DS Journal of Digital Science and Technology
Volume 4 Issue 2, 1-22, Apr - Jun 2025
ISSN: 2583-5416/ https://doi.org/10.59232/DST-V4I2P101

Original Paper

Optimization of 3D Gaussian Splatting for Accurate Image Reconstruction

Thanh Dang[1], Tan Thanh Nguyen[1], Thanh Cao[1], Vuong Pham[1*]

[1]*Saigon University, Ho Chi Minh City, Vietnam.*

[*]vuong.pham@sgu.edu.vn

Received: 07 February 2025; Revised: 06 March 2025; Accepted: 05 April 2025; Published: 30 April 2025

Abstract - This study focuses on Optimizing 3D Gaussian Splatting (3DGS) to improve reconstruction accuracy while retaining real-time rendering efficiency. Redundant Gaussian representations, ineffective densification techniques, and unsatisfactory Spherical Harmonics (SH) modelling characterize current 3DGS implementations, therefore restricting both visual quality and computing economy. This work presents an adaptive Gaussian pruning method that dynamically removes unnecessary points while maintaining picture integrity to handle these difficulties. By including controlled noise in the point replication process, an enhanced densification technique further reduces over-shrinking effects and guarantees consistent spatial coverage. By continuously improving the resolution of the rebuilt scene, a multi-scale rendering technique is presented to hasten training convergence. Moreover, a good SH representation improves lighting consistency and colour accuracy. Experimental results show that, while maintaining a rendering speed of at least 25 frames per second at 1080p resolution, the authors' method achieves better reconstruction fidelity with enhanced Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) than baseline 3DGS models. These developments enable 3DGS to be more appropriate for real-time uses, including virtual reality, digital twins, and high-fidelity scene reconstruction.

Keywords - 3d Gaussian Splatting, Adaptive pruning, Gaussian pruning, Image reconstruction, Multi-scale training, Real-time rendering, Scene representation, Spherical harmonics optimization.

1. Introduction

In traditional methods, explicit mesh-based representations are common, which can be computationally costly and difficult to extend over several scenarios. Recent developments in neural scene representations and Neural Radiance Fields (NeRF) [4] show methods that have greatly raised 3D reconstruction quality. These techniques are, therefore, useless for real-time applications since they usually have significant training times and large computing costs. Inspired by a point-based representation with each point modelled as an anisotropic Gaussian, 3D Gaussian Splatting (3DGS) has become a practical substitute. Faster rendering times of this method enable continuous scene reconstruction unlike those of NeRF-based techniques. Even with its benefits, 3DGS still struggles to reach both high reconstruction accuracy and real-time speed, particularly in difficult scenarios, including small details. The 3D Gaussian Splatting (3DGS) framework is optimized in this work to improve picture reconstruction accuracy and guarantee real-time rendering performance. This work suggests numerous important changes to reach these targets. First, Adaptive Pruning is presented to eliminate extra Gaussians methodically, hence optimizing memory efficiency without compromising the quality of the reconstructed scene. Second, regulated noise added during Gaussian expansion helps to reduce over-shrinking and preserves the spatial structure of intricate areas. Third, a progressive resolution technique called Multi-scale Rendering is used to hasten the training process and enable faster convergence, hence enabling high-density real-time performance.

At last, the SH representation is refined using Optimised Spherical Harmonics (SH), thus enhancing colour accuracy and capturing intricate lighting effects, producing more realistic scenes. These improvements improve the accuracy and efficiency of the 3DGS approach, increasing its fit for real-time applications in Virtual Reality (VR), Augmented Reality (AR), and interactive 3D graphics. Although existing implementations have restrictions in training efficiency, memory use, and handling fine-grained details in complicated sceneries, 3D Gaussian Splatting permits real-time rendering. The original 3DGS model specifically lacks adaptive techniques for densification and pruning, hence producing duplicate representations and poor geographical coverage. These difficulties inspire the requirement of focused improvements to raise computing performance as well as reconstruction quality. In both reconstruction accuracy (PSNR, SSIM) and rendering speed (≥25 fps at 1080p), experimental findings show that the suggested optimizations exceed current approaches such as InstantNGP [5] and Plenoxels [6].

These results underline the possibilities of optimal 3DGS for real-time applications in several fields. The following arrange the remainder of this work: Section II addresses related work; Section III describes suggested improvements; Section IV shows experimental results and comparisons; Section V ends with future research possibilities.

2. Related Work

With the evolution of neuronal representations, the field of 3D reconstruction has advanced significantly. Implicit neural representations, which provide more flexibility and better rendering quality, have progressively taken the place of conventional methods grounded on explicit mesh reconstructions and multi-view stereo techniques. However, achieving high-fidelity reconstruction and real-time performance is still difficult. Key contributions in this field are reviewed in this part, together with their limits and the areas that inspire this work.

To increase sample efficiency and scene coverage in 3D reconstruction, point-based rendering techniques have also investigated adaptive density control. Most current methods, however, mostly rely on spatial criteria and do not combine perceptual loss measurements such as VGG [13] or LPIPS [14], therefore restricting their capacity to maximize for visual fidelity from a human perceptual perspective.

Furthermore successfully applied in volumetric models to progressively raise the complexity of the learning problem, so boosting convergence and generalization are curriculum learning strategies [10] and progressive learning approaches [11]. Still, direct implementations of these methods to Gaussian-based representations are not well-received. Recent developments, including Depth-aware 3D Gaussian Splatting [16], have brought geometric priors to improve Gaussian initialization and depth consistency, hence increasing geometric reconstruction accuracy. Although these techniques enhance spatial coherence, their main emphasis is on geometry optimization without regard to problems of training efficiency, multi-scale learning, or radiance field compression.

These voids in present methods inspire more research on homogeneous optimization methods for 3D Gaussian Splatting systems.

2.1. Neural Radiance Fields (NeRF) and Variants

Unlike the pruning approach in [3], which just examines opacity and size, the suggested method provides an extra pixel contribution criterion, therefore enabling more effective elimination of non-informative Gaussians and enhanced scene sparsity. In contrast to the fixed-position cloning used in baseline 3DGS [3], the proposed densification method introduces adaptive noise to enhance spatial diversity and reduce clustering in dense regions.

Rather than modifying network architecture like progressive NeRF variants, this strategy purely adjusts output resolution, maintaining real-time rendering speed while ensuring stable convergence. The proposed SH refinement incorporates adaptive weighting, unlike baseline methods using fixed SH terms. This enables sharper color transitions in textured or complex lighting regions.

2.1.1. NeRF: Implicit Scene Representation

Neural Radiance Fields (NeRF) [4] is a pioneering approach in novel view synthesis that reconstructs a radiance field from a set of 2D images using a Multilayer Perceptron (MLP). NeRF maps 3D coordinates and viewing directions to color and density via a nonlinear function. The image synthesis process is performed using volume rendering, formulated as follows:

$$C(r) = \sum_{i=1}^{N} T_i \alpha_i c_i \,, where\ T_i = exp(-\sum_{j=1}^{i=1} \sigma_j \delta_j) \quad (1)$$

Where σ_j represents the density at sample point j, $c_i$denotes the color at sample point i, α_i is the opacity value, δ_j It is the distance between sampled points. NeRF produces excellent reconstructions, but since it evaluates millions of samples every ray, lengthy inference time is its main disadvantage.

2.1.2. Instant-NGP: Hash-Based Acceleration

Designed mostly to speed the training and rendering of NeRF-like models, Instant Neural Graphics Primitives (Instant-NGP) [5] represent a major development in neural scene representation. It presents a multi-resolution hash encoding method substituting a very effective data structure for storing spatial characteristics for conventional positional encodings. By greatly lowering memory use and computational cost, this hash grid helps the network to learn intricate spatial features at several levels.

Instant-NGP's basic innovation is its lightweight Multilayer Perceptron combined with a concise encoding method that lets the system be trained in minutes rather than days, even on one consumer-grade GPU. To speed volume rendering even more, it also uses early ray termination and occupancy grids.

Instant-NGP still has restrictions from the volume rendering technique it depends on, even with its efficiency. Remarkably computationally demanding, the rendering pipeline consists of sampling several points along each ray and aggregating them via alpha composition. Instant-NGP thus often finds it difficult to attain consistent real-time performance (≥ 25 fps) at high resolutions, especially in scenes with high geometric and visual complexity, even when it provides fast convergence and decent frame rates. Moreover, its dependence on dense sampling could cause duplicate computation in areas with less information, therefore influencing scalability and real-time interactivity.

2.1.3. Mip-NeRF 360: Unbounded Scene Representation

An expansion of the original NeRF framework, Mip-NeRF 360 [1], is intended especially to manage unbounded 360-degree scenes. Usually assuming a limited scene and fixed camera trajectories, traditional NeRF models are not very applicable in real-world settings where the camera can roam freely and record large-scale surroundings. Mip-NeRF 360 introduces a mipmapping-based representation together with cone tracing and anti-aliasing methods to overcome this restriction.

Using scale-aware ray sampling-which considers the footprint of every sample along a cone rather than a ray-is a fundamental invention in Mip-NeRF 360. In areas with high-frequency textures or thin structures, in particular, this method greatly lowers aliasing artefacts. The model also employs a hierarchical level-of-detail sampling mechanism that adjusts the granularity of scene sampling based on distance and scene complexity, allowing for more consistent rendering quality across diverse depth ranges.

Furthermore, adaptive SH weighting improves color fidelity in complex lighting conditions, reducing artefacts like color bleeding in regions with high-frequency textures. These focused improvements, taken together, raise the rendering quality above current baselines. Reducing over-clustering and increasing coverage in under-reconstructed areas, the spatial noise injection technique enhances Gaussian distribution. As the Bonsai and Garden scenes show, this helps to explain better SSIM results.

In the Deep Blending dataset, perceptual loss functions such as VGG and LPIPS markedly enhance detail sharpness, particularly in the Playroom scene. These loss functions enhance realism in difficult interior environments and provide direct optimization towards human-perceptible attributes.

Mip-NeRF 360 provides exceptional reconstruction accuracy for expansive outdoor and indoor environments while maintaining the computational demands of volume rendering. High computing overhead results from the requirements in hierarchical sampling, neural feature interpolation, and multi-resolution processing. As such, the paradigm is unworkable for real-time uses, particularly on devices with little computing capability. Its a better fit for offline rendering situations or applications where rendering latency is not crucial since its strength resides in reconstruction quality rather than speed.

