3D Gaussian Splatting Registration

Liyuan Zhu
Stanford University
liyzhu@stanford.edu

Abstract

Photorealistic scene reconstruction is paramount for many AR/XR applications and it has been made possible by 3D Gaussian Splatting (3DGS). However, the incremental reconstruction of 3DGS remains a challenging problem. We propose a method to align two sets of 3D Gaussians that are captured and optimized separately. We consider 3DGS registration as the problem of view localization and leverage the fast rasterization and multi-view observation for robust and accurate registration of 3D Gaussian splatting. We demonstrate the effectiveness of our method on pairwise registration benchmark, as well as the downstream application of loop closure in Gaussian Splatting SLAM.

1. Introduction

Neural fields [26, 30, 37, 44] have become one of the trendiest 3D representations in 3D vision due to their simple design and state-of-the-art performance. Neural fields have introduced many important applications, *e.g.*, 3D/4D scene reconstruction [28, 30, 53, 57], novel view synthesis [3, 15, 26] and generative modelling [21, 22, 27]. Neural fields usually encode the scene information (*i.e.*, color, geometry and etc.) into a neural network, *i.e.*, Multi-layer Perceptron (MLP) or Triplanes, and reconstruct the 3D scene using volume rendering. However, the heavy compute and slow speed of such methods are not optimal for real-time rendering and other down-stream applications.

Kerbl *et al*. [18] introduced rasterization-based [48] 3D Gaussian Splatting (3DGS), which explicitly represents the scene as a mixture of 3D Gaussians, showing superior rendering speed and comparable performance in novel view synthesis (NVS) w.r.t. neural fields. 3DGS takes as input sparse point clouds and calibrated cameras from Structure-from-Motion (SfM) and optimizes the 3D Gaussians (color, position and opacity) during training. Once trained, it is hard to add new observations to the scene, unless re-do everything *i.e.*, from SfM to optimization. This drawback hinders the process of merging two sets of 3DGS that are captured at different times and optimized separately, under different coordinate systems. We consider merging two sets of unaligned 3DGS an essential problem as it can facilitate many down stream applications, *e.g*. large-scale incremental reconstruction [6, 35] and SLAM [17, 25]. Chen *et al*. [5] propose to register two Neural Radiance Fields (NeRFs) by extracting their explicit surfaces and learning to align them, similar to the methods in point cloud registration [14, 31]. VF-NeRF [36] introduces normalizing flow to NeRF training to determine how well a 3D point is observed by training views, improving the performance NeRF registration. However, DReg-NeRF and VF-NeRF are specially designed for NeRF and thus not applicable to 3DGS representation. Compared to NeRF, 3DGS representation explicitly stores the color and geometry in 3D Gaussians, somewhat equivalent to point cloud representation with additional attributes. In this paper, we try to answer the research question of:

"How to efficiently register two sets of 3D Gaussians?"

To answer this question, we consider 3DGS registration as a mulit-camera relocalization problem by mutually localizing the training views of one 3DGS in the other 3DGS. Our insight is that rendering loss is a good indicator of pose accuracy and we use it to quantify the confidence of camera localization for registration. We showcase its performance on pairwise 3DGS registration, as well as mutli-view registration and its application for loop closure in Gaussian-Splatting SLAM [24] .

2. Related work

Neural scene representation. Neural scene representation has emerged as a pivotal area of research in computer vision and graphics, aiming to capture and reconstruct complex 3D environments using neural networks. One foundational approach in this domain is Neural Radiance Fields (NeRF), which utilizes volumetric rendering to generate high-fidelity 3D scenes from a sparse set of 2D images [26]. NeRF and its extensions have demonstrated remarkable capabilities in synthesizing novel views of scenes with intricate details and realistic lighting [2, 23]. Subsequent advancements include optimization techniques for faster training and inference [11], as well as incorporating multi-modal

supervision such as depth maps and point clouds to improve reconstruction accuracy [10]. Another promising technique is 3D Gaussian splatting, which represents scenes as a collection of 3D Gaussians, allowing for efficient and flexible rendering of complex geometries and textures [18].

