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

Pou-Chun Kung[1], Seth Isaacson[1], Ram Vasudevan[1], and Katherine A. Skinner[1]

***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 the 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.**

## I. Introduction

GAUSSIAN Splatting (GS) [1] marks a recent paradigm shift in the field of computer vision and novel view synthesis. 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 [2, 3]. RGBD (color and depth) sensors are widely used in GS-based SLAM to achieve real-time indoor reconstruction and pose estimation [4, 5, 6, 7, 8]. 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 Gaussian splat [9, 10, 11, 12].

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(Figure 1). This is usually acceptable when synthesizing views from perspectives near training views. However, it implies the wrong 3D geometry that overfits the designed loss function. This paper proposes a shape-aligned depth-supervised method to obtain precise 3D geometry. The accurate geometry is essential for downstream tasks such acdeep learning data augmentation or robot path planning and manipulation.

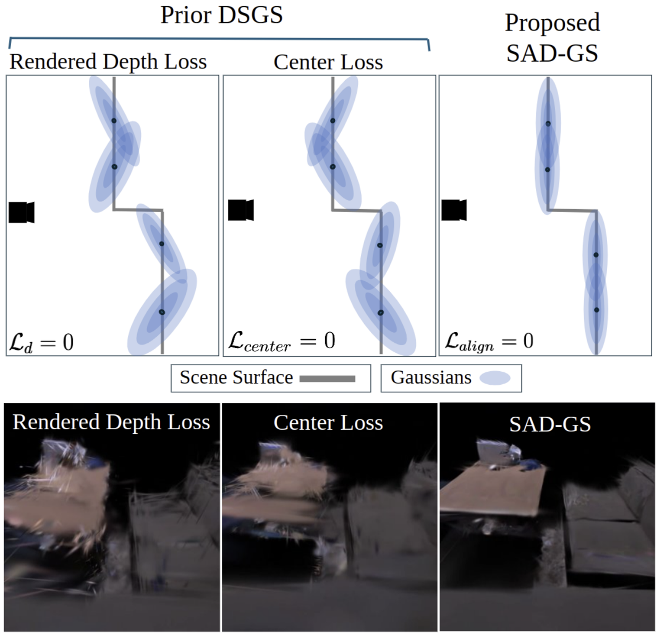

Fig. 1. The illustration of the comparison between proposed and prior methods.

## II. Related works

A naive depth-supervised GS can be intuitively done by using the depth point cloud as the initial means of Gaussians. This method is used in [2, 13, 8] 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 shape-misaligned 3D reconstruction. To solve this problem, the rendered depth loss similar to NeRF is widely used in GS-based SLAMs using RGBD cameras [6, 4, 5] or few-shot GS using monocular depth estimation [9, 10, 11, 12]. To avoid the geometry ambiguity problem of rendered depth loss as shown in Figure 2, [5] proposes a deleting step to degrade all Gaussians

This research was funded by Mcity, University of Michigan.

[1] P. Kung, S. Isaacson, R. Vasudevan, and K. A. Skinner are with the Department of Robotics, University of Michigan, Ann Arbor, MI 48109. {pckung, sethgi, ramv, kskin}@umich.edu.

