

# SPARE-GS: Structural Parsimony and Resource Efficiency for 3D Gaussian Splatting

Zhang Chen, Shuai Wan, Fuzheng Yang, Jiazhi Xia, Weiyao Lin, *Senior Member, IEEE*  
and Junhui Hou, *Senior Member, IEEE*

**Abstract**—3D Gaussian Splatting (3DGS) achieves high-fidelity novel view synthesis in real-time; however its training efficiency and representation compactness are hindered by excessive primitive proliferation. To address this challenge, we formulate the structural evolution of 3DGS as a global budget-constrained optimization problem and derive an optimality condition, which requires the marginal utility of structural resources to be balanced across spatial regions under a finite primitive budget. Based on this formulation, we propose SPARE-GS, a general *plug-and-play* framework that dynamically aligns the distribution of 3D Gaussian primitives with regional representational demand. SPARE-GS estimates capacity-normalized regional demand, assigns adaptive target quotas, and uses regional budget deviations to coordinate densification, pruning and adaptive termination toward a more balanced structural allocation. Extensive experiments across standard, accelerated, and structure-enhanced 3DGS pipelines demonstrate that SPARE-GS reduces the Gaussian count and training time by an average of 30.38% and 23.81%, respectively, while improving the average PSNR. Moreover, the resulting compact representations reduce downstream processing time and improve the rate-distortion performance of diverse compression and pruning methods, demonstrating the broad applicability of global structural budget regulation. We refer readers to the project page at <https://zhangchen2022.github.io/SPARE-GS.github.io/> for the source code and more visual results.

**Index Terms**—3D Gaussian Splatting; Novel View Synthesis; Budgeted Optimization; Structural Parsimony; Efficiency; Compression

## I. INTRODUCTION

3D Gaussian Splatting (3DGS) [1] has emerged as an effective approach for novel view synthesis, achieving a favorable balance between photo-realistic quality and real-time rendering. By explicitly representing scenes with anisotropic

Zhang Chen is with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China, and also with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China. (email: chen-zhang@mail.nwpu.edu.cn)

Shuai Wan is with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China (e-mail: swan@nwpu.edu.cn).

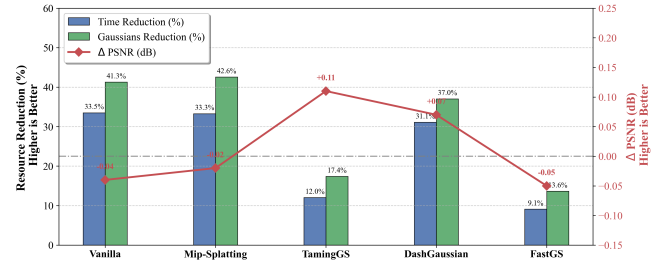
F. Yang is with the School of Telecommunication Engineering, Xidian University, Xi'an 710071, China (e-mail: fzhyang@mail.xidian.edu.cn).

J. Xia is with the School of Computer Science and Engineering, Central South University, China (email: xiajiazhi@csu.edu.cn)

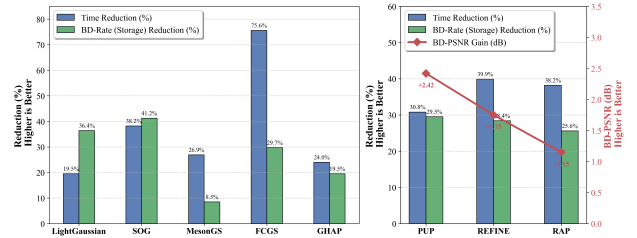
W. Lin is with the Department of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China (e-mail: wylin@sjtu.edu.cn).

J. Hou is with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China (e-mail: jh.hou@cityu.edu.hk).

This work was supported in part by the TCL Science and Technology Innovation Fund, the NSFC under Grants 62371358 and 62422118, and the Hong Kong Research Grants Council under Grants 11220426, 11219324, and N\_CityU1114/25.



(a) Efficiency and Quality Gains during 3DGS Training



(b) Benefits on Post-processing Compression & Pruning

Fig. 1: Performance gains of incorporating our SPARE-GS into diverse off-the-shelf pipelines under various tasks. (a) Training: across diverse 3DGS pipelines, SPARE-GS reduces training time and Gaussian counts without sacrificing rendering quality. (b) Post-Processing: this compact representation benefits downstream tasks, delivering substantial storage and latency savings for various compression and pruning frameworks.

