GDA Project: 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Pierrick Bournez
École Normale Supérieure
Paris-Saclay, France

Amruth Srivathsan
École Normale Supérieure
Paris-Saclay, France

Paul Tabbara
École Normale Supérieure
Paris-Saclay, France

Figure 1: Visualization of 3D Gaussians on a baseline scene

Abstract

Three-dimensional (3D) Gaussian splatting is a novel technique for representing and rendering volumetric data using 3D Gaussians. This paper describes an optimized framework for generating and manipulating 3D Gaussian splats, enabling highly efficient rendering while preserving fidelity. We test the model on several experiments, such as changing the input of the structure from motion, adding a depth regularization or changing the number of spherical harmonics. Overall, the proposed methods are robust under high quality input and paved the way to numerous follow-up articles

CCS Concepts

• **Computing methodologies** → **Rendering**; **Rasterization**.

Keywords

3D Gaussian,Rendering,Neural fields,Spherical Harmonics

1 Introduction

Gaussian splatting [Kerbl et al. 2023] is an innovative approach to scene rasterization, utilizing Gaussian functions as input primitives instead of traditional geometric representations like polygons or voxels. Departing from conventional rendering techniques, this method projects 3D Gaussian distributions directly into 2D space as smooth, continuous blobs. This process significantly enhances rendering efficiency, providing smoother visual outputs while achieving substantial speed improvements by allowing for parallelized computing. These advantages position Gaussian splatting as a promising technique for real-time applications such as augmented reality, virtual reality, and video games [Shao et al. 2024; Zhai et al. 2024], where achieving low-latency rendering is of paramount importance.

In this work, we review existing methods for rendering 3D scenes from multiple views. We analyze the techniques presented in the original 3D Gaussian Splatting paper and highlight its limitations and potential extensions.

We identified the following limitations and proposed corresponding extensions:

- **Under-represented viewpoints**: Missing viewpoints in the scene result in holes in the 3D reconstruction (see Section 4.1). We addressed this by using generative image techniques to fill in these gaps (see Section 5.1).
- **Rendering boundaries and discontinuities**: 3D Gaussian Splatting struggles to accurately render sharp boundaries and discontinuities (see Section 4.2).
- **View artifacts and chromatic aberrations**: Artifacts and aberrations appear under certain angles (see Section 4.3). To mitigate this, we experimented with adding noise and adjusting the degree of spherical harmonics (see Section 5.4).
- **Depth regularization**: We introduced depth information to the input images to regularize the training process and improve results (see Section 5.2).
- **SfM solver comparison**: we compared more extensively than the original paper the performance of two SfM solvers, GLOMAP and COLMAP, to evaluate their impact on Gaussian Splatting (see Section 5.3).

1.1 Related Works

Prior to the emergence of Gaussian splatting, two primary approaches dominated the synthesis of novel views from input images: Neural Rendering and Radiance Fields (e.g., NeRF), and Point-based Rendering.

1.1.1 Point-Based Rendering. Point-based methods efficiently render disconnected geometric samples[Grossman and Dally 1998] by computing a point cloud. Each point represents a shape that spans more than a pixel, such as circular [Botsch et al. 2005] or elliptical representations [Zwicker et al. 2001b]. While faithful to the underlying data, point samples suffered from discontinuities and visible holes. There have also been efforts to augment points with neural features to enhance rendering. [Rückert et al. 2022]. Although these methods share similarities with the foundational ideas of Gaussian

splatting, they lacked a ready-to-use, fast rendering algorithm like the one proposed in the initial Gaussian splatting paper.

1.1.2 Neural Rendering and Radiance Fields. Deep learning methods for novel-view synthesis were adopted early, with CNNs used to estimate blending weights [Hedman et al. 2018]. Then, Neural Radiance Fields (NeRF) [Mildenhall et al. 2021] introduced techniques like importance sampling and positional encoding to enhance quality but relied on large networks, which significantly impacted computational speed. The success of NeRF spurred an explosion of follow-up methods, with MiP-NeRF360 [Barron et al. 2022] representing the state of the art at the time. While achieving impressive rendering quality, these methods required extensive training and relied on computationally expensive sampling during rendering.