Point cloud registration. Point cloud registration is a fundamental task in 3D vision and robotics, aiming to find a rigid transformation between two partially overlapping point clouds in different reference frames. Several hand-crafted 3D feature descriptors have been developed for local feature matching, such as FPFH [33] and SHOT [40]. With advancements in deep learning, methods like 3DMatch [51], PerfectMatch [12], and RPMNet [46] focus on learning-based descriptors. Predator [14] incorporates attention mechanisms [41] to enhance 3D correspondences, especially in low-overlapping regions. Transformer architectures are further refined for superpoint matching in works like [32, 47]. Additionally, methods such as [16, 34, 50] leverage prior information like surface curvature, 2D image overlap, and scene structure. Beyond correspondence matching methods, another approach involves learning equivariant representations to solve the relative pose. Wang *et al*. [42] develop rotation-equivariant descriptors using group equivariant learning [7]. Yu *et al*. [49] propose a rotation-invariant transformer method to handle pose variations in point cloud matching. Zhu *et al*. [56] directly address pairwise rotation using equivariant embeddings.

NeRF registration. Given NeRF is a different scene representation than point clouds and its popularity, the registration of it has become a new research direction. iNeRF [45] proposes a method to perform image-to-NeRF registration by back-propagating the photometric loss through the fixed NeRF weights to optimize camera pose. NeRF2NeRF [13] is a method that aligns two NeRFs using surface fields but it requires user input for initialization. DReg-NeRF [5] proposed an attentaion-based neural network to find the correspondences in two NeRFs' surfaces and tackle NeRF registration as point cloud registration. FV-NeRF [36] introduces the Normalizing Flow [29] to NeRF registration to implicitly quantify how well each point is observed by the training views and then uses the best-viewed points to improve the registration performance.

3. Methodology

We first introduce preliminary of 3D Gaussian Splatting and definition of 3DGS registration. Then we elaborate our complete pipeline in threefold : i) finding the overlapping region. ii) registration by camera pose estimation on 3D Gaussian splatting. iii) fusing multiple viewpoints for robust estimation. (*c.f*. Fig. 1)

Preliminary. Consider two sets of 3D Gaussians $\mathbf{P}$ and $\mathbf{Q}$, and their training views $\mathbf{V_P}$ and $\mathbf{V_Q}$. $(\mathbf{P}, \mathbf{V_P})$ and $(\mathbf{Q}, \mathbf{V_Q})$ are captured and optimized separately.

$$\mathbf{P} = \{\mathbf{p}_\mathrm{i} = (\mathbf{x}, \mathbf{s}, \mathbf{r}, \alpha, \mathbf{c})_\mathrm{i} | \mathrm{i} = 0..M\}, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the position vector, $\mathbf{s} \in \mathbb{R}^3_+$ denotes the scaling vector, $\mathbf{r} \in \mathbb{R}^4$ denotes the rotation, represented in unit quaternion, $\alpha \in [0, 1]$ denotes the opacity and $\mathbf{c}$ the spherical harmonics (SH) coefficients. $\mathbf{Q}$ is defined similarly as $\mathbf{Q} = \{\mathbf{q}_\mathrm{i} = (\mathbf{x}, \mathbf{S}, \mathbf{R}, \alpha, \mathbf{c})_\mathrm{i} | \mathrm{i} = 0..M'\}$. We also define the viewpoints used for training each 3DGS as:

$$\mathbf{V^P} = \{\mathbf{v}^\mathbf{P}_\mathrm{i} = (\mathbf{I}, \mathbf{T_v})_\mathrm{i} | \mathrm{i} = 0..N\} \tag{2}$$

where $\mathbf{I} \in \mathbb{R}^{W \times H \times 3}$ is the image captured at this viewpoint with W, H being its dimensions. $\mathbf{T_v} \in SE(3)$ denotes the rigid transformation from the world reference frame to the camera reference frame. Viewpoints for $\mathbf{Q}$ are defined as $\mathbf{V^Q} = \{\mathbf{v^q}_\mathrm{i} = (\mathbf{I}, \mathbf{T})_\mathrm{i} | \mathrm{i} = 0..N'\}$.

Problem definition. Given $\mathbf{P}$ and $\mathbf{Q}$ in different reference frames, our goal is to recover a rigid transformation $\mathbf{T}^\mathbf{Q}_\mathbf{P} = [\mathbf{R}^\mathbf{Q}_\mathbf{P} | \mathbf{t}^\mathbf{Q}_\mathbf{P}] \in SE(3)$ that aligns $\mathbf{P}$ to $\mathbf{Q}$ and optimize a new 3DGS representation $\mathbf{Q}'$ by fusing $\mathbf{P}$ and $\mathbf{Q}$. We assume $\mathbf{P}$ and $\mathbf{Q}$ have spatial overlap and the scene is rigid.

