GSplatLoc : Ultra-Precise Pose Optimization via 3D Gaussian Rendering

https://github.com/Atticuszz/GsplatLoc

Atticus Zhou, Atticus Zhou, Atticus Zhou, Atticus Zhou

September 3, 2024

Abstract

We present GSplatLoc, an innovative pose estimation method for RGB-D cameras that employs a volumetric representation of 3D Gaussians. This approach facilitates precise pose estimation by minimizing the loss based on the rendering of 3D Gaussians from real depth maps captured from the estimated pose. Our method attains rotational errors close to zero and translational errors within 0.01mm, representing a substantial advancement in pose accuracy over existing point cloud registration algorithms, as well as explicit volumetric and implicit neural representation-based SLAM methods. Comprehensive evaluations demonstrate that GSplatLoc significantly improves pose estimation accuracy, which contributes to increased robustness and fidelity in real-time 3D scene reconstruction, setting a new standard for localization techniques in dense mapping SLAM.

1 Introduction

Visual localization, the task of estimating camera position and orientation for a given image within a known scene, is a fundamental challenge in computer vision. This problem is crucial for enabling truly autonomous robots, such as self-driving cars, and serves as a prerequisite for Augmented and Virtual Reality systems. Recent advancements in Dense Visual Localization and Mapping (Visual SLAM) have shown significant progress in simultaneously performing pose tracking and scene mapping for various applications, including virtual/augmented reality (VR/AR), robot navigation, and autonomous driving.

Traditional visual SLAM systems [1] have demonstrated accurate tracking performance across diverse environments. However, their underlying 3D representations (e.g., point clouds, meshes, and surfels) exhibit limitations in facilitating highly flexible scene exploration, such as photorealistic scene touring and fine-grained map updating. The introduction of Neural Radiance Fields (NeRF) [2] for surface reconstruction and view rendering has inspired novel NeRF-based SLAM methods [3]. These approaches have shown promising results in tracking, surface modeling, and novel view synthesis. Nevertheless, existing NeRF-based methods rely on a differential volume rendering pipeline that is computationally intensive and time-consuming, limiting their ability to perform real-time tracking and mapping.

The recent development of 3D Gaussian Splatting [4] for efficient novel view synthesis has shown great potential in addressing the inherent challenges of NeRF-based SLAM. Its rasterization-based rendering pipeline allows for faster image-level rendering, making it an attractive option for real-time applications. However, integrating 3D Gaussian fields into SLAM systems presents several challenges, including overfitting to input images due to strong anisotropy and the lack of explicit multi-view constraints.

Current Gaussian Splatting-based SLAM methods, such as RTG-SLAM [5] and GS-ICP-SLAM [6], primarily rely on ICP-based techniques for pose estimation. Other approaches, like Gaussian-SLAM [7], adapt traditional RGB-D odometry methods. While these methods have shown promising results, they may not fully exploit the differentiable nature of the Gaussian Splatting representation for pose estimation.

In this paper, we introduce GSplatLoc, a novel camera localization method that leverages the differentiable properties of 3D Gaussian Splatting for efficient and accurate pose estimation. Our approach is designed to address the limitations of existing methods by focusing solely on the localization aspect of SLAM, allowing for better utilization of the scene representation and camera properties. By developing a fully differentiable pipeline, GSplatLoc can be easily integrated into existing Gaussian Splatting SLAM frameworks or other deep learning tasks.

Our main contributions are as follows:

1. We present a GPU-accelerated framework for real-time camera localization based on a comprehensive theoretical analysis of camera pose derivatives in 3D Gaussian Splatting.

2. We propose a novel optimization approach that focuses on camera pose estimation given a 3D Gaussian scene, fully exploiting the differentiable nature of the rendering process.
3. We demonstrate the effectiveness of our method through extensive experiments, showing competitive or superior tracking results compared to state-of-the-art SLAM approaches utilizing advanced scene representations.

By addressing the challenges of localization in Gaussian Splatting-based scenes, GSplatLoc opens new avenues for high-precision camera pose estimation in complex environments. Our work contributes to the ongoing advancement of visual localization systems, pushing the boundaries of accuracy and real-time performance in 3D scene understanding and navigation.

2 Related Work

2.1 Classical RGBD dense SLAM

Dense visual SLAM focuses on reconstructing detailed 3D maps, unlike sparse SLAM methods which excel in pose estimation [[8], [9], [10]] but typically yield maps useful mainly for localisation. In contrast, dense SLAM creates interactive maps beneficial for broader applications, including AR and robotics. Dense SLAM methods are generally divided into two primary categories: Frame-centric and Map-centric. Frame-centric SLAM minimises photometric error across consecutive frames, jointly estimating per-frame depth and frame-to-frame camera motion. Frame-centric approaches [[11], [12]] are efficient, as individual frames host local rather than global geometry (e.g. depth maps), and are attractive for long-session SLAM, but if a dense global map is needed, it must be constructed on demand by assembling all of these parts which are not necessarily fully consistent. In contrast, Map-centric SLAM uses a unified 3D representation across the SLAM pipeline, enabling a compact and streamlined system. Compared to purely local frame-to-frame tracking, a map-centric approach leverages global information by tracking against the reconstructed 3D consistent map. Classical map-centric approaches often use voxel grids [[13], [14],[15], [16]] or points [[17], [18], [19]] as the underlying 3D representation. While voxels enable a fast look-up of features in 3D, the representation is expensive, and the fixed voxel resolution and distribution are problematic when the spatial characteristics of the environment are not known in advance. On the other hand, a point-based map representation, such as surfel clouds, enables adaptive changes in resolution and spatial distribution by dynamic allocation of point primitives in the 3D space. Such flexibility benefits online applications such as SLAM with deformation-based loop closure [[18], [19]]. However, optimising the representation to capture high fidelity is challenging due to the lack of correlation among the primitives. Recently, in addition to classical graphic primitives, neural network-based map representations are a promising alternative. iMAP [20]demonstrated the interesting properties of neural representation, such as sensible hole filling of unobserved geometry. Many recent approaches combine the classical and neural representations to capture finer details [[21], [3], [22], [23]]; however, the large amount of computation required for neural rendering makes the live operation of such systems challenging.

