IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

SAGA-SLAM: Scale-Adaptive 3D Gaussian Splatting for Visual SLAM

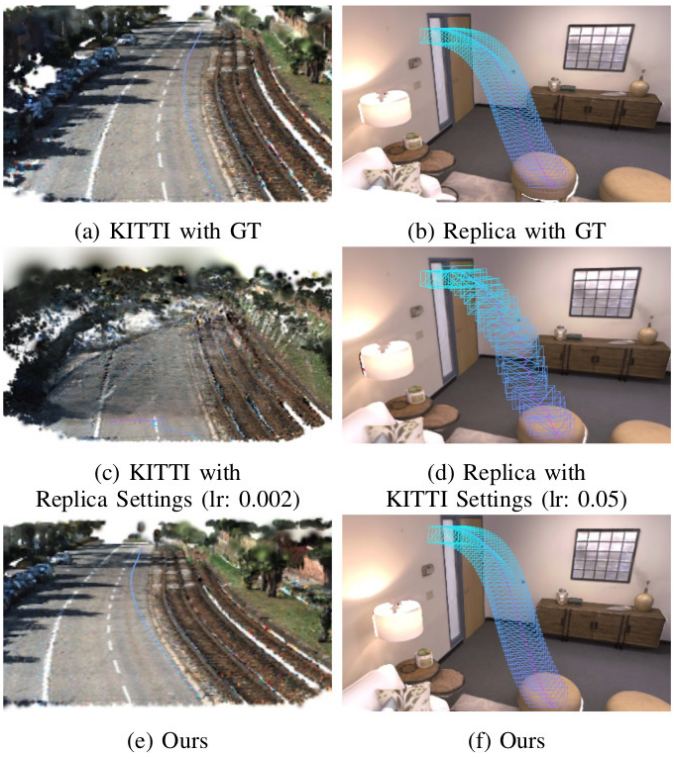
Kun Park[1] and Seung-Woo Seo[1]

***Abstract*—3D Gaussian Splatting (3DGS) has recently emerged as a powerful technique for representing 3D scenes. Its superior high-fidelity rendering quality and speed have driven its rapid adoption in many applications. Among them, Visual Simultaneous Localization and Mapping (VSLAM) is the most prominent application, as it requires real-time simultaneous mapping and position tracking of navigating objects. However, from our comprehensive study, we observed a fundamental hurdle in directly applying the current 3DGS technique to VSLAM, which we define as the scale adaptation problem. The scale adaptation problem refers to the inability of existing 3DGS-based SLAM methods to address varying scales, specifically the extent of camera pose difference from the perspective of tracking, and environmental size in terms of mapping and the addition of new 3D Gaussians. To overcome this limitation, we propose SAGA-SLAM, the first scale-adaptive RGB-D Dense SLAM framework based on 3DGS. We optimize the tracking and mapping stages robustly over various scales by utilizing the Polyak step size and momentum. Additionally, we present gaussian fission method to address the scale problem during the addition of 3D Gaussians. Experiments show that our method achieves state-of-the-art results robustly on both large and small scales, such as KITTI, Replica, and TUM-RGBD. By adapting without the need for hyperparameter tuning, our method demonstrates both superior performance and practical applicability.**

***Index Terms*—RGB-D Perception; SLAM; 3D Gaussian Splatting;**

I. Introduction

3DGS has gained significant attention in 3D reconstruction. By modeling each point in 3D space as a Gaussian distribution, it enables a smooth and continuous representation of the environment. Its fast rendering speed and photorealistic quality have further driven its adoption in neural rendering, novel view synthesis, and immersive virtual environments. Among these applications, Visual SLAM stands out as a key task, constructing maps while simultaneously estimating pose. It plays a fundamental role in autonomous systems, enabling perception, navigation, and interaction with their environments. Given the cost-effectiveness and accessibility of visual sensors, this approach is suited for various robots and is crucial in fields such as augmented reality and digital twins. Leveraging 3DGS as a map, 3DGS-based SLAM methods [1]–[5] operate in three stages: tracking, mapping, and addition of new 3D Gaussians. In tracking, the camera pose is optimized relative to the map. During mapping, 3D Gaussians are further refined to enhance the map's accuracy. In the addition stage, new 3D Gaussians are allocated to represent newly discovered space. These methods not only generate high-quality photorealistic maps but also achieve superior tracking performance, establishing 3DGS-based SLAM as a leading approach in learning-based SLAM and motivating diverse follow-up research.



(a) KITTI with GT (b) Replica with GT

(c) KITTI with Replica Settings (lr: 0.002) (d) Replica with KITTI Settings (lr: 0.05)

(e) Ours (f) Ours

Fig. 1: **Difficulty of Adaptation Across Environments of Varying Scales.** (a) and (b) show the result of SplaTAM on KITTI and Replica with ground truth pose. (c) and (d) present the SplaTAM results when the settings are mismatched to the scale. (e) and (f) are KITTI and Replica results using our method.

However, as illustrated in Fig. 1, we observed a critical scale adaptation issue in these methods through diverse experiments. To explore the reason for this issue, we investigate how environmental changes influence the performance of existing

Manuscript received: February, 10, 2025; Revised May, 1, 2025; Accepted June, 18, 2025.

This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-00207391, Development of Hashgraph-based Blockchain Enhancement Scheme and Implementation of Testbed for Autonomous Driving) (*Corresponding author: Seung-Woo Seo.*)

[1]The authors are with the Department of Electrical and Computer Engineering, and Institute of New Media and Communications, Seoul National University, Republic of Korea {gun615, sseo}@snu.ac.kr

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

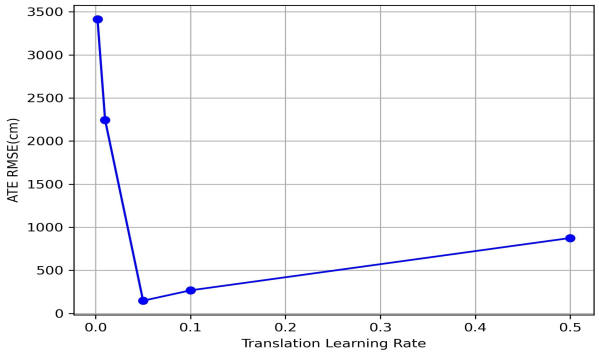


Fig. 2: **Tracking Performance of SplaTAM with Varying Translation Learning Rates on KITTI.** The x-axis represents the translation learning rate, and the y-axis represents the tracking error. The best performance is observed at a learning rate of 0.05, with the error increasing as the learning rate decreases or increases.

