SAD-GS: Shape-aligned Depth-supervised Gaussian Splatting

Pou-Chun Kung, Seth Isaacson, Ram Vasudevan, and Katherine A. Skinner
University of Michigan, 2505 Hayward St, Ann Arbor, MI 48109
pckung@umich.edu, sethgi@umich.edu, ramv@umich.edu, kskin@umich.edu

Abstract

This paper proposes SAD-GS, a depth-supervised Gaussian Splatting (GS) method that provides accurate 3D geometry reconstruction by introducing a shape-aligned depth supervision strategy. Depth information is widely used in various GS applications, such as dynamic scene reconstruction, real-time simultaneous localization and mapping, and few-shot reconstruction. However, existing depth-supervised methods for GS all focus on the center and neglect the shape of Gaussians during training. This oversight can result in inaccurate surface geometry in the reconstruction and can harm downstream tasks like novel view synthesis, mesh reconstruction, and robot path planning. To address this, this paper proposes a shape-aligned loss, which aims to produce a smooth and precise reconstruction by adding extra constraints to the Gaussian shape. The proposed method is evaluated qualitatively and quantitatively on two publicly available datasets. The evaluation demonstrates that the proposed method provides state-of-the-art novel view rendering quality and mesh accuracy compared to existing depth-supervised GS methods. A project page is available at https://umautobots.github.io/sad_gs.

1. Introduction

3D Gaussian Splatting (GS) [12] marks a recent paradigm shift in the field of computer vision and novel view synthesis. It offers a powerful way to represent scenes and render novel views without relying on neural networks such as Neural Radiance Fields (NeRF) [15], significantly accelerating both rendering and training. GS has been successfully implemented in various domains, such as virtual and augmented reality [10], robot manipulation [1, 13], and autonomous navigation [3].

In several recent works, depth supervision is introduced to GS to improve scene reconstruction accuracy when applying GS to different use cases, such as dynamic scenes, real-time systems, or few-shot reconstruction. For instance, LiDARs have been integrated with GS to reconstruct highly dynamic scenes for autonomous driving [23, 27]. RGBD (color and depth) sensors are widely used in GS-based simultaneous localization and mapping (SLAM) frameworks to achieve real-time indoor reconstruction and pose estimation [8, 11, 14, 22, 25]. Furthermore, due to the advancement of monocular depth estimation, many few-shot GS systems leverage monocular depth to reduce the number of input images required to train a Gaus-

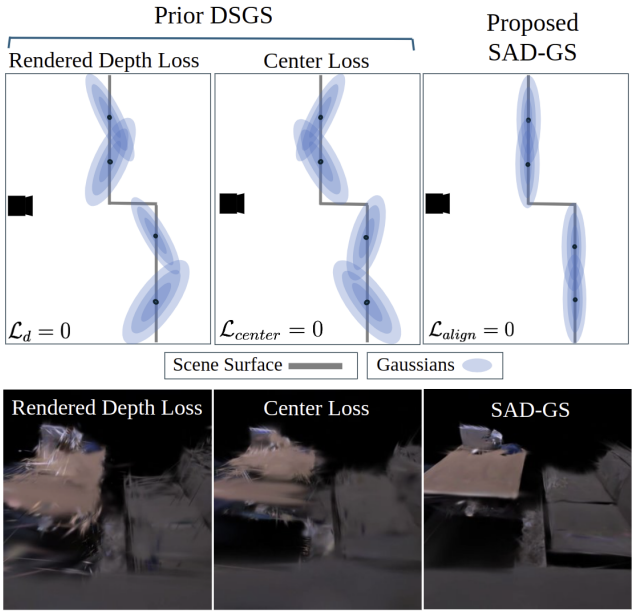

Figure 1. The illustration of the comparison between proposed and prior methods. SAD-GS achieves shape-aligned reconstruction by adding an extra constraint to the shape of scene Gaussians. Existing methods fail to align Gaussian shapes with scene geometry captured from depth information, which produces artifacts and blurring in the reconstruction. The figure shown in the second row is a novel view rendering. SAD-GS outperforms prior methods by offering crisp and smooth geometry.

sian splat [4, 21, 24, 29].

Despite the common use of depth information in GS, current depth-supervised GS (DSGS) methods do not utilize depth accurately. As a result, these methods often use depth information for just initialization or with relatively low training weight, continuing to rely on multi-view RGB images to obtain precise 3D geometry. Specifically, after projecting 3D Gaussians onto the image plane, current DSGS methods only consider the mean position of the Gaussians and ignore their shapes (see Figure 1). This is usually acceptable when synthesizing views from perspectives near training views. However, it implies the wrong 3D geometry that overfits to the training data, which can lead to inaccurate reconstruction, especially for novel views rendered farther from the training views. Inaccurate 3D reconstruction can cause problems for downstream tasks that rely on accurate 3D geometry.