2.2 NeRF-based RGBD dense SLAM

The classical method for creating a 3D representation was to unproject 2D observations into 3D space and to fuse them via weighted averaging [[24], [14]]. Such an averaging scheme suffers from over-smooth representation and lacks the expressiveness to capture high-quality details. To capture a scene with photorealistic quality, differentiable volumetric rendering [[25]] has recently been popularised with Neural Radiance Fields (NeRF) [[2]]. Using a single Multi-Layer Perceptron (MLP) as a scene representation, NeRF performs volume rendering by marching along pixel rays, querying the MLP for opacity and colour. Since volume rendering is naturally differentiable, the MLP representation is optimised to minimise the rendering loss using multiview information to achieve high-quality novel view synthesis. The main weakness of NeRF is its training speed. Recent developments have introduced explicit volume structures such as multi-resolution voxel grids [[26],[27], [28]] or hash functions [[29]] to improve performance. Interestingly, these projects demonstrate that the main contributor to high-quality novel view synthesis is not the neural network but rather differentiable volumetric rendering, and that it is possible to avoid the use of an MLP and yet achieve comparable rendering quality to NeRF [[26]]. However, even in these systems, per-pixel ray marching remains a significant bottleneck for rendering speed. This issue is particularly critical in SLAM, where immediate interaction with the map is essential for tracking. In contrast to NeRF

The classical method for creating a 3D representation was to unproject 2D observations into 3D space and to fuse them via weighted averaging [16, 23]. Such an averaging scheme suffers from over-smooth representation and lacks the expressiveness to capture high-quality details. To capture a scene with photorealistic quality, differentiable volumetric rendering [24] has recently been popularised with Neural Radiance Fields (NeRF) [17]. Using a single Multi-Layer Perceptron (MLP) as a scene representation, NeRF performs volume rendering by marching along pixel rays, querying the MLP for opacity and colour. Since volume rendering is naturally differentiable, the MLP representation is optimised to minimise the rendering loss using multiview information to achieve high-quality novel view synthesis. The main weakness of NeRF is its training speed. Recent developments have introduced explicit volume structures such as multi-resolution voxel grids [6, 14, 34] or hash functions [19] to improve performance. Interestingly, these projects demonstrate that the main contributor to high-quality novel view synthesis is not the neural network but rather differentiable volumetric rendering, and that it is possible to avoid the use of an MLP and yet achieve comparable rendering quality to NeRF [6]. However, even in these systems, per-pixel ray marching remains a significant bottleneck for rendering speed. This issue is particularly critical in SLAM, where immediate interaction with the map is essential for tracking.

2.3 Gaussian-based RGBD dense SLAM

In contrast to NeRF, 3DGS performs differentiable rasterisation. Similar to regular graphics rasterisations, by iterating over the primitives to be rasterised rather than marching along rays, 3DGS leverages the natural sparsity of a 3D scene and achieves a representation which is expressive to capture high-fidelity 3D scenes while offering significantly faster rendering. Several works have applied 3D Gaussians and differentiable rendering to static scene capture [30], [31], and in particular more recent works utilise 3DGS and demonstrate superior results in vision tasks such as dynamic scene capture [[32],[33]], and 3D generation [[34], [35]]. Our method adopts a Map-centric approach, utilising 3D Gaussians as the only SLAM representation. Similar to surfel-based SLAM, we dynamically allocate the 3D Gaussians, enabling us to model an arbitrary spatial distribution in the scene. Unlike other methods such as ElasticFusion [[19]] and PointFusion [17], however, by using differentiable rasterisation, our SLAM system can capture highfidelity scene details and represent challenging object properties by direct optimisation against information from every pixel.

3DGS performs differentiable rasterisation. Similar to regular graphics rasterisations, by iterating over the primitives to be rasterised rather than marching along rays, 3DGS leverages the natural sparsity of a 3D scene and achieves a representation which is expressive to capture high-fidelity 3D scenes while offering significantly faster rendering. Several works have applied 3D Gaussians and differentiable rendering to static scene capture [11, 38], and in particular more recent works utilise 3DGS and demonstrate superior results in vision tasks such as dynamic scene capture [15, 42, 44] and 3D generation [35, 45]. Our method adopts a Map-centric approach, utilising 3D Gaussians as the only SLAM representation. Similar to surfel-based SLAM, we dynamically allocate the 3D Gaussians, enabling us to model an arbitrary spatial distribution in the scene. Unlike other methods such as ElasticFusion [41] and PointFusion [9], however, by using differentiable rasterisation, our SLAM system can capture highfidelity scene details and represent challenging object properties by direct optimisation against information from every pixel

3 Method

Overview. The GSplatLoc method presents an innovative approach to camera localization, leveraging the differentiable nature of 3D Gaussian splatting for efficient and accurate pose estimation.

