RGBD GS-ICP SLAM

Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu

Sungkyunkwan University, Suwon, South Korea
{sobo3607,wcr12st,hwyu}@skku.edu

Abstract. Simultaneous Localization and Mapping (SLAM) with dense representation plays a key role in robotics, Virtual Reality (VR), and Augmented Reality (AR) applications. Recent advancements in dense representation SLAM have highlighted the potential of leveraging neural scene representation and 3D Gaussian representation for high-fidelity spatial representation. In this paper, we propose a novel dense representation SLAM approach with a fusion of Generalized Iterative Closest Point (G-ICP) and 3D Gaussian Splatting (3DGS). In contrast to existing methods, we utilize a single Gaussian map for both tracking and mapping, resulting in mutual benefits. Through the exchange of covariances between tracking and mapping processes with scale alignment techniques, we minimize redundant computations and achieve an efficient system. Additionally, we enhance tracking accuracy and mapping quality through our keyframe selection methods. Experimental results demonstrate the effectiveness of our approach, showing incredibly fast speeds up to 107 FPS (for the entire system) and superior quality of the reconstructed map.
The code is available at: https://github.com/Lab-of-AI-and-Robotics/GS-ICP-SLAM
Video is: https://youtu.be/ebHh_uMMxE

Keywords: Coordinate-based 3D Representation · G-ICP · SLAM

1 Introduction

Visual Simultaneous Localization and Mapping (SLAM) is an algorithm that constructs maps of unknown environments while localizing poses of vision sensors simultaneously. As the influence of 3D visual SLAM in the fields of robotics, Virtual Reality (VR), and Augmented Reality (AR) has increased, higher rendering performance and more accurate trajectory are required. Thus, 3D reconstruction methods such as Signed Distance Field (SDF), and Truncated Signed Distance Field (TSDF) are utilized for traditional dense visual SLAM [3, 5, 10, 27, 30, 42].

Recently, coordinate-based 3D Implicit Neural Representation (INR) [23] has been proposed for representing spatial information using neural radiance fields, showcasing high novel view synthesis capabilities and high-fidelity spatial representation prowess. Various approaches [8, 18, 32, 39, 45, 47] have been attempted to utilize INR for the real-time SLAM mapping process. However, INR requires computationally intensive raycasting to synthesize images, thus the rendering process in INR-based SLAM incurs significant time overhead, slowing down map optimization and rendering-loss based tracking. In contrast, coordinate-based 3D explicit representation such as 3D Gaussian

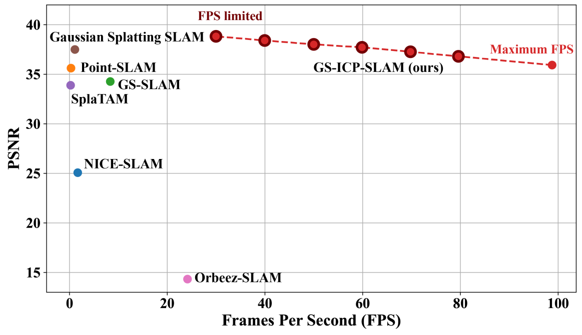

Fig. 1: A comparison of PSNR with respect to FPS of entire system in recent research on SLAM algorithm utilizing dense representation such as neural scene representation and 3D Gaussian representation. Our method achieves state-of-the-art performance in rendering evaluation and FPS of entire system. Note that this FPS represents the overall system performance. Reported values are average of Replica 8 scenes.

Splatting (3DGS) [15] represents the 3D space using 3D Gaussians as primitives, allowing for rendering speeds faster than NeRF using the rasterization method [43]. This rapid rendering capability of 3DGS results in fast optimization of 3D spatial information, making it suitable for dense representation in SLAM [12, 14, 22, 44].

Although 3DGS-based SLAM methods take advantages of the high-speed rendering, they fail to address the fundamental issue: the inability to directly utilize 3D explicit representations and the indirect tracking of 3D space through 2D image rendering. Even with the majority of current 3DGS-based SLAM [12, 14, 22, 32, 44] utilizing RGB-D data, the use of explicit representations is overlooked. For example, INR-based methods [32, 39, 47] and 3DGS-based methods [14, 22, 44] employ photometric-error based techniques, which estimate the optimal pose by iteratively minimizing the error between rendered and observed 2D images. However, due to limitations in tracking speed and performance, decoupled approaches [8, 12, 18] have been proposed incorporating well-crafted visual odometries [4,26] into the tracking process. Although decoupled methods exhibit better performances in tracking by separating mapping and tracking processes, they require additional computational resources for the independent stages. Moreover, these methods need to store features in 3D space, unrelated to the 3DGS map.

There exists a method that allows Gaussian, an explicit representation, to be directly used for tracking. The well-known Generalized Iterative Closest Point (G-ICP) [17, 33] from the 3D scan matching family is simple yet efficient for fast tracking of 3D point clouds. During preprocessing, it only requires computing Gaussians for the current frame and the map. Given that the map in 3DGS utilizes Gaussians as an explicit representation in 3D space, using G-ICP for tracking allows for the direct utilization of the 3DGS map without the need for post-processing. Furthermore, the Gaussians of the current frame computed during tracking with G-ICP can also be directly utilized as an explicit representation in the 3DGS map, without additional computations.

Therefore, we propose a dense representation SLAM framework that integrates G-ICP and 3DGS, allowing them to complement each other by sharing explicit representations. Unlike traditional methods, our approach actively leverages 3D information through G-ICP for tracking. The proposed method is a coupled approach that shares a single map during tracking and mapping processes, while maintaining fast tracking speed akin to decoupled approaches. Previous works using decoupled methods [8, 12, 18] require separate maps containing ORB feature information for tracking, along with additional resources and computations needed to obtain ORB features. Moreover, the values obtained during tracking are not utilized in mapping. Our approach exploits the covariance of each point computed during the G-ICP-based tracking process as the initial state of the 3DGS mapping. Meanwhile, the 3D Gaussians in the 3DGS map are also mutually used as 3D points and their covariances which are essential for G-ICP-based tracking, discarding the necessity to recalculate the covariances of the map. This is possible because the covariances of points computed during the G-ICP process and the 3D Gaussians representing the map commonly contain information about the surrounding space. In other words, G-ICP and 3DGS can share the same Gaussian world. Therefore, our system minimizes unnecessary computations and facilitates efficient system configuration by mutually utilizing the key elements, Gaussians, between tracking and mapping processes. To ensure optimal performance in tracking and mapping sharing information between G-ICP and 3DGS, we also introduce several techniques such as scale alignment. To summarize, our contributions are as follows:

- We present a real-time dense representation SLAM that combines G-ICP and 3DGS, achieving extremely high speed of the entire system (up to 107 FPS) and superior quality of the map.
- By incorporating G-ICP for tracking, our system utilizes 3D information actively and significantly reduces the time required for the tracking process.
- Reduction of computational cost and facilitating rapid convergence of primitives of 3DGS is achieved by sharing the covariances of G-ICP and 3DGS with scale aligning techniques.

