Color-cued Efficient Densification Method for 3D Gaussian Splatting

Sieun Kim Kyungjin Lee Youngki Lee
Seoul National University
Republic of Korea
{sieunk, jin11542, youngkilee}@snu.ac.kr

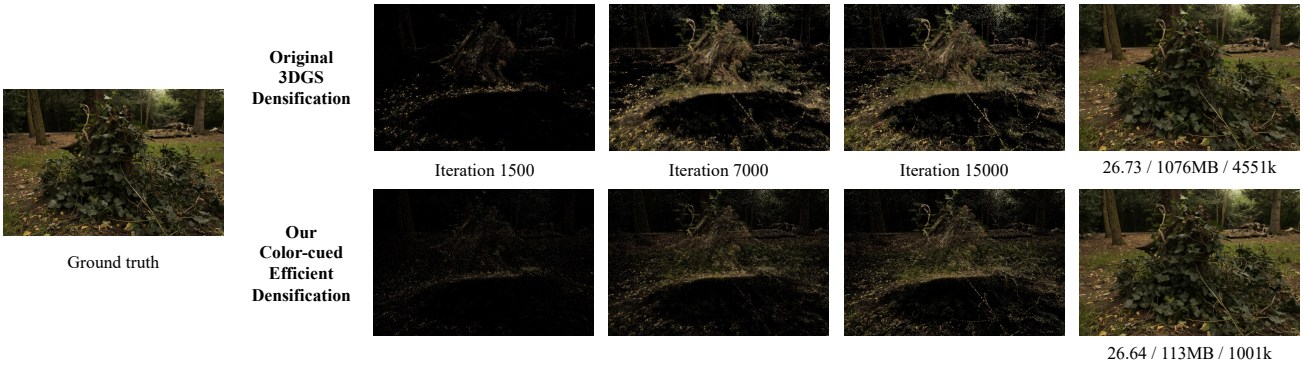

Figure 1. Comparison of the densification process and the final image of the original 3D Gaussian Splatting (3DGS) and our efficient densification method. Intermediate iterations are shown with spheres of equal opacity to demonstrate the density of the Gaussians. The right-most column indicates the final image with PSNR, data size, and the number of Gaussians, respectively.

Abstract

Many variants of Neural Radiance Fields (NeRF) have been explored in pursuit of high-quality results with reasonable data size and real-time rendering speed. 3D Gaussian Splatting (3DGS) gained popularity due to its ability to render quality images in real-time; however, it still faces challenges with large data sizes. Meanwhile, the densification process of 3DGS plays a large role in deciding the quality and the data size of a model. Hence, it is crucial to devise a densification method that can populate Gaussians efficiently so that quality can be enhanced and fewer Gaussians are used. An efficient densification method that results in fewer Gaussians can also promote efficiency in training time, GPU memory usage, and rendering speed.

Hence, we propose a novel, efficient densification method based on color cues, aiming to achieve a more compact Gaussian model without sacrificing image quality. By expanding the original 3DGS densification scheme, we identify weaknesses in the original method that lead to redundant Gaussians and compromise quality. In contrast to the original approach, which relies solely on the 2D position gradient, our method additionally leverages the spherical harmonics (SH) gradient to consider color cues. This approach resolves the inefficiencies of the original densification by aligning with the expanded scheme. Our method achieves at least 9× data size reduction with increased perceptual quality, accompanied by additional efficiencies in training time, GPU memory usage, and rendering speed.

1. Introduction

Recently proposed 3D Gaussian Splatting (3DGS) [12] is one of the most effective methods for novel view synthesis (NVS), achieving photo-realistic quality and real-time rendering. The anisotropic Gaussians employed by 3DGS contribute to its rendering quality, while the differentiable rasterizer, optimized with CUDA, enables real-time performance. However, the requirement for numerous Gaussians to represent a scene, each with multiple features, leads to a challenge of large model size. To promote practical usages of 3DGS for various applications of NVS, enhancements that can generate smaller models without compromising on quality or rendering speed are necessary.

The densification process in 3DGS is crucial as it enhances the model by adding Gaussians to a coarser representation, resulting in finer quality. The densification method significantly influences both the data size and quality of the model. Failing to create Gaussians that express an object can cause a severe drop in quality, and adding

multiple unnecessary Gaussians can increase the model size without improving the final image. The original 3DGS densification scheme exhibits weaknesses by overlooking specific scenarios, generating redundant Gaussians, and failing to reconstruct fine patterns. Meanwhile, existing works that target to reduce the model size of 3DGS suggest using vector quantization or multi-layer perceptrons (MLP) to compress the features of a Gaussian [6, 14, 16]. These compression-based approaches are not lossless and can inevitably result in the degradation of scene quality. Therefore, it is necessary to devise a densification process capable of populating Gaussians at adequate positions to maintain a small model size while enhancing quality.