3D Gaussian primitives and utilizing a differentiable rasterization pipeline, 3DGS has become a widely adopted method for 3D scene representation [2]–[5]. However, this explicit representation introduces considerable resource overhead [6]–[8]. High-fidelity reconstructions typically require massive Gaussian primitives to capture complex geometries and fine view-dependent appearances [9], [10]. This dense primitive count increases model size, posing challenges for mobile deployment and large-scale scene modeling, while raising memory consumption and training time [11]–[14]. Furthermore, the conventional fixed-iteration training schedule often continues optimization even after the scene structure and reconstruction loss have converged. Consequently, advancing the practicality of 3DGS becomes a problem of comprehensive cost control across training time, computational resources, memory footprint, and storage capacity, a challenge that has driven recent efforts to explore various optimization strategies [15], [16].

Existing methods have attempted to improve 3DGS efficiency from two aspects: optimization of structural evolution and representation compression. Structural evolution optimized methods improve 3DGS efficiency by reducing optimization cost, refining density control, or redesigning the underlying representation. Existing approaches employ strategies such as resolution scheduling, sparse computation, iteration reduction, multi-view-consistent densification, targeted pruning, and modified rendering formulations [17]–[22]. Although these approaches improve different aspects of training, most structural decisions still rely on local heuristics and lack explicit spatial budget coordination.

Representation compression primarily targets storage and transmission costs. It can be broadly divided into training-integrated and post-training approaches. Training-integrated methods, such as HAC [23], jointly optimize compact representations, masking or quantization mechanisms, and entropy models during scene reconstruction. Post-training methods instead operate on a pretrained Gaussian representation, including compression frameworks that reorganize, transform, quantize, or entropy-code Gaussian attributes [24]–[27], as well as post-hoc pruning methods that remove low-importance primitives and optionally refine the remaining representation [28]–[30]. These two categories are closely related to the structural organization of the Gaussian representation. Post-training compression is strongly influenced by the redundancy and spatial organization inherited from the pretrained Gaussian representation.

Overall, the organization of Gaussian primitives directly affects the computational cost of optimization and the amount of redundancy inherited by downstream compression and pruning. However, this structure is typically governed by local density-control decisions during training or treated as a fixed input afterward, rather than being explicitly managed as a finite global resource. This motivates a fundamental question: *how can the structural budget of 3DGS be globally regulated to improve training efficiency and representational compactness?*

We argue that achieving structural parsimony in 3DGS requires formulating structural optimization through the lens of budget management, rather than relying on unconstrained primitive-level decisions. Each Gaussian primitive represents a unit of computational and memory cost, while different spatial regions exhibit varying representational demands. Regions with complex geometry, high-frequency textures, or strong multi-view visibility naturally justify higher budget utilization; conversely, regions that already contain sufficient primitives offer diminishing quality gains from further densification. Consequently, conventional gradient-threshold-based densification fails to enforce global budget regulation, which inevitably leads to redundant primitive proliferation in some areas and structural under-representation in others, ultimately inflating the overall cost.

Based on this perspective, we formulate the structural evolution of 3DGS as a global budget-constrained optimization problem and introduce SPARE-GS, a *plug-and-play* framework for structural regulation. SPARE-GS estimates regional representational demand from aggregated multi-view statistics, converts it into capacity-aware target quotas, and uses regional

budget deviations to coordinate densification and pruning. Photometric and structural stability further provide a principled signal for adaptive termination.

Extensive experiments across diverse 3DGS training pipelines demonstrate substantial reductions in training time and Gaussian count, while improving rendering fidelity. The resulting compact representations also improve the efficiency and rate-distortion performance of downstream compression and pruning methods, as shown in Fig. 1.

The main contributions are summarized as follows.

- We formulate the structural evolution of 3DGS as a global budget-constrained optimization problem and derive a optimality condition, which requires the marginal utility of structural resources to be balanced across spatial regions under a finite primitive budget.
- Guided by this condition, we propose SPARE-GS, which estimates regional marginal demand, constructs capacity-aware target quotas, and uses regional budget deviations to steer densification, pruning, and adaptive termination toward a more balanced structural allocation.
- We validate SPARE-GS across diverse 3DGS training pipelines and downstream tasks, demonstrating substantial reductions in training time and Gaussian count while preserving rendering fidelity and improving the efficiency of subsequent post-processing.