Motivation. Recent advancements in 3D scene representation, particularly the 3D Gaussian Splatting technique [4], have opened new avenues for efficient and high-quality 3D scene rendering. By adapting this approach to the task of camera localization, we aim to exploit its differentiable properties and speed advantages to achieve robust and real-time pose estimation.

Problem formulation. Our objective is to estimate the 6-DoF pose $(\mathbf{R}, \mathbf{t}) \in SE(3)$ of a query depth image D_q, where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation vector in the camera coordinate system. Given a 3D representation of the environment in the form of 3D Gaussians, let $\mathcal{G} = \{G_i\}_{i=1}^{N}$ denote a set of N 3D Gaussians, and posed reference depth images $\{D_k\}$, which together constitute the reference data.

3.1 Scene Representation

Building upon the Gaussian splatting method [4], we adapt the scene representation to focus on the differentiable depth rendering process, which is crucial for our localization task. Our approach utilizes the efficiency and quality of Gaussian splatting while tailoring it specifically for depth-based localization.

3D Gaussians. Each Gaussian G_i is characterized by its 3D mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$, 3D covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times3}$, opacity $o_i \in \mathbb{R}$, and scale $\mathbf{s}_i \in \mathbb{R}^3$. To represent the orientation of each Gaussian, we use a rotation quaternion $\mathbf{q}_i \in \mathbb{R}^4$.

The 3D covariance matrix $\boldsymbol{\Sigma}_i$ is then parameterized using $\mathbf{s}_i$ and $\mathbf{q}_i$:

$$\boldsymbol{\Sigma}_i = \mathbf{R}(\mathbf{q}_i)\mathbf{S}(\mathbf{s}_i)\mathbf{S}(\mathbf{s}_i)^\top\mathbf{R}(\mathbf{q}_i)^\top$$

where $\mathbf{R}(\mathbf{q}_i)$ is the rotation matrix derived from $\mathbf{q}_i$, and $\mathbf{S}(\mathbf{s}_i) = \mathrm{diag}(\mathbf{s}_i)$ is a diagonal matrix of scales.

Projecting 3D to 2D: To project these 3D Gaussians onto a 2D image plane, we follow the approach described by [4]. The projection of the 3D mean $\boldsymbol{\mu}_i$ to the 2D image plane is given by:

$$\boldsymbol{\mu}_{I,i} = \pi(\mathbf{P}(\mathbf{T}_{wc}\boldsymbol{\mu}_{i,\text{homogeneous}}))$$

where $\mathbf{T}_{wc} \in SE(3)$ is the world-to-camera transformation, $\mathbf{P} \in \mathbb{R}^{4\times4}$ is the projection matrix [36], and $\pi : \mathbb{R}^4 \to \mathbb{R}^2$ maps to pixel coordinates.

The 2D covariance $\mathbf{\Sigma}_{I,i} \in \mathbb{R}^{2\times2}$ of the projected Gaussian is derived as:

$$\mathbf{\Sigma}_{I,i} = \mathbf{J}\mathbf{R}_{wc}\mathbf{\Sigma}_i\mathbf{R}_{wc}^\top\mathbf{J}^\top$$

where $\mathbf{R}_{wc}$ represents the rotation component of $\mathbf{T}_{wc}$, and $\mathbf{J}$ is the affine transform as described by [37].

3.2 Depth Rendering

We implement a differential depth rendering process, which is crucial for our localization method as it allows for gradient computation throughout the rendering pipeline. This differentiability enables us to optimize camera poses directly based on rendered depth maps.

Compositing Depth: For depth map generation, we employ a front-to-back compositing scheme, which allows for accurate depth estimation and edge alignment. Let d_n represent the depth value associated with the n-th Gaussian, which is the z-coordinate of the Gaussian's mean in the camera coordinate system. The depth $D(\mathbf{p})$ at pixel $\mathbf{p}$ is computed as [4]:

$$D(\mathbf{p}) = \sum_{n\leq N} d_n \cdot \alpha_n \cdot T_n, \quad \text{where } T_n = \prod_{m<n}(1-\alpha_m)$$

Here, α_n represents the opacity of the n-th Gaussian at pixel $\mathbf{p}$, computed as:

$$\alpha_n = o_n \cdot \exp(-\sigma_n), \quad \sigma_n = \frac{1}{2}\mathbf{\Delta}_n^\top\mathbf{\Sigma}_I^{-1}\mathbf{\Delta}_n$$

where $\mathbf{\Delta}_n$ is the offset between the pixel center and the 2D Gaussian center $\boldsymbol{\mu}_I$, and o_n is the opacity parameter of the Gaussian. T_n denotes the cumulative transparency product of all Gaussians preceding n, accounting for the occlusion effects of previous Gaussians.