methods. Datasets commonly used for evaluation in existing research pipelines, such as Replica [6], TUM [7], and ScanNet [8], exhibit characteristics such as indoor scenes with relatively small spatial extents, captured by handheld cameras. In contrast, KITTI [9], which was captured by vehicle-mounted cameras, features outdoor scenes with expansive spatial extents. These various properties result in scale differences across the three stages of 3DGS-based methods. In this paper, we define the scale as follows:

- From the perspective of tracking, scale refers to the extent of camera pose difference between consecutive input image frames, influenced by the linear and angular speed of the camera.
- In the context of mapping and addition of new 3D Gaussians, scale pertains to the size of the surrounding environment.

Such scale variations result in performance deficiencies in two categories of components: (1) optimization of tracking and mapping, and (2) the addition of new 3D Gaussians.

First, for the optimization of tracking and mapping, existing methods employ Adam [10] with learning rates tailored for small-scale environments. Nonetheless, as shown in Fig. 1, a mismatch between the learning rate and the scale can result in a significant degradation in performance. For instance, a relatively small translation learning rate of 0.002 demonstrated remarkable performance for SplaTAM [2] on the Replica. However as illustrated in Fig. 2, a higher translation learning rate of 0.05 is more suitable for the KITTI, which represents a larger-scale environment. In the case of SplaTAM, it requires up to seven different learning rates for tracking and mapping, making it exceedingly difficult to identify the optimal learning rates for each environment. Moreover, even if appropriate learning rates are found, they vary at every timestep due to the dynamic changes in object speed and the surrounding environment. Therefore, existing methods were developed under the assumption that the environmental scale would remain constant at each timestep, which is inherently unrealistic.

Second, the adaptation issue also arises during the addition of new 3D Gaussians. In this stage, 3DGS-based methods allocate new 3D Gaussians to new areas. The number of new 3D Gaussians is determined by the number of pixels in the input image. Each pixel generates one 3D Gaussian, which is then unprojected into 3D using the depth input. As a result, regardless of whether the scale of the new space is large or small, the number of 3D Gaussians that can be added is limited. Therefore, in indoor environments with a short maximum depth, all objects can receive a sufficient number of 3D Gaussians. However, in outdoor environments with a long maximum depth, as objects move farther from the camera, their corresponding pixel count decreases, leading to an insufficient allocation of 3D Gaussians. Such insufficiency diminishes the map representation capability.

To address these issues, we propose SAGA-SLAM, the first **S**cale-**A**daptive 3D **Ga**ussian Splatting-based **SLAM**. Our system optimizes tracking and mapping by comparing the RGB-D rendered image with the ground truth. To tackle the scale problem in these stages, we integrate a method [11] that combines the Polyak step size [12] and momentum. Polyak step size is a subgradient method for scale-adaptive learning rates, which allows adaptation to various scales, and momentum enhances convergence speed and robustness. Additionally, we present gaussian fission to overcome the limitation during the addition of new 3D Gaussians. This approach, inspired by nuclear fission, splits 3D Gaussians to provide a sufficient number effectively representing the environment. We conducted experiments on datasets of various scales, achieving state-of-the-art performance in learning-based SLAM models with respect to both tracking and mapping. Overall, our contributions can be summarized as follows:

- To the best of our knowledge, this is the first study to address the scale adaption issue in existing learning-based SLAM systems.
- We leverage the Polyak step size and momentum to enable tracking and mapping to adapt to diverse environments and varying object movement scales, eliminating the need for a learning rate.
- We propose gaussian fission to address the issue of insufficient 3D Gaussians caused by scale variations. This module ensures effective mapping performance, even in large-scale environments.

II. Related Works

A. SLAM using 3D Gaussian Splatting

Early learning-based SLAM methods [13]–[16] leveraged Neural Radiance Fields (NeRF) [17] to model the surrounding environment. Following the construction of the environmental model, NeRF was further utilized to estimate the camera pose, earning recognition for its capability to generate high-quality dense maps. Nevertheless, these implicit representation methods face slow inference speed. To overcome this challenge, several representative studies have utilized 3DGS in SLAM including SplaTAM [2], MonoGS [3], and others [4], [5]. Despite variations in naming and implementation details, they generally follow a common architectural framework: tracking, mapping, and adding new 3D Gaussians. Among

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

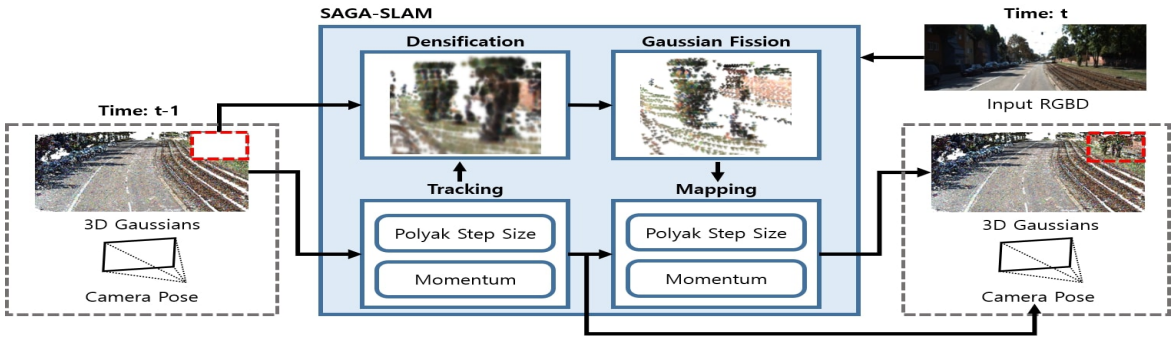


Fig. 3: **Framework Overview.** Given an RGB-D input, four stages—tracking, mapping, densification, and gaussian fission-are performed for each frame, producing a photorealistic dense map and camera pose as outputs. The **tracking and mapping** stages update camera pose and 3D Gaussians using the Polyak step size and momentum. In **densification stage**, 3D Gaussians are unprojected from the input RGB-D image. New module, **gaussian fission**, is proposed to enhance scale adaptability when adding 3D Gaussians.

these, SplaTAM [2] referred to the process of adding new 3D Gaussians as densification, which we adopted for our implementation. Subsequent studies have focused on further improving real-time performance. Photo-SLAM [18] employs ORB-SLAM3 [19] features to achieve more efficient and robust optimization during tracking. Similarly, HI-SLAM2 [20] leverages DROID-SLAM [21] for online tracking. CaRtGS-SLAM [22] introduces a computational alignment strategy that effectively tackles issues related to insufficient and long-tail optimization.