The rest of this paper is organized as follows. Section II reviews previous work in 3DGS. Section III outlines the formulation of the problem and the fundamental principles behind SPARE-GS. The details of SPARE-GS are described in Section IV. Section V introduces extensive experimental results. Finally, Section VI concludes this paper.

## II. RELATED WORK

### A. Optimization of 3DGS

Existing methods improve 3DGS efficiency by jointly reducing optimization cost and refining the evolution of Gaussian representations. Speedy-Splat [18] exploits sparse pixels and primitives to avoid unnecessary computation, while Dash-Gaussian [17] coordinates rendering resolution and primitive growth throughout optimization. FastGS [19] introduces multi-view-consistent densification and targeted pruning to suppress irrelevant primitives and reduce training cost. Other studies focus more directly on density control. Revising Densification [20] introduces a pixel-error-driven criterion, Pixel-GS [21] incorporates pixel coverage into multi-view gradient aggregation, and AbsGS [31] alleviates gradient collisions using homodirectional view-space gradients. GaussianPro [22] progressively propagates Gaussians using multi-view stereo priors, whereas TamingGS [32] performs constructive densification under a prescribed global Gaussian budget. MCMC-based 3DGS [33] further reformulates Gaussian evolution as a sampling process through transitions and relocalization.

Several methods improve training behavior by redesigning the representation or rendering formulation [34]. Mip-Splatting [35] introduces 3D smoothing and 2D Mip filtering to reduce aliasing under varying sampling rates. Scaffold-GS [36] organizes local Gaussians around neural anchors

and predicts view-dependent attributes on demand, while 2DGS [37] replaces volumetric Gaussians with oriented Gaussian disks to improve geometric consistency. Although these methods improve training efficiency or rendering quality, explicit coordination of a finite global budget across spatial regions according to estimated demand remains insufficiently explored.

### B. Compression of 3D Gaussian Representations

Representation compression targets the storage, transmission, and deployment costs of 3DGS. It can be broadly divided into training-integrated and post-training approaches. Training-integrated methods jointly optimize the scene representation and its coding model. Scaffold-GS [36] provides a structured anchor-based representation, upon which HAC [23] models spatial context through a hash grid for entropy coding. HAC++ [38] extends this formulation with inter- and intra-anchor context modeling, adaptive quantization, and masking. Related approaches employ structured parameterizations, hierarchical spatial organization, or compact basis functions to reduce representation redundancy [14], [39].

Post-training compression operates on an already optimized Gaussian representation. Attribute-oriented frameworks reduce storage by transforming, quantizing, or entropy-coding Gaussian parameters. Compact-GS [40] and CompGS [41] compress Gaussian attributes through compact parameterization and quantization, while SOG [24] maps Gaussian attributes onto structured grids for video-based coding. Light-Gaussian [25] combines importance-based pruning, recovery, spherical-harmonic distillation, and vector quantization. MesonGS [26], FCGS [15], and GHAP [27] further improve rate-distortion performance through attribute transformation, structured coding, or geometry-aware processing. A complementary group of post-hoc pruning methods focuses on reducing the number of primitives before or alongside subsequent coding. PUP [28] estimates the reconstruction sensitivity of Gaussian spatial parameters and applies iterative prune-refine operations. REFINE [29] employs an efficient rendering-aware importance approximation, while RAP [30] removes redundant primitives according to its pruning criterion. Other methods use opacity, scale, gradients, or rendering contribution to rank and eliminate low-importance Gaussians [42]–[44]. The effectiveness of post-training compression and pruning is influenced by the redundancy and spatial organization inherited from the pretrained representation.

Overall, the Gaussian structure affects the computational cost during optimization and determines the redundancy presented to downstream compression and pruning. However, this structure is commonly shaped by local density-control decisions or treated as a fixed input after training, rather than being explicitly managed as a finite global resource.