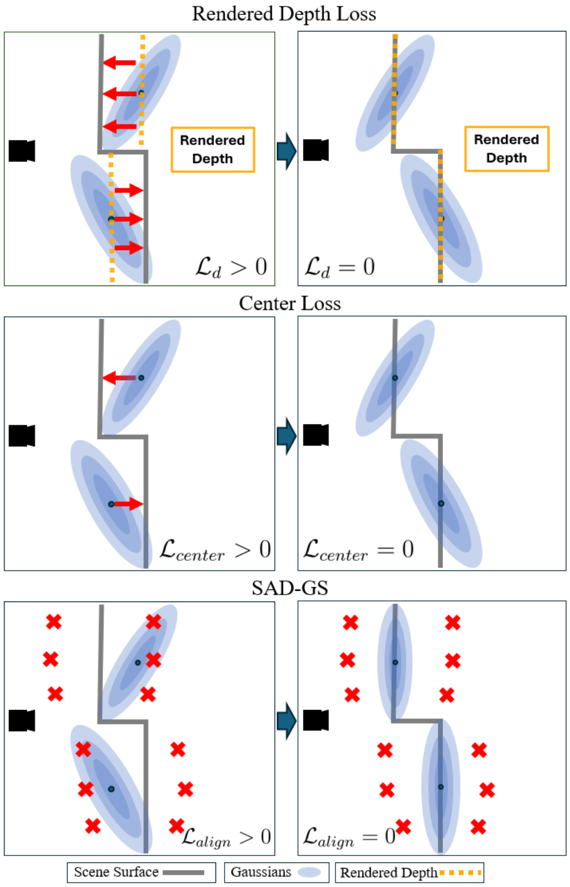

Fig. 2. Illustration of different DSGS methods. The rendered depth loss uses the Gaussian center as the depth of the entire Gaussian. The center loss uses the difference between the Gaussian center and its nearest point in a sensed point cloud for training. However, both existing methods ignore the Gaussian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

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 [3] 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.

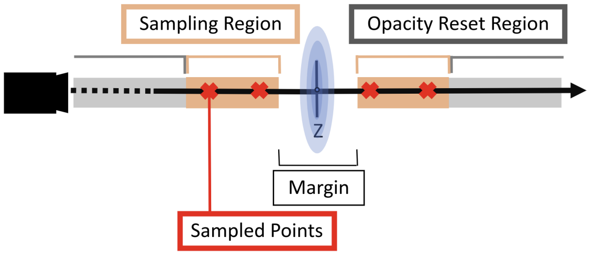

Fig. 3. Illustration of our proposed strategy.

## III. Method

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. Finally, the opacity reset region is situated beyond the sampling region and serves to degrade all Gaussians within its boundaries.

### A. *Shape-aligned Depth Supervison*

**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 $\epsilon$ 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 $\delta$ 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 [14]. We further remove sample points located in the margin region.

**Sampled Point Training:** 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 voxelized the space with grid size $M$ and computed each weight for each sampled point only using the located voxel. The weight of each point contributed by each Gaussian can be computed by:

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

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

$$w_i = \sum_k \mathcal{G}_k(x_i) \tag{2}$$

To make the sampled position have zero density, we define the loss function as the L1 norm of $w_i$. In our training step, the loss function is the color loss combined with the shape-aligned loss:

$$\mathcal{L} = \lambda_c \mathcal{L}_{color} + \mathcal{L}_{align} \tag{3}$$

| Eval. | Metrics | Methods | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathcal{L}_c$ | $\mathcal{L}_d$ | $\mathcal{L}_{center}$ | $D/$ | $\mathcal{L}_d + D/$ | 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 | 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 I
Rendering Evaluation on Replica

| Eval. | Metrics | Methods | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathcal{L}_c$ | $\mathcal{L}_d$ | $\mathcal{L}_{center}$ | $D/$ | $\mathcal{L}_d + D/$ | 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 | 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 II
Rendering Evaluation on TUM

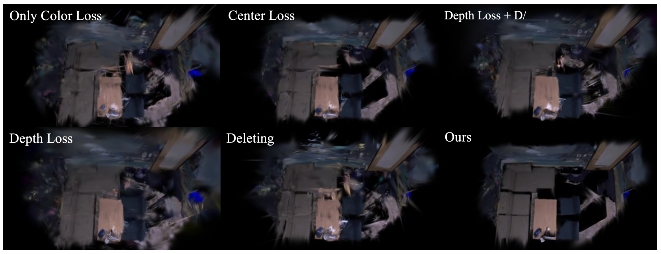

Fig. 4. Rendering from bird-eye-view on Replica. Our method shows the crisp rendering from an extremely different perspective, while other methods produce unpleasant artifacts.