To address these limitations, Instant NGP [Müller et al. 2022] introduced hash grids and occupancy grids to accelerate computation. Similarly, Plenoxels [Fridovich-Keil et al. 2022] leveraged a sparse voxel grid to interpolate a continuous density field. These approaches were the primary solutions for engineers prior to the publication of the 3D Gaussian Splatting method. The 3D Gaussian Splatting approach further improved on these techniques by offering significantly faster training and rendering times compared to traditional NeRF methods.

2 Target Applications for this method

This method is applicable to various domains aiming to generate 3D scene from images. For instance, gaming and virtual reality, where 3D Gaussian splatting enhances the rendering of complex environments. Its ability to handle large-scale 3D data at real-time speeds with smooth transitions between objects and surfaces makes it ideal for immersive experiences, such as VR simulations [Jiang et al. 2024] or augmented reality interfaces [Franke et al. 2024], where responsiveness and visual fidelity are critical.

Another significant application lies in autonomous systems and robotics [Zhu et al. 2024], where 3D Gaussian splatting can be used for real-time scene reconstruction and mapping without any need for heavy sensors [Salzinger 2024]. Moreover, in medical imaging, this technique can help with novel Computed Tomography projections, but it should be noted that applying SfM to computed tomography images is especially challenging [Nikolakakis et al. 2024].

There are many uses for this technique in digital twins [Lin et al. 2024], where high-precision 3D duplicates of real-world objects are required for simulation in real-time tasks.

3 Methodology of the original paper

In this section, we shall describe and explain the method presented in the paper [Kerbl et al. 2023]. The input is a set of images from a static scene. A structure from motion is then applied to it. From these points, they create a set of 3D Gaussians, defined by a position (mean), covariance matrix, colors, and opacity. The directional appearance component (color) of the radiance field is represented via spherical harmonics (SH). These parameters are optimized in an iterative process, as described in Section 3.4. The primary contribution of the initial paper is a rasterizer that avoids the computationally expensive sampling required by NeRF [Mildenhall et al. 2020] by approximating the projection process.

3.1 Initialization

We assume that we start with images of a static scene. As input, we take a set of images depicting a finite set of different viewpoints, V, of this static scene, where $V \in \mathbb{N}$. We then apply Structure from Motion (SfM), where the input consists of these V images, yielding the output consisting of camera calibration data and a sparse point cloud $P \in \mathbb{N}$ that will be the input of the gaussian splatting algorithm.

3.2 Gaussian: a building block of 3D Gaussian

The fundamental building block of this paper is the 3D Gaussian. One of the key properties of Gaussians is that 3D Gaussians can be projected and rendered as 2D Gaussians [Zwicker et al. 2001a]. Each gaussian will be represented with a covariance matrix Σ, a mean μ, an opacity α and colors:

$$\mathcal{G}(x) = exp(-1/2(x)^T \times \Sigma^{-1}(x)) \tag{1}$$

The corresponding 2D Gaussian for rendering is a Gaussian with covariance σ' given by :

$$\sigma' = JW\Sigma W^T J^T$$

with the corresponding Jacobian J of the projective transformation and W the viewing transformation. Opacity α and colors will be used in the rendering section described in section 3.4.

3.3 Training and Geometry Optimizations

3.3.1 Training. Training is performed iteratively by rendering images from the 3D scene of Gaussians and comparing them with the real input training images. The Gaussians are then optimized using a loss function that measures the difference between the rendered and training images.

In the original paper, the following loss function is used to compare the rendering and training views:

$$L = (1 - \lambda)L_1 + \lambda L_{D-SSIM} \tag{2}$$

Here, L_1 represents the L_1 norm loss, which measures pixel-by-pixel differences, while L_{D-SSIM} refers to a structural dissimilarity term derived from the Structural Similarity Index Measure (SSIM). Unlike L_1, L_{D-SSIM} considers the neighborhood of pixels to capture inter-pixel dependencies.

3.3.2 Adding Geometry. The paper also introduces methods to add or remove geometrical features. They add features in regions where gaussians fail to reconstruct local geometry either by under-reconstruction (regions with missing features) or by over-reconstruction (regions where one gaussian covers too big of an area). In both cases, the gaussians are replaced by pairs of Gaussians, reducing their scale in the case of over-reconstruction.