## III. PROBLEM FORMULATION

This section establishes the theoretical foundation of SPARE-GS by framing 3DGS structural evolution as a constrained optimization problem. We first formulate the densification and pruning process as a budget-constrained mixed-integer nonlinear programming (MINLP) task, uncovering the

limitations of conventional heuristic-based training (Section III-A). To address these inefficiencies, we relax this formulation into a regional budget optimization framework governed by the principle of diminishing marginal utility (Section III-B). By deriving the optimal structural state through Karush-Kuhn-Tucker (KKT) conditions [45], we prove that structural parsimony is achieved when the marginal rendering utility is balanced across different spatial regions (Section III-C). These insights provide the guiding principle for the practical resource-regulation mechanism detailed in Section IV.

### A. Rethinking of 3DGS Training

A 3D scene in 3DGS is represented by a set of  $N$  Gaussian primitives  $\mathcal{G} = \{G_i = (\mu_i, s_i, q_i, c_i, \alpha_i)\}_{i=1}^N$ . Each primitive is defined by its center  $\mu_i \in \mathbb{R}^3$ , spatial scaling  $s_i \in \mathbb{R}^3$ , rotation  $q_i \in \mathbb{R}^4$ , color spherical harmonics coefficients  $c_i \in \mathbb{R}^{3 \times 16}$ , and opacity  $\alpha_i \in \mathbb{R}$  [1]. During rendering, these primitives are projected and depth-sorted to perform alpha-blending along each pixel ray  $p$ , computing the final color  $C(p)$  via the accumulated transmittance  $T_i(p)$  [46], formulated as

$$C(p) = \sum_{i \in \mathcal{N}} \tilde{c}_i \tilde{\alpha}_i(p) T_i(p), \quad T_i(p) = \prod_{j=1}^{i-1} (1 - \tilde{\alpha}_j(p)), \quad (1)$$

where  $\mathcal{N}$  is the depth-sorted set of primitives overlapping pixel  $p$ ,  $\tilde{c}_i$  is the color computed via spherical harmonics, and  $\tilde{\alpha}_i(p)$  is the Gaussian footprint evaluated in pixel  $p$  multiplied by the standalone opacity  $\alpha_i$ .

During training, 3DGS optimizes these parameters by minimizing the photometric loss between the synthesized and ground-truth images over  $\mathcal{V}$  viewpoints:

$$\mathcal{L}(\mathcal{G}) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} L(\mathcal{R}_v(\mathcal{G}), \mathcal{I}_v), \quad (2)$$

where  $\mathcal{R}_v(\cdot)$  denotes differentiable Gaussian rasterization at view  $v$ ,  $\mathcal{I}_v$  is the corresponding ground-truth image, and  $L(\cdot)$  denotes the photometric loss.

Minimizing Eq. (2) is not a standard continuous optimization problem; it couples parameter tuning with discrete structural operations, such as primitive cloning, splitting, and pruning. To describe this structural decision process, we consider a structural update at training step  $t$ . Let  $\mathcal{A}^{(t)}$  denote the set of Gaussian primitives that are active before the update, and  $\mathcal{C}^{(t)}$  the set of new candidate primitives proposed by densification. The complete set of admissible primitives for the structural decision is formulated as:

$$\bar{\mathcal{G}}^{(t)} = \mathcal{A}^{(t)} \cup \mathcal{C}^{(t)} = \left\{ G_i^{(t)} \right\}_{i=1}^{\bar{N}^{(t)}}, \quad (3)$$

where  $\bar{N}^{(t)}$  is the total number of admissible primitives.

For each primitive  $G_i^{(t)} \in \bar{\mathcal{G}}^{(t)}$ , we introduce a binary structural variable  $\sigma_i^{(t)} \in \{0, 1\}$ . If  $\sigma_i^{(t)} = 1$ , the primitive is selected to remain active after the update, participating in subsequent rendering and optimization. Conversely, if  $\sigma_i^{(t)} = 0$ , it is deactivated and excluded from training. The updated active Gaussian set is calculated by:

$$\tilde{\mathcal{G}}^{(t)}(\sigma^{(t)}) = \left\{ G_i^{(t)} \in \bar{\mathcal{G}}^{(t)} \mid \sigma_i^{(t)} = 1 \right\}, \quad (4)$$

where  $\sigma^{(t)} = \{\sigma_i^{(t)}\}_{i=1}^{\bar{N}^{(t)}}$  denotes the structural activation states. The assignment of  $\sigma^{(t)}$  governs the entire structural evolution: it determines whether existing primitives in  $\mathcal{A}^{(t)}$  are retained or pruned, and whether new candidates in  $\mathcal{C}^{(t)}$  are accepted or discarded. Consequently, densification and pruning are components of a unified selection process over the admissible set  $\bar{\mathcal{G}}^{(t)}$ .