B. Adaptive Optimization Methods

Manually tuning the learning rate is both inefficient and computationally expensive. In this context, adaptive learning rate methods have proven effective in enhancing optimization processes by automating the adjustment of learning rates. Nesterov Accelerated Gradient (NAG) [23] enhances momentum-based optimization by calculating gradients at a future position. RMSProp [24] stabilizes updates by normalizing gradients with a moving average of their squared values. Adam [10] combined RMSProp with momentum, providing robust optimization across diverse tasks. However, its coupled weight decay mechanism limited generalization.

Polyak step size [12], originally designed for convex, non-smooth optimization problems, introduced a theoretically grounded method for adaptively determining step sizes based on the gap between the current loss and the optimal loss. Stochastic Polyak Step-size (SPS) [25] extended this concept to Stochastic Gradient Descent. Building on this, MoMo [11] integrates the Polyak step size with momentum-based optimization, offering a more robust and efficient adaptive learning rate mechanism.

III. PRELIMINARIES

A. 3D Gaussian Splatting

Following the approach used in SplaTAM [2], 3D Gaussians are projected onto the image plane, where they undergo a weighted summation process. In this process, the contribution of each Gaussian to the final pixel value is determined by a set of coefficients computed based on the 3D Gaussian's properties. These coefficients are derived from the position, radius, opacity, and RGB values of the 3D Gaussians, ensuring that each contributes proportionally to its influence within the projection area, as follows:

$$w(\mathbf{x}) = o \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}\|^2}{2r^2}\right), \tag{1}$$

where $\mathbf{x}$ is the projected position, $\boldsymbol{\mu}$ is the center position of the 3D Gaussian, r is the radius, and o is the opacity. Using Eq. (1), color and depth can be rendered as follow:

$$\hat{C}(\mathbf{p}) = \sum_{i=1}^{n} c_i w_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - w_j(\mathbf{p})), \tag{2}$$

$$\hat{D}(\mathbf{p}) = \sum_{i=1}^{n} d_i w_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - w_j(\mathbf{p})), \tag{3}$$

where $\mathbf{p}$ corresponds to the 2D pixel, n is the number of 3D Gaussians along the ray at the pixel, c_i represents the RGB value of each 3D Gaussian, and d_i represents the depth value of each 3D Gaussian.

B. Polyak Step Size

The Polyak step size [12] is an adaptive subgradient method for setting the step size in convex, non-smooth optimization problems. It utilizes the current objective function value $f(x_k)$ and the known optimal value f^*. The step size can be defined as:

$$\alpha_k = \frac{f(x^k) - f^*}{\|\nabla f(x^k)\|^2}, \tag{4}$$

where k is the current iteration, α_k represents the step size, $\nabla f(x_k)$ is the gradient at x_k, and f^* is the optimal value of the objective function. In this study, f is defined as the loss function, and its optimal value is assumed to be zero, as suggested in SPS [25].

IV. METHODS

Fig. 3 illustrates an overview of the SAGA-SLAM framework, which processes input RGB-D data in four stages: tracking, mapping, densification, and gaussian fission. The tracking and mapping stages leverage the 3DGS rendering technique to rasterize RGB-D images. During tracking, the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

camera pose is optimized by comparing the rendered images with the ground truth, while during mapping, the parameters of the 3D Gaussians are refined. To enable effective optimization across varying scales, we propose an approach that integrates Polyak step size with momentum. Following the tracking stage, changes in the camera pose may uncover previously unseen areas. In the densification stage, new 3D Gaussians are added to represent these regions. Finally, we introduce gaussian fission, a novel methodology that addresses the scale adaptability of densification by splitting large-radius 3D Gaussians, which result from their allocation in distant regions.

A. Tracking and Mapping

In the tracking stage, 3D Gaussian parameters are fixed while the camera pose is estimated and adjusted by comparing the rendered RGB-D image with the ground truth. In the mapping stage, the process is reversed. The pose from tracking is fixed, and the 3D Gaussian parameters are optimized. The error between the rendered RGB-D and the ground truth serves as the loss function:

$$\mathcal{L} = \sum_{\mathbf{p}} \left\| \mathbf{C}(\mathbf{p}) - \hat{\mathbf{C}}(\mathbf{p}) \right\|_1 + \left\| \mathbf{D}(\mathbf{p}) - \hat{\mathbf{D}}(\mathbf{p}) \right\|_1. \quad (5)$$

The scales of tracking and mapping directly affect the magnitude of the parameters optimized by this function. Thus, the optimum of this function must be found in a scale adaptive manner, which will be discussed in Sec. IV-B.