3.3.3 Removing Geometry. One issue with this process is the potential creation of an excessive number of Gaussians. To address this, the authors introduced a pruning step during the training process,

which removes Gaussians with an opacity lower than a specified threshold, as they contribute minimally to rendering.

To further prevent the over-creation of Gaussians in cases of under- or over-reconstruction, the authors set all opacity coefficients α close to zero after every N optimization steps. The optimization process then naturally increases the α coefficients for necessary Gaussians, while the pruning step automatically removes those deemed unnecessary.

3.4 Rendering

We explain here how the 3D gaussian will be splatted into the 2D space to create an image.

3.4.1 Storing RGB value for every Gaussian. To encode color information, each Gaussian stores view-dependent RGB values using spherical harmonics along with an opacity value, α. "Spherical harmonics" are a widely used method [Green 2003] for representing colors on 3D objects, allowing the color to vary depending on the camera's viewpoint.

Mathematically, we store the spherical harmonics up to a maximum degree l_{max}. If we denote the view direction of point i in image j as v_i^j Then the color of the point[Zhang et al. 2022] is given by :

$$c_i^j = \sum_{l=0}^{l_{max}} \sum_{m=-l}^{l} c_m^l * \times Y_l^m(v_i^j)$$

c_m^l is a learnable parameter per gaussian with $Y_l^m(v_i^j)$, the corresponding spherical harmonics.

3.4.2 Rendering. To determine which pixels a Gaussian in 3D space will impact, we compute its projection onto the 2D image plane. For a Gaussian and an image j, its projection p_{ij} is calculated in 2D space, along with its 2D covariance matrix σ'. If the projection falls within the image boundaries, we calculate a rectangle based on the eigenvalues of the covariance matrix σ'. This rectangle defines the region of the image influenced by the Gaussian during rendering.

Since multiple Gaussians can influence the same pixel in a 2D image, it is necessary to introduce a parameter to control how these Gaussians interact in the pixel space. To address this, the authors included an α parameter, which determines the contribution of each Gaussian to the final color of the pixel.

The final color of a pixel i in an image j is given by:

$$c_i^j = \sum_{i=1}^{N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{3}$$

where α_i represents the opacity weights, and N is the number of Gaussians whose rectangles intersect the pixel in the image space, sorted depth-wise.

3.5 Rasterization and backpropagation

Since 2D images must be generated repeatedly from the 3D scene, a fast algorithm for rasterization is essential. The main goal is to balance quality with speed while maximizing the parallelization of the rendering process. In the original paper, the authors divided the 2D screen into $n \times n$ tiles and retained only the Gaussians likely to fall within the view frustum. For each tile, they processed every pixel within it, sorting the Gaussians by depth and accumulating the color and alpha values depth-wise (from front to back). The process stops once the accumulated alpha value in a pixel reaches a saturation threshold α. Unlike traditional ray tracing methods, this approach does not rely on sampling and uses saturation as the sole stopping criterion.

In the backpropagation step, the per-pixel loss must be converted into the 3D Gaussian space. For each tile, the process starts with the last Gaussian that affected any pixel. Each pixel then tests and processes Gaussians whose depth is less than that of the last contributing Gaussian. This approach enables direct differentiation, ensuring efficient gradient computation.

4 Limitations

In our experiments with the method, we identified several limitations of the original paper. Specifically, missing viewpoints can result in "holes" in the rendered 3D scene (see Section 4.1). Additionally, we observed that using Gaussians to represent information in 3D space results in artifacts when rendering boundaries and discontinuities (see Section 4.2). Finally, we found that the use of view-dependent spherical harmonics introduces view artifacts and chromatic aberrations (see Section 4.3).

4.1 "Holes" Due to Missing Camera Views

A notable limitation of Gaussian splatting is the appearance of "holes," rendered as black patches in regions not observed by the training camera, as illustrated in Figure 2. Since Gaussian splatting relies on information from multiple viewpoints to reconstruct the scene, areas outside the field of view of the training images lack the necessary data to generate accurate visual representations. Consequently, these regions remain undefined or are rendered as black patches, as the algorithm cannot infer color or texture without sufficient spatial information. This limitation underscores a fundamental challenge in 3D reconstruction and view synthesis: the output quality is heavily dependent on the completeness of input camera coverage during training, with no extrapolation from other parts of the scene.

