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

https://github.com/Atticuszz/GsplatLoc

Atticus Zhou, Atticus Zhou, Atticus Zhou, Atticus Zhou

August 12, 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 reprojection 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

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 reprojection 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.

2 Related Work

Accurate visual localization commonly relies on estimating correspondences between 2D pixel positions and 3D scene coordinates. Such approaches detect, describe [7,49], and match [32,46,48,73,81,96] local features, maintain an explicit sparse 3D representation of the environment, and sometimes leverage image retrieval [33,86] to scale to large scenes [32,57,70,75,82,87]. Recently, many of these components have been learned with great success [2,23,25,58,60,65,67,71,95], but often independently and not end-to-end due to the complexity of such systems. Here we introduce a simpler alternative to feature matching, finally enabling stable end-to-end training. Our solution can learn more powerful priors than individual blocks, yet remains highly flexible and interpretable. End-to-end learning for localization has recently received much attention. Common approaches encode the scene into a deep network by regressing from an input image to an absolute pose [35,37,59,66,90] or 3D scene coordinates [9,13,16,17,80]. Pose regression lacks geometric constraints and thus does not generalize well to novel viewpoints or appearances [76,78], while coordinate regression is more robust. Both do not scale well due to the limited network capacity [11,82] and require for each new scene either costly retraining or adaptation [16,17]. ESAC [11] improves the scalability by training an ensemble of regressors, each specialized in a scene subset, but is still significantly less accurate than feature-based methods in larger environments. Differently, some approaches regress a camera pose relative to one or more training images [5,24,42,97], often after an explicit retrieval step. They do no memorize the scene geometry and are thus scene-agnostic, but, similar to absolute regressors, are less accurate than feature-based methods [76,97]. Closer to ours, SANet [93] takes the scene representation out of the network by regressing 3D coordinates from an input 3D point cloud. Critically, all top-performing learnable approaches are at least trained per-dataset, if not per-scene, and are limited to small environments [37,80]. In this work we demonstrate the first end-to-end learnable network that generalizes across scenes, including from outdoor to indoor, and that delivers performance competitive with complex pipelines on large real-world datasets, thanks to a differentiable pose solver. Learning

camera pose optimization can be tackled by unrolling the optimizer for a fixed number of steps [21,51,53, 83,91,92], computing implicit derivatives [13,15,18,34,68], or crafting losses to mimic optimization steps [88,89]. Multiple works have proposed to learn components of these optimizers [21,51,83], with added complexity and unclear generalization. Some of these formulations optimize reprojection errors over sparse points, while others use direct objectives for (semi-)dense image alignment. The latter are attractive for their simplicity and accuracy, but usually do not scale well. Like their classical counterparts [26,38], they also suffer from a small basin of convergence, limiting them to frame tracking. In contrast, PixLoc is explicitly trained for wide-baseline cross-condition camera pose estimation from sparse measurements (Figure 2). By focusing on learning good features, it shows good generalization yet learns sensible data priors that shape the optimization objective.

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 [1], 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 $(R, t) \in SE(3)$ of a query depth image D_q, where R is the rotation matrix and 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 [1], 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 $\mathbf{\Sigma}_i \in \mathbb{R}^{3\times3}$, 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 $\mathbf{\Sigma}_i$ is then parameterized using $\mathbf{s}_i$ and $\mathbf{q}_i$:

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

where $R(\mathbf{q}_i)$ is the rotation matrix derived from $\mathbf{q}_i$, and $S(\mathbf{s}_i) = \text{diag}(\mathbf{s}_i)$ 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 [1]. The projection of the 3D mean $\boldsymbol{\mu}_i$ to the 2D image plane is given by:

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

where $T_{wc} \in SE(3)$ is the world-to-camera transformation, $P \in \mathbb{R}^{4\times4}$ is the projection matrix [2], 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:

$$\mathbf{\Sigma}_{I,i} = JR_{wc}\mathbf{\Sigma}_i R_{wc}^T J^T$$

where R_{wc} represents the rotation component of T_{wc}, and J is the affine transform as described by [3].

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 $D(p)$ at pixel p is computed as [1]:

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

Here, α_n represents the opacity of the n-th Gaussian at pixel p, computed as:

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

where Δ_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 $\alpha(p)$ for each pixel:

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

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

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

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.

Methods	Avg.	R0	R1	R2				
ICP	0.38	0.53	0.38	0.45	**0.35**	**0.24**	0.36	0.33	**0.43**
Vox-Fusion	3.09	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94
NICE-SLAM	1.06	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13
ESLAM	0.63	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63
Point-SLAM	0.52	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72
SplaTAM	0.36	0.31	0.40	**0.29**	0.47	0.27	0.29	0.32	0.55
Ours	**0.10**	**0.20**	**0.30**	0.40	0.50	0.60	**0.20**	**0.30**	0.50

Table 1: Replica[4] (ATE RMSE ↓[cm])

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:

$$L = \lambda_1 \cdot L_{\text{depth}} + \lambda_2 \cdot L_{\text{contour}}$$

where L_{depth} represents the L1 loss for depth accuracy, and L_{contour} focuses on the alignment of depth contours or edges. Specifically:

$$L_{\text{depth}} = \sum_{i \in M} |D_i^{\text{rendered}} - D_i^{\text{observed}}|$$

$$L_{\text{contour}} = \sum_{j \in M} |\nabla D_j^{\text{rendered}} - \nabla D_j^{\text{observed}}|$$

Here, M denotes the rendered alpha mask, indicating which pixels are valid for comparison. Both L_{depth} and L_{contour} 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_{contour} is computed using the Sobel operator [6], 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 images, 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:

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

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 Experiments

Datasets: The Replica dataset [4] comprises high-quality 3D reconstructions of a variety of indoor scenes. We utilize the publicly available dataset collected by Sucar et al . [7], which provides trajectories from an RGBD sensor. Further, we demonstrate that our framework achieves SOTA results on real-world data by using the TUM-RGBD [5]. The poses for TUM-RGBD were captured using an external motion capture system.

Metrics: We quantitatively evaluate reconstruction quality using different 3D metrics. Given 3D triangle meshes, we compute mapping Accuracy [cm], Completion [cm], and Completion Ratio [<5cm %]. Following NICE-SLAM [58], we discard unobserved regions that are not in any viewpoints. As for tracking performance, we measure ATE RMSE [41] for estimated trajectories **Baselines:** We primarily consider state-of-the-art NeRF-SLAM works, including NICE-SLAM [58], Co-SLAM [47], Point-SLAM [34], and Vox-Fusion [53], as baselines. For a fair comparison, we reproduced all results from these baselines and reported their reconstruction performance with the same evaluation mechanism. We also add some concurrent manuscripts such as GS-SLAM [51] and SplaTAM [20] for reference, and we directly report the results in their papers. # Conclusion

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 reprojection 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] 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.

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

[3] 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.

[4] 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

[5] 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.

[6] 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.

[7] 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

Methods	Avg.	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/off.
Kintinous	4.84	3.70	7.10	7.50	2.90	3.00
ElasticFusion	6.91	2.53	6.83	21.49	1.17	2.52
ORB-SLAM2	**1.98**	**1.60**	**2.20**	**4.70**	**0.40**	**1.00**
NICE-SLAM	15.87	4.26	4.99	34.49	31.73	3.87
Vox-Fusion	11.31	3.52	6.00	19.53	1.49	26.01
Point-SLAM	8.92	4.34	4.54	30.92	1.31	3.48
SplaTAM	5.48	3.35	6.54	11.13	1.24	5.16

Table 2: TUM[5] (ATE RMSE ↓[cm])