3.1. Finding the overlap

As pointed out by Huang *et al*. [14], finding the overlap between source $\mathbf{P}$ and target $\mathbf{Q}$ is essential in point cloud registration. Given that 3D Gaussians can be understood as an point cloud with additional attributes, we believe this insight still holds in 3D Gaussian splatting registration. 3DGS is that it not only encodes the scene geometry but also encodes the appearances. Instead of finding the similarity in 3D structure, we find the viewpoints in both 3DGS that share the same visual content, which is closer to image retrieval. Here, we use NetVLAD [1] to training views $\mathbf{V_P}$ and $\mathbf{V_Q}$ and retrieve two image pairs as the cadidate view for later camera localization.

3.2. Camera localization as registration

We consider 3D Gaussian registration as a camera relocalization problem and leverage the multi-camera observation to quantify the uncertainty in camera localization to find the best transformation between $\mathbf{P}$ and $\mathbf{Q}$. We first demonstrate that estimating relative transformation $\mathbf{T}^\mathbf{Q}_\mathbf{P}$ between $\mathbf{P}$ and $\mathbf{Q}$ is equivalent to localizing the viewpoints $\mathbf{V_P}$ in $\mathbf{Q}$. Consider a viewpoint $\mathbf{v}^\mathbf{P}_\mathrm{i}$ and its corresponding image $\mathbf{I}^\mathbf{P}_\mathrm{i}$. After training, we can render image $\hat{\mathbf{I}}^\mathbf{P}_\mathrm{i}$ at $\mathbf{v}^\mathbf{P}_\mathrm{i}$ on $\mathbf{P}$. We first transform all the Gaussians to the camera frame of the viewpoint and render $\hat{\mathbf{I}}^\mathbf{P}_\mathrm{i}$ using differentiable rasterization [18]:

$$\hat{\mathbf{I}}^\mathbf{P}_\mathrm{i} = \Phi\left(\mathbf{T_v}\mathbf{P}\right). \tag{3}$$

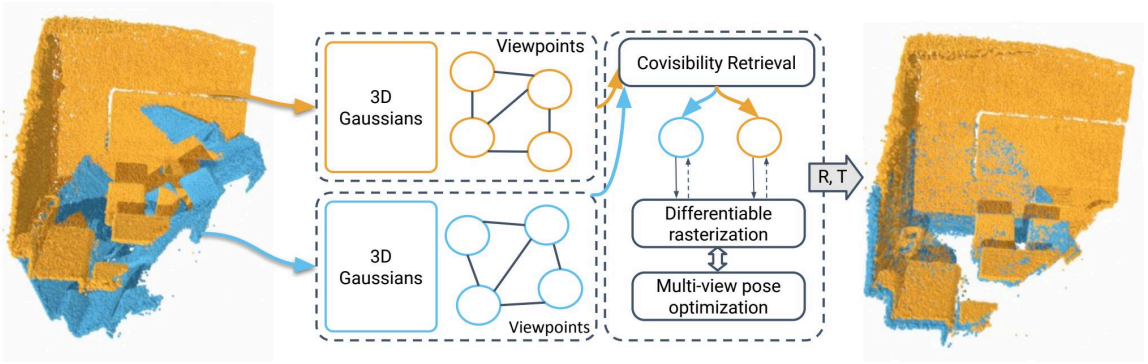

Figure 1. **Method overview**. We represent source 3DGS in blue and target 3DGS in yellow. For simplification, we do not visualize the points with their raw appearance. The arrows demonstrate the flow of the method. We first find the viewpoints in $\mathbf{P}$ and $\mathbf{Q}$ with the most covisibility. Then we use the render the source viewpoints in target 3DGS and back-propagate the loss using differentiable rasterizer of [24] to optimize the pose of the viewpoints. After getting the relative transformation of selected viewpoints, we fuse them as the final estimate, leveraging multiview information.