In our work, we discover a simple but effective modification to the densification process that populates Gaussians in a way that can reduce the data size but improve the rendering quality, preserving finer details. The original 3DGS incorporates a densification process where 2D position gradients detect under/over-reconstruction of geometry. However, relying solely on the 2D position gradient neglects the need for more Gaussians from the color difference and amplifies the necessity for more Gaussians when unnecessary. Instead, we propose a novel densification method for 3DGS that leverages the view-independent (0th) spherical harmonics (SH) coefficient gradient to discern cues from color, measuring the necessity for additional Gaussians. Concurrently, densification using the 2D position gradient is employed more coarsely, focusing on details in areas where structure-from-motion (SfM) fails to provide fine structure.

The results showed that our densification method could reduce the data size by at least 9× with further quality improvement. While the metric-based quality results remain comparable, our approach demonstrates quality enhancement in visualization results. By reducing the number of Gaussians, the new method also results in faster rendering, faster training, and less GPU memory usage during training.

In summary, our contributions are as follows:

- We suggest a thorough densification scheme for 3DGS that identifies the limitations of the original method.
- We propose a simple and effective 3DGS densification method utilizing color cues that are complementary to existing 3DGS compression schemes.
- We achieve at least 9× reduction of model size while improving the perceptual rendering quality.
- We also achieve significant improvement in many efficiency metrics, including training time, GPU memory, and rendering speed.

2. Related Works

We briefly overview 3D Gaussian Splatting and its densification method and review various techniques previously applied to NeRF variants to reduce model size.

2.1. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [12] represents a scene with multiple 3D Gaussians that can be rendered into an image through "splatting" [24, 25] to 2D space. Each Gaussian is described by its position x, rotation q, scaling s, opacity α, and spherical harmonics (SH) coefficients. These features are optimized through training using a differentiable rendering process. The 3DGS framework employs a fast differentiable rasterizer with CUDA kernels, enabling real-time image rendering.

One notable feature of the original 3DGS is its densification process, also referred to as the adaptive density control. 3DGS training begins with an initial set of sparse points obtained from structure-from-motion (SfM) and densifies over iterations, enabling a denser set that better represents the scene. The original 3DGS focuses on regions where geometric features are under or over-represented. From observation, the original 3DGS intuitively uses 2D position gradients to identify candidates for densification. Gaussians with average 2D position gradient over a threshold τ_{pos} are densified. Candidate Gaussians with small scales are cloned identically, while candidate Gaussians with large scales are split into smaller Gaussians with shifted positions.

2.2. Compact NeRF Variants

Many approaches have been made to address the challenge of reducing the size of NeRF models, enabling their practical deployment in diverse scenarios, including network streaming, mobile devices, and resource-constrained environments like microcontrollers (MCUs). Approaches to compress the original network-based NeRF models have employed various techniques, such as neural weights quantization [8, 21], network distillation [18], and the use of entropy-penalized functions [3].

As notable variants of NeRF emerged to tackle NeRF's remaining challenges, a subsequent wave of research has delved into compression techniques aimed at reducing model size. Plenoxels [7] introduced the concept of using a sparse voxel grid with density and spherical harmonic coefficients for sample point computation through trilinear interpolation. InstantNGP [15] suggested utilizing a hash grid and an occupancy grid to accelerate computation. TensoRF [4] proposed factorizing the full volume field into multiple compact tensors for memory efficiency. These grid-based approaches inspired further research in compression. Vector quantization, as seen in works like [11, 20, 22], and efficient pruning, as demonstrated in [5, 22, 23], were commonly deployed to reduce model size. Other techniques include combining a low-resolution 3D grid and higher-resolution 2D planes [17], leveraging the level of detail for voxel and octree [19], and baking sparse voxel features into a 3D texture atlas [10].