2 Related Work

G-ICP Scan-matching focuses on the registration of two point clouds that observe similar environments by selecting a transformation matrix that minimizes errors between two point clouds [2]. Iterative Closest Point (ICP) [1] algorithm is a widely used and influential method in the scan-matching area. ICP iteratively estimates point correspondences and finds a transformation that minimizes Euclidean distance between corresponding points, thereby optimizing the registration of point clouds. Because of simplicity and speed of ICP, there are many follow-up studies [7,20,33,34,41]. To improve robustness, Trimmed-ICP [7], proposed a method for correspondence selection. Point-to-plane ICP [20] augmented robustness and accuracy by taking point-to-plane distance as objective function. G-ICP [33] introduced probabilistic models to generalize ICP yielding notable improvements in robustness and accuracy. Scan matching is a common technique used in SLAM for pose tracking [6,11,21,28,29], and for enhanced robustness, feature-based scan matching is often employed [19,35,36,46].

SLAM with Dense Representation SLAM with dense representation [9, 16] aims to construct maps in a dense form to enable interaction with the map in tasks such as AR, robotics, etc. To achieve this, classic approaches [3,5,10,27,30,42] creates dense maps by representing space in Signed Distance Field (SDF), and Truncated Signed Distance Field (TSDF), rather than in sparse forms such as point clouds or grids.

Recently, dense representation SLAM methods represent maps by utilizing NeRF [23], which demonstrates high spatial representation capabilities have been proposed. iMAP [39], NICE-SLAM [47], Point-SLAM [32], ESLAM [13] perform tracking by optimizing camera poses by reducing errors between synthesized and observed images. These methods have limitations in tracking speed and performance. Alternatively, Orbeez-SLAM [8], vMAP [18] incorporate well-crafted visual SLAM techniques into the tracking process, making mapping and tracking operate independently. Orbeez-SLAM utilizes ORB-SLAM2 [26] for tracking and instant-ngp [24] for learning spatial information to perform mapping. In a similar context, vMAP utilizes ORB-SLAM3 [4] for tracking while employing separate Multi-layer perceptrons [40] for each object to effectively represent the entire space.

3DGS represents space in explicit form using 3D gaussians as primitives. It offers a level of high-fidelity spatial representation similar to NeRF but provides significantly faster rendering speeds. To leverage this advantage to dense representation SLAM, several approaches [12, 14, 22, 44] utilizing 3DGS for space representation have been proposed. Among them, GS-SLAM [44], SplaTAM [14], Gaussian Splatting SLAM [22] perform tracking using dense photometric error. While 3DGS shows significantly improved rendering performance compared to NeRF, the tracking methods based on dense photometric error still suffer from tracking speed. To tackle this problem, Photo-SLAM [12] integrates ORB-SLAM3 for tracking, in a similar manner to Orbeez-SLAM and vMAP.

3 Method

To mutually benefit tracking and mapping, we introduce the fusion of G-ICP and GS, each representing the tracking and mapping processes, respectively. The key insight of our approach is that covariance can be considered as a fundamental common factor for this fusion. Suppose we have a point set (point cloud) $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_m\}_{m=1,\dots,M}$ and its corresponding covariance set $\boldsymbol{\mathcal{C}} = \{C_m\}_{m=1,\dots,M}$, where $\boldsymbol{x} = [x, y, z]^T$. The covariance C of one 3D point $\boldsymbol{x}$ is given by computing covariance matrix of k-nearest neighbors of $\boldsymbol{x}$. Let us define $\boldsymbol{G} = \{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{C}}\}$ as a set of Gaussians. G-ICP aims to find a transformation $\mathbf{T}$ that maximally aligns the source Gaussians (current frame) $\boldsymbol{G}^s = \{\boldsymbol{\mathcal{X}}^s, \boldsymbol{\mathcal{C}}^s\}$ and the target Gaussians (map) $\boldsymbol{G}^t = \{\boldsymbol{\mathcal{X}}^t, \boldsymbol{\mathcal{C}}^t\}$. Assume we know the correspondences between $\boldsymbol{\mathcal{X}}^s$ and $\boldsymbol{\mathcal{X}}^t$ determined by nearest neighbor search. For example, we have $\{\boldsymbol{x}_i^s\}_{i=1,\dots,N} \subset \boldsymbol{\mathcal{X}}^s$, $\{C_i^s\}_{i=1,\dots,N} \subset \boldsymbol{\mathcal{C}}^s$ and $\{\boldsymbol{x}_i^t\}_{i=1,\dots,N} \subset \boldsymbol{\mathcal{X}}^t$, $\{C_i^t\}_{i=1,\dots,N} \subset \boldsymbol{\mathcal{C}}^t$, where $\boldsymbol{x}_i^s$ is associated with $\boldsymbol{x}_i^t$. To find the optimal transform $\mathbf{T}^*$, we exploit not the single point but the distribution of that point defined as a Gaussian distribution: $\boldsymbol{x}_i \sim \mathcal{N}(\hat{\boldsymbol{x}}_i, C_i)$. Let $d_i = \boldsymbol{x}_i^t - \mathbf{T}\boldsymbol{x}_i^s$ be the error term, and if we assume that there is an optimal transformation $\mathbf{T}^*$, it is clear that $\boldsymbol{x}_i^t = \mathbf{T}^*\boldsymbol{x}_i^s$ thus $\hat{d}_i = \mathbf{0}$. Since we assume $\boldsymbol{x}$ is a Gaussian random variable, d_i is also a Gaussian random variable as the following:

$$\begin{aligned} d_i &\sim \mathcal{N}(\hat{d}_i, C_i^t + \mathbf{T}^* C_i^s (\mathbf{T}^*)^T) \\ &= \mathcal{N}(\hat{x}_i^t - \mathbf{T}^* \hat{x}_i^s, C_i^t + \mathbf{T}^* C_i^s (\mathbf{T}^*)^T) \\ &= \mathcal{N}(0, C_i^t + \mathbf{T}^* C_i^s (\mathbf{T}^*)^T). \end{aligned}$$

To find the optimal transform $\mathbf{T}^*$ for $\boldsymbol{\mathcal{X}}^s$ and $\boldsymbol{\mathcal{X}}^t$, we use maximum likelihood estimation (MLE) as the following:

$$\begin{aligned} \mathbf{T}^* &= \underset{\mathbf{T}}{\operatorname{argmax}} \prod_i^N p\left(d_i\right) = \underset{\mathbf{T}}{\operatorname{argmax}} \sum_i^N \log p(d_i) \\ &= \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i^N d_i^T \left(C_i^B + \mathbf{T} C_i^A \mathbf{T}^T\right)^{-1} d_i. \end{aligned} \tag{1}$$

Therefore, $\mathbf{T}^*$ can be used as the relative pose between the current frame $\boldsymbol{G}^s = \{\boldsymbol{\mathcal{X}}^s, \boldsymbol{\mathcal{C}}^s\}$ and the map $\boldsymbol{G}^t = \{\boldsymbol{\mathcal{X}}^t, \boldsymbol{\mathcal{C}}^t\}$.