In practice, these missing regions could provide an important signal to users, highlighting areas where additional camera views are required. Identifying such gaps could prompt users to capture supplementary imagery, thereby improving the reconstruction.

In scenarios where acquiring more data is not feasible, an interesting extension to Gaussian splatting could involve methods for interpolating missing regions. We propose an approach to address this issue, as detailed in Section 5.1.

4.2 Difficulty Rendering Boundaries and Discontinuities

The Gaussian splatting algorithm also faces challenges in handling discontinuities and boundaries, particularly at the edges between objects, as shown in Figure 3. These discontinuities are common in scenes with sharp transitions between objects or materials, where the smooth, probabilistic nature of Gaussian splatting fails to accurately capture abrupt changes in geometry or color.

Figure 2: Black patches resulting from areas not observed by the initial cameras during scene reconstruction.

Figure 3: A viewpoint showing boundary and discontinuous regions have significant artifacts

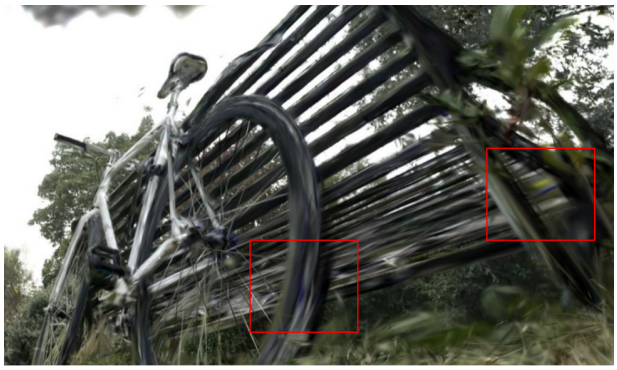


Figure 4: Chromatic Aberration

A plausible explanation for this limitation is that the Gaussian splatting algorithm inherently assumes a degree of continuity in the spatial distribution of Gaussians. While this assumption performs well for smooth surfaces, it proves insufficient for accurately representing sharp edges.

However, upon reviewing relevant further literature, we found that some efforts have been made to address these issues. For example, approaches discussed in [Qu et al. 2024] and [Chelani et al. 2024] propose modifications to the algorithm and introduce new techniques to better preserve discontinuities and sharp edges in this algorithm. These methods employ strategies such as edge-aware optimization or adaptive Gaussian placement to mitigate the challenges posed by discontinuities.

In our experimentation, we decided to take another approach by testing whether adding noise to modify the "detail" in the image could help rectify this issue. The key takeaways from this investigation are presented in Section 5.4.2.

4.3 Color artifacts in Rendered Viewpoints

In this section, we detail several color anomalies observed while exploring the rendered scenes.

First, as described in Equation 3.4.1, spherical harmonics provide a view-dependent color representation. While this approach is effective for thoroughly scanned 3D objects, visualizing novel viewpoints not captured during scanning can result in blurry artifacts, as demonstrated in Figure 6.

Additionally, the blending of Gaussians at object boundaries can introduce chromatic aberrations, which hinder high-fidelity reconstruction in scenes with complex structural details, as shown in Figure 4.

Furthermore, in regions with sparse or no training views, black specks (Gaussian artifacts) were observed in areas devoid of objects, as illustrated in Figure 10a.

We hypothesize that these issues arise because the Gaussians are optimized specifically for rendering the training scenes, without fully capturing the true 3D structure of the environment.

Figure 5: A fine view of part of a scene

Figure 6: Another angle from the same scene produces blurry artifacts.

Figure 7: First try of filling gaps in images with Stable Diffusion with the prompt:" fill in the missing gap sections". The input image is figure 2, the bike is not present anymore.

