, 2024], incorporating both color loss and geometric depth loss, rather than relying solely on color loss. As shown in Table S6, we analyze the impact of varying the number of iterations for the final refinement (2K, 5K, 10K, and 26K iterations). While rendering accuracy improves with more iterations, geometric accuracy declines. This decline is attributed to continued training on regions with low confidence in the MVS, which negatively impacts depth consistency. To strike a balance between rendering quality, geometric accuracy, and computational efficiency, we select 2K iterations as it provides the optimal trade-off.

### B.7 Disparity Map Utilization in the MVS Module

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Depth L1 $\downarrow$	FPS $\uparrow$
w/o disparity	38.864	0.975	0.031	0.067	9.556
w/ disparity (Ours)	39.283	0.976	0.029	0.050	9.180

Table S7: Effectiveness of using disparity map in the MVS module. We report the rendering performance with and without the use of disparity maps.

Instead of simply using a Multi-View Stereo (MVS) model, we integrated the disparity map estimated from DROID-SLAM into a lower layer of the MVS model (Appendix A.2).

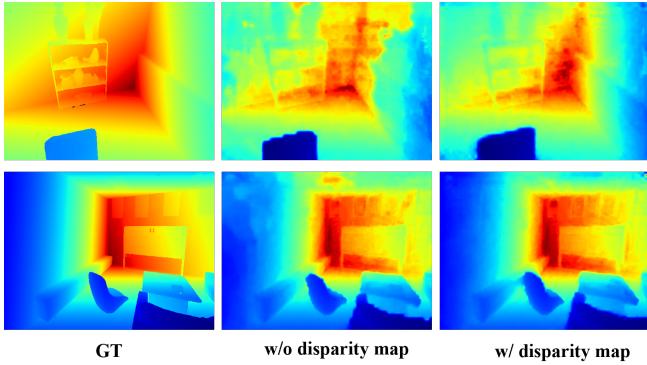


Figure S2: Effectiveness of using disparity map in the MVS module. We visualize output depths on the Replica dataset.

As shown in Table S7 and Fig. S2, this approach significantly enhances both reconstruction quality (Fig. S2) and rendering quality (Table S7).

## B.8 Comparison of Other View Selection

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Depth L1 $\downarrow$
Random	43.288	0.985	0.020	0.044
Quality(PSNR)	43.352	0.987	0.017	0.042
Uncertainty(Ours)	<b>43.932</b>	<b>0.988</b>	<b>0.016</b>	<b>0.034</b>

Table S8: Ablation Study of NVS on "office0" in the Replica dataset [Straub *et al.*, 2019]

We conducted experiments comparing the method of randomly selecting novel views, the PSNR-based selection method [Jin *et al.*, 2024], and our approach. As shown in Table S8, the results indicate that our method outperforms the others in both rendering and reconstruction quality, thereby demonstrating the effectiveness of our uncertainty-based approach.

## B.9 Efficiency Evaluation

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	GPU Usage [GIB] $\downarrow$	Avg. FPS $\uparrow$
MonoGS	23.53	0.75	0.33	14.62	0.32
Splat-SLAM	32.25	0.96	0.09	17.57	1.24
Ours	34.88	0.96	0.04	16.34	10.77

Table S9: Comparison of rendering accuracy and model efficiency (GPU usage and FPS) on the Replica "room 0".

As shown in Table. S9, our method not only achieves high rendering quality but also demonstrates superior runtime efficiency compared to existing approaches. Specifically, it delivers a significantly higher frame rate than MonoGS and Splat-SLAM, while maintaining competitive GPU memory usage. Despite the additional computational cost introduced by NVS, our method still achieves a practical frame rate suitable for real-time applications. This highlights its effectiveness in balancing accuracy and efficiency under resource-constrained settings.

## B.10 Runtime Analysis

	Tracking	MVS	GBA	NVS	3DGS Mapping	3DGS Opt.
Runtime (sec)	0.048	0.206	4.620	3.500	0.014	0.251 (per frame)
Invocation Interval (sec)	0.050	1.010		33.344		Consistent Optimization

Table S10: We report the average runtime and invocation interval of each module on the Replica "office 0". Modules separated by bold vertical lines run independently in parallel, while those with dashed lines are interdependent. As each module's runtime is shorter than its invocation interval, the system avoids bottlenecks.

Table. S10 summarizes the average runtime and invocation interval of each module in our system on the Replica "office 0". As shown in the table, all modules operate within their respective invocation intervals without exceeding them. This indicates that no single component becomes a performance bottleneck, enabling smooth and stable real-time operation.

Notably, the 3DGS optimization runs continuously and independently of the frontend modules such as tracking or MVS, ensuring consistent mapping quality without interrupting the overall pipeline. This design guarantees both efficiency and robustness, making our system suitable for long-term operation in complex environments.