B. Polyak Step Size with Momentum

In the tracking and mapping stages, we aim to solve the following optimization problem by effectively updating θ using scale-adapted learning rates:

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(x, \theta), \quad \mathcal{L}(x, \theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[\mathcal{L}(x, \theta) \right]. \quad (6)$$

Here, $\mathcal{L}(x, \theta)$, defined in Eq. 5, represents the loss function, where x denotes the input RGB-D image, and θ refers to the parameters being optimized—camera pose in tracking and 3D Gaussians in mapping. We can approximate this problem utilizing the Quadratic Taylor Series:

$$\mathcal{L}(\theta^{k+1}) \approx \mathcal{L}(\theta^k) + \nabla \mathcal{L}(\theta^k)^\top \left(\theta^{k+1} - \theta^k\right) + \frac{1}{2} \left(\theta^{k+1} - \theta^k\right)^\top \nabla^2 \mathcal{L}(\theta^k) \left(\theta^{k+1} - \theta^k\right).$$

And by assuming that the hessian matrix takes the form $\mathbf{H} = \alpha \mathbf{I}$ in the same manner as proximal point method [26], where $\mathbf{I}$ is the identity matrix and α is the constant, the expression can be simplified as follows:

$$\mathcal{L}(\theta^{k+1}) \approx \mathcal{L}(\theta^k) + \nabla \mathcal{L}(\theta^k)^\top \left(\theta^{k+1} - \theta^k\right) + \frac{1}{2\alpha_k} \|\theta^{k+1} - \theta^k\|^2. \quad (7)$$

In this context, α_k denotes the learning rate, which we aim to make scale-adaptive using the Polyak step size. At this point, the expectation in Eq. (6) can be expressed as follows:

$$\mathcal{L}(x, \theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[\mathcal{L}(x, \theta) \right] \approx \frac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} \mathcal{L}(x_j, \theta), \quad (8)$$

Algorithm 1 Update with Polyak Step Size and Momentum.

Default settings: $\alpha_k = 1$, $\beta = 0.9$.
Input: $x_k \in \mathbb{R}^{H*W*C}, \theta_k \in \mathbb{R}^n$.
Init: $\mathcal{L}_0 = \mathcal{L}(x_1, \theta^1)$, $d_0 = \nabla \mathcal{L}(x_1, \theta^1)$, $\gamma_0 = \langle d_0, x_1 \rangle$.
for $k = 1$ to $K - 1$ **do**
  $\bar{\mathcal{L}}_k = (1 - \beta)\mathcal{L}(x_k, \theta^k) + \beta \bar{\mathcal{L}}_{k-1}$
  $\gamma_k = (1 - \beta)\langle \nabla \mathcal{L}(x_k, \theta^k), \theta^k \rangle + \beta \gamma_{k-1}$
  $d_k = (1 - \beta)\nabla \mathcal{L}(x_k, \theta^k) + \beta d_{k-1}$
  $\theta^{k+1} = \theta^k - \min \left\{ \alpha_k, \frac{\bar{\mathcal{L}}_k + \langle d_k, \theta^k \rangle - \gamma_k}{\|d_k\|^2} \right\} d_k$
end for

where, ρ is the coefficient, which serves as a constant for applying momentum in subsequent steps. Applying Eq. (8) to Eq. (7), we obtain the following equation:

$$\mathcal{L}(\theta^{k+1}) \approx \frac{1}{\rho_k} \left(\sum_{j=1}^{k} \rho_{j,k} \left(\mathcal{L}(x_j, \theta^j) + \langle \nabla \mathcal{L}(x_j, \theta^j), \theta^k - \theta^j \rangle \right) \right) + \frac{1}{2\alpha_k} \|\theta^{k+1} - \theta^k\|^2.$$

Solving this equation in closed form yields the following expression:

$$\theta^{k+1} = \theta^k - \alpha_k \frac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} \nabla \mathcal{L}(x_j, \theta^j). \quad (9)$$

We can define α_k to apply Polyak step size, Eq. (4), as follows:

$$d_k := \frac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} \nabla \mathcal{L}(x_j, \theta^j),$$

$$\bar{\mathcal{L}}_k := \frac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} \mathcal{L}(x_j, \theta^j),$$

$$\gamma_k := \frac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} \langle \nabla \mathcal{L}(x_j, \theta^j), \theta^j \rangle.$$

$$\alpha_k := \frac{\bar{\mathcal{L}}_k + \langle d_k, \theta^k \rangle - \gamma_k}{\|d_k\|^2}. \quad (10)$$

Additionally, for momentum, We apply ρ_k in the same way as used in Adam [10]:

$$\rho_{j,k} = (1 - \beta)\beta^{k-j}, \quad j = 1, \ldots, k. \quad (11)$$

Finally, by substituting Eq. (10) and Eq. (11) into Eq. (9), we derive the final expression:

$$\theta^{k+1} = \theta^k - \frac{\bar{\mathcal{L}}_k + \langle d_k, \theta^k \rangle - \gamma_k}{\|d_k\|^2} \frac{1}{\rho_k} \sum_{j=1}^{k} (1 - \beta)\beta^{k-j} \nabla \mathcal{L}(x_j, \theta^j).$$

This equation guides the parameter updates in the tracking and mapping modules, as detailed in Algorithm 1. By leveraging this approach, we achieve mathematically robust optimization, effectively adapting to loss functions that vary across different scales.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

C. Densification

This is the stage where new 3D Gaussians are allocated. When the camera illuminates a new space, the existing 3D Gaussians struggle to represent the newly observed area. Therefore, new 3D Gaussians are initialized in the unexplored regions to prepare for mapping. We have added new 3D Gaussians using the densification methodology from SplaTAM [2], among 3DGS-based SLAM [2]–[5]. Through Eq. 1, a silhouette image can be rendered to determine visibility:

$$S(\mathbf{p}) = \sum_{i=1}^{n} w_i(\mathbf{p}) \prod_{j=1}^{i-1} \left(1 - w_j(\mathbf{p})\right). \tag{12}$$

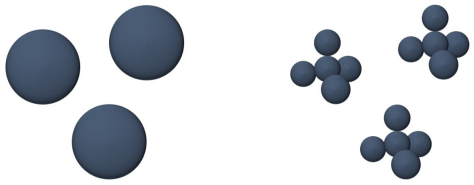
If a pixel $\mathbf{p}$ contains sufficient information from the current map, the value of the silhouette increases. Based on this value, a criterion can be established for identifying new regions illuminated by the camera's movement. Additionally, if the estimated depth exceeds the ground truth, it is assumed that there is likely an object in front of the camera, leading to the addition of a new 3D Gaussian at the corresponding pixel. In other words, if either the silhouette condition or the depth condition is satisfied, it is determined that the corresponding pixel lacks sufficient information, prompting densification. To add a 3D Gaussian, the ground truth depth at the corresponding pixel is utilized to unproject it into 3D space.