Our paper has three main contributions: First, we propose a novel shape-aligned depth-supervised method for Gaussian Splatting. Inspired by the sample-based loss in NeRF, our method adds additional constraints to the shape of Gaussians by sampling points close to the measured surface along the ray. Second, we demonstrate the importance of Gaussian shape alignment for novel view rendering. Third, through experiments, we demonstrate that our method reconstructs more accurate 3D geometry on one-shot RGBD data compared to previous approaches, as shown in Figure 1. Our method also improves the quality of novel view synthesis, especially for novel viewpoints that are distant from the training view. The proposed method is evaluated qualitatively and quantitatively on publicly available simulated and real-world RGBD datasets.

The remainder of this paper is organized as follows: In Section 2, we review related works about depth-supervised NeRF and GS. In Section 3, we describe our proposed method, SAD-GS. In Section 4, we evaluate SAD-GS on two datasets against prior works, and in Section 5, we conclude and discuss current limitations.

2. Related works

2.1. Depth-Supervised NeRF

Depth supervision is very common in NeRF training. Depth-supervised NeRF frameworks can be divided into two classes. The first type renders depth images directly from the 3D radiance field, then uses the difference between rendered and sensed depth as a loss to learn geometry from 2D depth images [19, 28]. The rendered depth is the expected termination depth of camera rays cast through the scene. As a result of taking an expectation, the rendered depth can sometimes be correct, while the scene geometry is incorrect. It often fails to remove translucent artifacts before the actual depth. This method is straightforward but leads to blurred geometry, as shown in [9].

To improve the depth supervision of NeRF, other works use depth information directly in 3D space to avoid the geometry ambiguity caused by the rendered depth [2, 5, 9, 16]. The main idea is to obtain a distribution along a ray from sampled points and then compute the difference between the rendered distribution and the desired distribution. Prior works use a normal distribution as the desired distribution [5]. Rematas et al. proposed to decrease the variance of the desired distribution progressively [16]. Isaacson and Kung et al. further presented a dynamic variance to reduce convergence time and adapt to incremental input [9].

2.2. Depth-Supervised GS

Unlike NeRF, which represents the scene with a neural network, GS estimates the same radiance function using Gaussian functions as a basis set [12]. In the rendering step, instead of sampling points, querying the neural network, and then rendering pixel-by-pixel as in NeRF, GS simply sorts all Gaussians with the depth of their means and projects all 3D Gaussians to 2D images for full image rendering. This approach leads to significantly faster rendering and training.

A naive depth-aware GS can be intuitively done by using the depth point cloud as the initial means of Gaussians. This method is used in [7, 8, 23] with dense point clouds provided by LiDAR or an RGBD camera. However, no geometry constraint from the depth measurement is applied in this method, so the geometry still relies only on multi-view RGB images. Also, this approach can easily overfit to color input and, as a result, may offer inaccurate 3D reconstruction. To solve this problem, a rendered depth loss similar to that used in NeRFs is widely used in GS-based SLAM frameworks with RGBD cameras [11, 22, 25] and in few-shot GS framewoks that leverage monocular depth estimation [4, 21, 24, 29]. To avoid the geometry ambiguity problem of rendered depth loss as mentioned in 2.1, [22] proposes a deleting step to degenerate all Gaussians before the ray terminates. However, the rendered depth uses the depth of the Gaussian center as the depth of the entire Gaussian. Thus, the rendered depth loss only constrains the Gaussian mean position to fit the geometry, and the Gaussian shape, including scale and orientation, is ignored. The shape-misaligned Gaussians lead to rough surface reconstruction and cause artifacts in the free space. Aside from training with rendered depth loss, the center loss introduced in [27]

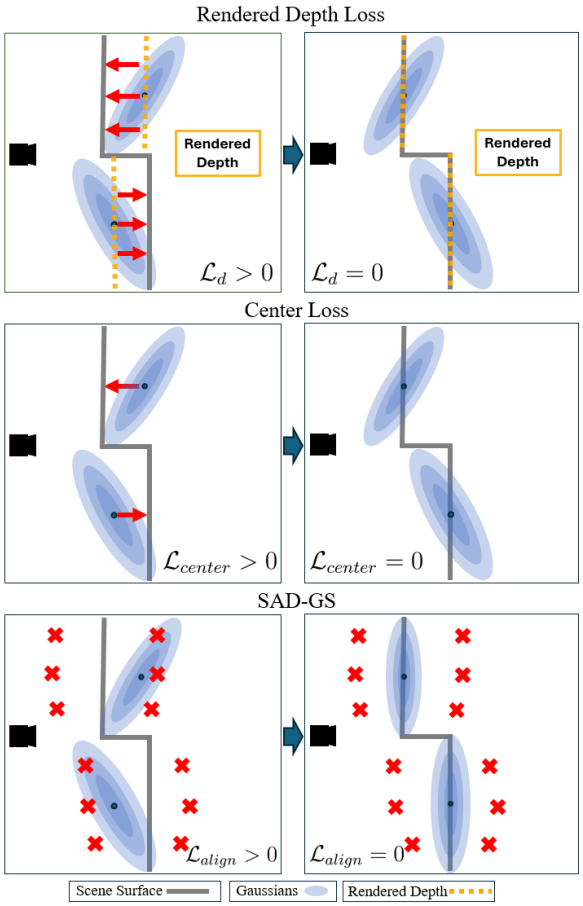

Figure 2. Illustration of different DSGS methods. The rendered depth loss uses the Gaussian center as the depth of the entire Gaussian. The center loss uses the difference between the Gaussian center and its nearest point in a sensed point cloud for training. However, both existing methods ignore the Gaussian shape in their loss functions. In contrast, the proposed SAD-GS achieves shape-aligned Gaussians by penalizing surface-misaligned Gaussians near sampled points (red cross).