Meanwhile, for mapping purposes, GS also relies on Gaussians $\boldsymbol{G} = \{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{C}}\}$ of the 3D scene representation. Different from G-ICP, GS aims to find the optimal coordinates of the Gaussians $\boldsymbol{\mathcal{X}}^* = \{\boldsymbol{x}_m^*\}_{m=1,\ldots,M}$ and the optimal covariances $\boldsymbol{\mathcal{C}}^* = \{C_m^*\}_{m=1,\ldots,M}$ as the following:

$$\boldsymbol{\mathcal{X}}^*, \boldsymbol{\mathcal{C}}^*, \boldsymbol{H}^*, \boldsymbol{O}^* = \underset{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{C}}, \boldsymbol{H}, \boldsymbol{O}}{\operatorname{argmin}} \lambda_{I_1} \boldsymbol{\mathcal{L}}_1\left(I, I_{gt}\right) + \lambda_{I_2} \boldsymbol{\mathcal{L}}_{D-SSIM}\left(I, I_{gt}\right) + \lambda_D \boldsymbol{\mathcal{L}}_1\left(D, D_{gt}\right)$$

where $\boldsymbol{H} = \{\boldsymbol{h}_m\}_{m=1,\ldots,M}$ and $\boldsymbol{O} = \{\boldsymbol{o}_m\}_{m=1,\ldots,M}$ are the color set and the opacity set of 3D points which are for the RGBD image rendering. I and D are the rendered RGB and depth images obtained by performing rasterization using $\boldsymbol{G}$, $\boldsymbol{H}$ and $\boldsymbol{O}$.

Note that in G-ICP and GS, the key common factor is Gaussians $\boldsymbol{G} = \{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{C}}\}$, allowing these Gaussians to be shared mutually. During G-ICP tracking, the covariance of each frame is computed. Hence, when adding keyframes to expand the 3DGS map, there is no need to recalculate $\boldsymbol{\mathcal{C}}$ for every expansion. Also, G-ICP does not need to compute the covariances of the map because our GS map already contains Gaussians. Moreover, G-ICP, by aligning frames based on 3D geometric structure, inherently initializes a certain number of points and their coordinates that aptly represent the 3D structure. Therefore, it brings about an effect where an appropriate number of points and their poses are initialized, well-suited for depicting the 3D structure, significantly reducing the learning time to find the optimal pose $\boldsymbol{\mathcal{X}}^*$ and the optimal covariance $\boldsymbol{\mathcal{C}}^*$ of Gaussians $\boldsymbol{G}$ in GS. Additionally, in the same vein, calculations like densifying or opacity reset to adjust the number of 3D points in GS become unnecessary. Consequently, by sharing a common source, each process becomes mutually beneficial, and the speed of execution accelerates due to reduced redundant computations. Our method can be simplified as follows:

1. Use G-ICP to align the current frame with the 3DGS map which contains covariance (solely need to compute the covariance for the current frame).

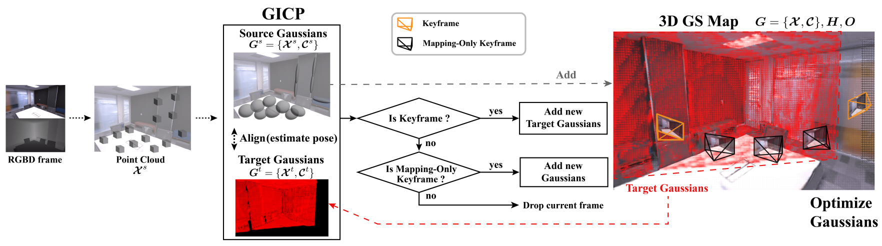
GICP
Source Gaussians $G^s = \{\mathcal{X}^s, \mathcal{C}^s\}$
Align(estimate pose)
Target Gaussians $G^t = \{\mathcal{X}^t, \mathcal{C}^t\}$
RGBD frame
Point Cloud $\mathcal{X}^s$
Keyframe
Mapping-Only Keyframe
Add
Is Keyframe ?
yes
no
Add new Target Gaussians
Is Mapping-Only Keyframe ?
yes
no
Add new Gaussians
Drop current frame
3D GS Map $G = \{\mathcal{X}, \mathcal{C}\}, H, O$
Target Gaussians
Optimize Gaussians

Fig. 2: System Overview. The input of our system is RGBD frame. We generate a point cloud by downsampling and reprojecting the current depth image and utilize it in the GICP process. During the GICP process, we create source Gaussians from the point cloud and estimate the current camera pose by aligning them with target Gaussians, which are a subset of the 3DGS map. If the current frame is identified as a keyframe or a mapping-only keyframe, we add the source Gaussians to the 3DGS map as new primitives. Meanwhile, in the mapping process, we optimize the Gaussians along with the color and opacity set of the Gaussians concurrently with the tracking process.

2. When adding keyframes to the 3DGS map, utilize the covariance computed in GICP during tracking (no need for densifying or opacity reset).
3. Repeat steps 1-2.

The overview of our system is shown in Fig. 2. The detailed implementations of the proposed technique are introduced in the following Section.

3.1 G-ICP Tracking

Scale Regularization When using G-ICP for tracking, aligning the scale of the current frame with the map enables high-performance camera pose estimation [33]. Given the covariance C, scale $\boldsymbol{S} = [s_2, s_1, s_0]^T$ is given by the singular value decomposition (SVD) as the following:

$$C = \boldsymbol{R}\boldsymbol{\Lambda}^2\boldsymbol{R}^T \tag{2}$$

where $\boldsymbol{\Lambda} = diag\,(s_2, s_1, s_0)\,\forall s_2 > s_1 > s_0$ is a scale matrix and $\boldsymbol{R}$ is the orientation of the Gaussian. To achieve the robust scan matching performance, point-to-plane ICP or voxelized point-to-plane ICP adopt a regularizing method that makes the scale as $\boldsymbol{S} = [1, 1, \epsilon]^T$ in order to treat each Gaussian as a plane-like distribution.