D. Gaussian Fission

In the densification stage, new 3D Gaussians are added with the color of the corresponding pixel, centered at the location of the unprojection of the pixel's depth, an opacity of 0.5, and radius is determined by dividing the depth by the focal length. This radius formulation implies that as the depth increases, the radius grows accordingly. Therefore, we propose a method called gaussian fission, where if the radius of a 3D Gaussian exceeds a certain threshold, it splits into smaller 3D Gaussians. In this process, a single 3D Gaussian divides into k smaller ones, preserving the same RGB values and opacity. For implementation, k is set to 5. Let the radius of the original 3D Gaussian be r_0 and its center be at the coordinates (x_0, y_0, z_0). Then, the radius and coordinates of the five subdivided 3D Gaussians are given as follows:

$$\begin{aligned}
r_k &= \frac{1}{3} r_0, \quad k = 0, \ldots, 4, \\
X_0 &= (x_0,\ y_0,\ z_0), \quad X_1 = (x_0,\ y_0,\ z_0 + 2r_k), \\
X_2 &= \left(x_0 - \frac{2\sqrt{6}}{3} r_k,\ y_0 - \frac{2\sqrt{2}}{3} r_k,\ z_0 - \frac{2}{3} r_k \right), \\
X_3 &= \left(x_0 + \frac{2\sqrt{6}}{3} r_k,\ y_0 - \frac{2\sqrt{2}}{3} r_k,\ z_0 - \frac{2}{3} r_k \right), \\
X_4 &= \left(x_0,\ y_0 + \frac{4\sqrt{2}}{3} r_k,\ z_0 - \frac{2}{3} r_k \right).
\end{aligned}$$

where X_i represents the center of each divided 3D Gaussian. The radii of the newly divided 3D Gaussians are all equal, and the centers of Gaussian$_{1\sim4}$ form a tetrahedron, ensuring that the 3D Gaussians are evenly distributed. These equations yield results consistent with the geometric shapes shown in Fig. 4.



(a) Before Gaussian Fission (b) After Gaussian Fission

Fig. 4: **Result of Gaussian Fission.** (a) represents the original large-radius 3D Gaussian, while (b) shows the new 3D Gaussians after fission.

gaussian fission, akin to nuclear fission, continues iteratively until the radius of the 3D Gaussian falls below a predefined threshold.

V. Experiments

A. Experimental Setup

Dataset: To evaluate SAGA-SLAM, we conduct experiments on the Replica [6], TUM-RGBD [7] and KITTI [9]. Replica and TUM-RGBD are the most commonly used datasets for evaluating existing learning-based SLAM methods. These datasets are small-scale, captured indoors with handheld cameras. KITTI, on the other hand, is used to validate performance in large-scale environments and is captured outdoors with cameras mounted on cars. For Replica and TUM-RGBD, evaluation was conducted on all scenes in the datasets. For KITTI, raw data was used for RGB images, and depth completion was applied for depth images. Additionally, considering the presence of dynamic objects, 5 scenes were selected for the experiments.

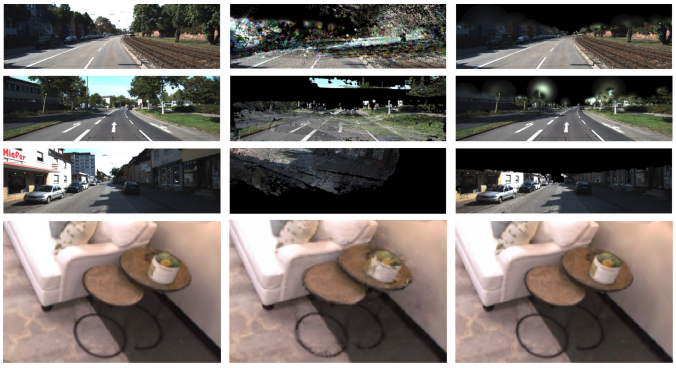
Metric: We follow all evaluation metrics for camera pose estimation and mapping performance as SplaTAM [2]. For camera pose estimation, we use the Average Absolute Trajectory Error (ATE) RMSE. We further evaluate the mapping performance using the Peak Signal-to-Noise Ratio (PSNR), SSIM, LPIPS as metrics for rendering quality, and L1 Depth error as a metric for 3D reconstruction accuracy.

Implementation Details: Our work is implemented in Python using the PyTorch framework, and the system is run on an Nvidia RTX 3090 GPU and an Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz. For gaussian fission, the radius threshold is set to 0.03. During the evaluation of SplaTAM [2] and MonoGS [3] on the KITTI dataset, the learning rates were kept consistent with the settings used for Replica. Furthermore, we performed hyperparameter tuning for SplaTAM [2] on KITTI and included the results in the comparison. The number of tracking/mapping iterations was set to 40/60 for Replica and 200/30 for TUM-RGBD, following the configuration used in SplaTAM. For KITTI and the Runtime and Memory experiments, the iterations were set to 50/50.

B. Evaluation of Localization and Mapping

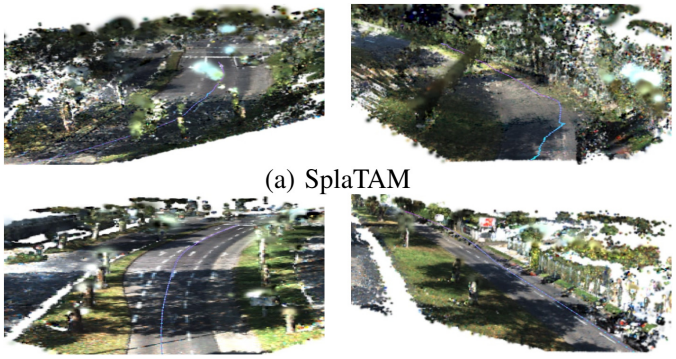
Evaluation on KITTI. Table I presents the results of tracking performance comparison on the large-scale KITTI dataset. SplaTAM and MonoGS show a complete failure in localization, which is due to their inability to adapt to the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) GT (b) SplaTAM (c) Ours

Fig. 5: **Rendering Results on KITTI and Replica.** The first to third rows correspond to the KITTI dataset, while the fourth row corresponds to the Replica dataset. (a) column shows the ground truth, (b) column shows the results of SplaTAM, and (c) column shows our results.