uses the distance between Gaussians to their nearest point cloud as the loss to force the Gaussian mean to align with the depth measurements. Yet, the alignment of the Gaussian shape with the surface is still missing.

In this paper, we propose a simple yet efficient shape-aligned loss that samples points near the sensed depth and applies an L1 loss to penalize Gaussians with a surface-misaligned shape. We demonstrate that the proposed loss function leads to better 3D reconstruction than previous DSGS loss functions. Figure 2 illustrates the difference between methods.

3. Method

3.1. 3D Gaussian Scene Representation

Gaussian Splatting models the 3D scene as a set of 3D Gaussians. Each 3D Gaussian is composed of the position μ, rotation quaternion q, scaling vector S, opacity α, and spherical harmonic (SH) coefficients sh:

$$\mathbf{G} = \{G_i : (\mu_i, q_i, s_i, \alpha_i, sh_i) \ | i = 1, ..., N\}. \quad (1)$$

The covariance of each Gaussian is parameterized by its Eigen decomposition

$$\Sigma = RSS^T R^T \quad (2)$$

where $S \in \mathbb{R}^3$ is a 3D scale vector with square roots of Σ's eigenvalues, and $R \in \mathrm{SO}(3)$ is rotation matrix computed from quaternion q.

GS [12] employs a point-based α-blending process to render the color $C(p)$ of a pixel p. Each 3D Gaussian is projected onto the 2D image plane by rendering along the ray direction [30]. Then, the color of one pixel is rendered by sorting the Gaussians in depth (d) order and performing front-to-back α-blending rendering. Considering the view direction, v, the pixel color is computed as:

$$\hat{C}(p) = \sum_{i \in N} c_i \alpha'_i \prod_{j=1}^{i-1} (1 - \alpha'_j) \quad (3)$$

where c_i is the color of a Gaussian obtained by sh_i and v. The opacity, α'_i, is the multiplication of opacity of Gaussian α_i and Gaussian:

$$\alpha'_i = \alpha_i \exp\left(-\frac{1}{2}(x' - \mu'_i)^T \Sigma_i^{-1} (x' - \mu'_i)\right) \quad (4)$$

where x' and μ' are the rendered pixel and Gaussian center in projected 2D image space. In the training step, the color loss is designed to minimize the difference between color in the image, C_{img}, and rendered color, $\hat{C}$. We use L1 loss and D-SSIM loss:

$$\mathcal{L}_{color} = (1 - \lambda_{ssim})\mathcal{L}_1 + \lambda_{SSIM}\mathcal{L}_{D-SSIM} \quad (5)$$

where

$$\mathcal{L}_1 = \|C_{img} - \hat{C}\|_1 \quad (6)$$

and λ_{ssim} is the weight of D-SSIM loss. We use $\lambda_{ssim} = 0.2$ in all our tests following [12]. To extend the color rendering to depth rendering, the rendered depth for each pixel is computed as:

$$\hat{D}(p) = \sum_{i \in N} d_i \alpha'_i \prod_{j=1}^{i-1} (1 - \alpha'_j) \quad (7)$$

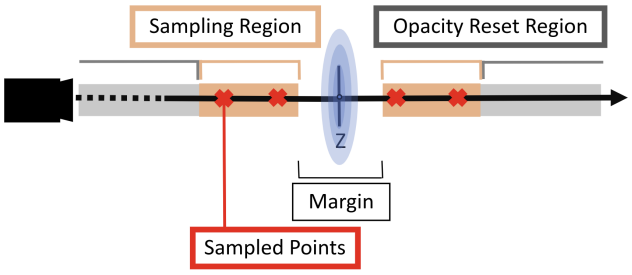

Figure 3. The distance along a ray is broken down into three sections. The area nearest to the measured depth is known as the margin, serving as the Gaussian shape tolerance. Beyond this is the sampling region, where points are sampled to penalize a Gaussian if it extends into this area. Finally, there is the opacity reset region, where we reduce the opacity of Gaussians within it.

where d_i is the depth of the center of the i-th Gaussian. To incorporate depth supervision, [4, 11, 21, 22, 24, 25, 29] design rendered depth loss as the difference between the depth image, D_{img}, and rendered depth, $\hat{D}$.

$$\mathcal{L}_d = \|D_{img} - \hat{D}\|_1 \tag{8}$$

However, using the depth of the Gaussian center as the depth for the entire Gaussian means that rendered depth loss only works if all Gaussians are parallel to the image surface or have extremely small scales. This is because after the projection step, the 3D shape of the Gaussian is collapsed onto the image plane, and the 3D shape is lost. Kerbl et. al. noted this problem by stating that the alpha blending does not consider a per-pixel depth ordering, making the alpha blending only approximate [12]. While this had a negligible effect without depth supervision, we found it to introduce larger problems once depth supervision is introduced. Thus, the rendered depth loss does not provide accurate geometry, and learning geometry still heavily depends on multi-view color supervision. Figure 2 illustrates why rendered depth loss fails to provide correct surface reconstruction.