Making 3DGS smaller is in its early stages, with ongo-

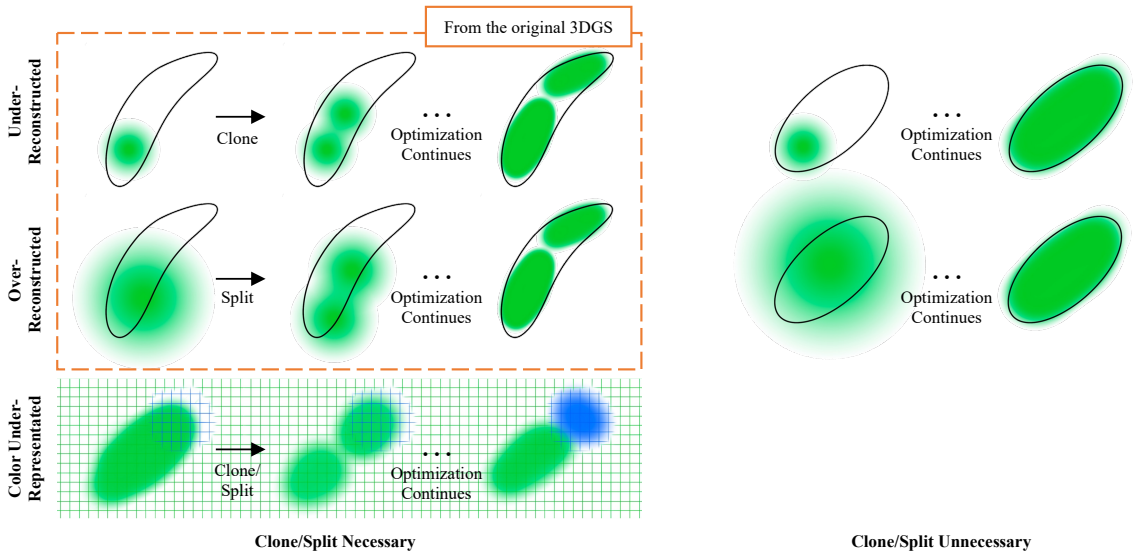

Figure 2. Our expanded densification scheme. The original 3DGS suggests the scenarios in the orange box. When true geometry is under-reconstructed by a smaller Gaussian, the Gaussian is cloned. When true geometry is over-reconstructed by a larger Gaussian, the Gaussian is split. However, the right column illustrates sub-cases where cloning or splitting is unnecessary despite under/over-reconstruction. In addition, the third row introduces a new scenario where color is under-represented, in contrast to the true color depicted by the grid.

ing efforts to adapt techniques previously employed in other NeRF variants. Preprints such as [6, 14, 16] showcase the utilization of Multi-Layer Perceptrons (MLPs) or quantization codebooks to reduce the amount of data required to represent a Gaussian. These approaches also incorporate trained binary masks or importance metrics to enhance pruning effectiveness [6, 14], and entropy encoding and run-length encoding to compress the list of Gaussians [16]. Many approaches aim to adapt previously developed methods to 3DGS, focusing on compressing the final Gaussians by altering the data representations to conserve memory. In contrast, our work focuses on providing a 3DGS-specific solution by introducing a novel densification method that changes the way Gaussians are populated.

3. Method

We propose a new densification method based on color cues to incorporate into the original 3DGS. By analyzing the inefficiencies in the current densification method relying on positional gradients, we introduce a novel architecture leveraging color-based gradients for densification. Additionally, we integrate a pruning method from previous work to enhance the opacity-based pruning of 3DGS and suggest minor precision adjustments.

3.1. Inefficiency Analysis of the Original 3DGS Densification

The original densification method has inefficiencies in two aspects. Firstly, the original densification overestimates the necessity for a new Gaussian. The orange box in Figure 2 shows the densification scheme used by the original 3DGS.

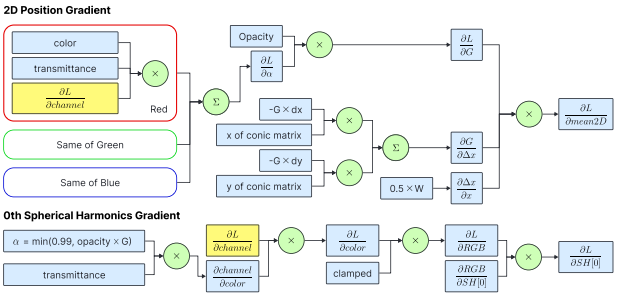

Figure 3. Breakdown of gradients mentioned in this paper. The upper section depicts the 2D position gradient of each Gaussian, while the lower section depicts the 0th spherical harmonics gradient of each Gaussian. Values for pixels are in yellow, while values for Gaussians are in blue.

The original 3DGS utilizes each Gaussian's 2D position gradient to determine whether a Gaussian is under/over-reconstructed and employs the scale of each Gaussian to differentiate between under and over-reconstruction. The right column in Figure 2 illustrates instances where the 2D position gradient can be large, but cloning or splitting is unnecessary. Optimization of scale and rotation of a single Gaussian is enough to represent the true geometry. Since there are no ground truth Gaussians to compare with, the original paper uses the 2D position gradient to roughly capture the difference between the true scene and each Gaussian, leading to aggressive densification that results in a large model with an inefficient Gaussian population.

Secondly, relying on the 2D position gradient as a cue is insufficient for addressing the 'color under-represented' scenario depicted in Figure 2. This scenario occurs when a single Gaussian cannot represent multiple colors in the

scene, necessitating the detection of significant color differences to create new Gaussians. This situation frequently arises when training real scenes with complex patterns. (e.g., for the Mip-NeRF 360 dataset, grass in 'bicycle', 'stump', and 'garden', and pavement in 'treehill') In Figure 3, the upper row demonstrates how the 2D position gradient incorporates Gaussian density (G), position, and color information during back-propagation. Since other factors dilute color information, it is hard to detect color differences using only the 2D position gradient. Consequently, the original 3DGS densification can lead to slower convergence, as additional Gaussians must be formed indirectly. Moreover, necessary Gaussians failing to form may result in a model with lower quality in areas with intricate patterns.