Here Φ denotes the rendering function. We only multiply $\mathbf{T_v}$ with the centers of $\mathbf{P}$. Then we insert an identity matrix $(\mathbf{T_P^Q}^{-1}\mathbf{T_P^Q})$ into Eq. (3) and get

$$\hat{\mathbf{I}}_{\mathrm{i}}^{\mathbf{P}} = \Phi\left((\mathbf{T_v^P T_P^Q}^{-1})(\mathbf{T_P^Q P})\right) \tag{4}$$

As shown above, to get the same rendering $\hat{\mathbf{I}}_{\mathrm{i}}^{\mathbf{P}}$ in the reference frame of $\mathbf{Q}$, the new viewpoint pose is $(\mathbf{T_v^P T_P^Q}^{-1})$. Therefore, the problem is now turned into finding the viewpoint in $\mathbf{Q}$ that generates the same rendering as in $\mathbf{P}$.

Camera pose optimization. Our goal is to find the best renders in $\mathbf{Q}$ for all viewpoints in $\mathbf{P}$. Here, we keep the parameters of $\mathbf{Q}$ fixed and optimize the camera poses $\{\mathbf{T_v^{P,i}} \in SE(3) | \mathrm{i} = 0..M\}$ to find the most similar render by minimizing the rendering loss [25]:

$$\min_{\substack{\mathbf{T_P^{Q,i}} \in SE(3), \mathbf{Q}, \\ \forall i \in [0,M]}} \sum_{i=0}^{M} E_{pho}^{\mathrm{i}} + E_{geo}^{\mathrm{i}}\,, \tag{5}$$

where E_{pho}^{k} is the photometric loss:

$$E_{pho} = \left\|\Phi(\mathbf{T_v^P Q}) - \mathbf{I}\right\|_1\,, \tag{6}$$

and E_{geo}^{k} is the geometric loss:

$$E_{geo} = \left\|\Phi^d(\mathbf{T_v^P Q}) - \mathbf{I}^d\right\|_1\,, \tag{7}$$

Φ^d renders depth image at given viewpoint and $\mathbf{I}^d$ is the rendered depth at its training view after optimization. We use the differentiable rasterizer implemented in [25], which provides analytical Jacobian of $SE(3)$ w.r.t. 3D Gaussians.

Optimization setting. We optimize each viewpoint for 100 steps and select the pose $\mathbf{T_v^{Q,i}}$ with the smallest rendering loss $\mathcal{L}^i$ as its camera pose in the target frame. We use Adam optimizer [19] and set the learning rate to be 0.003. After the per-viewpoint optimization, we get the poses of each viewpoint in both $\mathbf{P}$ and $\mathbf{Q}$. Then we can get the relative transformation for registration $\hat{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q},\mathrm{i}} = \mathbf{T_v^{Q,i}}^{-1}\mathbf{T_v^{P,i}}$.

3.3. Fusing multi-camera

We get multiple estimates $\{(\hat{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q},\mathrm{i}}, \mathcal{L}_{\mathrm{i}}) | \mathrm{i} = 0..M\}$ and the corresponding rendering loss for Gaussian Splatting registration. Now we proceed to fuse them and output the final estimate. Our key insight is that the rendering loss of a viewpoint is an approximate of the uncertainty in its pose estimation. We map the rendering loss of all viewpoints into a probabilistic distribution of camera pose:

$$\sigma(\mathcal{L}_{\mathrm{i}}) = \frac{1/\mathcal{L}_{\mathrm{i}}}{\sum_{j=1}^{K} 1/\mathcal{L}_j} \quad for\ \mathrm{i} = 0, 1, \ldots, M. \tag{8}$$

Then we compute the mode of the distribution as the final transformation:

$$\hat{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}} = \sum_{i=0}^{M} \sigma(\mathcal{L}_{\mathrm{i}}) \hat{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q},\mathrm{i}} \tag{9}$$

To keep $\mathbf{T_P^Q}$ on the $SE(3)$ manifold in Eq. (9), we average the estimated transformations in $\mathbf{T_P^Q} = [\mathbf{R} \in SO(3) | \mathbf{t} \in \mathbb{R}^3]$ separately. For $\mathbf{R}$, we apply rotation averaging [4]. Here, we complete the methodology for pairwise 3D Gaussian splatting registration. Next, we showcase one downstream application of 3DGS registration in Gaussian-Splatting-based SLAM. This is an application of our GS registration method for multi-view 3DGS registration.

