An Evolutionary Boost to Gaussian Splatting Optimization

Berk Kıvılcım[0009−0003−8159−7434], Erchan Aptoula[0000−0001−6168−2883], and Huseyin Ozkan[0000−0002−5539−9085]

Sabanci University, Istanbul, Türkiye
{berk.kivilcim, erchan.aptoula, huseyin.ozkan}@sabanciuniv.edu

Abstract. The introduction of Gaussian Splatting has had a notable impact on 3D reconstruction and novel view synthesis, enabling real-time rendering with photorealistic quality. However, despite advances, it still relies heavily on gradient-based optimization and gradient-driven densification strategies such as splitting, cloning and pruning, that are inherently accompanied by limitations such as susceptibility to local minima, insufficient densification in low-gradient yet high-frequency regions, and under-representation. The periodic application of opacity reset has been shown to mitigate some of these issues via Gaussian importance redistribution and refreshing gradient flow. In this paper, a gradient-free, post opacity reset, modular and complementary evolutionary optimization step is proposed for Gaussian Splatting. The resulting hybrid optimization scheme enables exploring a wider extent of Gaussian configurations, leading to PSNR improvements of up to 1.12dB across 3DGS, PixelGS and MipGS variants, the most notable gains observed in low-frequency and large homogenous regions.

Keywords: Genetic Algorithms · Gaussian Splatting · 3D Reconstruction · Novel View Synthesis

1 Introduction

Novel view synthesis and 3D reconstruction have long been established as two fundamental computer vision problems [9]. Novel view synthesis aims to generate realistic images from previously unseen viewpoints, while 3D reconstruction seeks to recover the underlying three-dimensional structure of a scene from 2D observations. They are usefull for a wide range of applications, including smart cities, animation, robotics, navigation, augmented and virtual reality, autonomous driving, and historical preservation [1, 5, 7, 13]. Following the success of neural radiance fields (NeRF) [16], Gaussian Splatting (GS) has emerged as an alternative approach that represents scenes using anisotropic Gaussian primitives by performing per-scene optimization (fitting the Gaussian parameters to a specific set of training views). This approach has initiated a rapidly growing line of research [14]. Unlike NeRF-based methods, GS leverages a differentiable rasterization pipeline to achieve substantially faster rendering, making it particularly suitable for real-time applications [14].

In GS, each Gaussian primitive is characterized by its 3D position, covariance parameters, opacity, and color. These primitive parameters are optimized through gradient-based learning, where gradients are computed based on a loss defined between rendered outputs and corresponding ground-truth images. While GS has shown strong empirical performance, several limitations of gradient-driven optimization have been highlighted [18, 27].

Large homogeneous regions, such as walls or sky, may ideally be represented by large-scale Gaussians [26]. However, such Gaussians can be incorrectly interpreted as sub-optimal and subjected to unnecessary split or prune operations [8]. This issue can lead to either under-representation or over-representation within different regions of the scene. In the case of under-representation, insufficiently or inaccurately placed Gaussians may result in distortions or blurring. In the case of over-representation, an excessive number of Gaussians in certain areas may lead to increased noise or artifacts. This issue largely stems from the densification mechanism's heavy dependence on scene geometry [2]. To mitigate these challenges, several alternative strategies have been proposed, including pixel-weighted gradients [21] and absolute gradient formulations [23]. Although these approaches refine the gradient signal, they still remain fundamentally tied to gradient-based optimization and geometry-dependent heuristics.

A complementary optimization paradigm, one that is independent of both explicit gradient signals and current geometric heuristics, has not yet been fully explored within GS. Motivated by this gap, this work integrates a Genetic Algorithm (GA) based optimization strategy [11] into the GS training process. Following the baseline architecture selection protocol of [22], the proposed method requires no architectural modifications and is integrated into three different GS pipelines: 3DGS [14], PixelGS [26], and MipGS [24]. By introducing an evolutionary search mechanism that does not rely directly on gradient magnitudes or densification heuristics, this study investigates whether a GA-based approach can reduce the previously discussed limitations and improve reconstruction quality. Experimental results indicate average PSNR improvements of 0.43 dB on the training set and 0.12 dB on the test set across all evaluated scenes for 3DGS adaptation.

2 Preliminaries

This section presents the fundamental concepts of the two main components of this study: GA and GS.

2.1 Genetic Algorithms

GA have long been established as a family of stochastic search procedures [11]. A GA iteratively searches for improved solutions by applying stochastic mutation and crossover operators, followed by the selection of individuals that better satisfy a predefined fitness function [6]. In this context, mutation refers to small stochastic modifications applied to the parameters of an individual, and crossover

refers to the construction of a new candidate solution by combining parameters from two parent individuals. For these operators to be applied, a GA operates over a set of candidate solutions simultaneously, and therefore a population of individuals is required [17]. Accordingly, the core components of a GA can be summarized as population initialization, fitness evaluation, selection, crossover and mutation. GA is used in a wide range of problems, including combinatorial optimization, network design, scheduling, reliability engineering, and logistics network optimization [4].