3.2. Efficient Densification Based on Color Cues

To overcome the inefficiencies inherent in the original densification method, we devised a new method capable of accommodating our expanded scheme. In order to capture the color under-represented scenario, we utilize the gradient of the 0th SH coefficient. Utilizing the 0th SH gradient is a practical choice for incorporating color cues. This is attributed to the fact that it is a constant multiple of the RGB gradient and is explicitly stored in the model, in contrast to the RGB gradient, which only serves as an intermediate value for back-propagation. If the accumulated gradient of the 0th SH is large enough, we infer the necessity for additional Gaussians in that region to represent multiple colors. Hence, we clone and split to make more Gaussians using the 0th SH gradient as the color cue. On the other hand, as we have analyzed redundancies in the 2D position gradient-based method, we adopt the position-based gradient more coarsely, only to strengthen geometric information.

The densification unfolds in two steps: Firstly, every 100 iterations, Gaussians with accumulated 0th spherical harmonics gradient larger than τ_{color} are identified as potentially color under-representing, and they are cloned or split based on scale. Secondly, on a more coarse level (every 700 iterations), Gaussians with substantial 2D position gradients are identified as potentially under/over-reconstructed and cloned or split based on scale. The selection of gradient thresholds can be generalized into two categories, depending on whether the scene is indoor or outdoor. The entire densification process is outlined in Algorithm 1.

The proposed method successfully overcomes the two inefficiencies of the original approach. Firstly, the addition of a color-cued densification step can cover broader cases than the original. By utilizing the 0th SH gradient as a cue, Gaussians that need to represent different colors but exhibit a small 2D position gradient—such as the example in the third row of Figure 2—can be duplicated to capture more diverse colors effectively. The lower part of Figure 3 illustrates the calculation of the 0th SH gradient, where the gradient of each pixel is propagated to the gradient of each Gaussian through color. This shows that the 0th SH gradient can better incorporate color information compared to the 2D position gradient, where color information is diluted.

Algorithm 1: 3DGS Training with Our Efficient Densification Method

```
for every iteration do
    render image
    calculate loss and back-propagate
    save the norm of 0th SH gradient
    save the norm of 2D position gradient

    if iteration % 100 == 0 then
        // color-based densification
        for Gaussian with accum. 0th SH grad. > τ_color do
            if scale < τ_scale then clone else split
        end
    end
    if iteration % 700 == 0 then
        // position-based densification
        for Gaussian with accum. 2D position grad. > τ_pos do
            if scale < τ_scale then clone else split
        end
    end
end
```

Secondly, the 2D position gradient is now applied more coarsely than in the original method, sharing its former role with color-cued densification and thereby eliminating redundancy. Relying solely on the 2D position gradient caused the formation of redundant Gaussians, as it couldn't discern when duplication was necessary (refer to the first two rows of Figure 2), consequently requiring aggressive duplication. The new method enables the 2D position gradient to split its role with the color-cued densification. The coarse 2D position gradient-cued densification contributes explicitly to reinforcing the structure of the scene, while the color-cued densification creates additional Gaussians to enhance representation capabilities.

3.3. Pruning and Compressed Representation

To investigate the combined impact of pruning and our novel densification method, we implemented a trainable pruning mask as suggested in [14]. This binary mask M can be calculated using the straight-through estimator [2] and an additional mask parameter $m \in \mathbb{R}^N$ as follows,

$$M_n = sg(\mathbf{1}[\sigma(m_n) > \epsilon] - \sigma(m_n)) + \sigma(m_n),$$

where the $sg(\cdot)$ is the stop gradient operator, $\mathbf{1}$ is the indicator function and $\sigma(\cdot)$ is the sigmoid function. Then the binary mask is used to regulate the scale $s \in \mathbb{R}^3$ and opacity $o \in [0,1]^N$ (e.g., $\hat{s_n} = M_n s_n$, $\hat{o_n} = M_n o_n$). We treat the mask parameter as an additional feature of a Gaussian. The mask values are trained and learned alongside other features. The loss of the 3DGS is modified to reflect the

Dataset	Mip-NeRF 360								Tank&Temples							
Method	SSIM	PSNR	LPIPS	Train	FPS	Disk	GPU Mem	Splats	SSIM	PSNR	LPIPS	Train	FPS	Disk	GPU Mem	Splats
Plenoxels†	0.626	23.08	0.463	25m49s	6.79	2.1GB			0.719	21.08	0.379	25m5s	13.0	2.3GB		
INGP-Base†	0.671	25.30	0.371	5m37s	11.7	13MB			0.23	21.72	0.330	5m26s	17.1	13MB		
INGP-Big†	0.699	25.59	0.331	7m30s	9.73	48MB			0.745	21.92	0.305	6m50s	14.4	48MB		
M-NeRF360†	0.792	27.69	0.237	48h	0.06	8.6MB			0.759	22.22	0.257	48h	0.14	8.6MB		
3DGS†	0.815	27.21	0.214	41m33s	134	734MB			0.841	23.14	0.183	26m54s	154	411MB		
3DGS	0.813	27.49	0.222	31m6s	104	747MB	8.54GB	3158k	0.845	23.68	0.178	15m48s	141	432MB	4.14GB	1825k
Ours	0.797	27.07	0.249	**20m42s**	**166**	**73MB**	**5.98GB**	**646k**	0.830	23.18	0.198	**11m33s**	**231**	**42MB**	**2.95GB**	**370k**