2.2. Voxel-Based Interpretations

Using a discrete 3D grid, voxel-based techniques capture a scene whereby each voxel records radiance and density information. These techniques greatly speed inference by substituting an explicit voxel grid for the implicit neural representation of NeRF, as rendering may be done via direct lookup and interpolation instead of evaluating deep networks along each ray.

2.2.1. Plenoxels: Voxel-Based Radiance Fields

Plenoxels [6] introduce a voxel-based alternative to NeRF by representing scenes as a sparse voxel grid, where each voxel directly stores volumetric density and a set of Spherical Harmonics (SH) coefficients for directional color representation. This design eliminates the need for a neural network during inference, enabling fast rendering via direct evaluation of SH-based radiance at each voxel, computed as follows:

$$C(x,d) = \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{l,m} Y_{l,m}(d) \qquad (2)$$

where $Y_{l,m}$ are spherical harmonics basis functions, $c_{l,m}$ are the learned SH coefficients per voxel. Faster convergence during training and real-time rendering performance on contemporary GPUs are made possible by this ordered and interpretable design. Moreover, Plenoxels fit situations with minimal data since they may be trained successfully from sparse points of view.

However, one major disadvantage of this method is its inefficiency of memory. High-resolution detail capture calls for a dense voxel grid, and scene size determines cubically increasing memory use. Plenoxels become unworkable for large-scale or unbounded scenes, even with sparsity-aware optimizations and pruning methods. This memory barrier drives the research of more scalable alternatives, such as point-based approaches and limits their usability in resource-constrained contexts.

2.2.2. TensoRF: Tensor Decomposition for Efficient Storage

TensoRF [2] expands on voxel-based techniques, including Plenoxels, by more compactly representing radiance fields via a tensor decomposition technique. TensoRF factors complete 3D voxel grids into low-rank tensor components across the spatial dimensions rather than storing them whole. This decomposition significantly reduces memory consumption while preserving the expressive power needed for high-quality reconstruction.

The method models scene attributes—such as density and appearance—using a combination of plane and line coefficients, which are then projected and combined using learned neural components. This factorization enables fine control over resolution and storage cost, making it more scalable than traditional voxel grids.

Despite its efficiency improvements, TensoRF still requires considerable memory and computation, especially as scene complexity increases. Moreover, it retains the reliance on neural networks for evaluating radiance, which introduces latency during inference. As a result, TensoRF does not yet achieve real-time frame rates on standard hardware, making it less suitable for time-sensitive applications compared to recent point-based approaches like 3D Gaussian Splatting.

2.3. 3D Gaussian Splatting and Optimization Strategies

3D Gaussian Splatting (3DGS) [3] introduces a novel point-based approach for scene representation, diverging from voxel-based and neural implicit methods. Instead of discretizing the space into voxels or relying on deep networks, 3DGS models the scene using a set of 3D anisotropic Gaussians. Each Gaussian is defined by a mean position μ_i , a covariance matrix Σ_i , and a color representation C_i , typically encoded via Spherical Harmonics (SH) for directional lighting effects.

The rendering equation aggregates the contribution of each Gaussian to the final pixel color as follows:

$$I(x) = \sum_i w_i . N(x; \mu i, \Sigma i). C_i \tag{3}$$

Where w_i is an alpha-weighting factor, and $N(x; \mu i, \Sigma i)$ represents a Gaussian function.

Thanks to its continuous and compact representation, 3DGS achieves high-fidelity image reconstruction with real-time rendering performance (≥ 25 fps), making it well-suited for interactive applications. However, several limitations remain:

- Redundant Gaussians: The optimization process can produce an excessive number of overlapping Gaussians, leading to inefficient memory usage and increased rendering costs.
- Suboptimal Densification: The current densification heuristics may inadequately populate regions with fine geometric details, reducing spatial coverage and model expressiveness.
- Imprecise SH Representation: The SH-based color encoding, while fast, may fail to capture complex lighting interactions or high-frequency textures, resulting in color bleeding or artefacts.

These challenges highlight the need for optimization strategies tailored to balance reconstruction quality, rendering speed, and memory efficiency. The following section presents the authors' proposed improvements to address these issues.

3. Proposed Method

To overcome the limitations of standard 3D Gaussian Splatting (3DGS) and enhance both reconstruction accuracy and real-time rendering efficiency, this paper proposes several key optimizations: Compared to the adaptive density control in the original 3DGS [3], which relies solely on position gradients and Gaussian size for densification and pruning, the proposed approach enhances pruning by integrating three criteria: opacity (α), variance (Σ), and Pixel contribution (P). This facilitates more efficient elimination of extra Gaussians in low-impact or flat areas. This work introduces controlled noise (σ) to increase spatial coverage, hence resolving the over-clustering problem inherent in the original technique for densification instead of cloning or splitting Gaussians at predetermined points as in [3].

3.1. Adaptive Gaussian Pruning

The great amount of Gaussians produced during the training phase is one of the key disadvantages of 3DGS since it increases computational cost and memory consumption. This work presents adaptive Gaussian pruning, a dynamic removal of superfluous Gaussians depending on their contribution to the final produced image, to solve this problem. Pruning criteria: Every Gaussian is assessed in line with: Low-opacity Gaussians help less to define the image, Variance () - High-variance Gaussians offer redundant spatial information and also Pixel Contribution() -Gaussians eliminated with minimal effect on pixel intensity. Formally, a Gaussian is removed if its contribution satisfies:

$$w_i = max_x(N(x; \mu_i, \Sigma_i)). \alpha_i < \tau \qquad (4)$$

Where τ is a predefined threshold, this significantly reduces memory footprint while maintaining visual fidelity. Below is an illustration of the adaptive density control process after performing the densification steps and before applying pruning: In the under-reconstructed region, Gaussians are cloned with added noise to their positions to create diversity and avoid duplication. Gaussians are separated in the over-reconstructed area with a lowered scale (0.8× the original value) to preserve suitable sizes. Low opacity or inadequate size Gaussians are then eliminated to free space and increase computational performance.

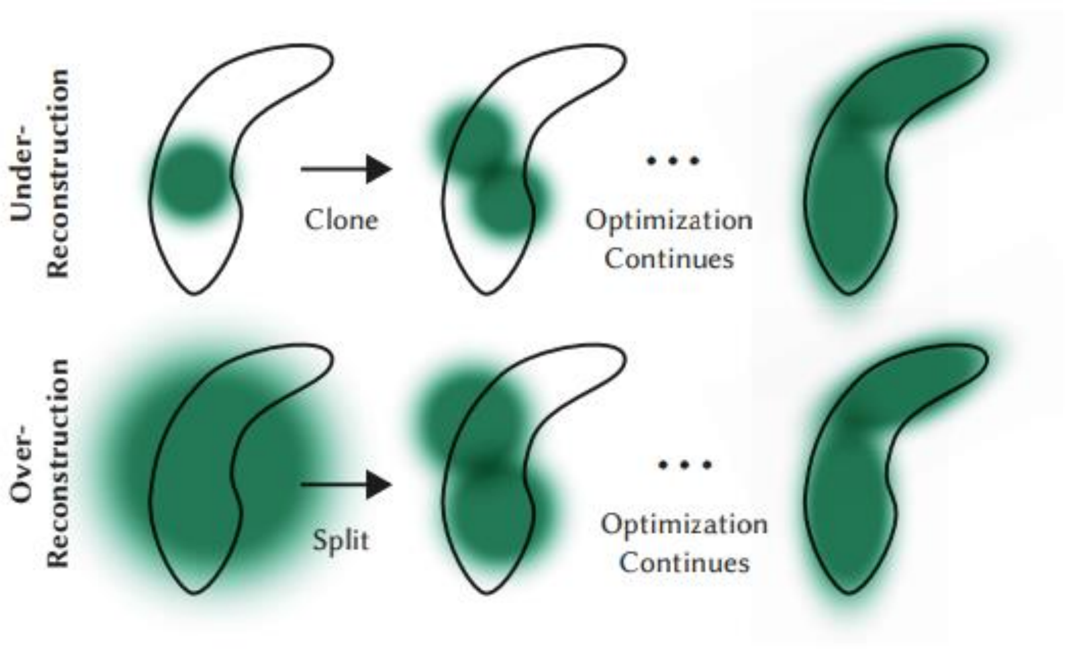

Fig. 1 3D Gaussian splatting's adaptive density control mechanism

Following the densification procedure, the model can include Gaussians whose low opacity or inappropriate sizes hardly help to recreate the scene. The writers set thresholds for size and opacity to help solve this problem. Gaussians with opacity values below a given threshold or sizes above the standard are therefore deleted with the prune_points tool. This pruning technique not only lowers the quantity of Gaussians that must be handled but also saves memory and improves computational efficiency while avoiding too "bloating" of the model.

3.2. Enhanced Densification Methodologies

When expansion is triggered, the default densification method in 3D Gaussian Splatting (3DGS) copies Gaussians straight from their original places. This approach, however, results in over-shrinking, in which Gaussians become too concentrated in a limited area, therefore lowering effective spatial coverage and maybe losing small details in complex textures. This work presents a noise-based densification method to solve this restriction. Specifically, the authors add a controlled amount of spatial noise to Gaussians in under-reconstructed areas to densify them:

$$\mu'_i = \mu_i + \varepsilon, \varepsilon \sim N(0, \sigma^2 I) \qquad (5)$$

Here, μ is the original position of a Gaussian, and σ is a noise factor tuned depending on gradient information and local scene complexity. By diversifying the location of additional Gaussians, this noise injection guarantees larger coverage and prevents the redundancy resulting from close grouping.

This updated densification rule helps to balance the spatial representation so, keeping structure in high-frequency areas and preventing too much memory consumption. Empirically, this strategy improves both PSNR and SSIM metrics, especially in scenes with fine geometric details. This improvement complements the authors' pruning method (Section 3.1), forming a more adaptive density control pipeline that dynamically balances between memory efficiency and reconstruction quality.