In our framework, we utilize target Gaussians $\boldsymbol{G}^t = \{\boldsymbol{\mathcal{X}}^t, \boldsymbol{\mathcal{C}}^t\}$ from the 3DGS map for tracking. Here, $\boldsymbol{\mathcal{C}}^t$ is optimized to represent the scene accurately, $\boldsymbol{G}^t$ involves not only planes but also lines, corners, and other features. Therefore, while tracking, instead of regularizing the scale into a plane-based form, preserving the original characteristics of the target Gaussians while performing regularization is more appropriate. Thus, in our framework, we propose the following ellipsoid regularization:

$$\boldsymbol{\Lambda}' = \frac{1}{median\,(\boldsymbol{S})} diag\,(s_2, s_1, s_0) \tag{3}$$

where $\boldsymbol{\Lambda}'$ is the ellipsoid-regularized scale matrix.

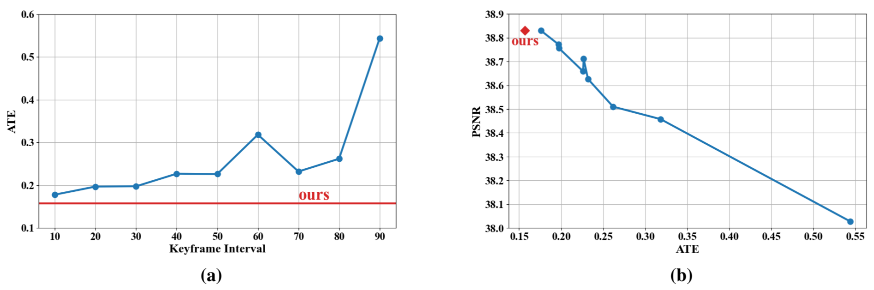

Fig. 3: Tracking Accuracy Comparison Based on Keyframe Selection Methods. The reported values represent the average results across eight scenes from the Replica dataset [37]. When selecting keyframes every n frame (depicted in blue), the tracking accuracy is notably low. Conversely, our keyframe selection method yielded the highest tracking accuracy.

Keyframe Selection Similar to [25,31], we perform dynamic keyframe selection. Considering the geometric structure is critical for both G-ICP tracking and GS mapping, we exploit the geometric correspondence computed from G-ICP. As Eqn. (1) denotes, we can get the distance between $\boldsymbol{x}_i^s$ and $\boldsymbol{x}_i^t$ as an interim result of G-ICP procedure without additional calculations. So we can efficiently determine the correspondence by setting the distance threshold. By considering the proportion of correspondences between the current frame and map, we select keyframes while taking into account the characteristics of the scene. In more detail, if the proportion in the current frame goes below specific thresholds, the frame is selected as a keyframe. This allows us to achieve consistent tracking performance while simultaneously maintaining a consistent density of Gaussians $\boldsymbol{G} = \{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{C}}\}$ added to the map. The effect of this keyframe selection is shown in Fig. 3 (a). Note that, from selected keyframe, only Gaussians that do not overlap with the existing current map are considered as target Gaussians. Errors from discrepancies during tracking are directly incorporated into the map, leading to accumulated inaccuracies and reduced tracking precision. This strategy ensures more accurate tracking results and improves the overall robustness of the system.

3.2 GS Mapping

Scale Aligning Suppose we have a fully trained 3DGS map. Since the 3DGS map is trained to represent the scene from any viewpoint, Gaussians $\boldsymbol{G}$ should be uniformly distributed according to the structure of the 3D space. However, single frames obtained from radiance sensors such as RGBD cameras or LiDAR yield sparse representations of the 3D space as the distance from the sensor increases, due to the inherent characteristics of the sensor. In other words, as the distance from the sensor increases, the spacing between 3D points widens, and the scale of the corresponding covariance calculated based on the k-nearest neighbors becomes excessively large. Adding such single frames as keyframes to the map during real-time training of GS mapping leads to an imbalance in the scale of Gaussians, ultimately resulting in a decrease in mapping performance . Therefore, to fundamentally alleviate the problem, our proposed method suggests the

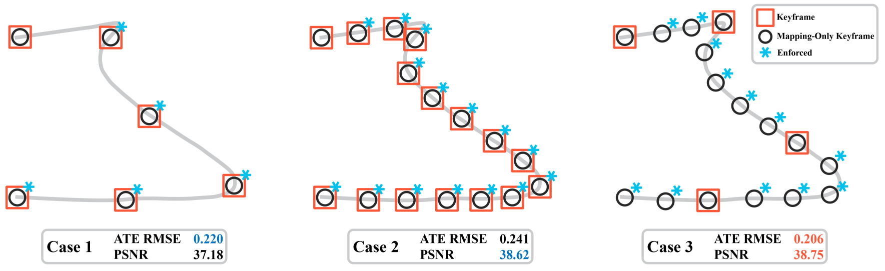
Keyframe
Mapping-Only Keyframe
Enforced
Case 1 ATE RMSE 0.220 PSNR 37.18
Case 2 ATE RMSE 0.241 PSNR 38.62
Case 3 ATE RMSE 0.206 PSNR 38.75

Fig. 4: Separated Keyframe Selection on Replica office4. We demonstrate that a small number of tracking keyframes yield accurate trajectory estimation (case 1), while a large number of mapping keyframes result in high rendering performance (case 2). Thus, our method adopts case 3 that select tracking keyframe and mapping keyframe separately at different intervals.

following scale normalization for the current frame:

$$\boldsymbol{\Lambda}'' = \frac{1}{z^p}\boldsymbol{\Lambda}.$$

Where $\boldsymbol{\Lambda}''$ is the normalized scale matrix and p is the parameter which is empirically determined. By adding such scale-normalized keyframes to the map to expand it, not only does the GS mapping performance increase, but also the performance of G-ICP tracking improves.

Additional Keyframe Selection for Mapping Mapping based on GS benefits from having a diverse set of frames from various viewpoints for training, as performance improves with a larger number of available frames. However, increasing the number of selected keyframes during tracking can lead to a degradation in tracking performance due to accumulated errors, as is common with G-ICP-based methods. To mitigate this issue, we propose adding mapping-only keyframe selection in addition to the existing keyframe selection process. In other words, tracking continues to proceed from selected keyframes in the existing tracking process, while mapping utilizes both the original keyframes and additional mapping-only keyframes.