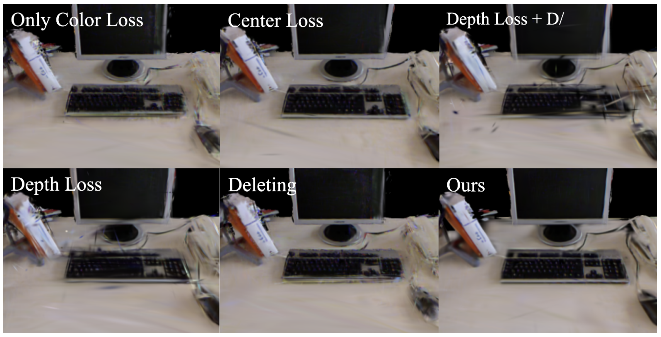

Fig. 5. Rendering from a zoomed-in view on TUM. Our method offers the best novel-view synthesis quality.

**Opacity Degrading Strategy:** The densification step in GS introduces randomness when new Gausians are added to the scene. This can induce floating Gaussians located before or after the sampling region. Inspired by the deleting strategy introduced in [5], 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 degradation outside the sampling region, which makes it less aggressive. We degrade the opacity by the factor of $\gamma$ for the distance that is closer or farther than the sampling region.

## IV. Experiments

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

### A. Experimental Setup

**Baselines:** We evaluate against DSGS methods used in previous papers. $\mathcal{L}_c$ 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 [2, 13, 8]. $\mathcal{L}_d$ is adding rendered depth loss used in [6, 4, 5, 9, 10, 11, 12] based on color loss. $\mathcal{L}_{center}$ is adding a term to minimize the distance between Gaussians to the nearest points in the depth point cloud. $D/$ is implementing only the deleting strategy introduced in [5]. Next, $\mathcal{L}_d + D/$ is the full method used in [5].

**Evaluation:** To evaluate the performance of the proposed SAD-GS, we conduct experiments on the simulated Replica dataset [15] and the real-world indoor TUM-RGBD dataset [16]. We evaluate the novel-view rendering performance using the peak signal-to-noise ratio (PSNR), Structural Similarity (SSIM) [17], and LPIPS [18]. For the simulated Replica dataset, we further use accuracy, completion, and Chamfer distance (CD) 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 $V = 0.05$, spatial voxel $M = 1$, $\epsilon = 0.03$, and $\delta = 0.05$. For GS configurations, we set the opacity reset interval to 1000 and run 2000 and 2200 iterations on Replica on TUM separately. We also follow the coarse meshing method used in SuGAR [19] to build the mesh.

### B. 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). On the other hand, the **Seen** evaluation means only evaluating the seen region. This evaluation focuses on the reconstruction within the FOV. 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 noises in masks. On the TUM dataset, we further apply an extra erosion step to shrink the seen mask. Because 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.

**Replica** Table I 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

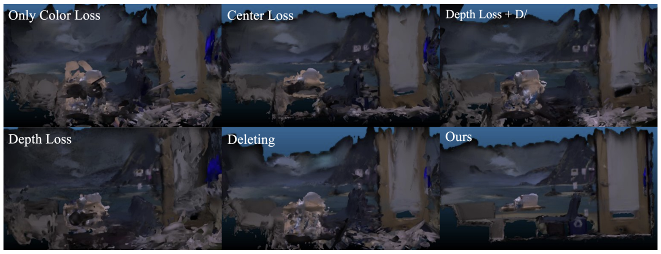

Fig. 6. Qualitative comparison on mesh reconstruction on Replica.

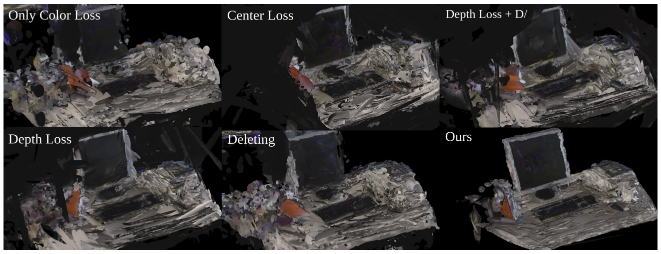

Fig. 7. Qualitative comparison on mesh reconstruction on TUM.

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 8 shows that our method is more robust to view changes compared to previous methods.

**TUM** Table II 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 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 8. Figure 5 demonstrates rendered results with a large view change.

### *C. Mesh Evaluation*

Table III 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 6 and Figure 7. The figure demonstrates that our method can significantly improve the estimated geometry of surfaces by aligning the shape of Gaussians with the surface.