Table 1. Quantitative evaluation of our method evaluated on Mip-NeRF 360 and Tank&Temples datasets. Baseline evaluations marked with † are adapted from the original 3DGS paper evaluated on an NVIDIA A6000 GPU. For comparison, we re-evaluate 3DGS and our approach on an NVIDIA 3090 GPU. 'Train', 'Disk', 'GPU Mem', 'Splats' denote training time, disk storage, peak GPU memory usage while training, and the number of Gaussians, respectively.

pruning mask as follows,

$$L = (1-\lambda)L_1 + \lambda L_{D-SSIM} + \lambda_{mask}\frac{1}{N}\sum_{n=1}^{N}\sigma(m_n)$$

We further optimize our approach by storing features in 2-byte floats instead of 4-byte floats. Our ablation studies demonstrate that this compression does not compromise image quality, as the precision was found to be excessive.

4. Experiments

4.1. Experimental Settings

4.1.1 Datasets and Metrics

We test our approach on 13 large-scale real scenes. 13 360-degree scenes include five outdoor and four indoor scenes from Mip-NeRF 360 dataset [1], as well as the two outdoor scenes from Tank&Temples dataset [13] and two indoor scenes from Deep Blending dataset [9] chosen by the original 3DGS paper. We compare rendering quality in peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and perceptual similarity through LPIPS. Training time, rendering FPS, and disk storage are also compared. For comparison with the original 3DGS, peak GPU Memory usage and the number of Gaussians are also examined.

4.1.2 Compared Baselines

We compare our approach with methods that can render novel views from large scenes with high quality. These include Plenoxels [7], InstantNGP [15], Mip-NeRF360 [1], and the original 3DGS [12]. The experimental results on baselines, excluding 3DGS, are adapted from the original 3DGS paper.

4.1.3 Implementation Details

Different sets of hyperparameters were chosen for indoor and outdoor datasets. The 2D position gradient threshold (τ_{pos}) was kept at 0.0002, consistent with the original 3DGS, and the densification interval based on the 2D position gradient was set to 700. For outdoor scenes, the 0th SH gradient threshold (τ_{color}) was set to 0.000002, and the λ_{mask} for loss was set as 0.05. For indoor scenes, the 0th SH gradient threshold (τ_{color}) was set to 0.00002, and the λ_{mask} for loss was set as 0.01. These hyperparameters were initially selected through a grid search and have demonstrated generalizability to scenes with shared indoor or outdoor characteristics.

Dataset	Deep Blending							
Method	SSIM	PSNR	LPIPS	Train	FPS	Disk	GPU Mem	Splats
Plenoxels†	0.795	23.06	0.510	27m49s	11.2	2.7GB		
INGP-Base†	0.797	23.62	0.423	6m31s	3.26	13MB		
INGP-Big†	0.817	24.96	0.390	8m	2.79	48MB		
M-NeRF360†	0.901	29.40	0.245	48h	0.09	8.6MB		
3DGS†	0.903	29.41	0.243	36m02s	137	676MB		
3DGS	0.900	29.44	0.247	25m31s	114	663MB	6.95GB	2803k
Ours	0.902	29.71	0.255	**15m16s**	**208**	**72MB**	**4.29GB**	**644k**

Table 2. Quantitative evaluation of our method on Deep Blending dataset. Baseline evaluations marked with † are adapted from the original 3DGS paper, and 3DGS is reevaluated for fairness.

4.2. Experimental Results

4.2.1 Quantitative Results

Quantitative results for Mip-NeRF360 and Tank&Temples are summarized in Table 1, and results for Deep Blending are summarized in Table 2. Results show that Mip-NeRF 360 excels in rendering quality but has an impractically low rendering speed below 1 FPS. Voxel-based approaches like Plenoxels and InstantNGP show moderate rendering speed but fall behind in quality and model size. While 3DGS outperforms many metrics compared to other NeRF variants, showing high fidelity with real-time rendering speed, it falls behind in model size. Our approach achieves quantitative image quality comparable to the 3DGS model but with a