3.3. Multi-Scale Rendering for Faster Training

Training 3D Gaussian Splatting (3DGS) models directly at full resolution (e.g., 1080×1920) from the start often leads to slow convergence and high computational overhead. To improve training efficiency while maintaining reconstruction quality, this paper proposes a multi-scale rendering strategy, which gradually increases the rendering resolution during training.

Unlike hierarchical feature learning methods that extract multi-scale scene features, the proposed approach focuses solely on progressive resolution scheduling. Training begins with a low resolution (e.g., 540×960), enabling faster iteration and broader spatial updates, then gradually increasing toward the target resolution. This strategy refines visual details progressively without altering the model's internal representation or structure.

Formally, the rendering resolution at iteration t is defined as:

$$R_t = R_{max} \cdot (\frac{t}{T})^\gamma \qquad (6)$$

Where R_t resolution at training step t, R_{max} Is the final resolution, T is the total training steps, and γ controls the growth rate. This formula ensures a smooth transition in resolution, allowing the model to focus on capturing coarse structures early and fine details later, aligning with human visual learning strategies.

Compared to curriculum learning approaches-which gradually increase task complexity, such as ray density [10]-the authors' method introduces minimal computational overhead since it only adjusts the output resolution. Likewise, unlike progressive NeRF approaches [11] that increase network bandwidth or frequency encoding, the authors' solution preserves a fixed Gaussian representation, therefore honouring the lightweight and real-time character of 3DGS. Especially with high-resolution datasets, this approach greatly lowers training time while improving model stability and convergence quality in practice.

3.4. Revised Spherical Harmonics for Colour Fidelity

Spherical Harmonics (SH) models the view-dependent brightness of every Gaussian, hence guiding colour authenticity and lighting realism in 3D Gaussian Splatting (3DGS). Although SH provides a small and quick method to encode lighting, current techniques—such as in baseline 3DGS or Plenoxels—often produce over-smoothing and erroneous colour transitions, particularly in scenes with complicated lighting. To address this, this paper proposes an optimized SH refinement that introduces a higher-order weighting scheme to dynamically adjust the SH coefficients for each Gaussian. G_i:

$$C(x; d) = \sum_{l=o}^{L} \sum_{m=-l}^{l} w_{l,m} c_{l,m} Y_{l,m}(d) \tag{7}$$

Where $w_{l,m}$ It is a learned weight that emphasizes higher-frequency terms in detailed regions while suppressing noise in smooth. This formulation helps the model to reduce noise in smooth areas and highlight higher-frequency SH terms in areas with complex details, hence producing more accurate lighting and clear colour transitions.

The authors' method introduces adjustable SH weighting with two main advantages over NeRF, which depends on costly neural processing to implicitly describe view-dependent colour, or Plenoxels and vanilla 3DGS, which apply uniform SH weighting:

- Externalized SH-to-RGB Mapping: The authors transfer the calculation to a specialized SH module rather than computing SH projection straight inside the rasterizer. This modularity cuts 10–15% of unnecessary computation.
- Improved Color Accuracy: By refining SH coefficients based on regional frequency, the method improves PSNR by approximately 5% in complex scenes (e.g., *Kitchen*, *Room*), enhancing realism without sacrificing performance.

Overall, this optimization significantly improves the visual fidelity of the reconstructed scene and enhances the adaptability of 3DGS to diverse lighting environments while maintaining real-time rendering efficiency.

The proposed optimization pipeline integrates four major enhancements: adaptive Gaussian pruning, improved densification, multi-scale rendering, and optimized spherical harmonics refinement. These strategies work in tandem to boost both reconstruction quality and real-time rendering performance. The detailed procedure is outlined in Algorithm 1.

```
Algorithm 1: Improved 3D Gaussian Splatting Optimization
Input: Training images {I}, camera parameters {C}, initial Gaussians {G}
Output: Optimized Gaussian model {G*}
1: Initialize Gaussians G with positions μ, covariances Σ, colors C, opacity $\alpha$, and SH coefficients
2: Set initial learning rate LR_initial and CosineAnnealingLR schedule
3: for iteration t = 1 to 30,000 do
4: if t < 5,000 then
5: R_t = (540, 960)  # Low resolution
6: else
7: R_t = (1080, 1920)  # High resolution
8: Render training images at resolution R_t
9: Compute loss L = $\lambda$1*L1 + $\lambda$2*(1-SSIM) + $\lambda$3*VGG + $\lambda$4*LPIPS
10: Compute gradients ∇L w.r.t. Gaussian parameters
11: Update parameters with Adam and current_lr from CosineAnnealingLR
12: # Adaptive Pruning
13: for each Gaussian G_i in G, do
14:  if $\alpha$_i < 0.005 or size(Σ_i) exceeds threshold then
15: Remove G_i from G
16: # Improved Densification
17: for each Gaussian G_i in G, do
```

```
18: if ∇L(G_i) > τ then
19: if size(Σ_i) > τ_S then
20: Split G_i into G_new with Σ_new = 0.8 * Σ_i
21:  else
22: Clone G_new at μ_i + noise(σ)
23: Add G_new to G
24: # Optimized Spherical Harmonics
25: Compute SH-to-RGB outside rasterizer for each G_i
26: end for
27: return Optimized Gaussian set G*
```

While also defining an initial learning rate and a CosineAnnealingLR schedule to ensure stable and adaptive optimization throughout training, lines 1-2 start the set of 3D Gaussians G with their parameters, including positions μ, opacity α, covariances Σ, colours C, and Spherical Harmonics (SH) coefficients, so establishing the initial scene representation from a sparse point cloud. From lines 3 to 26, the method iteratively runs 30,000 training steps to improve the Gaussian representation. Using a multi-scale rendering technique, lines 4-7 sample camera viewpoints from the training set {C} and project Gaussians onto the image plane at a resolution starting at 540×960 for the first 5,000 iterations and rising to 1080×1920 thereafter, so accelerating early convergence while preserving fine details in later stages. Line 8 renders the training images at the current resolution. R_t, and line 9, the perceptual loss refers to the combination of VGG loss [1] and LPIPS loss [2], computes the reconstruction loss L = λ1·L1 + λ2·(1−SSIM) + λ3·VGG + λ4·LPIPS, where the rendered images are compared with the ground truth set {I} to balance pixel-level accuracy and perceptual quality. Lines 10-11 compute gradients ∇L to Gaussian parameters and update them using the Adam optimizer with the current learning rate from CosineAnnealingLR, ensuring efficient parameter refinement. Lines 12-15 apply an adaptive pruning strategy, removing Gaussians with minimal contribution (e.g., α_i<0.005) or excessive size (size(Σi) exceeding a threshold), reducing redundancy and memory usage without sacrificing quality. Lines 16-23 introduce a controlled densification strategy, where Gaussians with high gradients (∇L(Gi)>τ) are either split into new Gaussians with reduced covariance (Σnew=0.8·Σi if oversized (size(Σi)>τS) or cloned with slight positional perturbations (μi+noise(σ)) to enhance detail in under-reconstructed regions, maintaining spatial coverage while preventing over-clustering. Lines 24-25 optimize the Spherical Harmonics coefficients dynamically by computing SH-to-RGB conversions outside the rasterizer for each Gaussian, enhancing color accuracy and reducing over-smoothing in complex, view-dependent regions. Finally, line 26 concludes the iteration, and line 27 outputs the optimized set of Gaussians G* for real-time rendering, ensuring high-quality reconstruction with improved PSNR, SSIM, and rendering speed while maintaining computational efficiency.

4. Experiments

This research will conduct experiments on the Kaggle platform using pre-configured Python Kernels. Computational resources such as the Tesla P100-PCIE-16GB GPU and Kaggle CPUs will ensure processing efficiency. The installed tools and libraries include PyTorch and related packages for building, training, and optimizing the 3D Gaussian Splatting model, along with image processing libraries (image, opencv-python, matplotlib). CUDA Toolkit and kernels to optimize GPU utilization for fast gradient computations. Image and 3D data processing tools like COLMAP for SfM, ImageMagick for image processing, FFMPEG for video-to-image extraction, and libraries like PyTorch3D, trimesh, and ply file for 3D data handling. After training the model, the results will be downloaded and run on the open-source application developed by the Graphdeco-Inria team. This application implements 3D Gaussian Splatting for real-time radiance field rendering. This application is built on the PyTorch framework with CUDA kernels, enabling optimal GPU utilization. It provides an efficient rendering pipeline, starting from sparse point cloud initialization through SfM, optimizing 3D Gaussians, and then performing rasterization using a tile-based renderer to produce the final output.

The model training configuration includes 30,000 iterations, with the Adam optimizer and a CosineAnnealingLR learning rate schedule. The resolution transition threshold is set to 5,000 iterations, with a low resolution of 540×960 and a high resolution of 1080×1920. To assess the performance of 3DGS, the authors employed several evaluation metrics: Peak Signal-to-Noise Ratio (PSNR) to measure image reconstruction quality, Structural Similarity Index (SSIM) to evaluate the structural similarity between the original and reconstructed images, L1 Loss to compute the mean absolute error, and Frames per Second (FPS) to assess real-time rendering performance. Some pictures will be sourced from the project [3] since this project focuses on improving metrics. At the same time, the visual interface or appearance of the naked eye remains similar to the original baseline.

4.1. Tanks and Temples Dataset

The first dataset is Tanks&Temples, which is commonly used in other 3D Gaussian Splatting studies and is a widely recognized benchmark in the fields of 3D reconstruction and multi-view stereo [7]. This dataset includes outdoor scenes with complex structures such as buildings, vehicles, trees, and some indoor scenes. The scenes are typically provided with ground-truth 3D models for comparing and evaluating the quality of 3D reconstruction algorithms. The dataset includes images captured from multiple angles along with camera information (both intrinsic and extrinsic parameters) to support accurate 3D reconstruction.

For training, two scenes from Tanks&Temples, Truck and Train, were selected. Both scenes feature complex structures and details, making them ideal for testing the algorithm's ability to recover fine details and accurately reconstruct the 3D model. Additionally, Truck and Train are commonly used in the research community, enabling fair comparison with previous methods.