Scaling Depth. To ensure consistent representation across the image, we normalize the depth values. First, we calculate the total accumulated opacity $\alpha(\mathbf{p})$ for each pixel:

$$\alpha(\mathbf{p}) = \sum_{n\leq N} \alpha_n \cdot T_n$$

The normalized depth $\text{Norm}_D(\mathbf{p})$ is then defined as:

$$\text{Norm}_D(\mathbf{p}) = \frac{D(\mathbf{p})}{\alpha(\mathbf{p})}$$

This normalization process ensures that the depth values are properly scaled and comparable across different regions of the image, regardless of the varying densities of Gaussians in the scene.

The differentiable nature of this depth rendering process is key to our localization method. It allows us to compute gradients with respect to the Gaussian parameters and camera pose, enabling direct optimization of the camera pose based on the rendered depth maps. This differentiability facilitates efficient gradient-based optimization, forming the foundation for our subsequent localization algorithm.

3.3 Localization as Image Alignment

Assuming we have an existing map represented by a set of 3D Gaussians, our localization task focuses on estimating the 6-DoF pose of a query depth image D_q within this map. This process essentially becomes an image alignment problem between the rendered depth map from our Gaussian representation and the query depth image.

Rotating with Quaternions. We parameterize the camera pose using a quaternion $\mathbf{q}_{cw}$ for rotation and a vector $\mathbf{t}_{cw}$ for translation. This choice of parameterization is particularly advantageous in our differential rendering context. Quaternions provide a continuous and singularity-free representation of rotation, which is crucial for gradient-based optimization. Moreover, their compact four-parameter form aligns well with our differentiable rendering pipeline, allowing for efficient computation of gradients with respect to rotation parameters.

Loss function. Our optimization strategy is designed to leverage the differentiable nature of our depth rendering process. We define our loss function to incorporate both depth accuracy and edge alignment:

$$\mathcal{L} = \lambda_1\mathcal{L}_d + \lambda_2\mathcal{L}_c$$

where $\mathcal{L}_d$ represents the L1 loss for depth accuracy, and $\mathcal{L}_c$ focuses on the alignment of depth contours or edges. Specifically:

$$\mathcal{L}_d = \sum_{i\in\mathcal{M}} |D_i^{\text{rendered}} - D_i^{\text{observed}}|$$

$$\mathcal{L}_c = \sum_{j\in\mathcal{M}} |\nabla D_j^{\text{rendered}} - \nabla D_j^{\text{observed}}|$$

Here, $\mathcal{M}$ denotes the rendered alpha mask, indicating which pixels are valid for comparison. Both $\mathcal{L}_d$ and $\mathcal{L}_c$ are computed only over the masked regions. λ_1 and λ_2 are weights that balance the two parts of the loss function, typically set to 0.8 and 0.2 respectively, based on empirical results.

The contour loss L_c is computed using the Sobel operator [38], which effectively captures depth discontinuities and edges.

This additional term in our loss function serves several crucial purposes. It ensures that depth discontinuities in the rendered image align well with those in the observed depth image, thereby improving the overall accuracy of the pose estimation. By explicitly considering edge information, we preserve important structural features of the scene during optimization. Furthermore, the contour loss is less sensitive to absolute depth values and more focused on relative depth changes, making it robust to global depth scale differences.

The optimization objective can be formulated as:

$$\min_{\mathbf{q}_{cw},\mathbf{t}_{cw}} \mathscr{L} + \lambda_q \|\mathbf{q}_{cw}\|_2^2 + \lambda_t \|\mathbf{t}_{cw}\|_2^2$$

where λ_q and λ_t are regularization terms for the quaternion and translation parameters, respectively.

Masking Uncertainty. The rendered alpha mask plays a crucial role in our optimization process. It effectively captures the epistemic uncertainty of our map, allowing us to focus the optimization on well-represented parts of the scene. By utilizing this mask, we avoid optimizing based on unreliable or non-existent data, which could otherwise lead to erroneous pose estimates.

Fine-tuning the Engine. The learning rates are set to 5×10^{-4} for quaternion optimization and 10^{-3} for translation optimization, based on empirical results. The weight decay values, serving as regularization to mitigate overfitting, are set to 10^{-3} for both quaternion and translation parameters. These parameters are crucial for balancing the trade-off between convergence speed and stability in the optimization process.

3.4 Pipeline

The GSplatLoc method streamlines the localization process by utilizing only posed reference depth images $\{D_k\}$ and a query depth image D_q. Its differentiability in rendering of 3D Gaussians facilitates efficient and smooth convergence during optimization.

Evaluation Scene. For evaluation consistency, we initialize 3D Gaussians from point clouds rendered by $\{D_k\}$. Each point corresponds to a Gaussian's mean $\boldsymbol{\mu}_i$. After outlier filtering, we set opacity $o_i = 1$ for all Gaussians. The scale $\mathbf{s}_i \in \mathbb{R}^3$ is initialized based on local point density:

$$\mathbf{s}_i = (\sigma_i, \sigma_i, \sigma_i), \text{ where } \sigma_i = \sqrt{\frac{1}{3}\sum_{j=1}^{3} d_{ij}^2}$$

Here, d_{ij} denotes the distance to the j-th nearest neighbor of point i, calculated using k-nearest neighbors ($k = 4$). This isotropic initialization ensures balanced representation of local geometry. We initially set rotation $\mathbf{q}_i = (1, 0, 0, 0)$ for all Gaussians.