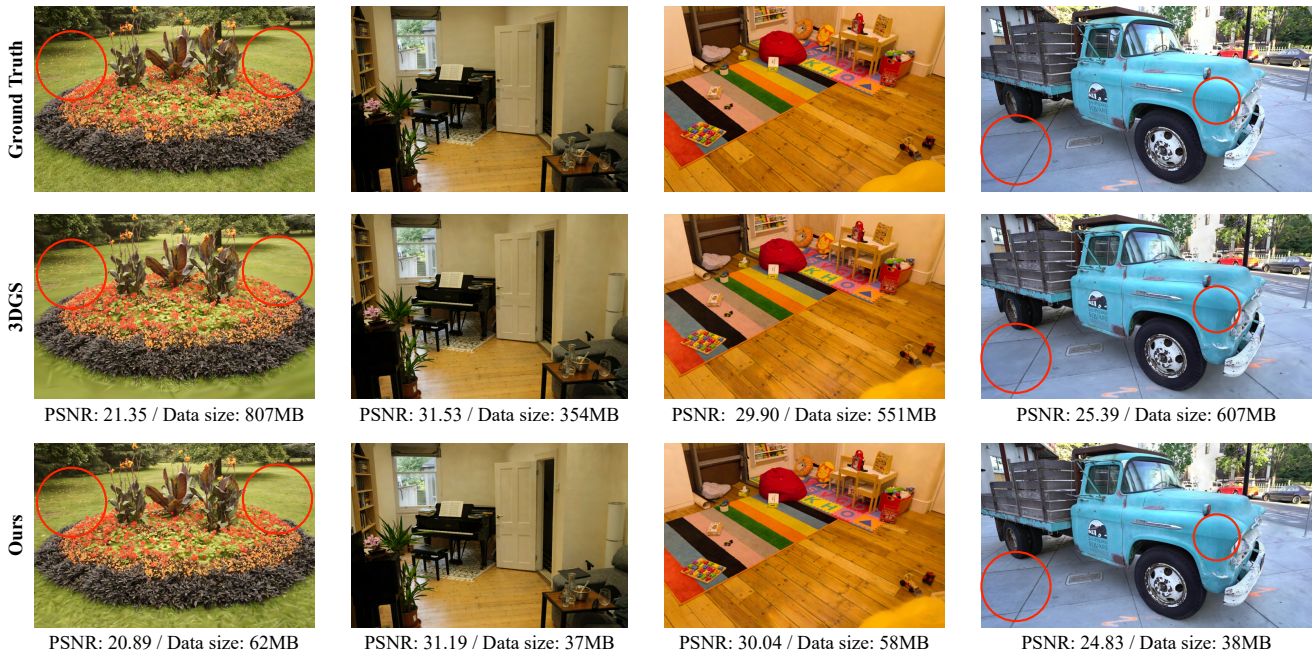

Figure 4. Qualitative comparison of our approach and the original 3DGS. Our approach shows comparable or better quality in metrics while having a smaller data size. Qualitatively, our method outperforms the original 3DGS in representing detailed regions, such as the grass behind the flower bed or the pavement below the truck. Details in this figure are best viewed zoomed in.

model size reduction of at least $9\times$. Notably, our method exhibits faster training times and smaller peak GPU memory usage during training than the original 3DGS, hence acquiring time and memory efficiency. This efficiency is attributed to the creation of fewer Gaussians during training, leading to reduced data and faster rendering for loss calculation. Additionally, the rendering FPS is higher for our approach than the original 3DGS because the 3DGS rendering pipeline includes a preprocessing stage done per Gaussian.

4.2.2 Qualitative Results

Figure 4 presents a qualitative comparison between our approach and the original 3DGS using examples selected from various datasets: 'flowers' from the outdoor Mip-NeRF 360, 'room' from the indoor Mip-NeRF 360, 'playroom' from Deep Blending, and 'truck' from Tank&Temples. Results demonstrate that our approach resulted in a model with at least $\times 9$ reduction in memory while maintaining comparable quality. Notably, our method improved quality for areas with small patterns, such as the grass behind the flower bed for the flowers dataset and the pavement and car stain for the truck dataset. Please refer to and zoom in on the red circles in Figure 4. This enhancement is a direct result of our novel densification method, which utilizes color cues to cover the expanded scenario illustrated in Figure 2. In the case of the playroom dataset, higher PSNR was achieved despite the smaller data size.

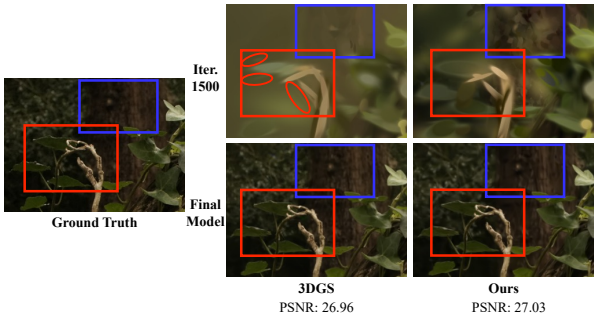

Figure 5. Magnified rendered image from iteration 1500 and the final model on 'stump' from MipNeRF-360. The blue box highlights the efficacy of our densification approach in representing complex patterns, and the red box illustrates the impact of our densification process in creating Gaussians only when necessary. Details in this figure are best viewed zoomed in.