The overview of the additional keyframe selection is illustrated in Fig. 4. Cases 1 and 2 represent scenarios without considering mapping-only keyframes. In Case 1, to increase the number of keyframes, if no keyframe satisfying the criteria is found up to the 30^{th} frame from the previous keyframe, the 30^{th} frame is selected as a keyframe. However, since this may not yield sufficient mapping performance, as shown in Case 2, adding an additional keyframe at the 10^{th} frame can be considered. While this approach improves mapping performance as desired, it leads to a decline in tracking performance due to accumulated scan matching errors. To alleviate this, as shown in Case 3, we propose adding the 10^{th} frame as a mapping-only frame, allowing us to maximize tracking performance while simultaneously maximizing mapping performance. The effect of the proposed scheme, Case 3, is shown in Fig. 3 (b). Note that since G-ICP, which performs tracking, computes covariance for all frames anyway, there is no difference in the overall computational load.

Avoiding Local Minima while Mapping GS SLAM differs from traditional GS in that it operates in real-time and often lacks sufficient observations of the scene. Moreover, since it operates in real-time, focusing solely on images from the current viewpoint or nearby viewpoints during training can easily lead to local minima and degrade mapping quality [22]. For instance, GS simply considers image rendering from specific viewpoints, thus if we continuously train only on a specific viewpoint, the scale of Gaussians in the map tends to elongate in the direction of that viewpoint, leading to local minima.

To overcome this challenge, at each training iteration, we adopt a strategy of randomly choosing one keyframe for learning among the selected keyframes so far, ensuring that both the current observed scene and the entire map are uniformly learned. Additionally, we prune Gaussians that fall into local minima during training to improve mapping performance and ensure robust tracking through geometric preservation.

4 Experiments

4.1 Experimental Setup

We evaluate our method on Replica dataset [37] and TUM dataset [38]. Replica dataset contains synthetic scenes and high-quality RGB/depth images rendered from these scenes. TUM dataset includes images captured in the real world, with significant noise and blur, resulting in poor quality. Particularly, many parts of the depth images suffer from information loss. We validate the effectiveness of our approach by evaluating it on both synthetic and real-world datasets. All experiments are performed on a desktop with a Ryzen 7 7800x3d CPU, 32GB RAM, and an NVIDIA RTX 4090 24GB GPU. To evaluate camera tracking accuracy, we use the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE). For the quality of the reconstructed map, we report standard photometric rendering quality metrics (PSNR, SSIM, and LPIPS).

Table 1: Tracking Performance on Replica and TUM-RGBD (ATE RMSE $\downarrow$ [cm]). Our method archives state-of-the-art tracking accuracy on Replica. On TUM, Among coupled methods, we achieve competitive performance. The results of GS-SLAM/Gaussian Splatting SLAM are taken from [44]/ [22]. And the results of ORB-SLAM3 and Photo-SLAM are obtained from [12].

Method		Replica									TUM			
		R0	R1	R2	Of0	Of1	Of2	Of3	Of4	Avg.	fr1/desk	fr2/xyz	fr3/office	Avg.
Decoupled	ORB-SLAM3 [4]	-	-	-	-	-	-	-	-	1.8	**1.7**	0.4	1.7	**1.3**
	Photo-SLAM [12]	-	-	-	-	-	-	-	-	**0.60**	2.6	**0.3**	**1.0**	**1.3**
Coupled	NICE-SLAM* [47]	1.61	1.48	1.61	0.95	0.81	1.46	1.76	1.69	1.42	2.8	2.1	7.2	4.0
	Point-SLAM* [32]	0.59	0.51	0.32	0.45	0.46	0.48	0.61	0.87	0.54	2.7	**1.3**	3.9	2.6
	GS-SLAM [44]	0.48	0.53	0.33	0.52	0.41	0.59	0.46	0.70	0.50	3.3	**1.3**	6.6	3.7
	SplaTAM* [14]	0.29	0.35	0.28	0.49	0.21	0.31	0.34	0.57	0.36	3.3	**1.3**	5.1	3.2
	Gaussian Splatting SLAM* [22]	0.33	0.22	0.29	0.36	0.19	0.25	**0.12**	0.81	0.32	**1.5**	1.4	**1.5**	**1.5**
	Ours (limited to 30 FPS)	**0.15**	**0.16**	**0.11**	**0.18**	**0.12**	**0.17**	0.16	**0.21**	**0.16**	2.7	1.8	2.7	2.4

* denotes the reproduced results by running official code.

4.2 Camera Tracking Accuracy

Tab. 1 shows the tracking accuracy of our method compared to other approaches on both synthetic [37] and real-world [38] datasets. On Replica [37], the proposed method achieved state-of-the-art (SOTA) performance across all scenes, reducing the trajectory error by more than 50% compared to the previous SOTA. This result arises from the fact that our proposed method actively utilizes 3D information by employing G-ICP [33] for tracking, unlike other methods that perform tracking based on errors in 2D space. On TUM [38], our method shows competitive tracking accuracy among coupled methods [14,22,32,44,47] which conduct tracking and mapping based on a single map. Note that the results of Gaussian Splatting SLAM are evaluated only on keyframes. Decoupled method [12] and its baseline [4] show better performance, but these approaches require additional resources for storing separate map information and costly computations for extracting features solely for the tracking process.

Table 2: Evaluation of System Speed and Map Quality on Replica. Our method outperforms all other frameworks in both system speed and quality of the reconstructed map.