## B.11 Additional Experiments

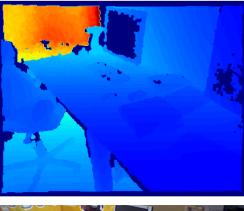
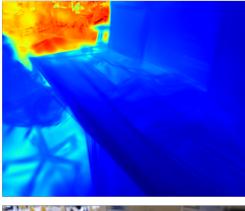
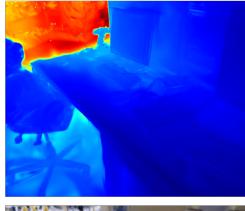
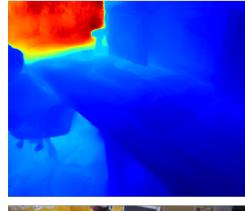
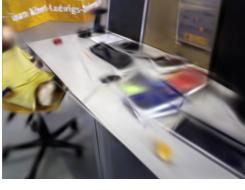
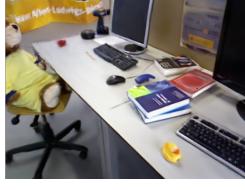
	GT	MonoGS	Photo-SLAM	Ours
				
				
Depth L1 [cm] ↓	27.00	38.6		22.56
PSNR ↑	19.03	19.51		28.10

Figure S3: Our system delivers superior scene reconstruction and rendering quality by utilizing enhanced depth estimation techniques and an optimized Gaussian learning approach. The results are averaged across all keyframes, with the experimental scene derived from the TUM-RGBD [Sturm *et al.*, 2012] fr3 office dataset.

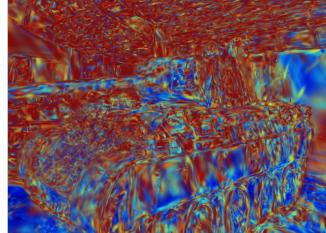
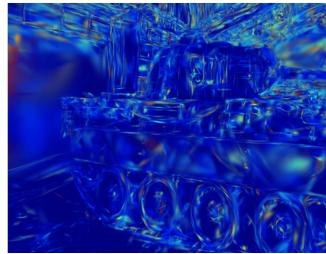
	Ground Truth	Rendering	Uncertainty Map
High Uncertainty			
Low Uncertainty			
	PSNR ↑	SSIM ↑	LPIPS ↓
High Uncertainty	19.609	0.632	0.193
Low Uncertainty	28.787	0.899	0.132
			Uncertainty Value ↓

Figure S4: High Uncertainty vs. Low Uncertainty. We visualize the ground-truth, rendered images, and corresponding uncertainty maps for high (top) and low (bottom) uncertainty views selected by our NVS module on scene "Panther" of the Tanks & Temples dataset [Knapitsch *et al.*, 2017]. As observed in the third column, the high uncertainty view contains many unexplored regions (denoted in red), while the low uncertainty view is already well-reconstructed (denoted by extensive blue regions). We further demonstrate the effectiveness of our proposed NVS through quantitative metrics, as shown in the table. The results indicate that the proposed uncertainty modeling method is highly accurate and plays a crucial role in our system.



Figure S5: Qualitative results on Replica [Straub *et al.*, 2019], TUM-RGBD [Sturm *et al.*, 2012] and Scannet [Dai *et al.*, 2017]. Our method surpasses other state-of-the-art models, such as Splat-SLAM [Sandström *et al.*, 2024], and occasionally produces rendering outputs that appear even more realistic than the ground truth.



(a) Truck



(b) Horse



(c) Barn

Figure S6: Qualitative results on Tanks and Temples [Knapitsch *et al.*, 2017].



(a) Scenario0



(b) Scenario1

Figure S7: Qualitative results on Aerial Dataset [Song *et al.*, 2021].