The impact of the novel densification process can be directly observed by examining the evolving Gaussians. In Figure 1, Gaussians are illustrated in equally sized spheres with constant opacity to convey the density of the scene. As seen from the figure, our approach successfully reconstructs the scene with similar quality with a much coarser model. Figure 5 shows the images of the same region from 'stump' of MipNeRF360 trained with 3DGS and our method. The blue box indicates a complex pattern on the tree bark. In our approach at iteration 1500, this region already exhibits multiple Gaussians to effectively represent the intricate pattern, resulting from using the 0th SH gradient as a cue for

Dataset	Stump of Mip-NeRF 360							
Method	SSIM	PSNR	LPIPS	Train	FPS	Disk	GPU Mem	Splats
3DGS	0.770	26.72	0.242	33m19s	89	1038MB	8.38GB	4387k
coarse 3DGS	0.752	26.27	0.279	21m43s	121	509MB	5.40GB	2152k
3DGS+pruning	0.772	26.67	0.249	27m	155	282MB	7.30GB	1193k
only color	0.748	26.14	0.264	31m8s	106	870MB	7.43GB	3679k
only color +pruning	0.752	26.22	0.267	25m55s	157	271MB	7.38GB	1148k
color+coarse	0.771	26.68	0.247	24m19s	126	620MB	6.02GB	2620k
color+coarse +pruning	0.771	26.70	0.252	22m9s	180	237MB	5.63GB	1001k
color+coarse +pruning+fp16	0.768	26.65	0.254	22m9s	178	113MB	5.63GB	1001k

Table 3. Quantitative evaluation of ablation studies. Bolded methods contain our approach. 'coarse' denotes coarse position-based densification, and 'pruning' denotes trainable pruning masks. 'fp16' denotes precision reduction to 16-bit floating points. Evaluated on an NVIDIA 3090 GPU.

densification. Conversely, the original approach at iteration 1500 struggles to create additional Gaussians to represent diverse colors, leading to slower convergence and lower quality. The red box indicates a region with branches and leaves. Both models start from the same SfM points to reconstruct branches. However, the original 3DGS at iteration 1500 creates redundant Gaussians, represented as the numerous semi-transparent thin Gaussians (circled with red ellipsoids, best viewed zoomed in) that contribute less to the final image. In comparison, the red box for our approach demonstrates the creation of only a few additional Gaussians, highlighting the strength of our new approach in adding Gaussians precisely when needed. The final model images of the two approaches are of similar quality, proving that the semi-transparent thin Gaussians were redundant in reconstructing the scene.

4.3. Ablation Studies

We conduct ablation studies to validate the contribution of our novel densification method compared to other techniques. Our contribution is to combine color and coarse position-based densification, while the effects of other techniques, such as pruning using a trained mask and precision reduction, are jointly observed.

4.3.1 Comparison with the Original 3DGS

As we have identified an inefficiency in the original densification method, which tends to create redundant Gaussians, tuning the hyperparameters to make the densification more conservative is a potential solution. To address this issue, we experimented by densifying every 300 iterations instead of the current frequency of 100 ('coarse 3DGS' in Table 3). Results in Table 3 exhibit worse image quality

Figure 6. Images from the ablation studies for 'stump'. Notice the change in detail in the shadows and bark. Details in this figure are best viewed zoomed in.

across all three metrics despite having more Gaussians than our approach ('color + coarse' in Table 3). This shows that position-based densification alone is insufficient for acquiring efficient Gaussians like our approach.

The results presented in Table 3 demonstrate that incorporating a trained pruning mask with the original 3DGS ('3DGS + pruning' in Table 3) needs more Gaussians to achieve comparable qualities to our approach ('color + coarse + pruning' in Table 3). Despite similar qualities, our approach results in fewer Gaussians, making a slight enhancement in terms of data size. Additionally, our approach outperforms the pruning approach in terms of training time by 20% and peak GPU memory usage by 21%. The difference arises from the intermediate Gaussian numbers during training. Although the final number of Gaussians is similar, Figure 8 shows that using a pruning mask with the original 3DGS densifies with redundant Gaussians in a similar manner to the original 3DGS, while our approach adds the Gaussians conservatively only when needed. As a result, our approach demonstrates efficiency gains in terms of training time and GPU usage.

Our method also demonstrates better qualitative performance in representing complex patterns, particularly when compared to pruning. Figure 7 illustrates the rendered images for the methods in ablation studies. Using a pruning mask with the original 3DGS leads to blurred regions, while our approach using color cues succeeds in reconstructing intricate patterns of the pavement.

4.3.2 Joint Effects of Our Approach

The results of using only the color-cued densification are presented in the 'only color' and 'only color + pruning' of Table 3. Using only color cues for densification results in lower quality metrics despite more Gaussians ('only color' and 'only color + pruning'), compared to our comprehensive approach that utilizes both color cues and 2D position

Figure 7. Images from the ablation studies for 'treehill'. Notice the change in detail in the zoomed-in pavement. Details in this figure are best viewed zoomed in.