Methods	Metrics	R0	R1	R2	Of0	Of1	Of2	Of3	Of4	Avg.
Orbeez-SLAM* [8]	PSNR[dB] ↑	12.13	15.28	15.87	17.59	19.26	10.30	11.55	12.65	14.33
	SSIM ↑	0.70	0.79	0.81	0.80	0.67	0.78	0.76	0.84	0.77
	LPIPS ↓	0.61	0.48	0.51	0.52	0.41	0.55	0.54	0.53	0.53
	FPS ↑	24.05	24.78	24.00	19.22	26.74	23.88	24.45	26.11	24.15
Point-SLAM* [32]	PSNR[dB] ↑	33.38	34.10	36.32	38.72	39.31	34.22	34.10	34.82	35.62
	SSIM ↑	**0.98**	**0.98**	**0.99**	**0.99**	**0.99**	0.96	0.96	**0.98**	**0.98**
	LPIPS ↓	0.10	0.12	0.10	0.09	0.11	0.15	0.12	0.13	0.11
	FPS ↑	0.26	0.30	0.31	0.33	0.34	0.30	0.28	0.30	0.30
GS-SLAM [44]	PSNR[dB] ↑	31.56	32.86	32.59	38.70	41.17	32.36	32.03	32.92	34.27
	SSIM ↑	0.97	0.97	0.97	**0.99**	**0.99**	**0.98**	**0.97**	0.97	**0.98**
	LPIPS ↓	0.09	0.08	0.09	0.05	**0.03**	0.09	0.11	0.11	0.08
	FPS ↑	8.34	-	-	-	-	-	-	-	8.34
SplaTAM* [14]	PSNR[dB] ↑	32.60	33.55	34.83	38.09	39.02	31.95	29.53	31.55	33.89
	SSIM ↑	**0.98**	0.97	0.98	0.98	0.98	0.97	0.95	0.95	0.97
	LPIPS ↓	0.07	0.10	0.07	0.09	0.09	0.10	0.12	0.15	0.10
	FPS ↑	0.24	0.19	0.19	0.20	0.22	0.27	0.26	0.24	0.23
Gaussian Splatting SLAM [22]	PSNR[dB] ↑	34.83	36.43	37.49	39.95	42.09	36.24	36.70	36.07	37.50
	SSIM ↑	0.95	0.96	0.97	0.97	0.98	0.96	0.96	0.96	0.96
	LPIPS ↓	0.07	0.08	0.08	0.07	0.06	0.08	0.07	0.10	0.08
	FPS ↑	-	-	-	-	-	1.1	-	-	-
Ours (no tracking speed limit)	PSNR[dB] ↑	32.20	35.36	34.42	40.31	40.75	33.85	34.08	36.47	35.93
	SSIM ↑	0.94	0.96	0.96	0.98	0.98	0.96	0.95	0.96	0.96
	LPIPS ↓	0.08	0.07	0.08	0.05	0.05	0.07	0.07	0.07	0.07
	FPS ↑	**100.98**	**84.92**	**103.38**	**99.10**	**107.06**	**95.60**	**97.20**	**96.66**	**98.11**
Ours (limited to 30 FPS)	PSNR[dB] ↑	**35.37**	**37.80**	**38.50**	**43.13**	**43.26**	**36.93**	**36.90**	**38.75**	**38.83**
	SSIM ↑	0.96	0.97	0.98	**0.99**	**0.99**	0.97	**0.97**	0.97	**0.98**
	LPIPS ↓	**0.05**	**0.05**	**0.05**	**0.03**	**0.03**	**0.04**	**0.04**	**0.05**	**0.04**
	FPS ↑	29.97	29.98	29.98	29.98	29.99	29.97	29.97	29.97	29.98

* denotes the reproduced results by running official code.
The result of GS-SLAM and Gaussian Splatting SLAM is taken from [44] and [22].

Table 3: Evaluation of System Speed and Map Quality on TUM-RGBD. Proposed method shows incredible system speed and competitive map quality.

Methods	PSNR[dB] ↑	SSIM ↑	LPIPS ↓	FPS ↑
NICE-SLAM* [47]	14.10	0.574	0.395	0.08
Point-SLAM* [32]	21.40	0.738	0.447	0.22
Photo-SLAM [12]	21.90	0.763	0.187	-
SplaTAM* [14]	**23.46**	**0.906**	**0.156**	0.32
Ours (unlimited tracking speed)	19.62	0.750	0.240	**73.92**
Ours (limited to 30 FPS)	20.72	0.768	0.218	29.99

* denotes the reproduced results by running official code.
The result of Photo-SLAM is taken from [12].

4.3 System Speed and Quality of Reconstructed Map

Tab. 2 and Tab. 3 demonstrate the FPS of the systems and quality of the reconstructed map on Replica [37] and TUM [38] dataset, respectively. FPS of the system is calculated by dividing the total number of frames by the total time. Note that this FPS represents the overall system performance including tracking and mapping processes, rather than just individual components. Photo-SLAM [12] reported the tracking FPS of their system, but did not include the system FPS, so we leave that field blank in the Tab. 3. Since our method implements mapping and tracking processes to operate in parallel, the number of map optimization iterations varies depending on the tracking speed, consequently affecting the quality of the reconstructed map. So we evaluate our system in two cases: one where the tracking speed is limited to the typical sensor input speed of 30 FPS, and the other where the tracking speed is not limited. Please refer to the supplementary material for results across various FPS and geometric quality evaluations.

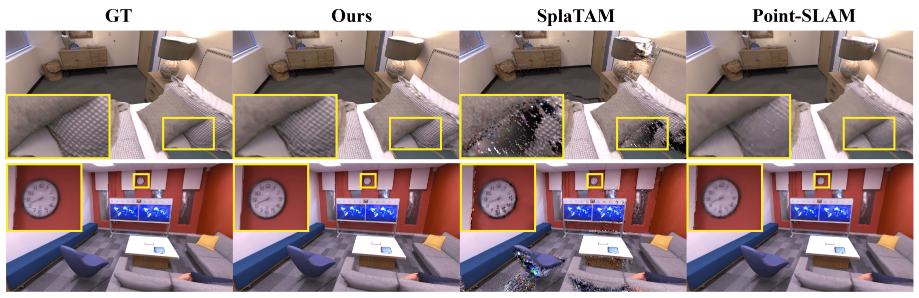

Fig. 5: Comparison of Rendering Results. In the first scene, SplaTAM [14] failed to reconstruct the pillow and lamp and Point-SLAM [32] failed to represent the detailed pattern of the pillow. In the second case, SplaTAM and Point-SLAM failed to accurately reconstruct the details of the clock. However, our method exhibits rendering results that closely resemble the ground truth image, demonstrating a high level of accuracy.

In the limited case (30 FPS), our system achieves SOTA map quality across all scenes in the Replica dataset. When the tracking speed is not limited, our method shows extremely fast system speed up to 107 FPS. This speed is more than four times faster than that of the Orbeez-SLAM [8], based on the ORB-SLAM2 [26]. Remarkably, even under this condition, our approach maintains superior map quality and outperforms other methods. The proposed method effectively integrates Gaussians for map representation with appropriate initial states and has better detailed representation capability compared to NeRF-based methods. As shown in Fig. 5, SplaTAM suffers from incorporating appropriate Gaussians into the map, leading to regions of poor quality. Point-SLAM exhibits areas where detailed representation is lacking. In TUM-RGBD, our method demonstrates remarkably fast system speed with competitive map quality, showcasing its practicality in real-world scenarios. Compared to SplaTAM [14], our approach shows a slight decrease in map quality, with a PSNR reduction of approximately 11.7%. However, we achieve significant speed improvements, with a speedup of approximately 91.6 times under limited conditions and up to 227 times at maximum speed. The reason is that, unlike Replica, TUM dataset is captured with old-fashioned sensors, resulting in noisy and substantial information loss in depth images. While SplaTAM treats these factors by adding new Gaussians only in regions with significant depth loss, our method utilizes all structural information from the depth image, focusing on accurate tracking and rapid system rather than treating such factors.