2.2 Gaussian Splatting

GS represents a scene using a set of 3D Gaussians, where each Gaussian is parameterized by its spatial position, covariance, color, and opacity. During training, these parameters are optimized via gradient-based methods to minimize the discrepancy between rendered images and ground-truth observations. In summary, the goal of this optimization process is to find the Gaussian configuration that best represents a given scene [14].

To render an image using the Gaussians, the 3D positions and covariance matrices of the Gaussians are first projected onto the 2D image plane [14]. Given the mean ($\boldsymbol{\mu}'$) and covariance ($\boldsymbol{\Sigma}'$) of a projected 2D Gaussian, its contribution at a pixel location $\mathbf{p} \in \mathbb{Z}^2$ on the 2D image can be formulated as

$$G(\mathbf{p}) = \exp\Big(-\tfrac{1}{2}(\mathbf{p}-\boldsymbol{\mu}')^\top \boldsymbol{\Sigma}'^{-1}(\mathbf{p}-\boldsymbol{\mu}')\Big). \tag{1}$$

Then, the resulting Gaussian response is combined with the learned opacity o to obtain the per-pixel alpha value used during alpha blending:

$$\alpha(\mathbf{p}) = o \cdot G(\mathbf{p}). \tag{2}$$

For each pixel, the final color value of the pixel p is then computed by blending the contributions of the N Gaussians overlapping that pixel in front-to-back order:

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

where $\mathbf{c}_i \in \mathbb{R}^3$ denotes the view-dependent color value of the i-th Gaussian, and N is the number of Gaussians overlapping pixel location $\mathbf{p}$. Consequently, more accurate opacity, position, covariance, and color values lead to a more accurate rendered output.

However, the Gaussian primitive parameter optimization problem is highly non-convex and the learned representation may become sensitive to local minima and suboptimal parameter updates [12]. To mitigate this issue, GS introduces adaptive density control, which refines the representation by splitting, cloning, pruning and opacity resetting during training. Splitting divides a single Gaussian into two smaller ones in regions where it covers a large area. Cloning duplicates a Gaussian in under-reconstructed regions where additional primitives are needed

to capture missing geometry. Pruning eliminates Gaussians whose opacity falls below a predefined threshold, since they contribute little to the rendered output and only increase computational cost. Opacity reset periodically lowers the opacity of all Gaussians to a small value. Nevertheless, such refinement is still driven primarily by gradient-based decisions and may not always be sufficient to explore alternative promising configurations. Motivated by this limitation, genetic optimization is incorporated in this work as an additional optimization process to enhance the scene representation.

3 Proposed Method: An evolutionary boost to GS

This section describes the core components of our GA-based proposed method (population initialization, fitness evaluation, selection, crossover, and mutation) and explains how they are mapped onto the GS pipeline. The integration of the GA into the GS pipeline is illustrated in Figure 1. In the proposed setting, an individual consists of multiple Gaussians that collectively form a candidate solution for representing a given scene. A population of P individuals is defined as $\mathcal{P} = \{\mathbf{ind}_1, \mathbf{ind}_2, \ldots, \mathbf{ind}_P\}$, where each individual is defined as the collection of N Gaussians and their primitive parameter vectors as $\mathbf{ind}_j = [\mathbf{v}_{j1}, \mathbf{v}_{j2}, \ldots, \mathbf{v}_{jN}]$. Each Gaussian is represented by a parameter vector $\mathbf{v}_{ji} = [\mu_{ji}, \mathbf{\Sigma}_{ji}, o_{ji}, \mathbf{k}_{ji}]$, where $\mu_{ji} \in \mathbb{R}^3$ denotes the position, $\mathbf{\Sigma}_{ji} \in \mathbb{R}^3$ denotes the scale, $o_{ji} \in \mathbb{R}$ denotes the opacity, and $\mathbf{k}_{ji} \in \mathbb{R}^{48}$ denotes the spherical harmonic coefficients encoding the view-dependent color ($\mathbf{c_{ji}}$) of the i-th Gaussian of the j-th individual. Since different individuals in a population encode the scene using different Gaussian configurations, they may produce varying rendering results. This is because the opacity, position, and scale values of a Gaussian are modified through mutation, and these parameters directly contribute to the rendered output.

3.1 Genetic Algorithm Pipeline

The Gaussian set that is optimized during the GS training is used as the initialization point for the GA after the opacity reset stages. This initial Gaussian set is referred to as the *Base Gaussian*. To generate diverse candidate solutions (a population) from this single base Gaussian, stochastic mutations are applied to the base. This mutation procedure is applied only once at the beginning of each GA invocation to create the initial population. To ensure that the initial individuals do not deviate excessively from the base Gaussian configuration, a smaller mutation rate than that used during the subsequent evolutionary iterations is employed. During the mutation process, only the position, scale, and opacity parameters are modified. Color parameters are intentionally excluded from mutation, since variations in color can significantly influence the rendered output without directly contributing to the geometric refinement of the scene. The mathematical details of the mutation process are provided in Section 3.2.