3.2. Shape-aligned Depth Supervison

In this section, we introduce a shape-aligned depth supervision method to reconstruct geometry. Figure 3 illustrates our proposed strategy. We divide the distance along a ray into three regions. The region nearest to a measured depth z is defined as the margin, representing the tolerance of the Gaussian shape. Beyond this is the sampling region, where we sample points to penalize a Gaussian if it occupies this area, which forces the shape of the Gaussian to align with the scene surface, as shown in Figure 1. Finally, the opacity reset region is situated beyond the sampling region and serves to degrade all Gaussians within its boundaries.

Sampled Points For Shape Constraint: First, we define a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o}$ is the sensor origin, $\mathbf{d}$ is the ray direction, and t is the distance along the ray. We divide the distance along the ray into three regions. First, with depth measurement z along the ray, we mark the margin region as $T_{margin} = [z-\epsilon, z+\epsilon]$, where ϵ is the tolerance distance.

To sample points from the sampling region, we define the far and near bound of the sampling region as $t_{far} = z + \epsilon + \delta$ and $t_{near} = z - \epsilon - \delta$, where δ is the size of the sampling region. We partition $[t_n, t_f]$ into N evenly-spaced bins and then draw one sample uniformly at random from within each bin using stratified sampling following [15]:

$$t_i \sim \mathcal{U}\big[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\big] \tag{9}$$

We further remove sample points located in the margin region:

$$T_s = \{x \in t_i \mid x \notin T_{margin}\} \tag{10}$$

The sampled points are used to penalize Gaussians for having shapes misaligned with the scene surface and occupying the sampling region.

Shape-aligned Loss: After we sample points for shape supervision, we query the Gaussian splat to get the estimated weight at those positions. To reduce the computational cost and avoid overhead gradient computation, we voxelize the space with grid size M and compute each weight for each sampled point only using the located voxel. The weight of each point contributed by each Gaussian can be computed by:

$$\mathcal{G}_k(x) = \alpha_k \exp\left(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right) \tag{11}$$

The total weight of a point can be obtained by summing the weight from individual Gaussians:

$$w_i = \sum_k \mathcal{G}_k(x_i) \tag{12}$$

To force the sampled position to have zero density for shape alignment, we define the loss function as the L1 norm of w_i.

$$\mathcal{L}_{align} = \|w_i\|_1 \tag{13}$$

In our training step, the loss function is the color loss combined with the shape-aligned loss:

$$\mathcal{L} = \lambda_c \mathcal{L}_{color} + \mathcal{L}_{align} \tag{14}$$

Opacity Reduction Strategy: The densification step in GS introduces randomness when new Gausians are added to the scene. This can introduce floating Gaussians located before or after the sampling region. Inspired by the deleting strategy introduced in [22], we also use an opacity degrading strategy to address this issue. However, since we have the extra shape-aligned constraint, we only need to apply the reduction outside the sampling region, which makes it less aggressive. We reduce the opacity by a factor of γ for the distance that is outside the margin and sampling region.

By projecting all centers of Gaussians back to the pixel coordinate, we get the corresponding depth measurement, z, and the depth of each Gaussian, d_i. The opacity of Gaussians are reduced as follows:

$$\alpha_i' = \begin{cases} \gamma\alpha_i & |z - d_i| > \epsilon + \delta \\ \alpha_i & \text{otherwise.} \end{cases} \tag{15}$$

4. Experiments

This section evaluates the novel view rendering and meshing quality of our method compared to prior DSGS methods on single-shot RGBD reconstruction.

4.1. Experimental Setup

Baselines: We evaluate against DSGS methods used in previous papers. **Color Loss** only uses a depth point cloud to initialize Gaussians of the scene, which is the simplest way to incorporate depth information into the GS used in [7, 8, 23]. **Depth Loss** is adding rendered depth loss used in [4, 11, 21, 22, 24, 25, 29] based on color loss. **Center Loss** is adding a term to minimize the distance between Gaussians to the nearest points in the depth point cloud as used in [27]. **Deleting** is implementing only the deleting strategy introduced in [22]. Next, **Depth Loss+D/** is the full method used in [22]. A summary of the baseline loss functions is shown in Table 1.