To enhance optimization stability, we apply standard Principal Component Analysis (PCA) for principal axis alignment of the point cloud. This process involves centering the point cloud at its mean and aligning its principal axes with the coordinate axes. The PCA-based alignment normalizes the overall scene orientation, providing a more uniform starting point for optimization across diverse datasets. This approach significantly improves the stability of loss reduction during optimization and facilitates the achievement of lower final loss values, particularly in the depth loss component of our objective function.

Optimization. We employ the Adam optimizer for both quaternion and translation optimization, with distinct learning rates and weight decay values for each. The optimization process benefits from the real-time rendering capabilities of 3D Gaussian Splatting. Each iteration of the optimizer is essentially limited only by the speed of rendering, which is extremely fast due to the efficiency of Gaussian splatting. This allows for rapid convergence of our pose estimation algorithm, making it suitable for real-time applications.

Convergence. To determine the convergence of the optimization process, we implement an early stopping mechanism based on the stabilization of the total loss. Extensive experimental results indicate that the total loss typically stabilizes after approximately 100 iterations. We employ a patience mechanism, activated after 100 iterations. If the total loss fails to decrease for a consecutive number of patience iterations, the optimization loop is terminated. The pose estimate corresponding to the minimum total loss is subsequently selected as the optimal pose.

This pipeline effectively combines the efficiency of Gaussian splatting with a robust optimization strategy, resulting in a fast and accurate camera localization method.

4 Evaluation

In this section, we delineate our experimental setup and validate that the proposed system achieves significant improvements in accuracy.

4.1 Experimental Setup

Implementation Details. We implemented our SLAM system on a laptop equipped with a 13*th* Gen Intel(R) Core(TM) i7-13620H CPU, 16*GB* of RAM, and an NVIDIA RTX 4060 8*GB* GPU. The optimization pipeline was implemented using Python with the PyTorch framework, while custom CUDA kernels were developed for rasterization and backpropagation operations.

Datasets. We utilized the Replica dataset [39] and the TUM-RGBD dataset [40] to evaluate our pose estimation accuracy. The Replica dataset, specifically designed for RGB-D SLAM evaluation, provides high-quality 3D reconstructions of various indoor scenes. We employed the publicly available dataset collected by Sucar et al. [20], which offers trajectories from an RGB-D sensor. The Replica dataset contains challenging purely rotational camera motions. The TUM-RGBD dataset, widely used in the SLAM field for evaluating tracking accuracy, represents real-world scenarios and provides precise camera poses captured by an external motion capture system.

Metrics. To assess camera pose estimation accuracy, we employed the average absolute trajectory error (ATE RMSE) and the absolute angular error (AAE RMSE). In the result tables, we highlight the best, second, and third performances.

Baselines. We evaluate our method against state-of-the-art SLAM approaches that leverage advanced scene representations, focusing on their localization components. Our comparison includes recent methods utilizing 3D Gaussian Splatting (3DGS) and Neural Radiance Fields (NeRF) for mapping, as well as established classical techniques. Specifically, we consider RTG-SLAM [5], which employs ICP for pose estimation within a 3DGS framework, and GS-ICP-SLAM [6], which utilizes Generalized-ICP [41] for alignment. We also include Gaussian-SLAM [7], which adapts Open3D [42] RGB-D Odometry, combining colored point cloud alignment [43] with an energy-based visual odometry approach [44]. To broaden our comparison, we incorporate ORBEE-SLAM [45], a NeRF-based approach built upon ORB-SLAM2 [46]. Additionally, we include ORB-SLAM3 [10] as a reference baseline, representing well-established feature-based methods. This selection enables a comprehensive evaluation of our approach against both cutting-edge neural implicit representation-based methods and classical techniques, with a focus on localization performance in the context of advanced mapping capabilities.

4.2 Localization Evaluation

To mitigate long-term drift accumulation, we focus on evaluating pose estimation between consecutive frames. For each frame, we initialize the pose using the ground truth pose of the previous frame.

Table 1: Replica[39] (ATE RMSE ↓[cm]).

Methods	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
RTG-SLAM*[5]	0.380	0.530	0.380	0.450	0.350	0.240	0.360	0.330	0.430
GS-ICP-SLAM*[6]	3.090	1.370	4.700	1.470	8.480	2.040	2.580	1.110	2.940
Gaussian-SLAM*[7]	1.060	0.970	1.310	1.070	0.880	1.000	1.060	1.100	1.130
ORB-SLAM3*[10]	0.630	0.710	0.700	0.520	0.570	0.550	0.580	0.720	0.630
Ours	**0.016**	**0.015**	**0.013**	**0.021**	**0.011**	**0.009**	**0.018**	**0.020**	**0.019**

The results presented in **Table 1.** demonstrate the exceptional performance of our proposed method on the Replica[39] dataset. Our approach achieves state-of-the-art camera pose estimation accuracy between consecutive frames, even in scenarios with significant camera movement. Notably, our method reduces the Average Trajectory Error (ATE) to the centimeter level (10^-2 cm), showcasing a substantial improvement over existing techniques. This remarkable precision underscores the effectiveness of our algorithm in handling challenging camera motions and maintaining accurate localization.