After generating the population (diverse candidate solutions), each individual is evaluated using the fitness function to determine whether any of them outper-

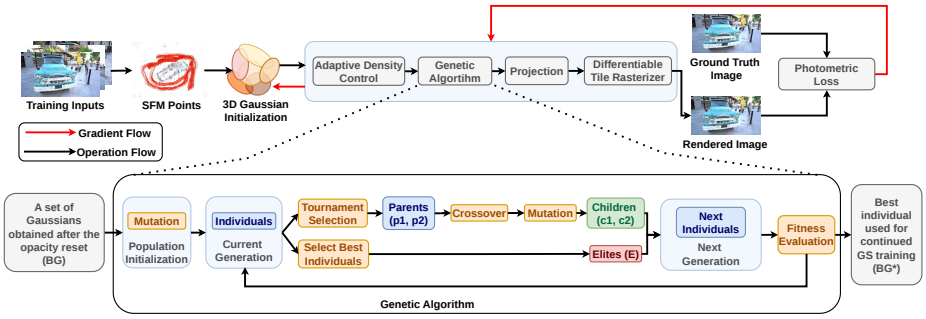

Fig. 1. Overview of the proposed GA pipeline integrated into GS. The GA is activated after each opacity reset. Mutations are first applied to the base Gaussians to create the initial population. The population is then iteratively refined through crossover and mutation, while a small number of best-performing individuals (elites) are carried over directly to the next generation. Once the predefined number of GA iterations is reached, the GS pipeline resumes training with the best-performing Gaussian set obtained from this process.

form the current base configuration. If a superior individual is found, the base configuration is replaced by that individual; otherwise, it is retained unchanged.

Subsequently, a fixed proportion of the population, consisting of the top-performing individuals, are carried over directly to the next generation without modification; these are referred to as *elites*. To generate the remaining individuals of the new population, a tournament selection procedure is applied to select parent candidates for crossover. In the tournament selection method, T individuals are randomly sampled from the population, where T denotes the tournament size, and the top-performing individual within this subset is selected as the first parent p_1. The same procedure is repeated independently to select the second parent p_2. After selecting p_1 and p_2, crossover is applied to generate two offspring candidates c_1 and c_2. After selecting the two parents, three crossover operators were evaluated: 1-point crossover, in which a single point splits the parent parameters into 2 segments that are swapped; 100-point crossover, in which 100 points split the parameters into 101 segments that are alternately exchanged; and uniform crossover, in which each parameter is independently inherited from either parent with equal probability. However, the accuracy differences across the different crossover variants were not significant. Therefore, 100-point crossover was selected for all subsequent experiments in this study. Subsequently, mutations are performed on the resulting offspring. This process produces two new individuals that are added to the next generation. The procedure is repeated until the newly formed next population reaches the predefined population size together with the elite individuals.

Once the next-generation population is created, all new individuals are evaluated to determine whether any of them outperform the current base configuration. If a superior individual is found, the base configuration is replaced

Algorithm 1 Genetic Algorithm for Gaussian Splatting Refinement

Require: Base Gaussian set BG, population size P, elite count E, tournament size T, number of genetic iterations K
Ensure: Refined Gaussian set BG^*

// **Create initial population by mutating the base**
$Population \leftarrow \emptyset$
for $i = 1$ **to** P **do**
  $individual_i \leftarrow \textsc{Mutate}(BG, \textit{initial mutation rate})$
  $Population \leftarrow Population \cup \{individual_i\}$
end for
$Population \leftarrow \textsc{EvaluateFitness}(Population)$ ▷ returns fitness-ordered population
if best individual in $Population$ is better than BG **then**
  $BG \leftarrow$ best individual in $Population$
end if
// **Evolutionary loop**
for $k = 1$ **to** K **do**
  $NextGeneration \leftarrow$ top E individuals of $Population$ ▷ elitism
  while $|NextGeneration| < P$ **do**
    $p_1 \leftarrow \textsc{TournamentSelection}(Population, T)$
    $p_2 \leftarrow \textsc{TournamentSelection}(Population, T)$
    $(c_1, c_2) \leftarrow \textsc{Crossover}(p_1, p_2)$
    $c_1 \leftarrow \textsc{Mutate}(c_1, \textit{mutation rate})$
    $c_2 \leftarrow \textsc{Mutate}(c_2, \textit{mutation rate})$
    $NextGeneration \leftarrow NextGeneration \cup \{c_1, c_2\}$
  end while
  $NextGeneration \leftarrow \textsc{EvaluateFitness}(NextGeneration)$ ▷ returns fitness-ordered population
  if best individual in $NextGeneration$ is better than BG **then**
    $BG \leftarrow$ best individual in $NextGeneration$
  end if
  $Population \leftarrow NextGeneration$
end for
return $BG^* \leftarrow BG$

by the best-performing individual. This evolutionary cycle with elite selection, crossover, and mutation is repeated until the predefined number of genetic iterations is completed. As a result, the search for improved solutions proceeds stochastically through crossover and mutation. Algorithm 1 provides the pseudocode for overall evolutionary refinement process.

3.2 Mutation Mechanism

Within each individual, a subset of Gaussians is randomly selected to undergo mutation. For each selected Gaussian, the primitive gene parameters to be mutated (opacity, position, and scale) are also randomly chosen. The magnitude of the applied mutation is controlled by four main factors:

1. the mutation intensity,
2. the gradient of the parameter to be mutated,
3. the standard deviation of that parameter,
4. a stochastic perturbation sampled from a normal distribution, which allows shifts in both positive and negative directions and increases the diversity of candidate solutions.