Datasets: To evaluate the performance of the proposed method, SAD-GS, we conduct experiments on the simulated Replica dataset [17] and the real-world indoor TUM-RGBD dataset [18]. We select a single RGBD frame in each sequence that captures most of the scene as the training image, and the rest of the frames are used as the testing data to evaluate the novel view rendering performance. In Replica, we select the frame index 790 in the office0 sequence as the training frame. In TUM, we select the frame index 0 in the sequence freiburg2 as the training frame. In TUM, we also set the maximum depth to 1.5 meters to remove noisy depth measurements from the RGBD sensor in the background region.

Baselines	Details
Color Loss	$\mathcal{L}_c$
Depth Loss	$\mathcal{L}_c + \mathcal{L}_d$
Deleting	$\mathcal{L}_c + Deleting$
Depth Loss + D/	$\mathcal{L}_c + \mathcal{L}_d + Deleting$

Table 1. Baselines details.

Evaluation Metrics: We evaluate the novel view rendering performance using the peak signal-to-noise ratio (PSNR), Structural Similarity (SSIM) [20], and LPIPS [26]. For the simulated Replica dataset, we further evaluate the mesh accuracy using the ground truth mesh to analyze reconstructed geometry. For metrics, we use accuracy (mean distance from each point in the estimated mesh to each point in the ground truth point cloud), completion (mean distance from each point in the ground truth point cloud to each point in the estimated mesh), and Chamfer distance (CD) (sum of accuracy and completion) to evaluate the similarity between the estimated and ground truth mesh.

Experimental Details. To initialize GS, we divide the space into voxels with size V (m), then compute the mean and covariance for each voxel. In experiments, we use initialization $V = 0.05$, $M = 1$, $\epsilon = 0.03$, $\delta = 0.02$, $\gamma = 0.01$, $N = 3$, and $\lambda_c = 1$. For GS configurations, we set the opacity reset interval to 1000 and run 2000 and 2200 iterations on Replica on TUM, respectively. We also follow the coarse meshing method used in SuGAR [6] to build the mesh.

4.2. Rendering Evaluation

For novel view rendering evaluation, we use all frames in a sequence and exclude the training view. We render images from unseen views and then compare the rendered images with captured images. The **Full** evaluation means the entire images rendered from testing views are evaluated. This evaluation can reflect the artifacts generated outside the field-of-view (FOV) of the training image. On the other hand, the **Seen** evaluation means only evaluating the seen region. This evaluation focuses on the reconstruction within the FOV of the training image. To generate the seen mask for each frame, we simply project the depth point cloud from the training frame to each testing view. Then, image erosion and dilation steps are applied to filter noise in the masks. On the TUM dataset, we further apply an extra erosion step to shrink the seen mask. Since the depth image captured from the RGBD sensor is noisy, the projected mask can also include unseen regions. We found applying an extra erosion step can largely reduce the problem.

Eval. Region	Metrics	Methods					
		Color Loss	Depth Loss	Deleting	Center Loss	Depth Loss+D/	Ours
Full	PSNR (↑)	26.31	21.20	26.43	28.32	24.97	**30.49**
	SSIM (↑)	0.90	0.81	0.90	0.92	0.89	**0.95**
	LPIPS (↓)	0.12	0.19	0.12	0.10	0.13	**0.07**
Seen Region	PSNR (↑)	30.00	27.03	28.64	31.02	29.05	**32.76**
	SSIM (↑)	0.93	0.91	0.92	0.94	0.93	**0.96**
	LPIPS (↓)	0.08	0.09	0.09	0.07	0.09	**0.05**

Table 2. Rendering Evaluation on Replica

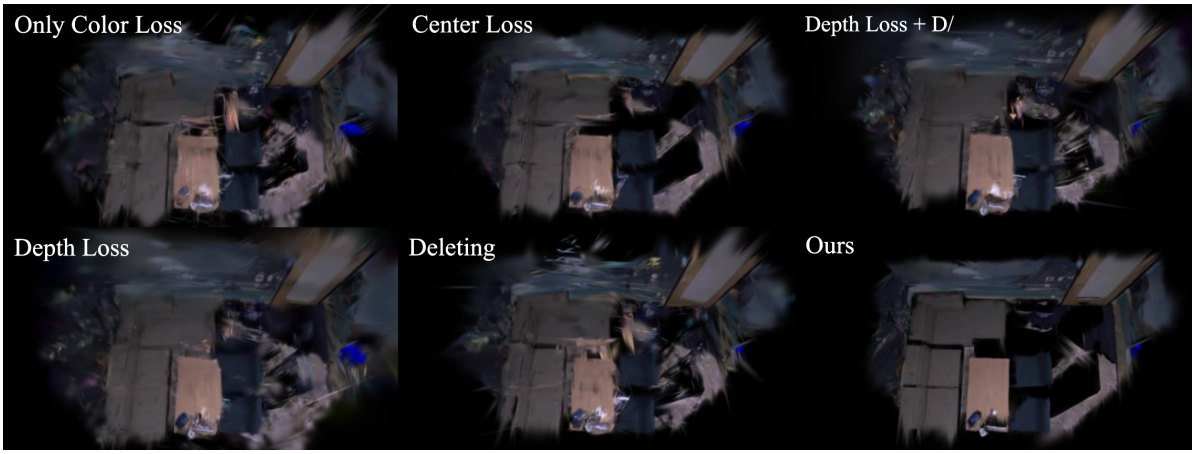

Figure 4. Rendering from bird's-eye-view on Replica.