Table 2: Replica[39] (AAE RMSE ↓[°]).

Methods	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
RTG-SLAM*[5]	0.380	0.530	0.380	0.450	0.350	0.240	0.360	0.330	0.430
GS-ICP-SLAM*[6]	3.090	1.370	4.700	1.470	8.480	2.040	2.580	1.110	2.940
Gaussian-SLAM*[7]	1.060	0.970	1.310	1.070	0.880	1.000	1.060	1.100	1.130
ORB-SLAM3*[10]	0.630	0.710	0.700	0.520	0.570	0.550	0.580	0.720	0.630
Ours	**0.009**	**0.007**	**0.008**	**0.010**	**0.009**	**0.009**	**0.011**	**0.009**	**0.011**

Table 2. presents the Absolute Angular Error (AAE) RMSE in degrees for various methods on the Replica dataset. Our approach demonstrates remarkably low rotational errors, even in challenging scenarios with purely rotational camera motions. This superior performance can be attributed to the inherent characteristics of our camera model and pose estimation algorithm. Unlike point cloud alignment-based methods such as RTG-SLAM[5], GS-ICP-SLAM[6], and Gaussian-SLAM[7], which solve for optimal poses from a spatial perspective, our approach leverages the camera's intrinsic rotational properties. By utilizing planar projections, which are inherently more sensitive to rotations than spatial transformations, our method achieves significantly higher accuracy in estimating angular displacements.

Table 3: TUM[40] (ATE RMSE ↓[cm]).

Methods	Avg.	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/off.
RTG-SLAM*[5]	4.840	3.700	7.100	7.500	2.900	3.000
GS-ICP-SLAM*[6]	6.910	2.530	6.830	21.490	1.170	2.520
Gaussian-SLAM*[7]	1.980	1.600	2.200	4.700	0.400	**1.000**
ORB-SLAM3*[10]	15.870	4.260	4.990	34.490	31.730	3.870
Ours	**0.810**	**0.930**	**1.010**	**0.670**	**0.250**	1.200

Table 3. presents the ATE RMSE in centimeters for various methods on the TUM-RGBD dataset [40]. This dataset provides a more challenging evaluation environment, as it more closely resembles real-world conditions with increased noise and environmental complexity compared to the Replica dataset [39]. As a result, the performance of our method, while still competitive, shows some variability across different sequences. This variability underscores the challenges posed by real-world data and highlights areas for potential future improvements in our algorithm's robustness to noise and environmental factors.

Table 4. expands upon the previous comparison by including additional state-of-the-art SLAM systems, namely Vox-Fusion and Point-SLAM, alongside the previously mentioned methods. This comprehensive comparison on the TUM-

Table 4: TUM[40] (AAE RMSE ↓[°]).

Methods	Avg.	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/off.
RTG-SLAM*[5]	4.840	3.700	7.100	7.500	2.900	3.000
GS-ICP-SLAM*[6]	6.910	2.530	6.830	21.490	1.170	2.520
Gaussian-SLAM*[7]	1.980	1.600	2.200	4.700	**0.400**	1.000
ORB-SLAM3*[10]	15.870	4.260	4.990	34.490	31.730	3.870
Ours	**0.980**	**1.130**	**1.270**	**0.910**	0.790	**0.810**

RGBD dataset [40] provides a broader context for evaluating our method's performance against a diverse range of approaches.

Our proposed method demonstrates competitive performance across various sequences, achieving an average ATE RMSE of 5.48 cm. This places our approach in the upper tier of performance among the compared methods. Notably, our method outperforms ORB-SLAM3 [10], Vox-Fusion, and Point-SLAM in terms of average error, showcasing its robustness and accuracy in real-world scenarios.

While Gaussian-SLAM [7] achieves the lowest average error, our method shows comparable or superior performance in several sequences. For instance, in the challenging fr1/room sequence, our approach (11.13 cm) significantly outperforms Gaussian-SLAM (4.70 cm), demonstrating its effectiveness in complex environments.

It's worth noting that our method's performance is particularly strong in sequences with varied camera motions and environmental complexities. This suggests that our approach effectively balances accuracy across different types of scenes and camera movements, a crucial attribute for real-world SLAM applications.

The results on both the Replica and TUM-RGBD datasets collectively demonstrate the efficacy of our proposed method. While achieving state-of-the-art performance on the more controlled Replica dataset, our approach also shows robust and competitive performance in the more challenging real-world scenarios presented by the TUM-RGBD dataset. This comprehensive evaluation underscores the potential of our method for accurate and reliable SLAM in diverse environments.

5 Conclusion

In this paper, we have presented GSplatLoc, a novel camera localization method that leverages the differentiable nature of 3D Gaussian splatting for efficient and accurate pose estimation. Our approach demonstrates significant improvements in pose accuracy, particularly in rotational precision, compared to existing point cloud registration algorithms and state-of-the-art SLAM methods. By utilizing a volumetric representation of 3D Gaussians and optimizing camera poses through differentiable depth rendering, GSplatLoc achieves rotational errors approaching zero and translational errors within 0.01mm on the Replica dataset, setting a new benchmark for localization accuracy in RGB-D SLAM systems.