The general mutation formulation is given below:

$$x^{\text{mutated}} = x^{\text{base}} + \alpha \cdot u_i \cdot \sigma_x \cdot \mathcal{N}(0,1), \tag{4}$$

where x^{base} denotes the original value of a Gaussian primitive parameter to be mutated, x^{mutated} is the resulting value after mutation, α denotes the mutation strength, $u_i \in (0,1)$ is a scaling factor derived from the derivative magnitude of the corresponding gene (smaller derivative magnitudes tend to yield smaller mutation effects, while larger derivatives lead to proportionally larger updates), σ_x denotes the standard deviation of x, which scales the sampled noise, and $\mathcal{N}(0,1)$ is a real-valued random variable drawn from a standard normal distribution with mean 0 and standard deviation 1.

Note that while the evolutionary search itself is independent of gradient-based optimization, accumulated gradient statistics are leveraged here as an optional side signal to inform the mutation magnitude. Indeed, this component can be removed entirely from the formulation, in which case the GA would still operate as a fully gradient-free method by setting $u_i = 1$ for all selected Gaussians. The scaling factor $u_i \in (0,1)$ is computed from the accumulated gradients through a sequence of transformations. Let $acc_i \in R$ denote the accumulated gradient of the i-th Gaussian over d_i contributing images. The mean gradient is then obtained by normalizing the accumulation over the number of contributing images as $g_i = acc_i/d_i$. Since gradient values may take both positive and negative values, and subsequent logarithmic operations require non-negative inputs, the gradient magnitude is computed using the ℓ_2 norm as $g_i \leftarrow \|g_i\|_2$. To control the influence of large gradient magnitudes, a logarithmic transformation $h_i = \log(1 + g_i)$ is applied; the term $1+g_i$ ensures numerical stability, since $h_i = 0$ when $g_i = 0$ and the logarithm remains well-defined. The values h_i are then standardized using their mean $\mu_h = \frac{1}{k}\sum_{i=1}^{k} h_i$ and standard deviation $\sigma_h = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(h_i - \mu_h)^2}$, where k is the number of selected Gaussians, yielding the standardized variable $z_i = (h_i - \mu_h)/\sigma_h$. Finally, z_i is mapped to the interval $(0,1)$ using a sigmoid function, $u_i = 1/(1 + e^{-z_i})$, which produces the bounded scaling factor used in the mutation formulation.

3.3 Fitness Function

In this study, the fitness function is defined based on the original GS loss function. For each individual, rendered images are first generated using the set of Gaussian primitives. The discrepancy between the rendered images and the corresponding ground-truth images is then computed as

$$\mathcal{L} = (1-\lambda)\,\|I_{\text{rendered}} - I_{\text{gt}}\|_1 + \lambda\,(1 - \text{SSIM}(I_{\text{rendered}}, I_{\text{gt}}))\,, \tag{5}$$

where I_{rendered} denotes the rendered image obtained from the Gaussian configuration, I_{gt} represents the ground-truth image, and λ controls the trade-off between the ℓ_1 reconstruction loss and the Structural Similarity Index Measurement (SSIM) term. However, since a large number of individuals are evaluated within each GA iteration, computing the fitness over all training images becomes computationally prohibitive. Therefore, only a limited number of training images are used during genetic optimization. The overall computational cost scales proportionally with the number of genetic iterations K, the population size P, and the number of images V rendered for fitness evaluation. Formally, the total runtime and GPU memory consumption can be approximated as $Time \propto K \times P \times V$ and $Memory \propto P \times V$, respectively. Therefore, a fixed subset of training images is recommended to be chosen. The selected subset is consistently used for fitness evaluation across all genetic iterations. To ensure that the selected images were well distributed across the scene and captured diverse structural elements, uniform sampling was performed over the dataset.

4 Experiments and Results

This section first describes the datasets, hyperparameters and training times used in this study. Both qualitative and quantitative results are then presented along with their discussion.

4.1 Dataset Selection

Real-world scenes from several datasets frequently used in the prior GS literature were employed in this study. From the Nerfstudio dataset [19], the Poster scene (an object-centric close-range capture) and the Library scene (a large open scene with a broader capture trajectory) were selected. In addition, the Truck and Train scenes from the Tanks and Temples dataset [15], Playroom and DrJohnson from the Deep Blending dataset [10], and Garden and Bicycle from the MipNeRF360 dataset [3] were chosen. The primary motivation for selecting these particular datasets is that they have also been extensively used previously [1, 5, 7, 13] for both qualitative and quantitative evaluations, thereby enabling more direct and meaningful comparisons. Following the standard evaluation protocol adopted in prior GS works [14], every eighth image of each scene is held out as the test set, and the remaining images are used for training [14, 22].

4.2 Experimental Setup

All hyperparameters were kept consistent with the original 3DGS, PixelGS and MipGS methods [14, 24, 26], and no modifications were made to the workflow of the baseline pipelines. The additional hyperparameters values for the GA component introduced and described in Table 1. The number of evaluation cameras was experimentally set to 6, and further increases were observed to provide no additional benefit. Both the mutation and crossover rates were adjusted in a

Table 1. Hyperparameters of the proposed GA-based refinement and their values used in all experiments.