(a) SplaTAM

(b) Ours

Fig. 6: **Visualization of 3D Gaussians and Camera Trajectories on the KITTI.** (a) visualizes the 3D results of SplaTAM on the KITTI dataset. While it fails to accurately capture the road-following scenario, (b) demonstrates that our method produces precise results.

Methods	Avg.	scene1	scene2	scene11	scene13	scene48
MonoGS	2446.68	3141.41	1611.24	2706.86	4229.46	544.44
HI-SLAM2	2304.68	2543.43	1769.70	2718.88	4011.15	480.23
SplaTAM	1549.24	2428.37	1089.93	1352.34	2719.15	156.39
SplaTAM (HPT)	543.91	1244.06	1155.71	79.62	147.59	92.55
Photo-SLAM	140.66	198.78	**36.56**	**46.66**	150.36	270.93
Ours	**61.71**	**69.31**	40.85	48.75	**116.55**	**33.07**

TABLE I: Tracking Results on KITTI. (ATE RMSE[cm] ↓)

scale of KITTI. While hyperparameter tuning improved the performance of SplaTAM, it remained insufficient to achieve competitive results. This limitation highlights the inadequacy of a fixed learning rate in representing the dynamically changing scales at each timestep, even with hyperparameter tuning. Notably, even within the same KITTI dataset, the most suitable learning rates varied considerably across different scenes. For instance, the translation learning rate values of 0.1, 0.5, 0.05, 0.05, and 0.05 yielded the best performance for their respective scenes. HI-SLAM2 exhibited a similar trajectory shape but failed to achieve proper scale alignment.

Methods	Metrics	Avg.	scene1	scene2	scene11	scene13	scene48
MonoGS	PSNR ↑	6.44	6.90	6.95	6.56	6.71	6.06
	SSIM ↑	0.01	0.01	0.01	0.01	0.01	0.01
	LPIPS ↓	0.84	0.84	0.80	0.87	0.89	0.78
HI-SLAM2	PSNR ↑	23.74	24.41	26.47	23.27	20.00	24.55
	SSIM ↑	0.778	0.79	0.80	0.79	0.69	0.82
	LPIPS ↓	0.268	0.26	0.24	0.23	0.34	0.20
SplaTAM	PSNR ↑	17.33	15.42	17.11	17.36	16.79	19.99
	SSIM ↑	0.66	0.60	0.62	0.65	0.64	0.78
	LPIPS ↓	0.43	0.49	0.50	0.45	0.46	0.28
SplaTAM (HPT)	PSNR ↑	27.37	26.10	26.45	28.46	26.55	29.27
	SSIM ↑	0.78	0.75	0.77	0.80	0.75	0.83
	LPIPS ↓	0.06	0.06	0.05	0.05	0.10	0.03
Photo-SLAM	PSNR ↑	13.84	13.39	14.49	12.20	11.07	16.04
	SSIM ↑	0.58	0.62	0.58	0.62	0.71	0.36
	LPIPS ↓	0.78	0.83	0.75	0.91	0.90	0.53
Ours	PSNR ↑	**31.27**	**31.62**	**31.70**	**31.41**	**28.85**	**32.79**
	SSIM ↑	**0.99**	**0.99**	**0.99**	**0.99**	**0.98**	**0.99**
	LPIPS ↓	**0.02**	**0.02**	**0.03**	**0.02**	**0.04**	**0.01**

TABLE II: Mapping Results on KITTI.

Methods	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
NICE-SLAM	1.06	0.97	1.31	1.07	0.88	1.01	1.06	1.10	1.13
ESLAM	0.63	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63
Point-SLAM	0.52	0.61	0.41	0.37	0.38	0.84	0.54	0.69	0.72
MonoGS	0.58	0.44	**0.32**	0.31	0.44	0.52	**0.23**	**0.17**	2.25
GS-SLAM	0.50	0.48	0.53	0.33	0.52	0.41	0.59	0.46	0.7
Photo-SLAM	0.60	0.54	0.39	0.31	0.52	0.44	1.28	0.78	0.58
CaRtGS	0.46	0.32	0.36	**0.19**	0.49	0.39	1.03	0.42	0.47
SplaTAM	0.36	0.31	0.40	0.29	0.47	0.27	0.29	0.32	0.55
MGS-SLAM [27]	0.32	0.36	0.35	0.32	**0.35**	0.28	0.26	0.32	**0.34**
Ours	**0.31**	**0.29**	0.50	0.24	0.45	**0.17**	0.26	0.22	0.36

TABLE III: Tracking Results on Replica. (ATE RMSE[cm] ↓)

Photo-SLAM showed competitive performance, which can be attributed to its use of ORB-SLAM3, known for its robustness in outdoor environments. SAGA-SLAM demonstrates stable performance on KITTI. Table II shows the results of mapping performance comparison on KITTI. Similar to the tracking results, the existing 3DGS-based SLAM fails in 3D reconstruction, whereas our work achieves superior performance. Even in the qualitative results shown in Fig. 5, it can be observed that our work produces higher-quality images compared to the rendering results of SplaTAM. Fig. 6 presents the results of 3D visualization, where the scene follows the trajectory of the path. The curve connecting the blue dots represents the estimated camera poses as a trajectory.

Evaluation on Replica and TUM-RGBD. Table IV and Table III present the results of tracking and mapping performance comparisons on the small-scale Replica dataset. Replica is one of the most commonly used datasets for evaluating learning-based SLAM. SAGA-SLAM demonstrates the best performance in both tracking and mapping on Replica as well. In particular, for mapping, the average PSNR exceeded 40, demonstrating outstanding performance.