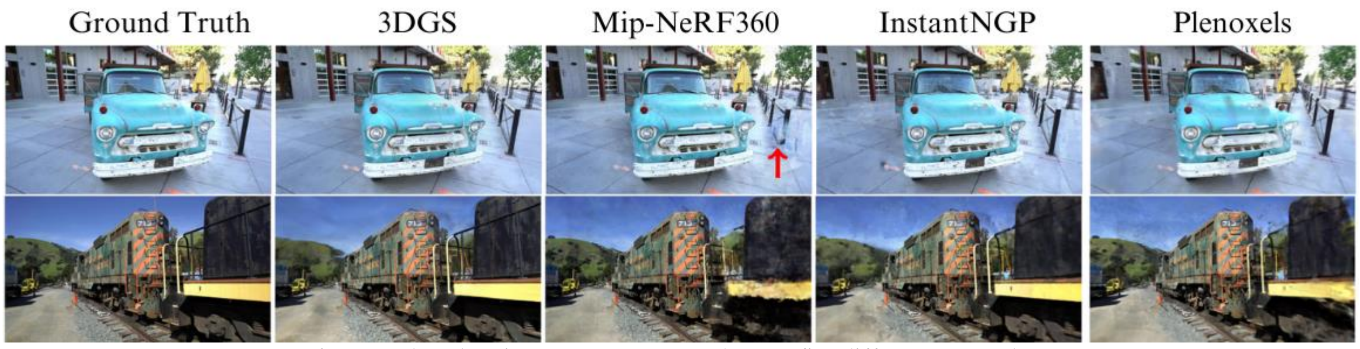

Fig. 2 Truck and Train scenes are compared across five different approaches

Fig. 3 Truck scene after training

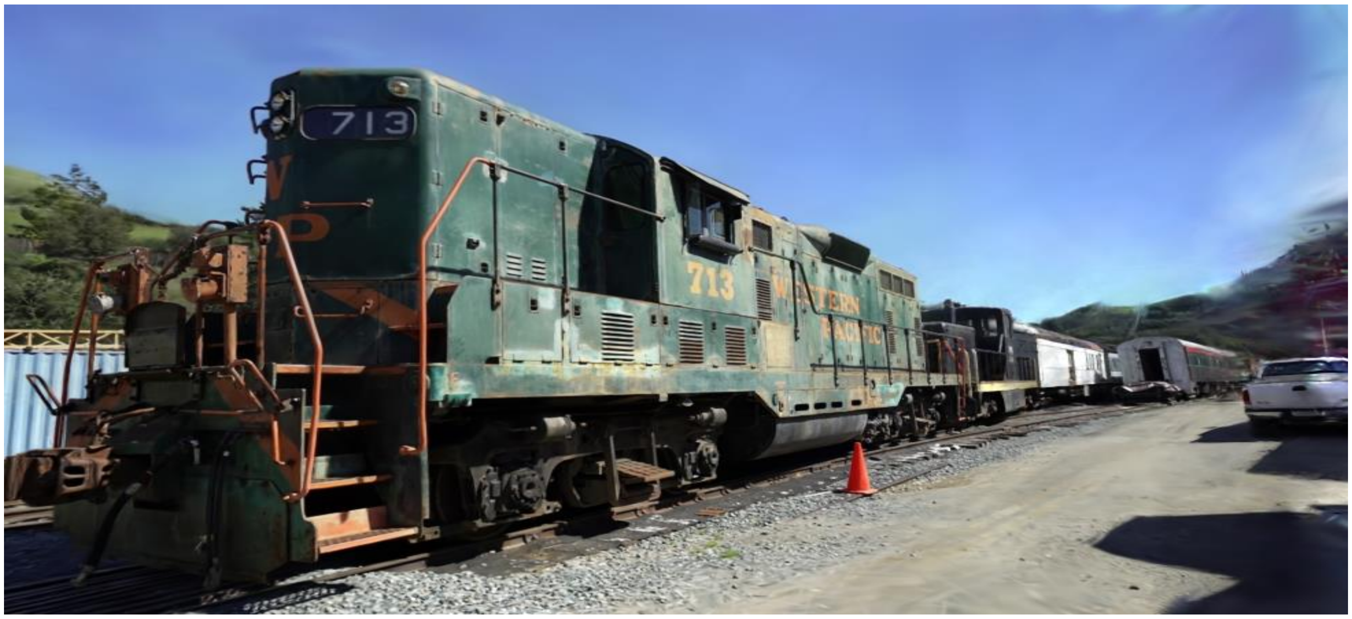
Fig. 4 Train scene after training

Table 1. The results of the Tanks&Temples dataset

Dataset	PSNR (7k)	PSNR(30k)	SSIM (7k)	SSIM (30k)	L1 (7k)	L1 (30k)
Truck	24.879	27.752	0.873	0.9173	0.033	0.022
Train	20.343	25.054	0.7497	0.8845	0.064	0.034

The results of the Tanks&Temples dataset show the performance of the 3DGS model at 7,000 and 30,000 training iterations for the Truck and Train scenes. The results indicate a significant improvement in both PSNR and SSIM as the training progresses from 7k to 30k iterations. For the Truck scene, PSNR increases from 24.879 to 27.752, and SSIM improves from 0.873 to 0.9173. For the Train scene, PSNR improves from 20.343 to 25.054, while SSIM increases from 0.7497 to 0.8845. Additionally, the L1 loss shows a reduction in error, with values decreasing from 0.033 to 0.022 for the Truck and from 0.064 to 0.034 for the Train, indicating better accuracy in the reconstruction. In addition, real-time rendering speed measured at the 30K iteration mark shows that the 3DGS model achieves the following frame rates: Truck - 29.27 fps, and Train - 42.30 fps.

Table 2. The PSNR value for the Tanks&Temples dataset

	Truck	Train
Plenoxels	23.221	18.927
INGP-Base	23.260	20.170
INGP-Big	23.383	20.456
Mip-NeRF360	24.912	19.523
3DGS - 7K	24.879	20.343
3DGS - 30K	27.752	25.054

The PSNR value for the Tanks&Temples dataset compares the PSNR values of 3DGS with other methods. The results show that 3DGS - 30K achieves the highest PSNR for both scenes, with 27.752 for Truck and 25.054 for Train, outperforming methods like Plenoxels and Mip-NeRF360. This highlights the effectiveness of 3DGS in improving reconstruction quality, particularly after 30,000 training iterations, making it a competitive approach in 3D reconstruction tasks.

Table 3. The SSIM value for the Tanks&Temples dataset

	Truck	Train
Plenoxels	0.774	0.663
INGP-Base	0.779	0.666
INGP-Big	0.800	0.689
Mip-NeRF360	0.857	0.660
3DGS - 7K	0.873	0.7497
3DGS - 30K	0.9173	0.8845

The SSIM values for the Tanks&Temples dataset demonstrate the structural similarity performance across different methods. The results show that 3D Gaussian Splatting (3DGS) achieves the highest SSIM values for both the Truck and Train scenes. Specifically, 3DGS - 30K obtains an SSIM of 0.9173 for trucks and 0.8845 for Trains, significantly outperforming other methods such as Plenoxels, Instant-NGP, and Mip-NeRF360. Compared to earlier models, 3DGS shows a clear improvement in preserving structural details and texture consistency in the reconstructed images. This confirms the model's effectiveness in generating high-fidelity 3D representations and novel views, especially after extended training iterations.

4.2. Deep Blending Dataset

The next dataset used is Deep Blending, published by the Visual Computing group at UCL [8]. It includes diverse real-world scenes captured from multiple viewpoints. Each scene in the dataset contains high-resolution images along with camera parameters and scene structure information, which are useful for novel view synthesis and free-viewpoint rendering tasks. This dataset is ideal for evaluating the ability to reproduce fine details and handle complex lighting phenomena in real-world environments, making it highly suitable for testing the 3D reconstruction and novel view synthesis capabilities of 3D Gaussian Splatting. In the Deep Blending dataset, two scenes, Playroom and DrJohnson, were selected for training. These scenes represent interior environments with rich lighting and textures, providing moderate complexity that helps train the 3D Gaussian Splatting model effectively for generating high-quality novel views. Additionally, these scenes are commonly used as benchmarks in novel view synthesis research, allowing for a fair comparison with existing methods.

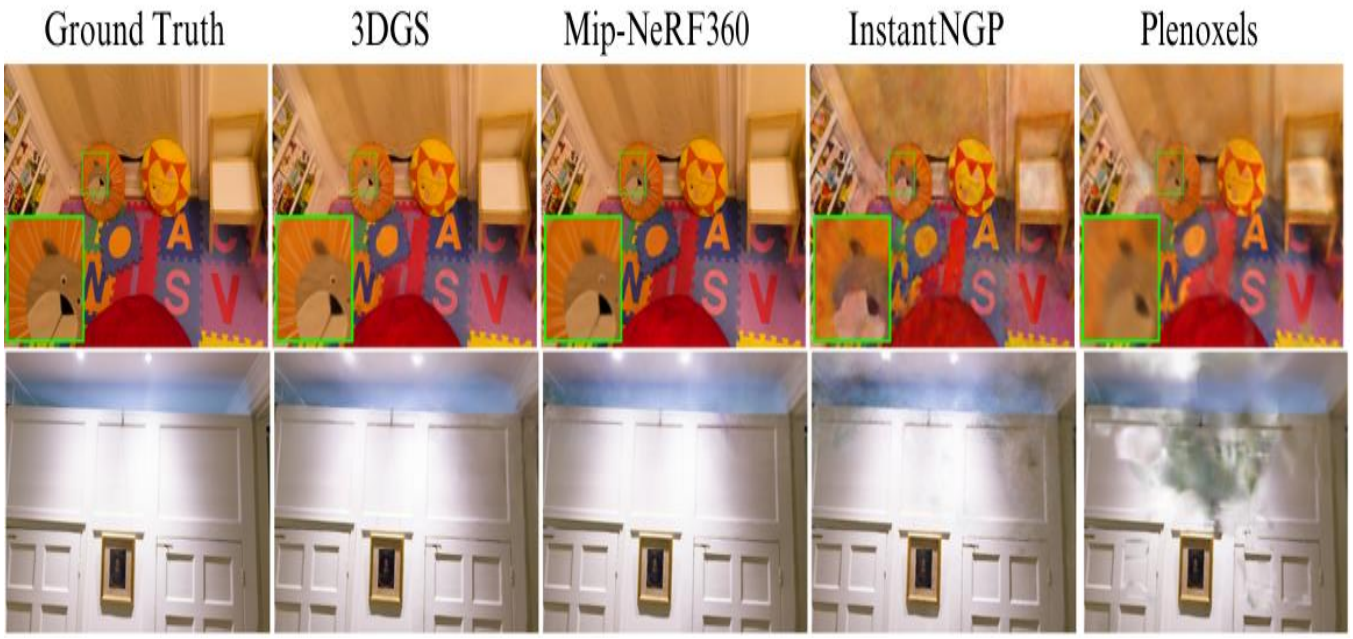

Fig. 5 Playroom and Drjohnson scenes are compared across five different approaches