Figure 8: Result with a more fine-tuned prompt: "Fill in the black sky". The bike is back again

5 Extensions

To address the limitations outlined in Section 4, we investigated several potential improvements. First, we tackled the issue of black patches by employing generative image techniques (see Section 5.1). Additionally, we incorporated depth regularization as an auxiliary loss term to enhance scene consistency (see Section 5.2). To experiment on improving the method's speed, we modified the initial structure-from-motion (SfM) solver (see Section 5.3). Finally, we adjusted color-related and input image parameters in an attempt to mitigate the color and boundary issues observed in the rendered scenes (see Section 5.4).

5.1 Fixing the Black Patches Issue with Generative Image Generation

As discussed in Section 4.1, under-represented viewpoints often result in black patches. To address this in a post-processing step, we experimented with generative image libraries like Stable Diffusion [Rombach et al. 2022] to fill in the gaps for key images. Initially, on Figure 2 we used a fixed prompt—"fill in the missing black sections"—to seamlessly complete the image. However, this approach produced incorrect results, as it inadvertently removed objects such as bicycle tires and benches (see Figure 7). We suspect the model interpreted the black areas too broadly.

To improve the output, we refined the prompt to make it more context-specific, changing it to "fill in the missing black sky". This adjustment led to a more satisfactory result, as it focused the model's attention on the sky, preserving other key elements of the image (see Figure 8).

While this method offers some improvement, it has significant limitations: it can only be applied to one image at a time at the end of the pipeline and must be repeated for each new viewpoint. A potential solution to this issue would involve retraining the model using these example-filled images. However, developing the algorithm to find the appropriate prompts and integrating these images into the training process is beyond the scope of this paper, though it presents a promising direction for future work.

(a) X_p **(b)** $\widetilde{\mathcal{D}}(X_p)$

Figure 9: A view and its estimated monocular depth map, obtained with Depth Anything v2.

5.2 Depth regularization

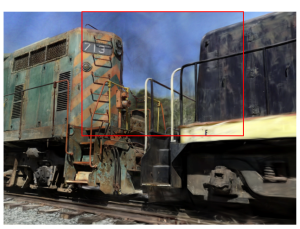
Based on Section 4.3 we hypothesized that ensuring proper depth positioning could resolve the problem by arranging the Gaussians in a more structured manner and eliminating floating artifacts when they don't align with actual geometry. One approach to achieve this could be to infer the depth of views and adjust the Gaussian geometry accordingly. So we wanted to add a depth regularization terms in the loss to account to some depth information.

To resolve the issue we noticed in Figure 10a, we used a depth normalization term in the loss as proposed by the authors of DNGaussian [Li et al. 2024], based on a pre-trained monocular depth estimator. See Figure 9 for the depth maps we created during the process.

We ran a training of a scene with and without a depth regularization term in the loss and rendered the results in Figure 10. We observed that some floating Gaussians, that can be interpreted as unwanted noise, were removed. We think this is an expected result of the depth regularization, as those weren't located near any object, they would introduce a high depth-wise loss term and their opacity may have then been lowered, leading to their culling in the training process.

5.3 Modifying the SfM Solver

As described in Section 3.1, an SfM algorithm must be applied to the input images. In scenarios where time is critical, such as online mapping in robotics [Zhu et al. 2024], users often need to repeatedly train Gaussian Splats. However, for online use cases, where real-time performance is essential, we must explore ways



(a) Rendering without depth regularization.

(b) Rendering with depth regularization.

Figure 10: Scenes rendered without and with depth regularization in the loss. We observe on the left some black noise between the cars that doesn't appear on the right.

to achieve incremental speedups. This motivates our investigation into optimizations that could accelerate Gaussian Splats without sacrificing the quality of the results.

COLMAP [Schönberger and Frahm 2016] was used as the Structure from Motion (SfM) solver by the authors in the original paper[Kerbl et al. 2023] to handle the camera calibration and 3D point cloud generation. COLMAP's output provides the necessary geometric information to initialize and optimize the Gaussians. However, since this algorithm is an incremental SfM algorithm, the cost of outputting high-quality 3D points is time. We compared this SfM algorithm with a global SfM algorithm GLOMAP [Pan et al. 2025], which is a faster algorithm to perform SfM, see table 1. The key difference between incremental COLMAP and global GLOMAP lies in when the bundle adjustment—a refinement of 3D point positions across different views—is performed. The incremental approach alternates iteratively between estimating absolute camera poses, performing triangulation, and applying bundle adjustment. In contrast, global methods solve the camera geometry for all input images simultaneously, prioritizing speed over reconstruction quality. Since the GLOMAP algorithm has not been speed-benchmarked on datasets commonly used for Gaussian Splatting, we conducted our own benchmarking on such a dataset.