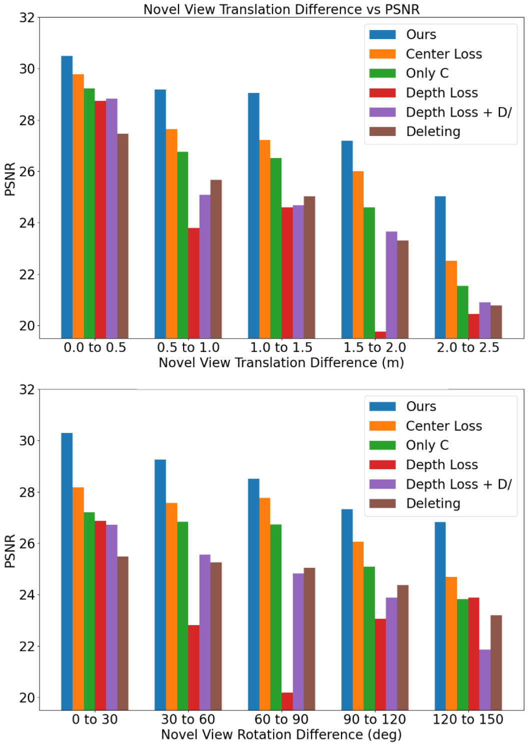

Fig. 8. PSNR of rendered images from novel views with different translation and rotation view change levels.

| Eval. | Metrics | Methods | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathcal{L}_c$ | $\mathcal{L}_d$ | $\mathcal{L}_{center}$ | $D/$ | $\mathcal{L}_d + D/$ | 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 | 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 III
Mesh Evaluation on Replica

## V. 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. It surpasses previous DSGS methods in novel-view synthesis and mesh accuracy on a single-shot RGBD reconstruction.

# REFERENCES

[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.

[2] Y. Yan, *et al.*, "Street gaussians for modeling dynamic urban scenes," *arXiv preprint arXiv:2401.01339*, 2024.

[3] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.-H. Yang, "Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes," *arXiv preprint arXiv:2312.07920*, 2023.

[4] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, "Gaussian-slam: Photo-realistic dense slam with gaussian splatting," *arXiv preprint arXiv:2312.10070*, 2023.

[5] C. Yan, *et al.*, "Gs-slam: Dense visual slam with 3d gaussian splatting," *arXiv preprint arXiv:2311.11700*, 2023.

[6] N. Keetha, *et al.*, "Splatam: Splat, track & map 3d gaussians for dense rgb-d slam," *arXiv preprint arXiv:2312.02126*, 2023.

[7] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, "Gaussian splatting slam," *arXiv preprint arXiv:2312.06741*, 2023.

[8] H. Huang, L. Li, H. Cheng, and S.-K. Yeung, "Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras," *arXiv preprint arXiv:2311.16728*, 2023.

[9] Z. Zhu, Z. Fan, Y. Jiang, and Z. Wang, "Fsgs: Real-time few-shot view synthesis using gaussian splatting," *arXiv preprint arXiv:2312.00451*, 2023.

[10] J. Chung, J. Oh, and K. M. Lee, "Depth-regularized optimization for 3d gaussian splatting in few-shot images," *arXiv preprint arXiv:2311.13398*, 2023.

[11] H. Xiong, S. Muttukuru, R. Upadhyay, P. Chari, and A. Kadambi, "Sparsegs: Real-time 360° sparse view synthesis using gaussian splatting," *arXiv e-prints*, pp. arXiv–2312, 2023.

[12] C. Yang, *et al.*, "Gaussianobject: Just taking four images to get a high-quality 3d object with gaussian splatting," *arXiv preprint arXiv:2402.10259*, 2024.

[13] S. Hong, *et al.*, "Liv-gaussmap: Lidar-inertial-visual fusion for real-time 3d radiance field map rendering," *arXiv preprint arXiv:2401.14857*, 2024.

[14] 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, p. 99–106, Dec 2021.

[15] J. Straub, *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[16] 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.

[17] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[18] 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)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2018, pp. 586–595. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00068

[19] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," *arXiv preprint arXiv:2311.12775*, 2023.