The comprehensive evaluations conducted on both the Replica and TUM-RGBD datasets validate the robustness and versatility of our method across various indoor environments and camera motion patterns. GSplatLoc's ability to maintain high accuracy in challenging scenarios, including purely rotational movements and complex real-world environments, underscores its potential for enhancing the overall performance of dense mapping SLAM systems.

While our current implementation focuses on frame-to-frame pose estimation to mitigate long-term drift, future work could explore integrating our method into a full SLAM pipeline with loop closure and global optimization. Additionally, extending GSplatLoc to handle outdoor scenes, dynamic objects, and varying lighting conditions would further broaden its applicability. Improving the method's robustness to sensor noise and environmental factors in real-world scenarios also presents an interesting avenue for future research. As the field of 3D scene representation and rendering continues to evolve, incorporating more advanced Gaussian splatting techniques or hybrid representations could potentially lead to even more accurate and efficient localization methods.

[1] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 2100–2106. doi: 10.1109/IROS.2013.6696650.

[2] 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," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, Jan. 2022, doi: 10.1145/3503250.

[3] 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*, 2023, pp. 18433–18444. Accessed: Aug. 22, 2024. [Online]. Available: http://openaccess.thecvf.com/content/ICCV2023/html/Sandstrom_Point-SLAM_Dense_Neural_Point_Cloud-based_SLAM_ICCV_2023_paper.html

[4] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023, doi: 10.1145/3592433.

[5] Z. Peng *et al.*, "RTG-SLAM: Real-time 3D Reconstruction at Scale using Gaussian Splatting." Accessed: Jul. 16, 2024. [Online]. Available: http://arxiv.org/abs/2404.19706

[6] S. Ha, J. Yeon, and H. Yu, "RGBD GS-ICP SLAM." Accessed: May 23, 2024. [Online]. Available: http://arxiv.org/abs/2403.12550

[7] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, “Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting.” Accessed: Jun. 09, 2024. [Online]. Available: http://arxiv.org/abs/2312.10070

[8] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017, doi: 10.1109/TPAMI.2017.2658577.

[9] C. Forster, M. Pizzoli, and D. Scaramuzza, “SVO: Fast semi-direct monocular visual odometry,” in *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2014, pp. 15–22. doi: 10.1109/ICRA.2014.6906584.

[10] 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, doi: 10.1109/TRO.2021.3075644.

[11] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, “Deepfactors: Real-time probabilistic dense monocular slam,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020, doi: 10.1109/LRA.2020.2965415.

[12] 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, Accessed: Jun. 15, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/89fcd07f20b6785b92134bd6c1d0fa42-Abstract.html

[13] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, “BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration,” *ACM Trans. Graph.*, vol. 36, no. 4, p. 1, Aug. 2017, doi: 10.1145/3072959.3054739.

[14] R. A. Newcombe *et al.*, “Kinectfusion: Real-time dense surface mapping and tracking,” in *2011 10th IEEE international symposium on mixed and augmented reality*, Ieee, 2011, pp. 127–136. doi: 10.1109/ISMAR.2011.6092378.

[15] V. A. Prisacariu *et al.*, “A Framework for the Volumetric Integration of Depth Images.” Accessed: Sep. 03, 2024. [Online]. Available: http://arxiv.org/abs/1410.0925

[16] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, “Real-time large-scale dense RGB-D SLAM with volumetric fusion,” *The International Journal of Robotics Research*, vol. 34, no. 4–5, pp. 598–626, Apr. 2015, doi: 10.1177/0278364914551008.

[17] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, “Real-time 3d reconstruction in dynamic scenes using point-based fusion,” in *2013 International Conference on 3D Vision-3DV 2013*, IEEE, 2013, pp. 1–8. doi: 10.1109/3DV.2013.9.

[18] T. Schops, T. Sattler, and M. Pollefeys, “Bad slam: Bundle adjusted direct rgb-d slam,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 134–144. Accessed: Sep. 03, 2024. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Schops_BAD_SLAM_Bundle_Adjusted_Direct_RGB-D_SLAM_CVPR_2019_paper.html

[19] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, “ElasticFusion: Dense SLAM without a pose graph.” in *Robotics: Science and systems*, Rome, Italy, 2015, p. 3. doi: 10.15607/RSS.2015.XI.001.

[20] 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*, 2021, pp. 6229–6238. Accessed: Aug. 10, 2024. [Online]. Available: http://openaccess.thecvf.com/content/ICCV2021/html/Sucar_iMAP_Implicit_Mapping_and_Positioning_in_Real-Time_ICCV_2021_paper.html

[21] 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*, 2023, pp. 17408–17419. Accessed: Sep. 03, 2024. [Online]. Available: http://openaccess.thecvf.com/content/CVPR2023/html/Johari_ESLAM_Efficient_Dense_SLAM_System_Based_on_Hybrid_Representation_of_CVPR_2023_paper.html

[22] Z. Zhu *et al.*, “Nice-slam: Neural implicit scalable encoding for slam,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12786–12796. Accessed: Sep. 03, 2024. [Online]. Available: http://openaccess.thecvf.com/content/CVPR2022/html/Zhu_NICE-SLAM_Neural_Implicit_Scalable_Encoding_for_SLAM_CVPR_2022_paper.html

[23] Z. Zhu *et al.*, “NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM,” in *2024 International Conference on 3D Vision (3DV)*, Mar. 2024, pp. 42–52. doi: 10.1109/3DV62453.2024.00096.