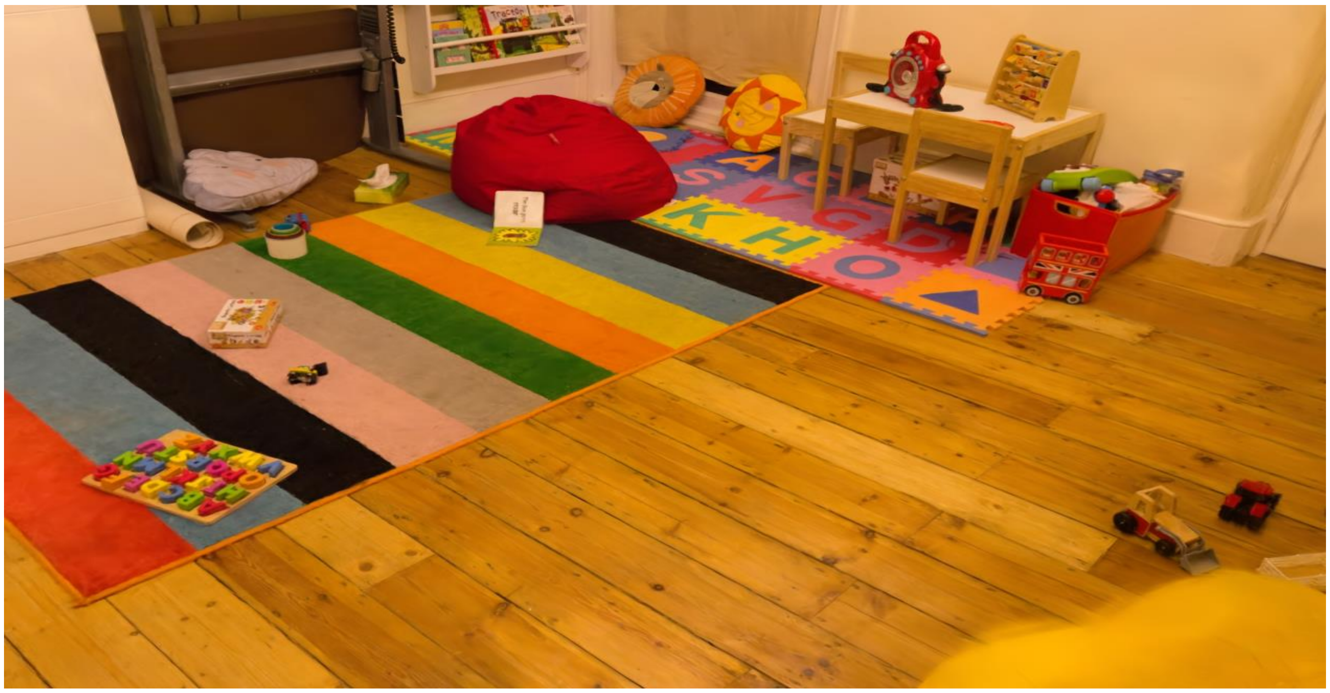
Fig. 6 Playroom scene after training

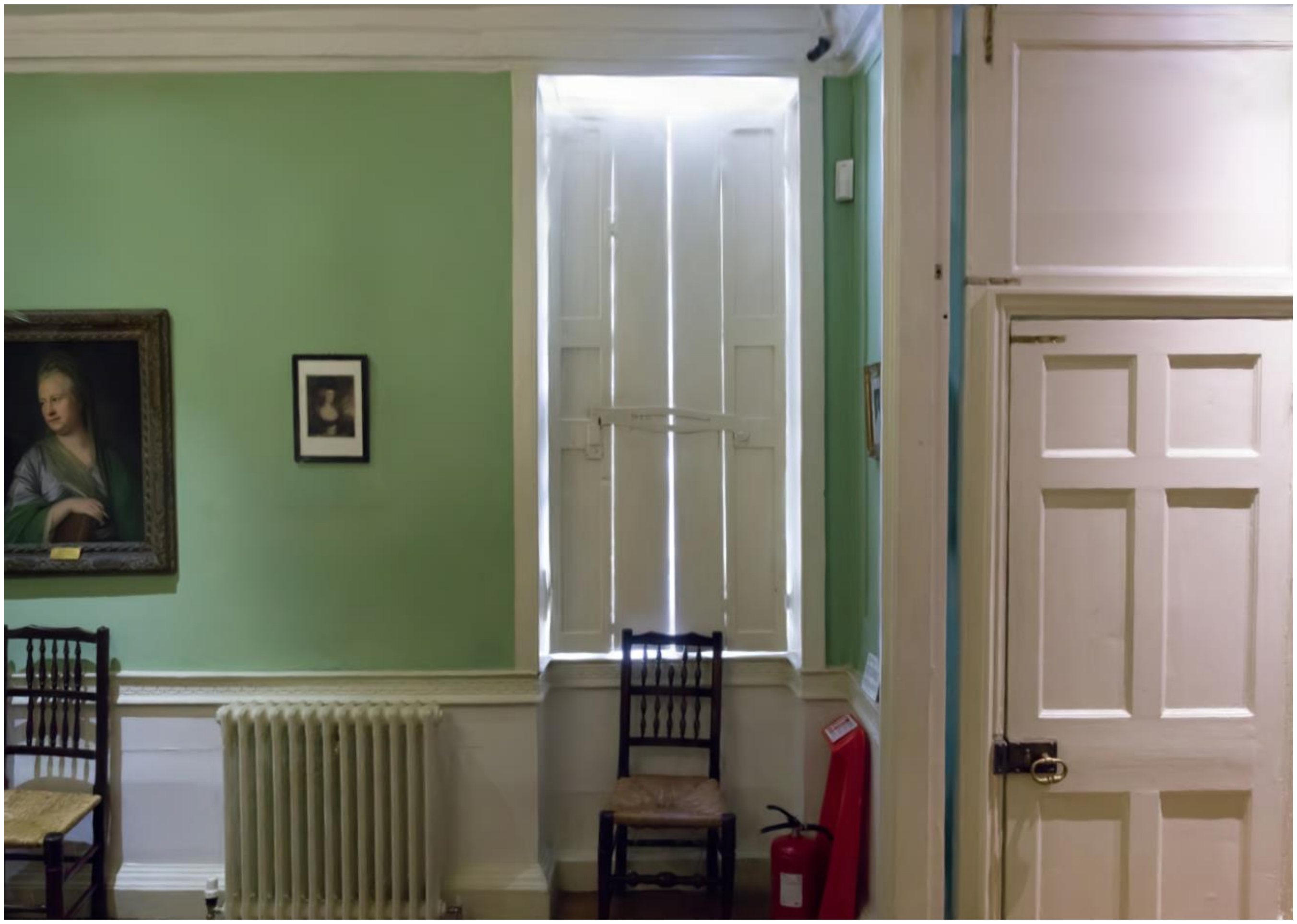
Fig. 7 DrJohnson scene after training

Table 4. The results of the deep blending dataset

Dataset	PSNR (7k)	PSNR (30k)	SSIM (7k)	SSIM (30k)	L1 (7k)	L1 (30k)
Dr Johnson	30.098	36.975	0.8869	0.9633	0.018	0.009
Playroom	31.933	36.335	0.9147	0.9506	0.016	0.010

For the DrJohnson scene, PSNR increases from 30.098 to 36.975, SSIM improves from 0.8869 to 0.9633, and L1 loss decreases from 0.018 to 0.009, reflecting enhanced reconstruction accuracy and image quality. Similarly, the Playroom scene shows PSNR improving from 31.933 to 36.335, SSIM from 0.9147 to 0.9506, and L1 loss reducing from 0.016 to 0.010. These results demonstrate the ability of 3DGS to effectively reconstruct complex indoor scenes with rich textures and lighting, producing high-quality novel views after sufficient training iterations. In addition, real-time rendering speed measured at the 30K iteration mark shows that the 3DGS model achieves the following frame rates: Dr Johnson - 18.59 fps, and Playroom - 27.78 fps.

Table 5. The PSNR value for the deep blending dataset

	Dr Johnson	Playroom
Plenoxels	23.142	22.980
INGP-Base	27.750	19.483
INGP-Big	28.257	21.665
Mip-NeRF360	29.140	29.657
3DGS - 7K	30.098	31.933
3DGS - 30K	36.975	36.335

For the DrJohnson scene, 3DGS - 30K reaches a PSNR of 36.975, surpassing Mip-NeRF360 (29.140) and other methods such as Plenoxels (23.142) and Instant-NGP (28.257). Similarly, in the Playroom scene, 3DGS - 30K achieves 36.335, outperforming Mip-NeRF360 (29.657) and other approaches. These results highlight the superior reconstruction quality of 3DGS, especially when trained for an extended number of iterations, making it highly effective for novel view synthesis in complex indoor environments.

Table 6. The SSIM value for the deep blending dataset

	Dr Johnson	Playroom
Plenoxels	0.787	0.802
INGP-Base	0.839	0.754
INGP-Big	0.854	0.779
Mip-NeRF360	0.901	0.900
3DGS - 7K	0.8869	0.9147
3DGS - 30K	0.9633	0.9506

For the DrJohnson scene, 3DGS - 30K achieves an SSIM of 0.9633, outperforming Mip-NeRF360 (0.901), Instant-NGP Big (0.854), and Plenoxels (0.787). In the Playroom scene, 3DGS - 30K also leads with an SSIM of 0.9506, surpassing Mip-NeRF360 (0.900) and other methods. These results confirm that 3DGS not only enhances image fidelity in terms of pixel accuracy but also excels in maintaining the structural consistency of the scene, making it particularly effective for high-quality novel view synthesis in indoor environments.