Table V presents the results of tracking performance comparisons on the TUM-RGBD dataset. Due to the poor qual-

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Methods	Metrics	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
NICE-SLAM	PSNR ↑	24.42	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94
	SSIM ↑	0.81	0.69	0.77	0.81	0.87	0.88	0.81	0.80	0.86
	LPIPS ↓	0.23	0.33	0.27	0.21	0.21	0.18	0.24	0.21	0.20
	Depth L1 [cm] ↓	3.53	-	-	-	-	-	-	-	-
Point-SLAM	PSNR ↑	35.17	32.40	34.08	35.50	38.26	39.16	33.99	34.48	33.49
	SSIM ↑	0.98	0.97	0.98	0.98	0.98	0.98	0.96	0.96	0.96
	LPIPS ↓	0.12	0.11	0.12	0.10	0.10	0.12	0.12	0.10	0.14
	Depth L1 [cm] ↓	0.44	-	-	-	-	-	-	-	-
MonoGS	PSNR ↑	37.50	34.83	36.43	37.49	39.95	42.09	**36.24**	**36.70**	**36.07**
	SSIM ↑	0.96	0.95	0.96	0.97	0.97	0.98	0.96	0.96	0.96
	LPIPS ↓	0.07	0.07	0.08	0.08	0.07	0.06	0.08	0.07	0.10
	Depth L1 [cm] ↓	36.59	-	-	-	-	-	-	-	-
GS-SLAM	PSNR ↑	34.27	31.56	32.86	32.59	38.70	41.17	32.36	32.03	32.92
	SSIM ↑	0.98	0.97	0.97	0.97	0.99	0.99	0.98	0.97	0.97
	LPIPS ↓	0.08	0.09	0.08	0.09	0.05	0.03	0.09	0.11	0.11
	Depth L1 [cm] ↓	-	-	-	-	-	-	-	-	-
Photo-SLAM	PSNR ↑	34.96	30.72	33.51	35.03	38.48	39.09	33.03	33.79	36.02
	SSIM ↑	0.94	0.90	0.93	0.95	0.96	0.96	0.94	0.94	0.95
	LPIPS ↓	0.06	0.08	0.06	0.04	0.05	0.05	0.08	0.07	0.05
	Depth L1 [cm] ↓	19.73	-	-	-	-	-	-	-	-
SplaTAM	PSNR ↑	34.11	32.86	33.89	35.25	38.26	39.17	31.97	29.70	31.81
	SSIM ↑	0.97	0.98	0.97	0.97	0.98	0.97	0.97	0.95	0.95
	LPIPS ↓	0.10	0.10	0.08	0.09	0.09	0.10	0.12	0.10	0.15
	Depth L1 [cm] ↓	0.72	-	-	-	-	-	-	-	-
Loopy-SLAM	PSNR ↑	29.32	31.80	32.70	32.70	38.66	15.96	15.00	33.61	34.15
	SSIM ↑	0.88	0.91	0.91	0.91	0.96	0.88	0.58	0.92	0.93
	LPIPS ↓	0.25	0.16	0.19	0.21	0.13	0.37	0.54	0.20	0.19
	Depth L1 [cm] ↓	**0.35**	**0.30**	**0.20**	0.42	0.23	0.46	0.60	**0.37**	**0.24**
Ours	PSNR ↑	**44.77**	**34.95**	**37.26**	**37.81**	**41.78**	**42.82**	34.45	33.69	35.33
	SSIM ↑	**0.98**	**0.98**	**0.99**	**0.99**	**0.99**	**0.99**	**0.98**	**0.98**	**0.97**
	LPIPS ↓	**0.05**	**0.04**	**0.04**	**0.04**	**0.03**	**0.02**	**0.05**	**0.06**	**0.09**
	Depth L1 [cm] ↓	0.43	0.38	0.23	**0.33**	**0.19**	**0.15**	**0.39**	0.95	0.83

TABLE IV: Mapping Results on Replica.

Methods	Avg.	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/office
ORB-SLAM3	-	**1.53**	-	-	**0.72**	**1.40**
DROID-SLAM	-	78.25	-	-	36.05	154.38
NICE-SLAM	15.87	4.26	4.99	34.49	31.73	3.87
Point-SLAM	8.92	4.34	**4.54**	30.92	1.31	3.48
MonoGS	-	1.84	-	-	1.71	1.74
Gaussian-SLAM	-	2.74	-	-	**0.96**	8.42
GS-ICP-SLAM [28]	-	3.26	-	-	2.26	3.07
SplaTAM	**5.48**	3.35	6.54	11.13	1.24	5.16
Ours	6.02	3.03	6.39	12.82	1.39	6.47

TABLE V: Tracking Results on TUM-RGBD. (ATE RMSE[cm] ↓). The best results are shown in bold, and the second-best results are marked with underlines.

ity of the RGB-D images, our method, like other learning-based approaches, did not outperform the feature-matching-based ORB-SLAM3. Nevertheless, it achieved performance comparable to other learning-based methods. Moreover, the other models underwent hyperparameter tuning to achieve higher performance, which caused them to overfit to the small scale. In contrast, our model achieves these results without the need for hyperparameter tuning. This confirms that our work achieves robust performance regardless of scale.

C. Ablation Study

We conducted an ablation study to evaluate the effectiveness of gaussian fission and Polyak step size. Gaussian fission is a methodology designed to address the limitations of densification in large-scale environments. Therefore, the ablation study was performed on the large-scale KITTI dataset. Fig. 7 presents a case where an insufficient number of 3D Gaussians are allocated due to the object being located far away. Our objective is to densify these sparsely allocated elements. By applying gaussian fission, which divides larger 3D Gaussians with excessive radius, a sufficient number is guaranteed. It leads to a detailed and refined representation through optimization during the mapping stage. Table VI shows that gaussian fission improves mapping performance across all scenes. Table VII shows a slight improvement in tracking performance when gaussian fission is applied. Although gaussian fission does not directly influence tracking, it can be inferred that the improved mapping quality indirectly enhanced tracking performance.

Tables VI and VII present the effects of the Polyak step size. In the tracking ablation study, the Polyak step size was removed from tracking to evaluate its impact, whereas in the

(a) Given Image

(b) Zoomed-in Image (c) w/o Gaussian Fission (d) Gaussian Fission

Fig. 7: **Ablation Study on Gaussian Fission (Prior to Optimization).** (a) represents the input image intended for densification. (b) shows a zoomed-in view of the red box area, highlighting objects located far away. (c) illustrates the 3D Gaussians without Gaussian Fission, where densification is insufficient. (d) demonstrates the result after applying Gaussian Fission, where a sufficient number of 3D Gaussians are allocated in preparation for optimization.