5.3.1 GLOMAP Versus COLMAP Runtime. To evaluate GLOMAP against the default implementation of COLMAP, we ran both algorithms on the entire Mip-NeRF 360 dataset. This dataset consists of high-quality images from 9 challenging outdoor 3D scenes, each with a 360-degree field of view [Barron et al. 2022]. It is widely used as a benchmark, including in the original paper [Kerbl et al. 2023]. The results, summarized in Table 1, show that GLOMAP delivers an average speedup of 293% over COLMAP. In all cases, GLOMAP significantly outperforms COLMAP, offering substantial reductions in runtime. For example, the "Kitchen" scene demonstrates the largest improvement, with GLOMAP reducing the processing time from 1176.02 seconds to just 173 seconds, achieving a remarkable speedup of 579.8%.

All experiments were conducted on an ArchLinux system equiped with an AMD EPYC 7773X processor, 2TB of RAM and a single NVIDIA A4000 16GB GPU.

Table 1: COLMAP and GLOMAP mapper run times (with default settings) for the Mip-NeRF 360 dataset.

Dataset	COLMAP (s)	GLOMAP (s)	Speedup (%)
Bicycle	73.74	**40.33**	82.8%
Bonsai	998.32	**232.32**	329.7%
Counter	605.83	**114.03**	431.3%
Flowers	68.48	**40.42**	69.4%
Garden	555.17	**94.10**	490.0%
Kitchen	1176.02	**173.00**	579.8%
Room	727.07	**168.50**	331.5%
Stump	50.16	**18.00**	178.7%
Treehill	120.26	**48.40**	148.5%
Average	**486**	**103**	**293**

5.3.2 Preparing A More Robust Comparison. In the original paper, the empirical test performed did not take into account how robust the algorithm was to various parameters of the SfM initialization. As the end user may have different preferences for these parameters, we experimented with 10 different initial parameter configurations for our comparison in section 5.3.3. These adjustments were intended to emulate and capture a broader range of use cases and variations.

5.3.3 Comparison of using GLOMAP instead of COLMAP in the Gaussian Splatting pipeline. As the ground truth images from Mip-NeRF 360 do not have XYZ/GPS Data, we compared between the output of Gaussian splatting using the two mappers across 10 initialization trials. We used the COLMAP initialization as our pseudo-ground truth data. To quantify these differences, we employed the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM). Together, these metrics provide a comprehensive assessment of both pixel-wise accuracy and the preservation of structural details. We then compute the average of this metric for 100 randomly chosen viewpoints in each of the 10 initializations and present the results in Table 2.

Table 2: Comparison of 3D Gaussian splatting outputs using COLMAP and GLOMAP over 100 rendered viewpoints, with 10 different SfM initialization settings. COLMAP output is used as pseudo-ground truth.

Dataset	PSNR (db)	SSIM
Bicycle	24.58 ± 3.2	0.742 ± 0.18
Bonsai	25.32 ± 1.5	0.676 ± 0.19
Counter	23.74 ± 2.1	0.648 ± 0.20
Flowers	26.15 ± 3.7	0.797 ± 0.07
Garden	24.98 ± 2.3	0.703 ± 0.16
Kitchen	22.57 ± 4.0	0.651 ± 0.19
Room	27.04 ± 3.1	0.778 ± 0.17
Stump	23.22 ± 2.8	0.611 ± 0.20
Treehill	25.63 ± 3.5	0.783 ± 0.15

We hypothesize that the differences in rendering of the background contribute heavily to the discrepancies observed in the

Figure 11: Looking at some differences in the background (on the right) between the generated COLMAP(above) and GLOMAP(below) initialized images

(a) Baseline **(b) 2 degree of spherical harmonics**

Figure 12: Scene rendered changing the number of spherical harmonics. No major changes can be seen.

results presented in Table 2. Specifically, we suspect that the initialization plays a critical role in determining where the majority of the "information" is concentrated. The initialization tends to focus on the foreground, as both initializations are heavily weighted towards capturing foreground features due to the SIFT computation on this given dataset. In contrast, the background contains fewer distinctive points, leading to less detailed representations. An example illustrating this phenomenon in a scene viewpoint can be seen in Figure 11, where we observe differences in certain background elements.