Hyperparameter	**Value**	**Description**
GA Iterations	100	Total number of evolutionary iterations per invocation.
Population Size	100	Number of candidate solutions (individuals) per generation.
Tournament Size	10	Number of individuals sampled per tournament selection.
Elite Fraction	0.10	Proportion of top-performing individuals carried over unchanged to the next generation.
Evaluation Cameras	6	Number of fixed training images used for fitness evaluation.
Selection Percentage	0.10	Fraction of Gaussians per individual subject to crossover and mutation per GA iteration.
Mutation Intensity	0.10	Magnitude of perturbation applied during mutation.
Initial Mutation Rate	0.03	Mutation rate used for generating the initial population.
Mutation Rate	15%/5%/2%	Determines the probability of mutation being applied to Gaussian parameters.
Crossover Rate	85%/70%/50%	Determines the probability of crossover being applied to generate offspring from selected parents.

piecewise manner: higher values were used for the first 30% of GA iterations, reduced for iterations between 30% and 70%, and further reduced for the remaining iterations. A lower mutation rate was employed for population initialization to prevent excessive deviation from the base Gaussian configuration. The GA-based refinement was applied at multiple stages of the GS training process, specifically before and after the opacity reset step, as well as before and after the densification step. The results consistently showed that only activating the GA after the opacity reset led to improvements in the final reconstruction quality. In contrast, applying the GA at other stages of training did not produce any measurable impact on the final results. One possible interpretation is that the GA becomes effective only after the opacity reset because, in other stages of training, the model is already relatively well optimized by gradient-based updates, leaving little opportunity for a random evolutionary search to produce further improvements. However, when the opacity reset is applied, this highly optimized structure is partially disturbed, creating a less stable but more adaptable optimization state. Therefore, in this study, all reported experiments employ the GA exclusively after opacity reset stages. Each experimental setup was repeated three times, and the average values of Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [20], and Learned Perceptual Image Patch Similarity (LPIPS) [25] were reported.

Table 2. Quantitative comparison on the training set. Higher values are better for PSNR and SSIM, while lower values are better for LPIPS. Δ values are computed as GA − baseline for PSNR and SSIM, and baseline − GA for LPIPS. Blue values indicate improvements.

	3DGS			3DGS + GA			PixelGS		
Dataset	PSNR	SSIM	LPIPS	ΔP	ΔS	ΔL	PSNR	SSIM	LPIPS
Poster	35.78	0.9724	0.141	+0.26	+0.0004	+0.002	36.43	0.9748	0.121
Library	29.62	0.9409	0.071	+0.94	+0.0062	+0.009	30.17	0.9463	0.063
Train	25.60	0.8457	0.185	+0.07	+0.0007	+0.003	26.10	0.8778	0.143
Truck	27.16	0.8942	0.139	+0.45	+0.0050	+0.006	27.73	0.9098	0.110
Playroom	37.99	0.9598	0.105	+0.21	+0.0004	0.000	38.39	0.9627	0.100
DrJohnson	37.70	0.9626	0.108	+0.34	+0.0011	+0.003	38.41	0.9680	0.095
Garden	29.76	0.9056	0.073	+0.50	+0.0026	+0.001	30.21	0.9182	0.063
Bicycle	25.91	0.8369	0.173	+0.69	+0.0080	+0.015	26.60	0.8683	0.137

	PixelGS + GA			MipGS			MipGS + GA		
Dataset	ΔP	ΔS	ΔL	PSNR	SSIM	LPIPS	ΔP	ΔS	ΔL
Poster	+0.13	+0.0005	+0.002	36.57	0.9752	0.123	+0.06	+0.0001	-0.002
Library	+0.79	+0.0050	+0.008	29.74	0.9394	0.078	+0.93	+0.0081	+0.015
Train	+0.30	+0.0042	+0.006	25.71	0.8667	0.163	+0.09	-0.0004	0.000
Truck	+0.64	+0.0073	+0.009	28.08	0.9141	0.112	+0.27	+0.0021	+0.002
Playroom	+0.34	+0.0007	+0.004	38.65	0.9654	0.102	+0.18	+0.0002	+0.002
DrJohnson	+0.67	+0.0030	+0.009	38.73	0.9696	0.098	+0.34	+0.0011	+0.004
Garden	+0.77	+0.0051	+0.004	30.65	0.9228	0.059	+0.51	+0.0011	+0.001
Bicycle	+0.84	+0.0138	+0.018	27.14	0.8901	0.105	+0.66	+0.0045	+0.003

4.3 Implementation Details

All experiments were conducted using an NVIDIA RTX 3090 GPU (24 GB VRAM). The GA was activated four times during training, specifically at the opacity reset iterations (i.e. at 3000, 6000, 9000, and 12000 iterations). For the Library dataset, the baseline method completes training in approximately 25 minutes, achieving a test PSNR of 29.10. When the proposed GA is applied with a population size of 50 and generation count of 50, the total training time increases to 43 minutes, with the PSNR improving to 29.89. Increasing both the population size and the generation count to 100 further extends the training time to 90 minutes, resulting in a test PSNR of 29.99. These results indicate that, while the GA introduces a substantial increase in training time, it can yield meaningful improvements in reconstruction quality for certain datasets.