[24] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, “Semanticfusion: Dense 3d semantic mapping with convolutional neural networks,” in *2017 IEEE International Conference on Robotics and automation (ICRA)*, IEEE, 2017, pp. 4628–4635. doi: 10.1109/ICRA.2017.7989538.

[25] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, “Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3504–3515. Accessed: Sep. 03, 2024. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2020/html/Niemeyer_Differentiable_Volumetric_Rendering_Learning_Implicit_3D_Representations_Without_3D_Supervision_CVPR_2020_paper.html

[26] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, “Plenoxels: Radiance fields without neural networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5501–5510. Accessed: Sep. 03, 2024. [Online]. Available: http://openaccess.thecvf.com/content/CVPR2022/html/Fridovich-Keil_Plenoxels_Radiance_Fields_Without_Neural_Networks_CVPR_2022_paper.html

[27] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, “Neural sparse voxel fields,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 15651–15663, 2020, Accessed: Sep. 03, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/b4b758962f17808746e9bb832a6fa4b8-Abstract.html

[28] C. Sun, M. Sun, and H.-T. Chen, “Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5459–5469. Accessed: Sep. 03, 2024. [Online]. Available: http://openaccess.thecvf.com/content/CVPR2022/html/Sun_Direct_Voxel_Grid_Optimization_Super-Fast_Convergence_for_Radiance_Fields_Reconstruction_CVPR_2022_paper.html

[29] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–15, Jul. 2022, doi: 10.1145/3528223.3530127.

[30] L. Keselman and M. Hebert, “Approximate Differentiable Rendering with Algebraic Surfaces,” in *Computer Vision – ECCV 2022*, vol. 13692, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., Cham: Springer Nature Switzerland, 2022, pp. 596–614. doi: 10.1007/978-3-031-19824-3_35.

[31] A. Wang, P. Wang, J. Sun, A. Kortylewski, and A. Yuille, “VoGE: A Differentiable Volume Renderer using Gaussian Ellipsoids for Analysis-by-Synthesis.” Accessed: Sep. 03, 2024. [Online]. Available: http://arxiv.org/abs/2205.15401

[32] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, “Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis.” Accessed: Sep. 03, 2024. [Online]. Available: http://arxiv.org/abs/2308.09713

[33] G. Wu *et al.*, “4d gaussian splatting for real-time dynamic scene rendering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20310–20320. Accessed: Sep. 03, 2024. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024/html/Wu_4D_Gaussian_Splatting_for_Real-Time_Dynamic_Scene_Rendering_CVPR_2024_paper.html

[34] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng, “DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation.” Accessed: Sep. 03, 2024. [Online]. Available: http://arxiv.org/abs/2309.16653

[35] T. Yi *et al.*, “GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models.” Accessed: Sep. 03, 2024. [Online]. Available: http://arxiv.org/abs/2310.08529

[36] V. Ye and A. Kanazawa, “Mathematical Supplement for the $\texttt{gsplat}$ Library.” Accessed: Jun. 29, 2024. [Online]. Available: http://arxiv.org/abs/2312.02121

[37] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, “EWA splatting,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 223–238, 2002, doi: 10.1109/TVCG.2002.1021576.

[38] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, “Design of an image edge detection filter using the Sobel operator,” *IEEE Journal of solid-state circuits*, vol. 23, no. 2, pp. 358–367, 1988, doi: 10.1109/4.996.

[39] J. Straub *et al.*, “The Replica Dataset: A Digital Replica of Indoor Spaces.” Accessed: Aug. 10, 2024. [Online]. Available: http://arxiv.org/abs/1906.05797

[40] 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*, IEEE, 2012, pp. 573–580. doi: 10.1109/IROS.2012.6385773.

[41] A. Segal, D. Haehnel, and S. Thrun, “Generalized-icp.” in *Robotics: Science and systems*, Seattle, WA, 2009, p. 435. doi: 10.15607/RSS.2009.V.021.

[42] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A Modern Library for 3D Data Processing.” Accessed: Aug. 20, 2024. [Online]. Available: `http://arxiv.org/abs/1801.09847`

[43] J. Park, Q.-Y. Zhou, and V. Koltun, “Colored point cloud registration revisited,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 143–152. Accessed: May 23, 2024. [Online]. Available: `http://openaccess.thecvf.com/content_iccv_2017/html/Park_Colored_Point_Cloud_ICCV_2017_paper.html`

[44] F. Steinbrücker, J. Sturm, and D. Cremers, “Real-time visual odometry from dense RGB-D images,” in *2011 IEEE international conference on computer vision workshops (ICCV Workshops)*, IEEE, 2011, pp. 719–722. doi: 10.1109/ICCVW.2011.6130321.

[45] C.-M. Chung *et al.*, “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)*, IEEE, 2023, pp. 9400–9406. doi: 10.1109/ICRA48891.2023.10160950.

[46] R. Mur-Artal and J. D. Tardós, “Orb-Slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017, doi: 10.1109/TRO.2017.2705103.