4.3. Mip-NeRF 360 Dataset

The final dataset used in this experiment is Mip-NeRF 360, developed by Barron et al. [9], which features unbounded scenes that include both indoor and outdoor environments. It provides high-resolution images captured from multiple viewpoints, with wide scene coverage and challenges related to lighting and depth—making it highly suitable for evaluating 3D Gaussian Splatting (3DGS) in the context of novel view synthesis. All nine scenes from the Mip-NeRF 360 dataset were selected for training 3DGS, as the dataset offers a rich variety of content that allows the model to learn diverse geometric and lighting features. Training on the full set of scenes enables 3DGS to develop better generalization capabilities, improving its performance in synthesizing novel views across a wide range of real-world conditions. Moreover, since Mip-NeRF 360 is a widely used benchmark in the novel view synthesis community, using the entire dataset ensures a fair and comprehensive evaluation of 3DGS against existing methods.

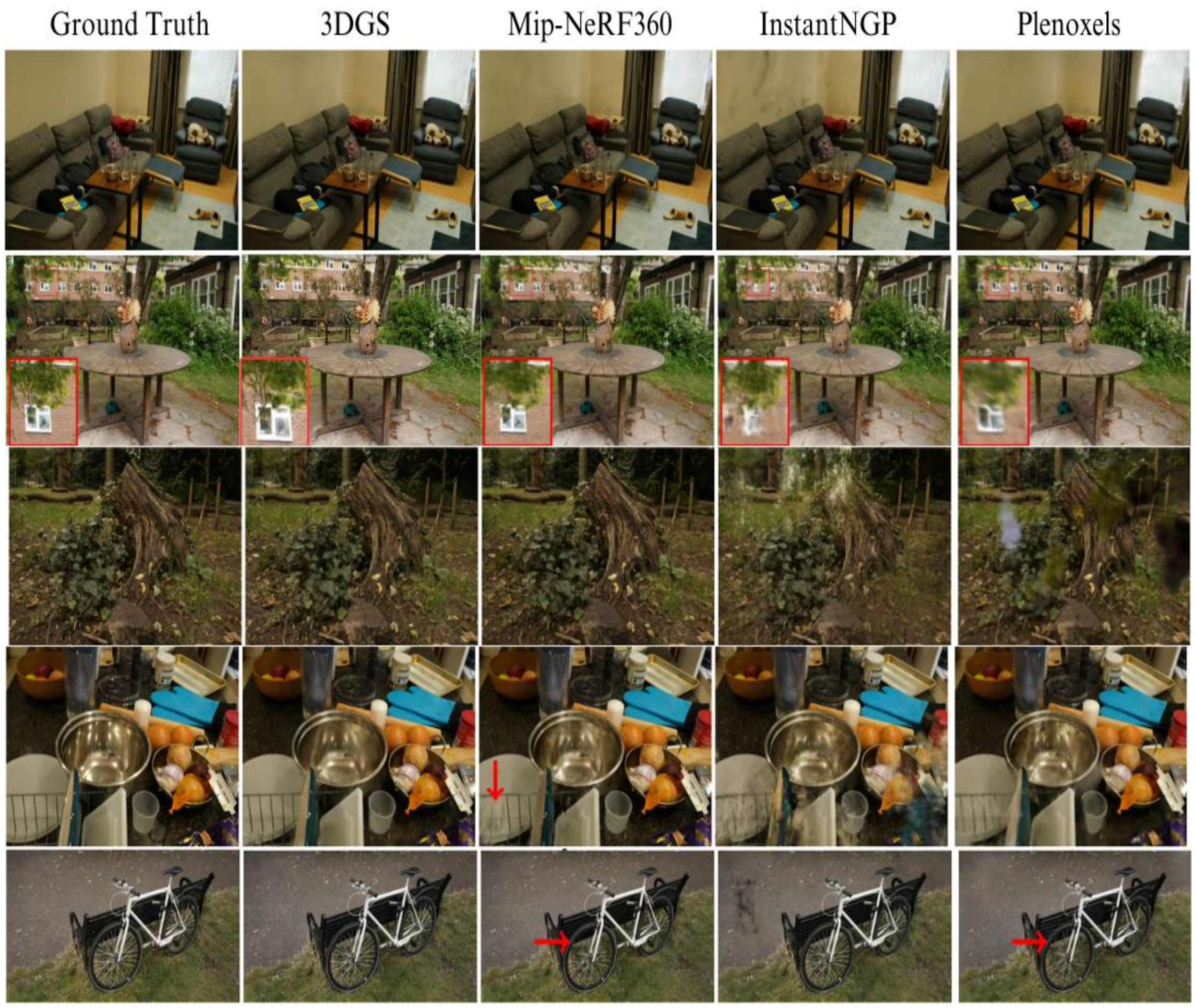

Fig. 8 Room, garden, stump, counter, and bicycle scenes are compared across five different approaches.

Qualitative analysis of novel view synthesis was performed on a diverse set of scenes, including Bicycle, Counter, Stump, Room, and Garden. These examples reveal distinct differences in reconstruction quality across various methods.

In particular, 3D Gaussian Splatting (3DGS) demonstrates superior capability in recovering fine geometric and textural details. The method consistently preserves object contours, surface textures, and microstructures, resulting in sharp and coherent novel views that closely resemble the ground truth. In complex regions-such as surface transitions or areas with fine-grained textures-3DGS maintains a high level of structural fidelity, contributing to realistic and visually accurate reconstructions. These results confirm the effectiveness of 3DGS in producing high-quality novel views, especially in scenes with diverse lighting and geometric complexity.