Methods	Avg.	scene1	scene2	scene11	scene13	scene48
w/o GF, Polyak: T	28.77	28.64	28.67	28.27	26.84	31.41
w/o GF, Polyak: T+M	29.53	30.02	29.67	29.10	27.50	31.37
Ours(GF, Polyak: M+T)	**31.274**	**31.62**	**31.70**	**31.41**	**28.85**	**32.79**

TABLE VI: Ablation Study on Mapping Performance. (KITTI, PSNR ↑).

Methods	Avg.	scene1	scene2	scene11	scene13	scene48
w/o GF, Polyak: M	949.842	897.32	1054.93	1483.11	991.46	322.39
w/o GF, Polyak: T+M	64.55	76.68	**40.00**	56.05	117.23	34.81
Ours(GF, Polyak: M+T)	**61.71**	**69.31**	40.85	**48.75**	**116.55**	**33.07**

TABLE VII: Ablation Study on Tracking Performance. (KITTI, ATE RMSE[cm] ↓).

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Methods	Tracking /Iteration[ms]	Mapping /Iteration[ms]	Tracking /Frame[s]	Mapping /Frame[s]	ATE [cm]	PSNR	Memory [MB]
SplaTAM	14.59	14.55	0.65	0.72	1244.06	22.22	3435.59
w/o GF	13.28	14.15	0.59	0.70	80.30	28.07	1647.30
Ours(λ=0.35)	15.01	17.33	0.67	0.81	77.56	29.66	1650.13
Ours(λ=0.1)	15.78	17.75	0.71	0.86	75.96	30.17	1796.19
Ours(λ=0.05)	16.01	18.46	0.72	0.92	72.41	30.16	2108.24

TABLE VIII: Comparison of runtime and memory usage on KITTI/Scene1 using an RTX 3090

mapping ablation study, it was removed from mapping. The Polyak step size leads to a significant improvement in tracking by successfully adapting to large-scale environments, and also enabled more accurate optimization in mapping.

D. Runtime and Memory Analysis

The model employing only the Polyak step size outperformed SplaTAM not only in terms of memory efficiency but also by achieving a reduction in runtime. This is attributed to the insufficient tracking performance of SplaTAM, which caused the 3D Gaussians to be scattered more sporadically, leading to a significant increase in memory consumption. However, enabling gaussian fission inevitably incurs additional computational costs. More frequent gaussian fission resulted in increased runtime and greater memory consumption. Nevertheless, we observed that both tracking and mapping performance improved as fission became more active. Thus, this method exhibits a clear trade-off between runtime and performance, and it would be beneficial to appropriately tune the radius threshold of the 3D Gaussians depending on the specific application requirements.

VI. Conclusions

To the best of our knowledge, we present SAGA-SLAM, the first scale-adaptive dense SLAM based on 3DGS. We address the challenges in the tracking and mapping stages by utilizing the Polyak step size and momentum. Furthermore, we propose gaussian fission, which improves the representation of objects located at distant points. We demonstrate its robustness across various scales and conditions, achieving state-of-the-art results in localization and scene reconstruction. SAGA-SLAM distinguishes itself from previous approaches by eliminating the need for hyperparameter tuning, a crucial factor in the practical applications of SLAM.

References

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering.," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[2] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "Splatam: Splat track & map 3d gaussians for dense rgb-d slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21357–21366, 2024.

[3] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, "Gaussian splatting slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18039–18048, 2024.

[4] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li, "Gs-slam: Dense visual slam with 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19595–19604, 2024.

[5] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, "Gaussian-slam: Photo-realistic dense slam with gaussian splatting," *arXiv preprint arXiv:2312.10070*, 2023.

[6] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 573–580, IEEE, 2012.

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[10] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] F. Schaipp, R. Ohana, M. Eickenberg, A. Defazio, and R. M. Gower, "Momo: Momentum models for adaptive learning rates," *arXiv preprint arXiv:2305.07583*, 2023.

[12] B. T. Polyak, "Introduction to optimization. translations series in mathematics and engineering. optimization software, inc., publications division, new york, 1987," *Translated from the Russian, With a foreword by Dimitri P. Bertsekas*.

[13] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6229–6238, 2021.

[14] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12786–12796, 2022.

[15] M. M. Johari, C. Carta, and F. Fleuret, "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17408–17419, 2023.

[16] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-slam: Dense neural point cloud-based slam," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18433–18444, 2023.

[17] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[18] H. Huang, L. Li, H. Cheng, and S.-K. 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*, pp. 21584–21593, 2024.

[19] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE transactions on robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[20] W. Zhang, Q. Cheng, D. Skuddis, N. Zeller, D. Cremers, and N. Haala, "Hi-slam2: Geometry-aware gaussian slam for fast monocular scene reconstruction," *arXiv preprint arXiv:2411.17982*, 2024.

[21] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16558–16569, 2021.

[22] D. Feng, Z. Chen, Y. Yin, S. Zhong, Y. Qi, and H. Chen, "Cartgs: Computational alignment for real-time gaussian splatting slam," *arXiv preprint arXiv:2410.00486*, 2024.

[23] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/k2)," in *Dokl. Akad. Nauk. SSSR*, vol. 269, p. 543, 1983.

[24] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, p. 2, 2012.

[25] N. Loizou, S. Vaswani, I. H. Laradji, and S. Lacoste-Julien, "Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence," in *International Conference on Artificial Intelligence and Statistics*, pp. 1306–1314, PMLR, 2021.

[26] H. Asi and J. C. Duchi, "Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity," *SIAM Journal on Optimization*, vol. 29, no. 3, pp. 2257–2290, 2019.

[27] P. Zhu, Y. Zhuang, B. Chen, L. Li, C. Wu, and Z. Liu, "Mgs-slam: Monocular sparse tracking and gaussian mapping with depth smooth regularization," *IEEE Robotics and Automation Letters*, 2024.

[28] S. Ha, J. Yeon, and H. Yu, "Rgbd gs-icp slam," in *European Conference on Computer Vision*, pp. 180–197, Springer, 2024.