## References

- [Bhat *et al.*, 2023] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023.
- [Bloesch *et al.*, 2018] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2560–2568, 2018.
- [Cao *et al.*, 2022] Chenjie Cao, Xinlin Ren, and Yanwei Fu. Mvsformer: Multi-view stereo by learning robust image features and temperature-based depth. *arXiv preprint arXiv:2208.02541*, 2022.
- [Dai *et al.*, 2017] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.
- [Eftekhar *et al.*, 2021] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021.
- [Engel *et al.*, 2017] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.
- [Feng *et al.*, 2024] Ziyue Feng, Huangying Zhan, Zheng Chen, Qingan Yan, Xiangyu Xu, Changjiang Cai, Bing Li, Qilun Zhu, and Yi Xu. Naruto: Neural active reconstruction from uncertain target observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21572–21583, 2024.
- [Forster *et al.*, 2014] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.
- [Hamdi *et al.*, 2024] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19812–19822, 2024.
- [Hu *et al.*, 2024] Yan Song Hu, Nicolas Abboud, Muhammad Qasim Ali, Adam Srebrnjak Yang, Imad Elhajj, Daniel Asmar, Yuhao Chen, and John S Zelek. Mgso: Monocular real-time photometric slam with efficient 3d gaussian splatting. *arXiv preprint arXiv:2409.13055*, 2024.
- [Hu *et al.*, 2025] Jiarui Hu, Xianhao Chen, Boyin Feng, Guanglin Li, Liangjing Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field. In *European Conference on Computer Vision*, pages 93–112. Springer, 2025.
- [Huang *et al.*, 2024] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21584–21593, 2024.
- [Jiang *et al.*, 2025] Wen Jiang, Boshu Lei, and Kostas Daniilidis. Fisherrf: Active view selection and mapping with radiance fields using fisher information. In *European Conference on Computer Vision*, pages 422–440. Springer, 2025.
- [Jin *et al.*, 2024] Rui Jin, Yuman Gao, Yingjian Wang, Yuze Wu, Haojian Lu, Chao Xu, and Fei Gao. Gs-planner: A gaussian-splatting-based planning framework for active high-fidelity reconstruction. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11202–11209. IEEE, 2024.
- [Kerbl *et al.*, 2023] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.
- [Knapitsch *et al.*, 2017] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.
- [Koestler *et al.*, 2022] Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In *Conference on Robot Learning*, pages 34–45. PMLR, 2022.
- [Lee *et al.*, 2024] Byeonggwon Lee, Junkyu Park, Khang Truong Giang, Sungho Jo, and Soohwan Song. Mvs-gs: High-quality 3d gaussian splatting mapping via online multi-view stereo. *arXiv preprint arXiv:2412.19130*, 2024.
- [Ma *et al.*, 2024] Yikun Ma, Dandan Zhan, and Zhi Jin. Fastscene: Text-driven fast 3d indoor scene generation via panoramic gaussian splatting. In *IJCAI*, 2024.
- [Matsuki *et al.*, 2024] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024.
- [Mildenhall *et al.*, 2021] 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*, 65(1):99–106, 2021.

- [Mur-Artal *et al.*, 2015] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.
- [Newcombe *et al.*, 2011] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtm: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.
- [Peng *et al.*, 2024] Chensheng Peng, Chenfeng Xu, Yue Wang, Mingyu Ding, Heng Yang, Masayoshi Tomizuka, Kurt Keutzer, Marco Pavone, and Wei Zhan. Q-slam: Quadric representations for monocular slam. *arXiv preprint arXiv:2403.08125*, 2024.
- [Ran *et al.*, 2023] Yunlong Ran, Jing Zeng, Shibo He, Jiming Chen, Lincheng Li, Yingfeng Chen, Gimhee Lee, and Qi Ye. Neurar: Neural uncertainty for autonomous 3d reconstruction with implicit neural representations. *IEEE Robotics and Automation Letters*, 8(2):1125–1132, 2023.
- [Ranftl *et al.*, 2021] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021.
- [Redmon, 2016] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [Rosinol *et al.*, 2023] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE, 2023.
- [Rota Bulò *et al.*, 2024] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. In *European Conference on Computer Vision*, pages 347–362. Springer, 2024.
- [Sandström *et al.*, 2024] Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R Oswald, and Federico Tombari. Splat-slam: Globally optimized rgb-only slam with 3d gaussians. *arXiv preprint arXiv:2405.16544*, 2024.
- [Scott *et al.*, 2003] William R Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys (CSUR)*, 35(1):64–96, 2003.
- [Song *et al.*, 2021] Soohwan Song, Daekyun Kim, and Sunghee Choi. View path planning via online multiview stereo for 3-d modeling of large-scale structures. *IEEE Transactions on Robotics*, 38(1):372–390, 2021.
- [Straub *et al.*, 2019] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.
- [Sturm *et al.*, 2012] 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*, pages 573–580. IEEE, 2012.
- [Teed and Deng, 2020] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.
- [Teed and Deng, 2021] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021.
- [Wang *et al.*, 2004] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [Wang *et al.*, 2020] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020.
- [Yan *et al.*, 2023] Dongyu Yan, Jianheng Liu, Fengyu Quan, Haoyao Chen, and Mengmeng Fu. Active implicit object reconstruction using uncertainty-guided next-best-view optimization. *IEEE Robotics and Automation Letters*, 2023.
- [Yao *et al.*, 2018] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018.
- [Zhang *et al.*, 2023] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3727–3737, 2023.
- [Zhang *et al.*, 2024] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, 2024.
- [Zhou *et al.*, 2023] Xingchen Zhou, Ying He, F Richard Yu, Jianqiang Li, and You Li. RePaint-NeRF: Nerf editting via semantic masks and diffusion models. In *IJCAI*, 2023.
- [Zhu *et al.*, 2024] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *2024 International Conference on 3D Vision (3DV)*, pages 42–52. IEEE, 2024.