4.4 Results

Quantitative Analysis: For each dataset, the quantitative results in terms of PSNR, SSIM, and LPIPS on training set are presented in Table 2. The corresponding test set results are provided in Table 3. Although the GA is guided by only six selected training images, the full training-set evaluation improves across

Table 3. Quantitative comparison on the test set. Higher values are better for PSNR and SSIM, while lower values are better for LPIPS. Δ values are computed as GA − baseline for PSNR and SSIM, and baseline − GA for LPIPS. Blue values indicate improvements.

	3DGS			3DGS + GA			PixelGS		
Dataset	PSNR	SSIM	LPIPS	ΔP	ΔS	ΔL	PSNR	SSIM	LPIPS
Poster	34.09	0.9664	0.156	+0.22	+0.0002	+0.004	34.60	0.9679	0.136
Library	29.10	0.9301	0.079	+0.89	+0.0072	+0.009	29.64	0.9362	0.070
Train	21.91	0.8003	0.207	+0.12	-0.0017	+0.003	22.31	0.8163	0.169
Truck	25.04	0.8618	0.146	+0.14	+0.0016	+0.004	25.28	0.8691	0.122
Playroom	30.39	0.9125	0.153	+0.20	-0.0008	+0.001	30.38	0.9119	0.148
DrJohnson	28.40	0.8639	0.199	-0.12	-0.0021	-0.001	28.24	0.8600	0.197
Garden	27.36	0.8570	0.089	-0.39	-0.0032	-0.001	27.48	0.8624	0.081
Bicycle	25.26	0.7526	0.198	-0.08	-0.0006	+0.008	25.37	0.7680	0.161

	PixelGS + GA			MipGS			MipGS + GA		
Dataset	ΔP	ΔS	ΔL	PSNR	SSIM	LPIPS	ΔP	ΔS	ΔL
Poster	+0.04	+0.0002	+0.001	33.93	0.9674	0.142	+0.01	0.0000	-0.001
Library	+0.71	+0.0048	+0.007	28.69	0.9261	0.090	+1.12	+0.0214	+0.016
Train	-0.23	-0.0044	+0.002	21.85	0.8113	0.190	-0.32	-0.0046	-0.002
Truck	-0.04	-0.0006	+0.005	25.37	0.8752	0.124	-0.11	-0.0008	+0.001
Playroom	+0.15	-0.0024	+0.001	30.39	0.9141	0.156	+0.03	-0.0010	+0.001
DrJohnson	-0.19	-0.0064	-0.004	27.77	0.8504	0.216	-0.09	-0.0021	-0.002
Garden	-0.68	-0.0080	-0.003	27.60	0.8686	0.077	-0.94	-0.0099	-0.009
Bicycle	-0.18	-0.0066	+0.009	25.55	0.7832	0.140	-0.21	-0.0043	-0.002

all eight scenes, with an average PSNR improvement of +0.43 dB compared to the original 3DGS, indicating that the refinement is not limited to the selected fitness images. Consistent trends are also observed in SSIM and LPIPS, where the GA-enhanced model generally achieves higher SSIM and lower LPIPS values on all training datasets. However, the inconsistent test-set behavior suggests that the method may still bias the representation toward the training distribution. A decrease in test performance is observed for several scenes, particularly those from the MipNeRF360 dataset. On the test set, PSNR improvements are observed on five out of eight scenes, with an average improvement of +0.12 dB in total and +0.31 dB across only the improving scenes. This suggests that, despite the consistent gains observed on the training set, the proposed integration does not generalize equally well to all scenes and remains limited in more complex scenes with finer structural details. Despite these test-set degradations, LPIPS improvements are observed in six out of eight scenes for both 3DGS and PixelGS. Since LPIPS [25] is a learned perceptual metric where lower values indicate better perceptual quality, these results suggest that the proposed refinement provides perceptual quality gains in most of the evaluated scenes. In contrast, when integrated into MipGS, both PSNR and LPIPS improvements are observed in only three out of eight scenes, suggesting that the effectiveness of the proposed strategy may depend on the underlying GS variant.