3.4. Loop closure in SLAM

In the SLAM system, the errors of motion estimation accumulate, causing the pose estimate to drift over time. When the sensor moves back to a previously visited location, it can use this information to correct its position estimate and adjust the map accordingly. Loop closure includes closure detection, pose graph construction and optimization. We use MonoGS [24] as the Gaussian Splatting SLAM system and follow Loopy-SLAM [20]'s method for loop edge detection and pose graph optimization. The only change we apply to MonoGS is to make it create submaps every 15 keyframes. Then we apply our 3DGS registration on the 3DGS submaps as the loop edges of the pose graph.

4. Experimental results

We evaluate our method on Objaverse and Replica, at object-level and scene-level, respectively.

4.1. Datasets

We evaluate our task on two datasets: Objaverse [9] and Scannet [8]. We also evaluate our method at scene-level, on Scannet [8]. We use the RGB-D camera trajectories in Scannet and run the Gaussian Splatting SLAM (MonoGS) [25] to generate submaps with 3D Gaussians and apply our registration to the submaps.

Objaverse. Follow [5], we use 1700+ objects from Objaverse [9], which is a large-scale 3D object dataset originally designed for text-to-3D tasks. There are 30+ categories and each categories has 40-80 objects. For each object, it contains 120 rendered images following the camera trajectory generated in [5]. The 120 images are then clustered in two parts using KMeans. With the images and camera poses, we use the 3DGS implementation [18] to get the two representations for each object.

Replica. Replica [38] is a dataset of high quality reconstruction of various indoor spaces. It includes detailed geometry, high-resolution textures, glass and mirror surface information, and both semantic and instance segmentation. The dataset is designed for machine learning and AI training purposes, compatible with the AI Habitat framework. Here we follow NICE-SLAM [57] and evaluate on 7 scenes in Replica.

4.2. Baseline methods

We compare our method with the following baselines: Fast global registration [54] (FGR) and DReg-NeRF [5]. FGR is a descriptor-based method that iteratively refine the matches on surface and remove false matches. DReg-NeRF is a learning based registration method for NeRF. To compare with DReg-NeRF, we use the same viewpoints and train a NeRF from them and apply same transformation to the NeRFs for fair comparison.

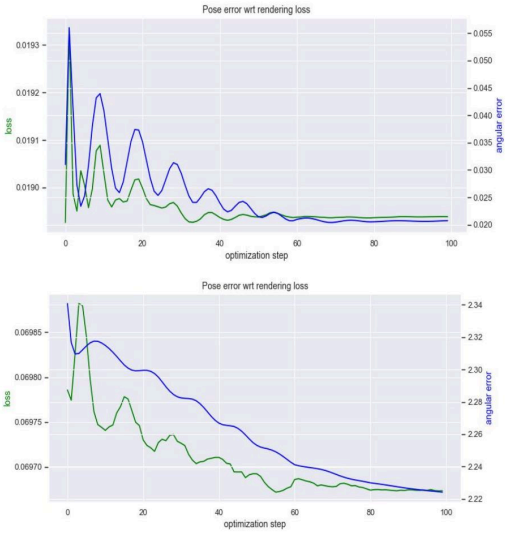

Figure 2. **Rendering loss v.s. iteration for camera localization.**

4.3. Evaluation metrics

Rotation error (RE). It measures the geodesic distance between two rotations:

$$\mathrm{RE} = \arccos\left(\frac{\mathrm{trace}\left(\mathbf{R}^{\mathrm{T}}\overline{\mathbf{R}}\right) - 1}{2}\right), \qquad (10)$$

where $\mathbf{R}$ denotes the predicted rotation matrix and $\overline{\mathbf{R}}$ the ground truth. $\mathrm{trace}()$ denotes the trace of a matrix.

Registration recall (RR). It is the fraction of rotation errors smaller than a threshold. We use as thresholds $\mathrm{RE} < 5°$ in our experiments. Follow [39, 57], we use ATE RMSE to evaluate.

Rendering. The rendering quality after registration are evaluated by peak signal-to-noise ratio (PSNR), SSIM [43] and LPIPS [52].

4.4. Object-level registration

We first show that there is a strong correlation between the rendering loss and rotation error in pose estimation. As shown in Fig. 3, as optimization goes on with more iterations and rendering loss decreases, the angular error in relative pose estimation decreases.

We use registration recall, MeanRE and MeanTE to evaluate the registration results on Objaverse. Fig. 4 is a qualitative result of the object-level registration. It showcases

Figure 3. **Qualitative results on scene-level 3D Gaussian Splatting Registration**.

Figure 4. Qualitative result on Objaverse. Left (input view), middle (estimated view) and right (target view).