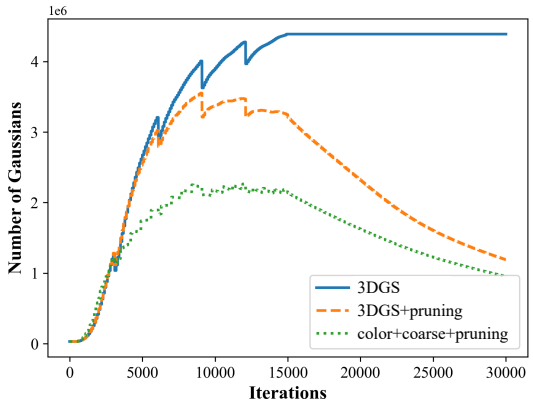

Figure 8. The number of Gaussians in intermediate iterations for the original 3DGS (3DGS), 3DGS with pruning mask (3DGS+pruning), and our approach (color+coarse+pruning) in the ablation studies.

gradients ('color + coarse' and 'color + coarse + pruning'). This is because the SH gradient cannot detect under/over-reconstructed regions, leading to an inaccurate reconstruction of position and geometry information from the coarse SfM points. The deficiency in geometric detail, visible in the barks (circled red) of 'Only Color' in Figure 6, supports the idea that the original 2D position gradient densification plays the role of reconstructing geometry. However, there are also quality gains in the shadows (circled blue) compared to the original 3DGS, as an artifact of using color-based densification. This validates the effectiveness of our comprehensive approach, where color-cued densification and 2D position-based densification serve distinct purposes, as detailed in 3.2.

Reducing the precision of Gaussian features can further reduce the data size. While different approaches like MLP or vector quantization are available, we took a simple method of reducing the feature precision to check the possibility of compression. Table 3 illustrates that compression can further reduce the data size to half without much reduction in quality metrics. This shows that other data compression methods in previous or future 3DGS works can be used on top of ours to gain further benefit.

5. Conclusion

In this work, we enhance the adaptive density control of the original 3DGS to populate Gaussians better for improved efficiency in both time and memory. We expand the densification scheme to cover a broader range of real image scenarios, and we suggest a new densification methodology that aligns with this expanded scheme. Our densification methodology leverages color cues and 2D position gradients to detect the need for additional Gaussians to represent color and complex geometry, respectively. By addressing the inefficiencies of the original densification method, our comprehensive approach significantly reduces the introduction of redundant Gaussians. Moreover, it achieves qualitative improvements in image quality, particularly in handling complex patterns. Jointly used with masked pruning and precision reduction, our novel densification method demonstrates substantial decreases in the number of Gaussians and data size. Reduction of the number of Gaussians inherits other efficiency benefits such as accelerated training time, reduced GPU memory usage, and faster rendering speed. Our novel methodology holds potential for adoption across various 3DGS variants, offering efficiency benefits in memory and time. Additionally, it will encourage discussion on strategies to populate Gaussians effectively to enhance quality.

Acknowledgements. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea Government(MSIT) (No. RS-2023-00218601; 2022R1A2C3008495) and by the Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOE) (P0025681, Semiconductor-Specialized University).

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 5

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4

[3] Thomas Bird, Johannes Ballé, Saurabh Singh, and Philip A Chou. 3d scene compression through entropy penalized neural representation functions. In *2021 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2021. 2

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2

[5] Chenxi Lola Deng and Enzo Tartaglione. Compressing explicit voxel grid representations: fast nerfs become also small. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1236–1245, 2023. 2

[6] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. 2, 3

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2, 5

[8] Cameron Gordon, Shin-Fang Chng, Lachlan MacDonald, and Simon Lucey. On quantizing implicit neural representations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 341–350, 2023. 2

[9] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 5

[10] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 2

[11] Seungyeop Kang and Sungjoo Yoo. Ternarynerf: Quantizing voxel grid-based nerf models. In *2022 IEEE International Workshop on Rapid System Prototyping (RSP)*, pages 8–14. IEEE, 2022. 2

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 1, 2, 5

[13] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 5

[14] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. *arXiv preprint arXiv:2311.13681*, 2023. 2, 3, 4

[15] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 2, 5

[16] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. *arXiv preprint arXiv:2401.02436*, 2023. 2, 3

[17] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 2

[18] Jinglei Shi and Christine Guillemot. Light field compression via compact neural scene representation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023. 2

[19] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 2

[20] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 2

[21] Yiying Yang, Wen Liu, Fukun Yin, Xin Chen, Gang Yu, Jiayuan Fan, and Tao Chen. Vq-nerf: Vector quantization enhances implicit neural representations. *arXiv preprint arXiv:2310.14487*, 2023. 2

[22] Zhixiang Ye, Qinghao Hu, Tianli Zhao, Wangping Zhou, and Jian Cheng. Mcunerf: Packing nerf into an mcu with 1mb memory. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9082–9092, 2023. 2

[23] Tianli Zhao, Jiayuan Chen, Cong Leng, and Jian Cheng. Tinynerf: Towards 100 x compression of voxel radiance fields. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3588–3596, 2023. 2

[24] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 2

[25] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 2