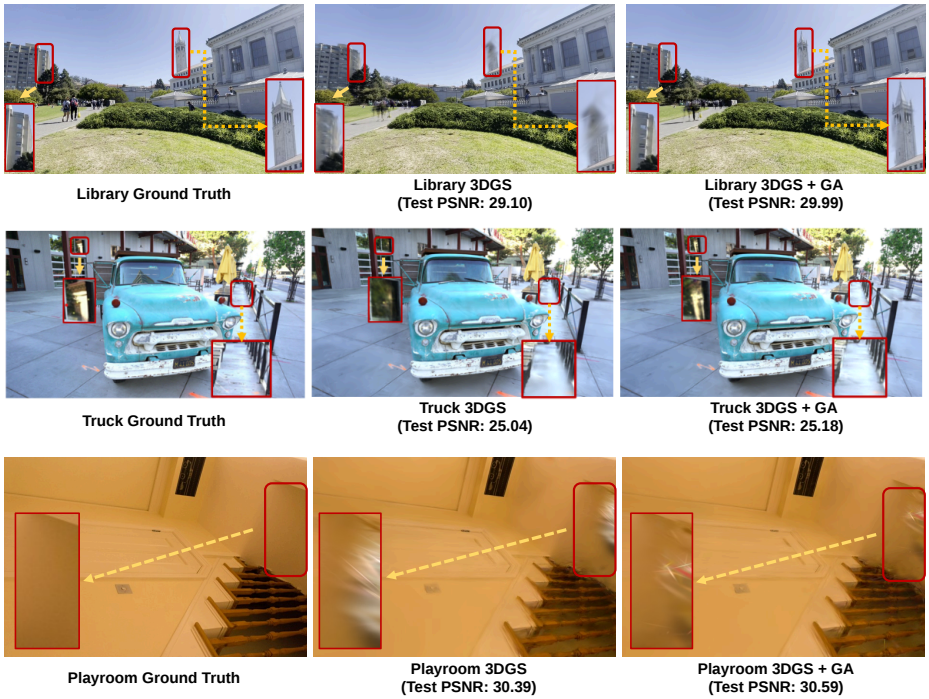

Fig. 2. Qualitative comparison between the baseline 3DGS and the proposed method (3DGS + GA) on selected scenes. Red boxes highlight the regions where the proposed method yields the most noticeable improvements.

Qualitative Analysis: The proposed method appears most effective on datasets characterized by large, low-frequency regions. For instance, as illustrated in Figure 2, notable improvements are observed for the Library dataset, both quantitatively and qualitatively. The same figure also shows notable gains for the Truck dataset, where the representation of low-frequency and large-scale regions (such as pavement structures) is improved. In addition, improvements are observed in the Playroom dataset. While the baseline method struggles with reconstruction in textureless wall regions and produces visible artifacts, the proposed approach mitigates these artifacts to some extent, leading to more consistent results in such low-frequency areas. For the Poster dataset, numerical improvements are also evident. However, since the baseline method already achieves a test PSNR above 34dB, the visual gains are difficult to distinguish through visual inspection.

On the other hand, the proposed method exhibits limitations in regions with high-frequency content and fine details, where the other methods also struggle to produce accurate reconstructions. In such cases, a degradation in test performance is observed. For example, as shown in Figure 3, the DrJohnson scene contains intricate textures that are difficult to reconstruct accurately for the orig-

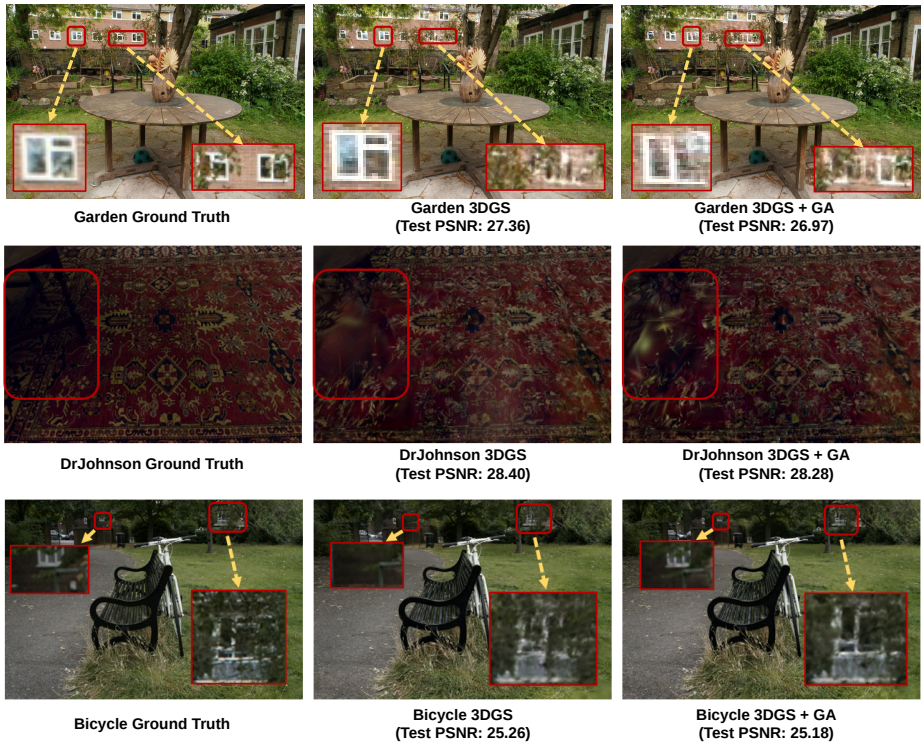

Fig. 3. Qualitative comparison on scenes where the proposed method degrades test-set reconstruction quality. Red boxes highlight the main differences between the baseline 3DGS and the proposed approach.

inal 3DGS method. While the GA attempts to refine these regions, it struggles to capture high-frequency variations, resulting in a slight overall performance drop. A similar behavior is observed in the Garden dataset, where distant window structures remain challenging to reconstruct, and the proposed method fails to improve upon the baseline.

5 Conclusion

In this work, a gradient-free evolutionary refinement was investigated as a complement to gradient-based optimization in GS. By applying a GA after each opacity reset stage, the proposed approach explores alternative Gaussian configurations. This work presents an initial investigation into evolutionary refinement for GS optimization, highlighting both its potential and its current limitations.