Eval. Region	Metrics	Methods					
		Color Loss	Depth Loss	Deleting	Center Loss	Depth Loss+D/	Ours
Full	PSNR (↑)	14.17	15.90	14.50	15.93	15.96	**16.06**
	SSIM (↑)	0.70	0.74	0.71	0.74	0.74	**0.77**
	LPIPS (↓)	0.24	0.21	0.23	0.19	0.21	**0.12**
Seen Region	PSNR (↑)	**18.29**	18.10	18.21	18.05	18.11	18.03
	SSIM (↑)	0.82	0.82	0.82	**0.83**	**0.83**	**0.83**
	LPIPS (↓)	0.11	0.11	0.11	0.10	0.12	**0.09**

Table 3. Rendering Evaluation on TUM

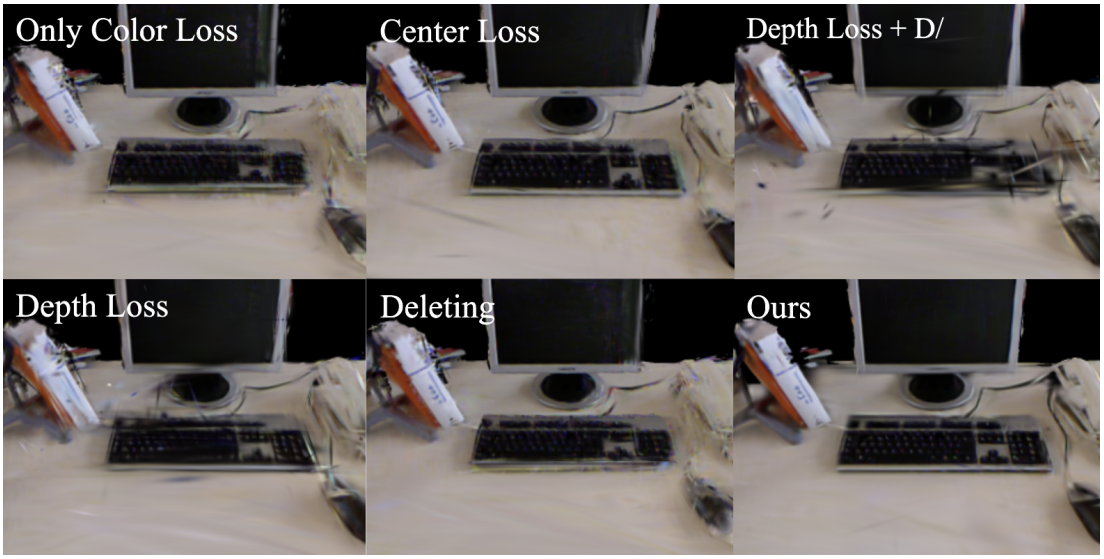

Figure 5. Rendering from a zoomed-in view on TUM.

Eval. Region	Metrics	Methods					
		Color Loss	Depth Loss	Deleting	Center Loss	Depth Loss+D/	Ours
Full	Acc.(↓)	0.148	0.241	0.118	0.078	0.168	**0.034**
	Comp.(↓)	0.035	0.030	0.030	0.030	0.033	**0.016**
	CD(↓)	0.183	0.271	0.148	0.109	0.202	**0.050**
Seen Region	Acc.(↓)	0.093	0.074	0.072	0.065	0.077	**0.027**
	Comp.(↓)	0.030	0.031	0.032	0.030	0.035	**0.027**
	CD(↓)	0.123	0.105	0.105	0.095	0.111	**0.054**

Table 4. Mesh Evaluation on Replica

Replica Table 2 compares rendering performance to the prior DSGS methods. Our method outperforms all existing DSGS methods in both Full and Seen mode. Our method performs better than other methods in the Full mode. This is because we generate fewer artifacts outside the FOV. As shown in Figure 4, our method has a crisp boundary at the edge of the FOV and produces less noise in occluded regions. In the Seen mode evaluation, our method provides the best rendering quality by constructing shape-aligned Gaussians, which leads to less noise and smoother surfaces. Figure 6 shows that our method is more robust to view changes compared to previous methods.

TUM Table 3 shows the rendering performance on the real-world TUM dataset. The overall performance on TUM is worse than Replica due to the noisy RGBD depth measurements. Our method still surpasses all existing methods in the Full with reduced noise.