4.4 Ablation study

Table 4: Scale Regularization Ablation on TUM-RGBD. Reported results are the average ATE RMSE $\downarrow$ [cm] of 3 scenes in TUM-RGBD dataset.

Scale Regularization	fr1-desk	fr2-xyz	fr3-office	Avg.
✗	136.32	196.61	376.40	236.54
Plane	65.63	5.53	16.21	29.12
ellipsoid	**2.66**	**1.77**	**2.67**	**2.37**

Scale Regularization Tab. 4 shows the result of the ablation study on scale regularization while tracking. We perform scale regularization to enhance tracking accuracy by ensuring that the scales of Gaussians in the current frame and the map are within a similar range. Despite the application of plane regularization resulting in some improvement in tracking accuracy, it remains inadequate. This is because the plane regularization method considers all the Gaussians as planes, ignoring the characteristics of Gaussians existing in the map, which are optimized to represent the surrounding space through the mapping process. On the other hand, when applying the proposed ellipsoid regularization, the best tracking accuracy is achieved. This is attributed to the ability of the proposed ellipsoid regularization to consider the characteristics of Gaussians existing in the map while performing regularization.
Scale Aligning We implement scale aligning to reduce the difference between scales

Table 5: Ablation of Scale Aligning on Replica. The results are the average of 8 scenes in the Replica dataset.

Covs from G-ICP	Re-init	ATE ↓	PSNR ↑	SSIM ↑	LPIPS ↓
✗	✗	8.893	24.81	0.840	0.380
✓	✗	0.258	33.21	0.922	0.177
✓	*constant*	0.158	37.33	0.964	0.074
✓	$z^{1.5}$	**0.157**	**38.83**	**0.975**	**0.041**

of existing Gaussians in the map and newly added Gaussians. By doing this, we ensure that scales of newly added Gaussians closely resemble the optimal value they will reach through optimization. To demonstrate the effectiveness of this approach, we tested our method in various cases and Tab. 5 shows the results. When not utilizing the covariances computed during the G-ICP [33] process, we calculate the scales of Gaussians by using simple-knn module of vanilla 3DGS [15], and set rotations as identity. In this case, a significant drop in tracking performance appeared. This is because the newly added Gaussians lack sufficient prior information about the 3D structure, causing them to be utilized as target Gaussians without being optimally optimized in space. In the case of using the computed covariance from the G-ICP process, the tracking and mapping performance improves. Furthermore, the best performance is observed when dividing the scale by $z^{1.5}$. These results indicate that utilizing covariances calculated during G-ICP procedure is effective for both tracking and mapping, and scale aligning facilitates smooth connection from G-ICP to 3DGS. Further analysis of the advantages of this connection is included in the supplementary material.

Table 6: Keyframe and Mapping-Only Keyframe Selection Ablation on Replica. Reported results are the average of 8 scenes in the Replica dataset.

Keyframe	Mapping-Only Keyframe	ATE [cm] ↓	PSNR ↑	SSIM ↑	LPIPS ↓
tracking + every 30 frame	-	0.174	37.52	0.968	0.055
tracking + every 10 frame	-	0.176	**38.83**	0.974	**0.041**
tracking	every 10 frame	**0.157**	**38.83**	**0.975**	**0.041**

Additional Keyframe Selection for Mapping For mapping quality, sufficient keyframes are required, but indiscriminate addition of keyframes causes a drop in tracking accuracy. To address this challenge and maximize both tracking accuracy and mapping quality, we pick mapping-only keyframes in addition to the original keyframes. Tab. 6 presents the result of the ablation study of this method. When using only keyframes without mapping-only keyframes, camera tracking accuracy suffers due to accumulated errors in scan-matching. However, employing our proposed method yields the best tracking accuracy and mapping quality. This shows that our method selectively adds keyframes essential for tracking, minimizing accumulated errors in scan-matching. While leveraging these benefits, we ensure sufficient primitives and images for training, which are crucial for map quality through the use of mapping-only keyframes.

Table 7: Ablation of Methods for Avoiding Local Minima While Mapping on Replica. The results are the average of 8 scenes in the Replica dataset.

(a)

Keyframe choice for learning	PSNR	SSIM	LPIPS
recent 1 keyframe	26.85	0.872	0.200
covisible keyframes	31.34	0.924	0.120
random keyframes	**38.83**	**0.975**	**0.041**

(b)

densifying	pruning	ATE [cm]	PSNR	SSIM	LPIPS
✓	✓	0.188	38.65	0.973	0.041
✓	✗	0.170	37.90	0.971	0.052
✗	✓	**0.157**	**38.83**	**0.975**	**0.041**
✗	✗	0.159	38.72	0.974	0.041

Avoiding Local Minima while Mapping We employ two methods to prevent Gaussians from overfitting to the training image, which results in elongation to the viewpoint direction: (1) training with randomly selected keyframes and (2) Gaussian pruning. Tab. 7 presents the results of the ablation study on these methods. The worst rendering performance is observed when repeatedly training with the recently added keyframe, while using covisible keyframes for training yields better results. Covisible keyframes offer more diverse viewpoints for the Gaussians compared to a single keyframe, yet they still provide limited diversity in viewpoints. Random keyframes provide the most diverse viewpoints compared to other methods, leading to a significant improvement in map quality. In the vanilla 3DGS [15], Gaussian densifying/pruning techniques are utilized to manage Gaussians. Tab. 7 (b) shows the results of an ablation test of Gaussian densifying/pruning in our method. Since our method appropriately supplies Gaussians for scene representation, densifying is unnecessary. However, pruning improves tracking accuracy and map quality by removing Gaussians that have become elongated due to overfitting or are no longer essential for representing the map.

5 Conclusion

In this paper, we have proposed RGBD GS-ICP-SLAM, a dense representation SLAM system that leverages 3D Gaussian representation for high-fidelity spatial representation. We demonstrate that a fusion of G-ICP and 3DGS that utilizes a single 3D Gaussian map for both tracking and mapping yields mutual benefits. The exchange of Gaussians between tracking and mapping processes with scale alignments minimizes redundant computations and constructs an efficient system. Moreover, our dynamic keyframe selection method enhances both tracking and mapping performance. Through extensive experiments, the proposed approach presents state-of-the-art performance in spatial representation, camera pose estimation, and total system speed.

Limitations. The proposed method achieved fast system speed by relying solely on depth for the 3D structure. In real-world, there are limitations in the quality of maps due to the depth noise in RGB-D sensors. Since the system speed is already exceptionally fast, it is expected that robust performance can be achieved by trading off a bit of speed to compensate for noisy depth images with relatively robust RGB information.