Method	RR [RE<5°] ↑	MeanRE [°] ↓	MeanTE ↓
FGR [55]	18.18	**2.45**	5.35
DRegNeRF [5]	20.45	2.67	1.87
Ours	**25.06**	3.06	**0.87**

Table 1. **Registration performance on Objaverse test set**.

that the proposed method can align two sets of 3DGS in the same reference frame, generating very similar renders from the same viewpoint. Our method outperforms the two baseline methods on Objaverse dataset on registration recall (RR) and MeanTE. Our method is also faster than the two baseline methods: GS-Registration takes around 1 second to register a pair while FGR and DRegNeRF take more than 2 seconds. In this experiment, we use the ground truth camera pose to train the 3D Gaussian Splatting for each object.

4.5. Scene-level registration

We also validate our method on scene-level dataset Replica. To simulate the real-world conditions, we do not use ground truth camera poses anymore, instead we estimate the camera pose using a Gaussian Splatting SLAM system [24]. We create a set of 3DGS from GS-SLAM every 15 keyframes created by the tracking system as a submap. We generate the ground truth transformation using GT camera poses. Gaussian Splatting SLAM only includes odometry and mapping, the loop closure detection and correction are missing. For loop closure detection, we use NetVLAD [1] and follow the standard method in [20]. To construct the pose graph, there are two types of edges: odometry edges and loop edges. For odometry edges, we use the estimates from Gaussian Splatting SLAM front-end

	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Mean
MonoGS(submap)	0.53	0.41	0.36	0.55	0.63	0.4	0.45	1.9	0.65
MonoGS(submap) + LC (FGR [55])	1.44	1.01	1.13	2.57	10.6	26.13	1.68	2.29	5.86
MonoGS(submap) + LC (ours-S)	0.4	0.44	0.47	0.31	0.54	0.35	0.45	0.68	0.46
MonoGS(submap) + LC (ours-M)	0.42	0.39	0.48	0.27	0.59	0.24	0.36	0.7	0.43

Table 2. **Absolute trajectory error (cm) on Replica dataset.** S denotes single viewpoint and M multiple viewpoints.

tracking system. For loop edges, once we detect the closure submap pair using NetVLAD image descriptors, we apply our 3DGS registration to detected loop edges. Here, our 3DGS registration serves as a basic module for loop closure in SLAM. We evaluate the ATE of MonoGS [24] submap system before and after loop closure in Tab. 2. We also directly apply fast global registration (FGR) [54] to loop edges as a baseline. From Tab. 2, we can see that the conventional registration algorithm for point cloud representation does not work very well because the centers of 3D Gaussians do not necessarily lie on the surface thus introducing error to the pose graph. As an ablation study, we compare the loop closure results with 3DGS registration using a single view or multiple viewpoints. Our full method using multi-view images has the best performance, demonstrating the effectiveness of design choices. Fig. 5 is the qualitative result of depth rendering after registering the source and target 3DGS, showing good alignment at the pixel level.

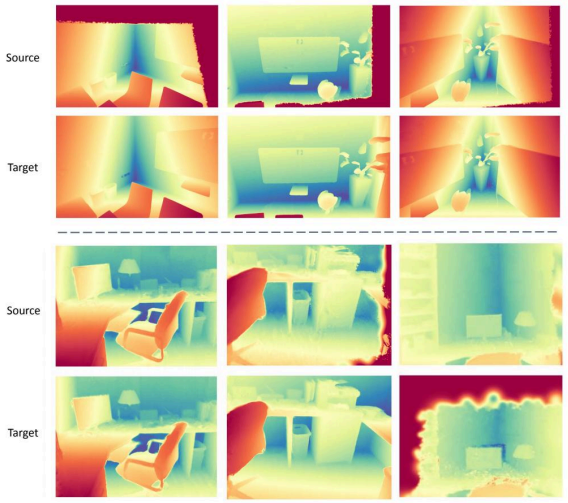

Figure 5. **Qualitative results on scene-level registration**.

5. Conclusion

In this project, we propose a method to register two sets of unaligned 3D Gaussian Splatting, leveraging the training viewpoints, fast differentiable rasterization and visibility selection for overlapping regions. We validate our method on both object-level and scene-level datasets. We finally showcase a potential application of our 3DGS registration on SLAM loop closure.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016.

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023.