Our PSNR value in the Seen evaluation is not the best compared to others. We assume this is also due to the noisy RGBD depth measurement. Since our method better fits the input depth data, our reconstruction is a bit deformed from ground truth geometry. In contrast, other methods construct more fuzzy geometry. This makes our method look worse when computing PSNR, which is based on mean squared error(MSE) and requires pixel alignment. However, for metrics like SSIM and LPIPS that evaluate overall quality, our method offers performance that is better or equivalent to others. We also found that the performance of our method and others is less different on TUM. We suppose this is because most of the testing views are close to the training view, while the proposed method offers more significant improvement at extreme view change, as shown in Figure 6. Figure 5 demonstrates rendered results with a large view change.

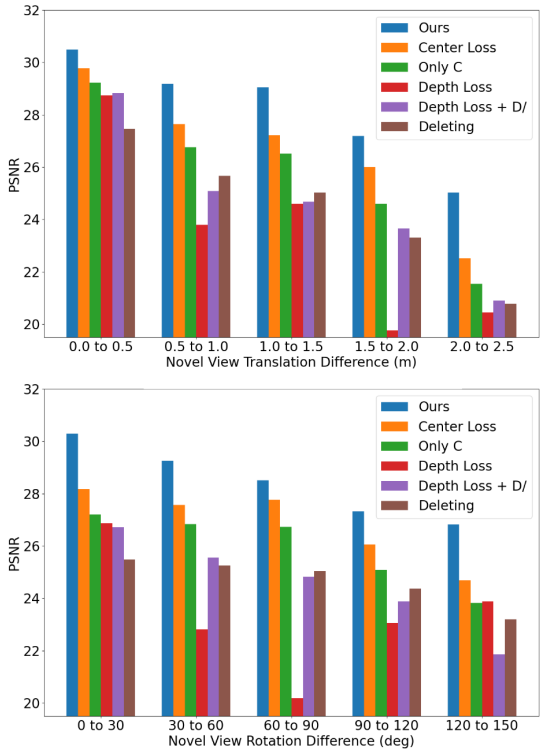

Figure 6. PSNR of rendered images from novel views with increasing translation and rotation view change levels from the training view.

4.3. Mesh Evaluation

Table 4 shows quantitative evaluation for mesh reconstruction performance on Replica. Our shape-aligned method offers the best geometry accuracy and completion against existing methods. Also, the mesh evaluation shows the same trend as that of the rendering evaluation. This indicates the importance of geometry to novel view rendering. The visualized meshes on Replica and TUM are shown in Figure 7 and Figure 8. The figure demonstrates that our method can significantly improve the estimated geometry of surfaces by aligning the shape of Gaussians with the surface.

4.4. Ablation Study

Table 5 provides an ablation study on Replica. $\mathcal{L}_{color}$ is only using color loss. +ORS and +$\mathcal{L}_{align}$ means applying the Opacity Reduction Strategy and $\mathcal{L}_{align}$ separately. The experiments show that adding ORS or $\mathcal{L}_{align}$ individually can worsen performance. This is because using ORS only is too aggressive without the

Figure 7. Qualitative comparison on mesh reconstruction on Replica.

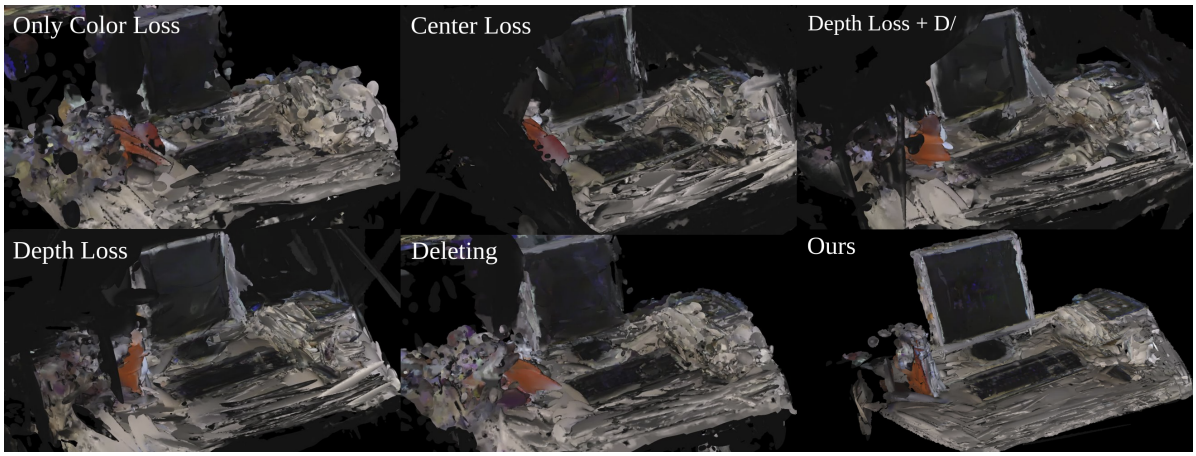

Figure 8. Qualitative comparison on mesh reconstruction on TUM.