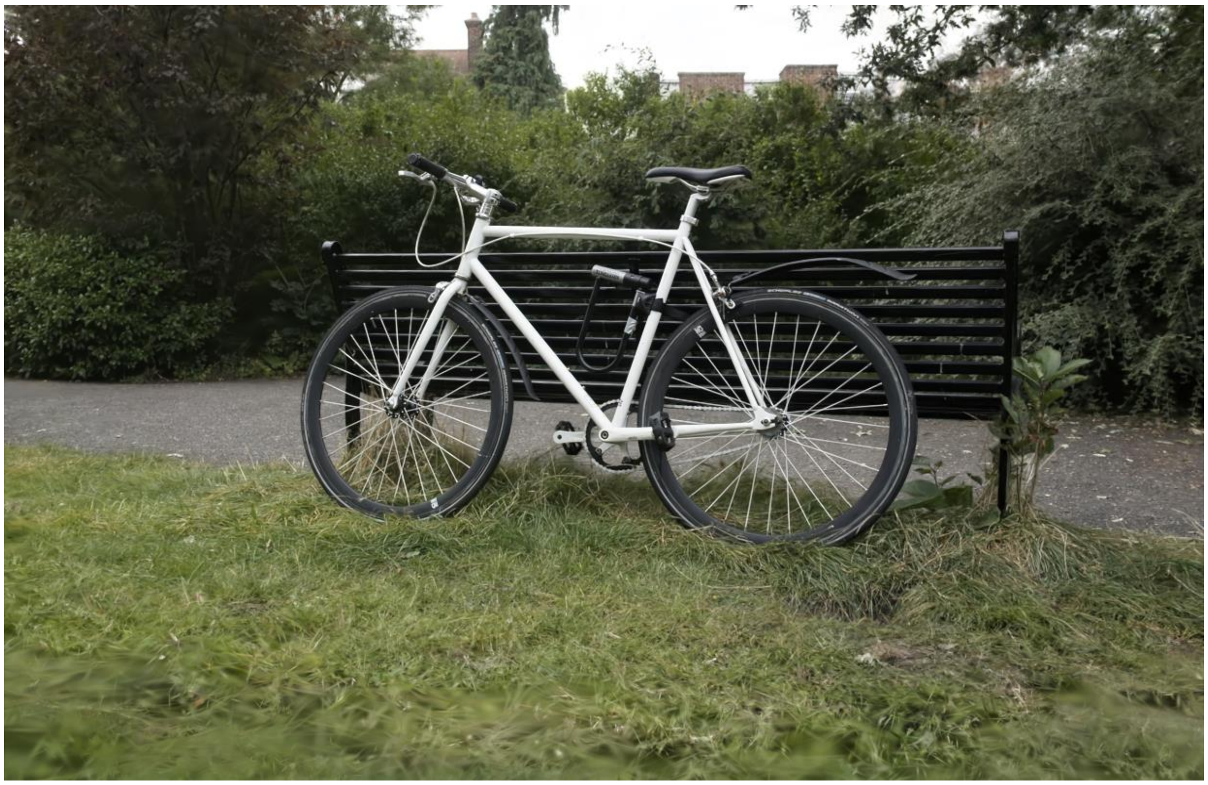
Fig. 9 Bicycle scene after training

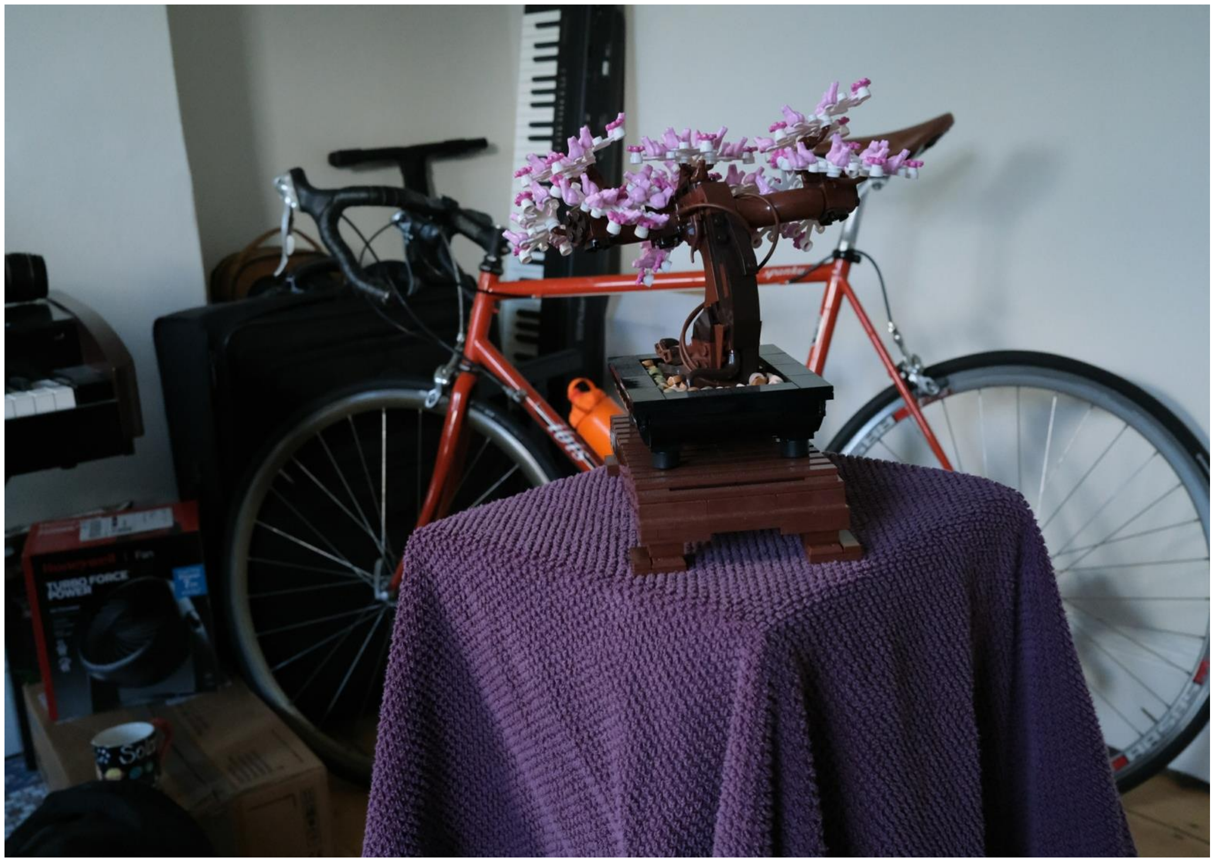
Fig. 10 Bonsai scene after training

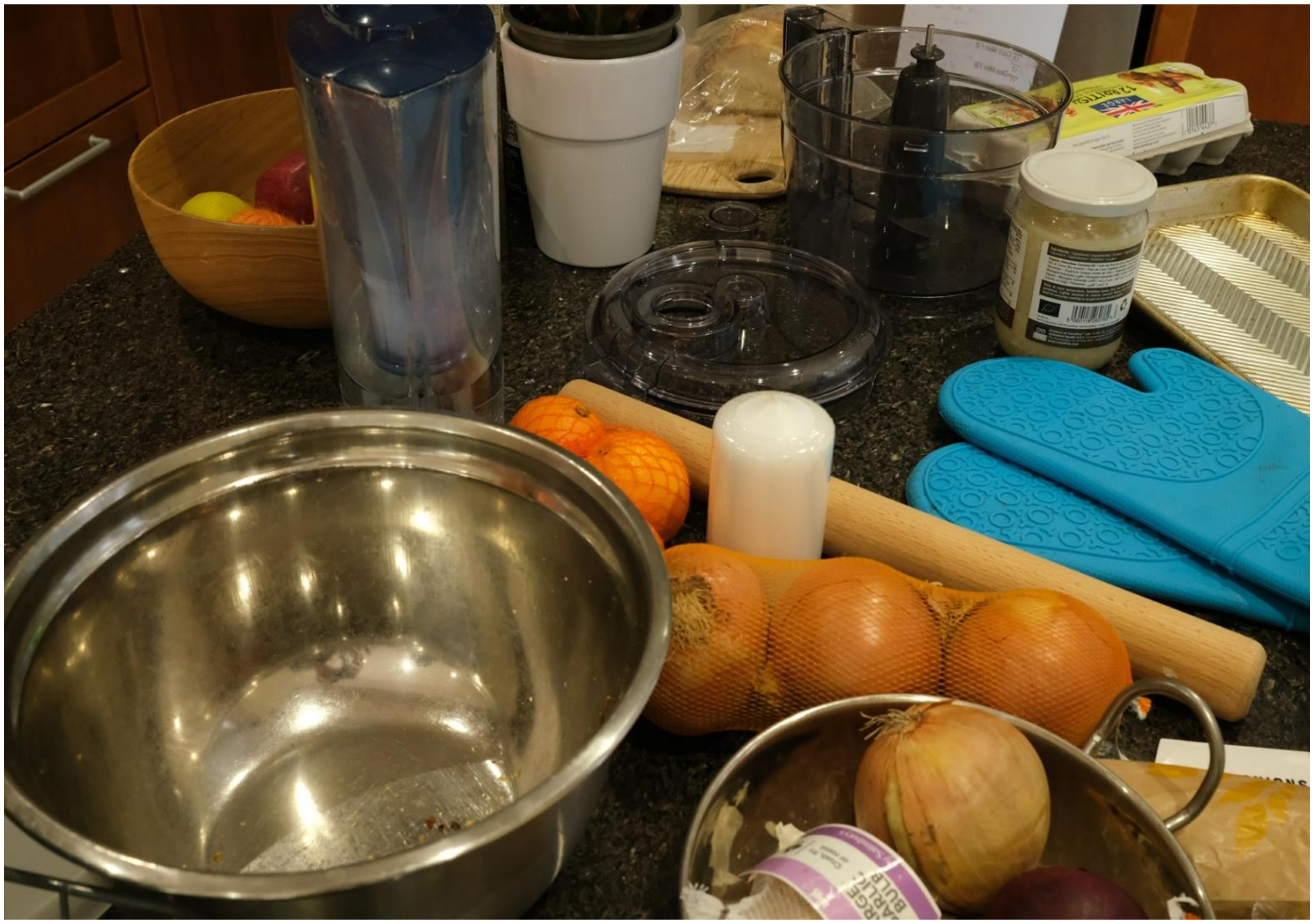

Fig. 11 Counter scene after training

Table 7. The results of the deep blending dataset

Dataset	PSNR (7k)	PSNR (30k)	SSIM (7k)	SSIM (30k)	L1 (7k)	L1 (30k)
Bicycle	24.263	27.859	0.7313	0.8608	0.090	0.024
Flowers	21.290	23.916	0.6169	0.7574	0.110	0.037
Garden	27.361	30.141	0.8545	0.9170	0.028	0.020
Stump	26.592	31.412	0.7664	0.9003	0.030	0.017
Treehill	22.352	24.833	0.6477	0.7702	0.042	0.030
Room	31.919	35.398	0.9191	0.9484	0.016	0.011
Counter	28.018	30.576	0.8970	0.9314	0.021	0.015
Kitchen	30.300	32.887	0.9027	0.9298	0.021	0.016
Bonsai	30.589	34.053	0.9391	0.9588	0.019	0.012

The results show that the 3DGS model consistently improves across all metrics with more training iterations, demonstrating strong reconstruction accuracy and structural consistency, especially in complex scenes such as Bonsai, Kitchen, and Room. In addition, real-time rendering speed measured at the 30K iteration mark shows that the 3DGS model achieves the following frame rates: Bicycle - 10.44 fps, Flowers - 21.31 fps, Garden - 13.51 fps, Stump -17.04 fps, Treehill - 19.41 fps, Room - 27.43 fps, Counter - 29.47 fps, Kitchen - 22.33 fps, and Bonsai - 37.86 fps.

Table 8. The results of bicycle, flowers, garden, stump, treehill

	Bicycle	Flowers	Garden	Stump	Treehill
Plenoxels	21.912	20.097	23.494	20.661	22.248
INGP-Base	22.193	20.348	24.599	23.626	22.364
INGP-Big	22.171	20.652	25.069	23.466	22.373
Mip-NeRF 360	24.37	21.73	26.98	26.40	22.87
3DGS - 7K	24.263	21.290	27.361	26.592	22.352
3DGS - 30K	27.859	23.916	30.141	31.412	24.833

This table compares the PSNR performance of 3D Gaussian Splatting (3DGS) with other state-of-the-art methods across five scenes: Bicycle, Flowers, Garden, Stump, and Treehill. The results demonstrate that 3DGS - 30K consistently achieves the highest PSNR values among all methods. For example, in the Stump scene, 3DGS - 30K reaches 31.412, significantly outperforming Mip-NeRF360 (26.40) and other baselines like INGP-Big (23.466) and Plenoxels (20.661). Similarly, in the Garden scene, 3DGS - 30K obtains a PSNR of 30.141, compared to 26.98 from Mip-NeRF360.

Table 9. The results of room, counter, kitchen, bonsai

	Room	Counter	Kitchen	Bonsai
Plenoxels	27.594	23.624	23.420	24.669
INGP-Base	29.269	26.439	28.548	30.337
INGP-Big	29.690	26.691	29.479	30.685
Mip-NeRF 360	31.63	29.55	32.23	33.46
3DGS - 7K	31.919	28.018	30.300	30.589
3DGS - 30K	35.398	30.576	32.887	34.053

Table 9 reports the PSNR values for the Room, Counter, Kitchen, and Bonsai scenes from the Mip-NeRF 360 dataset, comparing the performance of 3D Gaussian Splatting (3DGS) with several baselines. The results demonstrate that 3DGS, when trained for 30,000 iterations, consistently achieves superior reconstruction quality. Specifically, 3DGS - 30K obtains the highest PSNR in all four scenes: 35.398 for Room, 30.576 for Counter, 32.887 for Kitchen, and 34.053 for Bonsai. These values outperform those of Plenoxels, Instant-NGP (Base and Big), and even Mip-NeRF360, which previously showed competitive performance. The significant improvement highlights the effectiveness of 3DGS in capturing fine details and complex indoor structures, reinforcing its state-of-the-art capability for novel view synthesis in realistic settings.

Table 10. Comparison table between baseline and incremental improvement methods of Truck sence.

Method	PSNR	SSIM
Baseline 3DGS	25.187	0.879
+ Pruning	25.381	0.881
+Densification	26.687	0.895
+ Multi-scale Render	26.450	0.890
+ Optimized SH	26.271	0.888
Full (Ours)	27.752	0.9173

In this study, the authors use the ‘Truck’ scene from the Tanks&Temples set because it is one of the scenes with relatively high geometric and textural complexity, which represents well the challenges in 3D reconstruction. The authors assess four enhancements-Adaptive Pruning, Densification, Multi-scale Rendering, and Optimized Spherical Harmonics (SH)-against the baseline 3D Gaussian Splatting (PSNR: 25.187, SSIM: 0.879). Individually, pruning yields a modest PSNR increase to 25.381, while densification excels with a PSNR of 26.687 and estimated SSIM of 0.895, outperforming Multi-scale Rendering (PSNR: 26.450, SSIM: ~0.890) and Optimized SH (PSNR: 26.271, SSIM: ~0.888). The "Full (Ours)" configuration, combining all methods, achieves the highest PSNR (27.752) and SSIM (0.9173), highlighting synergy. Densification proves the most effective standalone enhancement, with the combined approach optimizing both metrics.