[4] Romain Brégier. Deep regression on manifolds: a 3D rotation case study. 2021.

[5] Yu Chen and Gim Hee Lee. Dreg-nerf: Deep registration for neural radiance fields. In *ICCV*, 2023.

[6] Yu Chen and Gim Hee Lee. Scalar-nerf: Scalable large-scale neural radiance fields for scene reconstruction. *arXiv:2311.16657*, 2023.

[7] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, 2016.

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023.

[10] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, pages 12882–12891, 2022.

[11] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14346–14355, 2021.

[12] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *CVPR*, 2019.

[13] Lily Goli, Daniel Rebain, Sara Sabour, Animesh Garg, and Andrea Tagliasacchi. nerf2nerf: Pairwise registration of neural radiance fields. In *ICRA*. IEEE, 2023.

[14] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *CVPR*, 2021.

[15] Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, and Or Litany. Neural lidar fields for novel view synthesis. In *ICCV*, 2023.

[16] Shengze Jin, Daniel Barath, Marc Pollefeys, and Iro Armeni. Q-reg: End-to-end trainable point cloud registration with surface curvature. *3DV*, 2023.

[17] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. *CVPR*, 2024.

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Lorenzo Liso, Erik Sandström, Vladimir Yugay, Luc Van Gool, and Martin R. Oswald. Loopy-slam: Dense neural slam with loop closures. In *CVPR*, 2024.

[21] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *NeurIPS*, 2024.

[22] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *ICCV*, 2023.

[23] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, June 2021.

[24] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *arXiv preprint arXiv:2312.06741*, 2023.

[25] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *CVPR*, 2024.

[26] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2021.

[27] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021.

[28] Yue Pan, Xingguang Zhong, Louis Wiesmann, Thorbjörn Posewsky, Jens Behley, and Cyrill Stachniss. Pin-slam: Lidar slam using a point-based implicit neural representation for achieving global map consistency. *arXiv:2401.09101*, 2024.

[29] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 2021.

[30] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020.

[31] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, Slobodan Ilic, Dewen Hu, and Kai Xu. Geotransformer: Fast and robust point cloud registration with geometric transformer. *IEEE TPAMI*, 2023.

[32] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *CVPR*, 2022.

[33] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3d registration. In *ICRA*, 2009.

[34] Sayan Deb Sarkar, Ondrej Miksik, Marc Pollefeys, Daniel Barath, and Iro Armeni. Sgaligner: 3d scene alignment with scene graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21927–21937, 2023.

[35] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. Lamar: Benchmarking localization and mapping for augmented reality. In *ECCV*, 2022.

[36] Leo Segre and Shai Avidan. Vf-nerf: Viewshed fields for rigid nerf registration. *arXiv preprint arXiv:2404.03349*, 2024.

[37] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *NeurIPS*, 2019.

[38] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[39] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012.

[40] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *ECCV*, 2010.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[42] Haiping Wang, Yuan Liu, Zhen Dong, and Wenping Wang. You only hypothesize once: Point cloud registration with rotation-equivariant descriptors. In *ACM Multimedia*, 2022.

[43] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 2004.

[44] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. 2022.

[45] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *(IROS)*, 2021.

[46] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *CVPR*, 2020.

[47] Zi Jian Yew and Gim Hee Lee. REGTR: End-to-end point cloud correspondences with transformers. In *CVPR*, 2022.

[48] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM TOG*, 2019.

[49] Hao Yu, Zheng Qin, Ji Hou, Mahdi Saleh, Dongsheng Li, Benjamin Busam, and Slobodan Ilic. Rotation-invariant transformer for point cloud matching. In *CVPR*, 2023.

[50] Junle Yu, Luwei Ren, Yu Zhang, Wenhui Zhou, Lili Lin, and Guojun Dai. PEAL: Prior-embedded explicit attention learning for low-overlap point cloud registration. In *CVPR*, 2023.

[51] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017.

[52] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[53] Xingguang Zhong, Yue Pan, Cyrill Stachniss, and Jens Behley. 3d lidar mapping in dynamic environments using a 4d implicit neural representation. *CVPR*, 2024.

[54] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV*, 2016.

[55] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV*, 2016.

[56] Minghan Zhu, Maani Ghaffari, and Huei Peng. Correspondence-free point cloud registration with SO(3)-equivariant implicit shape representations. In *CoRL*, 2022.

[57] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *CVPR*, 2022.