Eval. Region	Metrics	Ablation			
		$\mathcal{L}_{color}$	+ORS	+$\mathcal{L}_{align}$	Proposed
Full	PSNR (↑)	26.31	23.86	25.10	**30.49**
	SSIM (↑)	0.90	0.88	0.89	**0.95**
	LPIPS (↓)	0.12	0.13	0.14	**0.07**
Seen Region	PSNR (↑)	30.00	24.21	30.15	**32.76**
	SSIM (↑)	0.93	0.89	0.93	**0.96**
	LPIPS (↓)	0.08	0.12	0.08	**0.05**

Table 5. Ablation Study on Replica

$\mathcal{L}_{align}$ constraint that forces Gaussians to stay in the margin area. On the other hand, using $\mathcal{L}_{align}$ alone fails to remove floating Gaussians outside the margin and sampling region. The proposed method combining both ORS and $\mathcal{L}_{align}$ yields the best performance.

5. Conclusion

This paper introduces a shape-aligned, depth-supervised approach for GS. Previous research only pays attention to the positioning of Gaussians, which leads to inaccurate surface geometry. Our proposed loss constrains Gaussian shapes and yields a surface-aligned reconstruction. Our method's effectiveness is demonstrated qualitatively and quantitatively, through testing on two public datasets. Our method surpasses previous DSGS methods in novel view synthesis and mesh accuracy on a single-shot RGBD reconstruction. The current method relies on accurate depth measurement and only tests with single-view training. We plan to take depth uncertainty into account and test with multi-view input in the future. In addition, our sampled points training is implemented in Pytorch using auto gradient for back-propagation, which leads to relatively slow training speed. We expect that CUDA acceleration is needed for large-scale scenes.

Acknowledgements

This research was funded by Mcity, University of Michigan.

References

[1] Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Sünderhauf. Physically embodied gaussian splatting: Embedding physical priors into a visual 3d world model for robotics. In *Conference on Robot Learning*, number 7th, 2023. 1

[2] Alexandra Carlson, Manikandasriram S. Ramanagopal, Nathan Tseng, Matthew Johnson-Roberson, Ram Vasudevan, and Katherine A. Skinner. Cloner: Camera-lidar fusion for occupancy grid-aided neural representations. *IEEE Robotics and Automation Letters*, 8(5):2812–2819, 2023. 2

[3] Timothy Chen, Ola Shorinwa, Weijia Zeng, Joseph Bruno, Philip Dames, and Mac Schwager. Splat-nav: Safe real-time robot navigation in gaussian splatting maps. *arXiv preprint arXiv:2403.02751*, 2024. 1

[4] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. *arXiv preprint arXiv:2311.13398*, 2023. 2, 4, 5

[5] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 2

[6] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023. 5

[7] Sheng Hong, Junjie He, Xinhu Zheng, Hesheng Wang, Hao Fang, Kangcheng Liu, Chunran Zheng, and Shaojie Shen. Liv-gaussmap: Lidar-inertial-visual fusion for real-time 3d radiance field map rendering. *arXiv preprint arXiv:2401.14857*, 2024. 2, 5

[8] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. *arXiv preprint arXiv:2311.16728*, 2023. 1, 2, 5

[9] Seth Isaacson, Pou-Chun Kung, Mani Ramanagopal, Ram Vasudevan, and Katherine A Skinner. Loner: Lidar only neural representations for real-time slam. *IEEE Robotics and Automation Letters*, 2023. 2

[10] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. *arXiv preprint arXiv:2401.16663*, 2024. 1

[11] 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. *arXiv preprint arXiv:2312.02126*, 2023. 1, 2, 4, 5

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 1, 2, 3, 4

[13] Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. *arXiv preprint arXiv:2403.08321*, 2024. 1

[14] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *arXiv preprint arXiv:2312.06741*, 2023. 1

[15] 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. *Commun. ACM*, 65(1):99–106, 2021. 1, 4

[16] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[17] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5

[18] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012. 5

[19] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time. *2021 IEEE/CVF International Conference on Computer Vision*, pages 6209–6218, 2021. 2

[20] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5

[21] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. Sparsegs: Real-time 360° sparse view synthesis using gaussian splatting. *arXiv e-prints*, pages arXiv–2312, 2023. 2, 4, 5

[22] Chi Yan, Delin Qu, Dong Wang, Dan Xu, Zhigang Wang, Bin Zhao, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. *arXiv preprint arXiv:2311.11700*, 2023. 1, 2, 4, 5

[23] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. *arXiv preprint arXiv:2401.01339*, 2024. 1, 2, 5

[24] Chen Yang, Sikuang Li, Jiemin Fang, Ruofan Liang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Gaussianobject: Just taking four images to get a high-quality 3d object with gaussian splatting. *arXiv preprint arXiv:2402.10259*, 2024. 2, 4, 5

[25] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023. 1, 2, 4, 5

[26] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, Los Alamitos, CA, USA, 2018. IEEE Computer Society. 5

[27] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. *arXiv preprint arXiv:2312.07920*, 2023. 1, 2, 5

[28] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[29] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00451*, 2023. 2, 4, 5

[30] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3): 223–238, 2002. 3