Acknowledgment. This work was supported by the National Research Foundation of Korea (NRF) grant (no. RS-2024-00359937).

References

1. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. Spie (1992)
2. Biber, P., Straßer, W.: The normal distributions transform: A new approach to laser scan matching. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453). vol. 3, pp. 2743–2748. IEEE (2003)
3. Bylow, E., Sturm, J., Kerl, C., Kahl, F., Cremers, D.: Real-time camera tracking and 3d reconstruction using signed distance functions. In: Robotics: Science and Systems. vol. 2, p. 2 (2013)
4. Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M., Tardós, J.D.: Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. IEEE Transactions on Robotics **37**(6), 1874–1890 (2021)
5. Canelhas, D.R., Stoyanov, T., Lilienthal, A.J.: Sdf tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3671–3676. IEEE (2013)
6. Ceriani, S., Sánchez, C., Taddei, P., Wolfart, E., Sequeira, V.: Pose interpolation slam for large maps using moving 3d sensors. In: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 750–757. IEEE (2015)
7. Chetverikov, D., Svirko, D., Stepanov, D., Krsek, P.: The trimmed iterative closest point algorithm. In: 2002 International Conference on Pattern Recognition. vol. 3, pp. 545–548. IEEE (2002)
8. Chung, C.M., Tseng, Y.C., Hsu, Y.C., Shi, X.Q., Hua, Y.H., Yeh, J.F., Chen, W.C., Chen, Y.T., Hsu, W.H.: Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 9400–9406. IEEE (2023)
9. Czarnowski, J., Laidlow, T., Clark, R., Davison, A.J.: Deepfactors: Real-time probabilistic dense monocular slam. IEEE Robotics and Automation Letters **5**(2), 721–728 (2020)
10. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (ToG) **36**(4), 1 (2017)
11. Grisetti, G., Kümmerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based slam. IEEE Intelligent Transportation Systems Magazine **2**(4), 31–43 (2010)
12. Huang, H., Li, L., Cheng, H., Yeung, S.K.: Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. arXiv preprint arXiv:2311.16728 (2023)
13. Johari, M.M., Carta, C., Fleuret, F.: 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)
14. Keetha, N., Karhade, J., Jatavallabhula, K.M., Yang, G., Scherer, S., Ramanan, D., Luiten, J.: Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. arXiv preprint arXiv:2312.02126 (2023)
15. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
16. Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for rgb-d cameras. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2100–2106. IEEE (2013)
17. Koide, K., Yokozuka, M., Oishi, S., Banno, A.: Voxelized gicp for fast and accurate 3d point cloud registration. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 11054–11059. IEEE (2021)

18. Kong, X., Liu, S., Taher, M., Davison, A.J.: vmap: Vectorised object mapping for neural field slam. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 952–961 (2023)
19. Lin, J., Zhang, F.: Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 3126–3131. IEEE (2020)
20. Low, K.L.: Linear least-squares optimization for point-to-plane icp surface registration. Chapel Hill, University of North Carolina **4**(10), 1–3 (2004)
21. Mallios, A., Ridao, P., Ribas, D., Hernández, E.: Scan matching slam in underwater environments. Autonomous Robots **36**, 181–198 (2014)
22. Matsuki, H., Murai, R., Kelly, P.H., Davison, A.J.: Gaussian splatting slam. arXiv preprint arXiv:2312.06741 (2023)
23. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
24. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
25. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics **31**(5), 1147–1163 (2015)
26. Mur-Artal, R., Tardós, J.D.: Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE transactions on robotics **33**(5), 1255–1262 (2017)
27. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE international symposium on mixed and augmented reality. pp. 127–136. Ieee (2011)
28. Nieto, J., Bailey, T., Nebot, E.: Recursive scan-matching slam. Robotics and Autonomous systems **55**(1), 39–49 (2007)
29. Palieri, M., Morrell, B., Thakur, A., Ebadi, K., Nash, J., Chatterjee, A., Kanellakis, C., Carlone, L., Guaragnella, C., Agha-Mohammadi, A.a.: Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time. IEEE Robotics and Automation Letters **6**(2), 421–428 (2020)
30. Prisacariu, V.A., Kähler, O., Golodetz, S., Sapienza, M., Cavallari, T., Torr, P.H., Murray, D.W.: Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. arXiv preprint arXiv:1708.00783 (2017)
31. Qin, T., Li, P., Shen, S.: Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics **34**(4), 1004–1020 (2018)
32. Sandström, E., Li, Y., Van Gool, L., Oswald, M.R.: Point-slam: Dense neural point cloud-based slam. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 18433–18444 (2023)
33. Segal, A., Haehnel, D., Thrun, S.: Generalized-icp. In: Robotics: science and systems. vol. 2, p. 435. Seattle, WA (2009)
34. Serafin, J., Grisetti, G.: Nicp: Dense normal based point cloud registration. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 742–749. IEEE (2015)
35. Shan, T., Englot, B.: Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4758–4765. IEEE (2018)
36. Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., Rus, D.: Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 5135–5142. IEEE (2020)

37. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., et al.: The replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019)
38. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: 2012 IEEE/RSJ international conference on intelligent robots and systems. pp. 573–580. IEEE (2012)
39. Sucar, E., Liu, S., Ortiz, J., Davison, A.J.: imap: Implicit mapping and positioning in real-time. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6229–6238 (2021)
40. Taud, H., Mas, J.: Multilayer perceptron (mlp). Geomatic approaches for modeling land change scenarios pp. 451–455 (2018)
41. Vizzo, I., Guadagnino, T., Mersch, B., Wiesmann, L., Behley, J., Stachniss, C.: Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way. IEEE Robotics and Automation Letters **8**(2), 1029–1036 (2023)
42. Whelan, T., Johannsson, H., Kaess, M., Leonard, J.J., McDonald, J.: Robust real-time visual odometry for dense rgb-d mapping. In: 2013 IEEE International Conference on Robotics and Automation. pp. 5724–5731. IEEE (2013)
43. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. arXiv preprint arXiv:2310.08528 (2023)
44. Yan, C., Qu, D., Wang, D., Xu, D., Wang, Z., Zhao, B., Li, X.: Gs-slam: Dense visual slam with 3d gaussian splatting. arXiv preprint arXiv:2311.11700 (2023)
45. Yang, X., Li, H., Zhai, H., Ming, Y., Liu, Y., Zhang, G.: Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In: 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 499–507. IEEE (2022)
46. Zhang, J., Singh, S.: Loam: Lidar odometry and mapping in real-time. In: Robotics: Science and systems. vol. 2, pp. 1–9. Berkeley, CA (2014)
47. Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M.R., Pollefeys, M.: 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)