Given that the resulting metrics and visuals of Gaussian Splatting using GLOMAP show competitive performance, particularly in terms of percentage time speedup compared to COLMAP, GLOMAP presents a promising alternative for accelerating computation.

5.4 Color and Boundary Repair Experiments

5.4.1 Adjusting Spherical Harmonics Parameter. As discussed in Section 4.1, spherical harmonics can introduce chromatic aberrations or blurry artifacts. To address this issue, we experimented with modifying the degree of spherical harmonics for each Gaussian, testing values of 1 and 2. However, overall, we did not observe significant visual differences (see for example Figure 12). We hypothesize that the large number of Gaussians in the scene (approximately 1.2 million) compensates for any potential loss of color information.

5.4.2 Effects of adding noise on images before training. In this section, we explored the impact of introducing noise to the input

(a) no noise **(b) $p = 0.005$** **(c) $p = 0.2$**

Figure 13: Scenes rendered with varying levels of salt-and-pepper noise. As the noise probability p increases, the boundaries become smoother and more defined, enhancing clarity. However, excessive noise (e.g., $p = 0.2$) leads to significant loss of detail and color fidelity, resulting in desaturation and visual degradation across the image.

images. Specifically, we added salt-and-pepper noise [Jayaraman et al. 2009] with probabilities $p = 0$, $p = 0.05$, and $p = 0.2$. The resulting loss values were affected, as illustrated in Figure 14. While some colors became slightly desaturated, the 3D scenes were still reconstructed correctly. We hypothesize that the diversity of input images compensates for the added noise, suggesting that the quantity of images plays a more critical role in maintaining reconstruction quality than the individual image fidelity.

Furthermore, as discussed in Section 4.2, we hypothesized that adding Gaussian noise to colors could act as a smoothing parameter by distorting details. This smoothing effect could potentially soften hard edges, thereby simplifying the geometry for the Gaussians to learn. Our observations in Figure 13 confirmed that adding noise with a small magnitude helped better reconstruct edges, such as those between the box, the door and the wall, as well as the upper part of the door. However, increasing the noise magnitude too much ultimately degraded the geometry of the edges. Based on these findings, we conclude that introducing noise to input images can serve as a useful parameter for improving the algorithm's ability to learn sharp geometries at a relatively low computational cost.

6 Conclusion

This paper provides a comprehensive analysis of 3D Gaussian Splatting, highlighting its limitations and proposing several extensions to address these challenges. Key limitations identified include the appearance of black patches in unobserved regions, difficulties in rendering sharp boundaries, and color artifacts in novel viewpoints. To mitigate some of these issues, we proposed a variety of solutions, such as using generative image techniques to fill in missing viewpoints, incorporating depth regularization during training, modifying the structure-from-motion (SfM) solver, adjusting spherical harmonics to reduce artifacts, and introducing noise to input images to test the model's robustness. For future work, we plan to investigate the potential of Gaussian Splatting for 3D mesh reconstruction, seeking to extend its utility beyond rendering tasks.

References

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings*

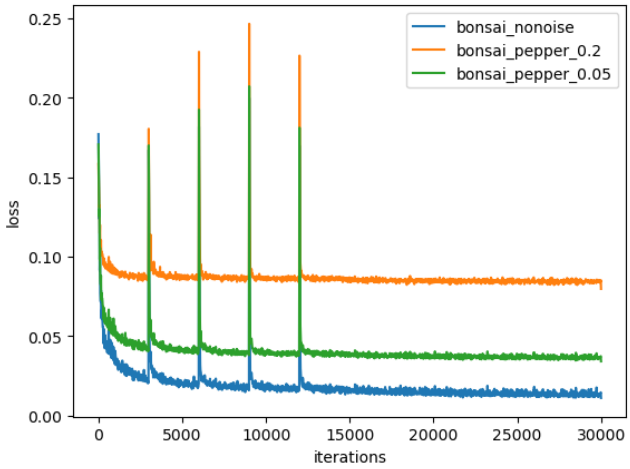


Figure 14: L1 loss for the different noise

of the IEEE/CVF conference on computer vision and pattern recognition. 5470–5479.

Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. 2005. High-quality surface splatting on today's GPUs. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* IEEE, 17–141.

Kunal Chelani, Assia Benbihi, Torsten Sattler, and Fredrik Kahl. 2024. EdgeGaussians – 3D Edge Mapping via Gaussian Splatting. arXiv:2409.12886 [cs.CV] https://arxiv.org/abs/2409.12886

Linus Franke, Laura Fink, and Marc Stamminger. 2024. VR-Splatting: Foveated Radiance Field Rendering via 3D Gaussian Splatting and Neural Points. arXiv:2410.17932 [cs.CV] https://arxiv.org/abs/2410.17932

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5501–5510.

Robin Green. 2003. Spherical harmonic lighting: The gritty details. In *Archives of the game developers conference*, Vol. 56. 4.

Jeffrey P Grossman and William J Dally. 1998. Point sample rendering. In *Rendering Techniques' 98: Proceedings of the Eurographics Workshop in Vienna, Austria, June 29—July 1, 1998 9*. Springer, 181–192.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 1–15.

Subramania Jayaraman, S Esakkirajan, and T Veerakumar. 2009. *Digital image processing*. Vol. 7014. Tata McGraw Hill Education New Delhi.

Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. 2024. VR-GS: a physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*. 1–1.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. 2024. DNGaussian: Optimizing Sparse-View 3D Gaussian Radiance Fields with Global-Local Depth Normalization. https://doi.org/10.48550/arXiv.2403.06912 arXiv:2403.06912.

Weikai Lin, Yu Feng, and Yuhao Zhu. 2024. RTGS: Enabling Real-Time Gaussian Splatting on Mobile Devices Using Efficiency-Guided Pruning and Foveated Rendering. arXiv:2407.00435 [cs.GR] https://arxiv.org/abs/2407.00435

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *CoRR* abs/2003.08934 (2020). arXiv:2003.08934 https://arxiv.org/abs/2003.08934

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

Emmanouil Nikolakakis, Utkarsh Gupta, Jonathan Vengosh, Justin Bui, and Razvan Marinescu. 2024. GaSpCT: Gaussian Splatting for Novel CT Projection View Synthesis. arXiv:2404.03126 [eess.IV] https://arxiv.org/abs/2404.03126

Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes L Schönberger. 2025. Global structure-from-motion revisited. In *European Conference on Computer Vision*. Springer, 58–77.

Haoxuan Qu, Zhuoling Li, Hossein Rahmani, Yujun Cai, and Jun Liu. 2024. DisC-GS: Discontinuity-aware Gaussian Splatting. arXiv:2405.15196 [cs.CV] https://arxiv.org/abs/2405.15196

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

Darius Rückert, Linus Franke, and Marc Stamminger. 2022. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 1–14.

Jules et al Salzinger. 2024. Pushing the limit to near real-time indoor LiDAR-based semantic segmentation. (2024).

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. 2024. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1606–1616.

Hongjia Zhai, Xiyu Zhang, Boming Zhao, Hai Li, Yijia He, Zhaopeng Cui, Hujun Bao, and Guofeng Zhang. 2024. Splatloc: 3d gaussian splatting-based visual localization for augmented reality. *arXiv preprint arXiv:2409.14067* (2024).

Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. 2022. Differentiable point-based radiance fields for efficient view synthesis. In *SIGGRAPH Asia 2022 Conference Papers*. 1–12.

Siting Zhu, Guangming Wang, Dezhi Kong, and Hesheng Wang. 2024. 3D Gaussian Splatting in Robotics: A Survey. arXiv:2410.12262 [cs.RO] https://arxiv.org/abs/2410.12262

Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001a. EWA volume splatting. In *Proceedings Visualization, 2001. VIS'01.* IEEE, 29–538.

Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001b. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 371–378.