The proposed method was integrated and tested on three different GS pipelines (3DGS, PixelGS, and MipGS) across eight scenes from four widely used datasets. Consistent improvements in reconstruction quality have been observed on all

training sets. On the test set, PSNR improvements have been obtained on five out of eight scenes for both the 3DGS, while LPIPS values have improved on six out of eight scenes. The proposed refinement appears most effective on scenes characterized by large, low-frequency regions, and less effective on scenes with high-frequency content and fine structural details, such as those from the Mip-NeRF360 dataset. When integrated into MipGS, improvements have been observed in only three out of eight scenes, suggesting that the effectiveness of the evolutionary refinement may also depend on the underlying GS variant.

A natural direction for future work might be the design of an error-guided variant, in which the selection of Gaussians to be mutated is informed by per-pixel reconstruction error maps. Preliminary experiments along this direction have yielded mixed results: training set accuracy improves further, but the gains do not consistently transfer to the test set. A more systematic investigation of this strategy remains an open question. A second direction concerns the initialization of the genetic population. In the current implementation, the initial population is generated through stochastic mutations applied to the base Gaussian configuration. Therefore, more informed initialization strategies could be explored to provide a more diverse and effective starting population.

6 Acknowledgement

This work was supported by The Scientific and Technological Research Council of Türkiye under Contract 125E729.

References

1. Bao, Y., Ding, T., Huo, J., Liu, Y., Li, Y., Li, W., Gao, Y., Luo, J.: 3D Gaussian Splatting: Survey, technologies, challenges, and opportunities. IEEE Transactions on Circuits and Systems for Video Technology **35**(7), 6832–6852 (2025)
2. Baranowski, B., Esposito, S., Gschoßmann, P., Chen, A., Geiger, A.: Conegs: Error-guided densification using pixel cones for improved reconstruction with fewer primitives. In: International Conference on 3D Vision (3DV). pp. 989–999. IEEE (2026)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: MiP-NeRF 360: Unbounded anti-aliased neural radiance fields. In: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR). pp. 5470–5479 (2022)
4. Bodenhofer, U.: Genetic algorithms: theory and applications. Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter **2004** (2003)
5. Chen, G., Wang, W.: A survey on 3D Gaussian Splatting. ACM Computing Surveys (2024)
6. De Jong, K.: Learning with Genetic Algorithms: An overview. Machine learning **3**(2), 121–138 (1988)
7. Gao, K., Gao, Y., He, H., Lu, D., Xu, L., Li, J.: Neural radiance fields in 3d vision: A comprehensive review. Computational Visual Media (2026)
8. Grubert, G., Barthel, F., Hilsmann, A., Eisert, P.: Improving adaptive density control for 3D gaussian splatting. arXiv preprint arXiv:2503.14274 (2025)

9. Han, X.F., Laga, H., Bennamoun, M.: Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(5), 1578–1604 (2019)
10. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics **37**(6), 257:1–257:15 (2018)
11. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Cambridge (1992)
12. Huang, Z., Chen, J., Liu, J., Ji, S.: Opt3DGS: Optimizing 3D Gaussian Splatting with adaptive exploration and curvature-aware exploitation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 40, pp. 5230–5238 (2026)
13. Kathariya, B., Lee, D.Y., Huang, T.W., Shao, T., Yin, P., Su, G.M.: Gaussian Splatting: State-of-the-arts and future trends. In: 2025 International Conference on Computing, Networking and Communications (ICNC). pp. 876–881. IEEE (2025)
14. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian Splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
15. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)
16. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
17. Mitchell, M.: An introduction to genetic algorithms. MIT press, Cambridge (1998)
18. Rota Bulò, S., Porzi, L., Kontschieder, P.: Revising densification in Gaussian Splatting. In: European Conference on Computer Vision. pp. 347–362. Springer (2024)
19. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., et al.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH conference proceedings. pp. 1–12 (2023)
20. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
21. Yan, K., Wang, H., Li, Z., Wang, Y., Li, S., Yang, H.: A large-scale 3D gaussian reconstruction method for optimized adaptive density control in training resource scheduling. Remote Sensing **17**(23), 3868 (2025)
22. Yang, H., Zhang, C., Wang, W., Volino, M., Hilton, A., Zhang, L., Zhu, X.: Improving gaussian splatting with localized points management. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 21696–21705 (2025)
23. Ye, Z., Li, W., Liu, S., Qiao, P., Dou, Y.: Absgs: Recovering fine details in 3D Gaussian Splatting. In: Proceedings of the 32nd ACM international conference on multimedia. pp. 1053–1061 (2024)
24. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: MiP-Splatting: Alias-free 3D Gaussian Splatting. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 19447–19456 (2024)
25. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
26. Zhang, Z., Hu, W., Lao, Y., He, T., Zhao, H.: Pixel-GS: Density control with pixel-aware gradient for 3D Gaussian Splatting. In: European Conference on Computer Vision. pp. 326–342. Springer (2024)
27. Zhou, Z., Xiong, Y.J., Xia, C.M., Zhang, J.C., Zhan, H.J.: Gradient-direction-aware density control for 3D Gaussian Splatting. arXiv preprint:2508.09239 (2025)