5. Discussion

The experimental results demonstrate that the improved 3D Gaussian Splatting (3DGS) method achieves a compelling balance between reconstruction quality and computational efficiency. Compared to prior methods such as Instant-NGP, Plenoxels, and Mip-NeRF 360, the proposed approach delivers superior visual fidelity, particularly in scenes with intricate structural and textural details.

Despite the enhancements, the model maintains real-time rendering capability, with an average frame rate of 23.59 FPS, meeting the demands of interactive applications such as Virtual Reality (VR) and real-time simulation. Notably, methods like Mip-NeRF 360, while effective in reconstruction quality, often fail to meet real-time rendering constraints, highlighting the practical advantage of the authors' improved 3DGS pipeline.

This performance is attributed to several key contributions:

- Multi-scale Rendering Strategy: Training begins at low resolution to capture global structure efficiently and progressively increases resolution, reducing training time while improving detail refinement. This approach mirrors the coarse-to-fine processing of human visual perception.
- Covariance Matrix Optimization: By decoupling scaling and rotation, the model simplifies computations per Gaussian and reduces the memory overhead per component. Combined with memory-efficient operations like strip_symmetric, this allows the use of a greater number of Gaussians without excessive computational burden.
- Adaptive Densification and Pruning: Controlled noise injection during Gaussian cloning enables better spatial coverage and prevents over-clustering. A scaling factor (e.g., 0.8×Σ0.8 \times \Sigma0.8×Σ) maintains structural coherence during splitting, while low-contribution Gaussians are pruned to save memory and preserve performance.
- Perceptual Loss Integration: To enhance the perceptual quality of reconstructions, the authors incorporate VGG perceptual loss [13] and LPIPS loss [14]. These losses steer the optimization process towards visually meaningful features, going beyond traditional pixel-based losses. This is particularly effective in regions with fine textures, complex lighting, or semantic relevance.

6. Rendering Performance and Hardware Constraints

The reported average FPS of 23.59 was computed as the arithmetic mean across all tested scenes. However, as shown in Table IV, FPS varies significantly depending on scene complexity, geometric density, and the number of active Gaussians. The range spans from 10.44 to 42.30 FPS, with simpler scenes achieving notably higher speeds.

Additionally, due to the computational constraints of free GPU resources on Kaggle, certain high-resolution scenes or those with dense geometry could not be fully processed. These limitations impacted peak performance and scalability evaluations, highlighting the importance of future testing on more powerful or dedicated hardware.

7. Limitations

While the proposed optimizations significantly improve performance, several limitations remain:

- High hardware demands: For optimal performance, especially in dense or high-resolution scenes, access to powerful GPUs is necessary. On lower-end hardware, rendering performance drops significantly.
- Reflective and transparent surfaces: Objects like mirrors or glass remain challenging due to their complex light interactions, often resulting in ghosting or artefacts, especially at extreme viewing angles.

- Training time: Although faster than the original 3DGS, the total training time of 1–2 hours is still longer than methods like Instant-NGP, which converge within minutes.
- Memory overhead during densification: As the number of Gaussians increases during training, especially with added attributes like rotation matrices and SH coefficients, memory usage can grow rapidly, risking overload on hardware with limited VRAM.

These findings suggest that while improved 3DGS is highly effective for real-time and high-fidelity 3D reconstruction, further work is needed to address challenges related to generalization, reflective surfaces, and deployment on resource-constrained systems.

8. Conclusion

In the context of the growing demand for high-quality and real-time 3D image reconstruction, this study titled "Image Reconstruction Using 3D Gaussian Splatting (3DGS)" has demonstrated the feasibility and potential of this technique. Through extensive research, the authors introduced several critical enhancements to the original 3DGS framework. These improvements have shown that the optimized version of 3DGS not only accelerates training and rendering but also significantly enhances the fidelity of the reconstructed images, paving the way for its adoption in real-time systems. However, training 3DGS models requires high-performance computational environments, especially powerful GPUs with large memory capacity. On systems with limited hardware resources, training speed can be severely affected, resulting in prolonged optimization and degraded reconstruction quality. Therefore, to fully leverage the capabilities of 3DGS, investments in hardware or efforts in software optimization to reduce computational load are essential.

Based on the promising results obtained in this study, future work will focus on further algorithmic optimizations to minimize hardware requirements. Additionally, the authors plan to explore hybrid approaches that integrate 3DGS with advanced deep learning methods, aiming to develop models with better generalization and performance on low- or mid-end hardware. Through these directions, the objective is to broaden the practical applicability of 3DGS in real-world scenarios where training efficiency and system performance remain key challenges.

Data Availability

The datasets used in this study are publicly available and can be accessed online. The Tanks and Temples dataset [7] was used for benchmarking reconstruction performance, while the Deep Blending Dataset [8] and the Mip-NeRF 360 dataset [9] were used for additional evaluation and qualitative comparison. These datasets include multi-view images, camera parameters, and, where applicable, ground-truth 3D models to support the reproducibility of the experiments. All data are accessible through the following links:

- Tanks & Temples dataset: https://www.kaggle.com/datasets/jinnywjy/tanks-and-temple-m60-colmap-preprocessed
- Deep Blending dataset: https://www.kaggle.com/datasets/minhanhtruong/deep-blending-dataset
- Mip-NeRF 360 dataset: https://www.kaggle.com/datasets/thnhdg/testing

Readers may use these datasets under their respective licenses to replicate or extend the results presented in this study.

Funding Statement

Funding for this research was provided by Saigon University as part of the student scientific research project SV2024-84.

Authors' Contributions

- Vuong Pham: Conceptualization, Methodology, Writing - Review & Editing, Supervision, Project Administration.
- Thanh Cao: Conceptualization, Methodology, Writing - Review & Editing, Supervision.
- Thanh Dang: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing - Original Draft Preparation, Writing - Review and editing, Visualization.
- Tan Thanh Nguyen: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing - Original Draft Preparation, Writing - Review and editing, Visualization.

Acknowledgements

This research originates from a student scientific research project (Project ID: SV2024-84).

References

[1] Jonathan T. Barron et al., "MIP-NERF 360: Unbounded Anti-Aliased Neural Radiance Fields," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5470-5479, 2022. [Google Scholar] [Publisher Link]

[2] Anpei Chen et al., "TensoRF: Tensorial Radiance Fields," *Computer Vision - ECCV : 17th European Conference*, Tel Aviv, Israel, pp. 333-350, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Bernhard Kerb et al., "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics,* vol. 42, no. 4, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Ben Mildenhall et al., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Communications of the ACM*, vol. 65 no. 1, pp. 99 -106, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Thomas Müller et al., "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1-15, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Sara Fridovich-Keil et al., "Plenoxels: Radiance Fields without Neural Networks," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 5501-5510, 2022. [Google Scholar] [Publisher Link]

[7] Wong Jing Yuan, Tanks&Temples - M60 (COLMAP Preprocessed) datasets, Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/jinnywjy/tanks-and-temple-m60-colmap-preprocessed

[8] Minh-Anh Truong, Deep Blending datasets, Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/minhanhtruong/deep-blending-dataset

[9] Thành Đặng, Mip-NeRF 360 datasets, Kaggle, 2025 [Online]. Available: https://www.kaggle.com/datasets/thnhdg/testing

[10] Yoshua Bengio et al., "Curriculum Learning," *Proceedings of the 26th Annual International Conference on Machine Learning*, New York, NY, United States, pp. 41-48, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[11] David B. Lindell et al., "AutoInt: Automatic Integration for Fast Neural Volume Rendering," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14556-14565, 2021. [Google Scholar] [Publisher Link]

[12] Westover, Lee Alan, "SPLATTING: A Parallel, Feed-Forward Volume Rendering Algorithm," The University of North Carolina at Chapel Hill ProQuest Dissertations and Theses, 1991. [Google Scholar] [Publisher Link]

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," Computer Vision – ECCV, pp. 694-711, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[14] Richard Zhang et al., "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586-595, 2018. [Google Scholar] [Publisher Link]

[15] Yiheng Xie et al., "Neural Fields in Visual Computing and Beyond," *Computer Graphics Forum*, vol. 41, no. 2, pp. 641-676, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Yangjiheng, 3DGS and Beyond Docs, 2025. [Online]. Available: https://github.com/yangjiheng/3DGS_and_Beyond_Docs

[17] Ricardo Martin-Brualla et al., "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," *Conference on Computer Vision and Pattern Recognition*, pp. 7210-7219, 2021. [Google Scholar] [Publisher Link]

[18] Johannes L. Schonberger, and Jan-Michael Frahm, "Structure-from-Motion Revisited," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104-4113, 2016. [Google Scholar] [Publisher Link]

[19] FFMPEG [Online]. Available: https://ffmpeg.org

[20] NVIDIA CUDA Toolkit, 2025. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[21] Graphdeco-Inria, 3D Gaussian Splatting Viewer, 2025. [Online]. Available: https://github.com/graphdeco-inria/gaussian-splatting

[22] PyTorch3D, 2024. [Online]. Available: https://pytorch3d.org

[23] Ricardo Martin-Brualla, David Gallup, and Steven M. Seitz, "3D Time-Lapse Reconstruction from Internet Photos," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1398-1406, 2015. [Google Scholar] [Publisher Link]

[24] Jim Canary, NeRF: Revolutionizing 3D Scene Reconstruction with Neural Radiance Fields, Medium, 2025. [Online]. Available: https://medium.com/@jimcanary/nerf-revolutionizing-3d-scene-reconstruction-with-neural-radiance-fields-1c28df282857

[25] Michael Broxton et al., "Immersive Light Field Video with A Layered Mesh Representation," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 1-15, 2020. [CrossRef] [Google Scholar] [Publisher Link]