By incorporating structural decisions into the photometric optimization, we reformulate 3DGS training as a resource-constrained MINLP task:

$$\begin{aligned} \min_{\bar{\mathcal{G}}^{(t)}, \sigma^{(t)}} \quad & \mathcal{L}(\bar{\mathcal{G}}^{(t)}(\sigma^{(t)})), \\ \text{s.t.} \quad & \sum_{i=1}^{\bar{N}^{(t)}} \sigma_i^{(t)} \leq B^{(t)}, \quad \sigma_i^{(t)} \in \{0, 1\}, \end{aligned} \quad (5)$$

where  $B^{(t)}$  is the maximum number of active primitives allowed in step  $t$  in the scene, and  $\sum_{i=1}^{\bar{N}^{(t)}} \sigma_i^{(t)} \leq B^{(t)}$  limits the structural complexity of the scene representation, which dictates memory consumption, rendering time, training overhead, or storage size.

Compared with Eq. (2), Eq. (5) models Gaussian primitives as finite resources bounded by  $B^{(t)}$ . By omitting this global constraint  $\sum_{i=1}^{\bar{N}^{(t)}} \sigma_i^{(t)} \leq B^{(t)}$ , the original 3DGS pipeline relies on isolated local rules, such as fixed gradient thresholds. Because these unconstrained heuristics operate without global awareness, they fail to balance structural growth across the scene, inevitably causing severe redundancy in high-frequency areas and under-representation elsewhere. Therefore, effective 3DGS optimization goes beyond merely reducing local loss; **it involves regulating structural evolution under a finite budget to achieve high representational efficiency.**

### B. Modeling via Marginal Utility

While Eq. (5) provides a formal objective for structural evaluation, solving this MINLP problem is computationally intractable during training. Therefore, we introduce a spatial relaxation: instead of formulating binary decisions for millions of primitives, we aggregate them into local spatial regions and optimize the continuous representational budget governed within each region.

Specifically, we subdivide the 3D scene into  $K$  disjoint spatial regions  $\{\Omega_r\}_{r=1}^K$ . Let  $n_r^{(t)} \geq 0$  represent the number of active primitives budgeted for region  $\Omega_r$  at training step  $t$ , and we approximate the overall optimization objective as the sum of independent regional contributions. Let  $U_r(n_r^{(t)})$  denote the expected local loss reduction (i.e., utility) achieved by dedicating a structural budget of  $n_r^{(t)}$  primitives to region  $\Omega_r$ . We simply Eq. (5) into a region-level budget optimization problem:

$$\begin{aligned} \max_{\{n_r^{(t)}\}_{r=1}^K} \quad & \sum_{r=1}^K U_r(n_r^{(t)}), \\ \text{s.t.} \quad & \sum_{r=1}^K n_r^{(t)} \leq B^{(t)}, \quad n_r^{(t)} \geq 0. \end{aligned} \quad (6)$$

During training, as more Gaussian primitives are added to  $\Omega_r$ , the ability to capture high-frequency details eventually saturates. As a result, the additional loss reduction provided by each added primitive decreases. This property corresponds to the law of diminishing marginal utility [47], expressed as

$$\frac{\partial U_r(n_r^{(t)})}{\partial n_r^{(t)}} \geq 0, \quad \frac{\partial^2 U_r(n_r^{(t)})}{\partial (n_r^{(t)})^2} \leq 0. \quad (7)$$

Given this diminishing nature, strategically distributing the finite structural budget becomes essential for achieving high representational efficiency. This objective motivates our formal derivation of the optimal representational state in the subsequent section.

### C. Optimal Regulation by KKT Conditions

To determine the optimal regional budget configuration  $(\mathbf{n}^*)^{(t)} = \{(n_1^*)^{(t)}, \dots, (n_K^*)^{(t)}\}$  at training step  $t$ , we construct the associated Lagrangian function:

$$\begin{aligned} \mathcal{J}(\mathbf{n}^{(t)}, \lambda^{(t)}, \boldsymbol{\nu}^{(t)}) \\ = \sum_{r=1}^K U_r(n_r^{(t)}) + \lambda^{(t)}(B^{(t)} - \sum_{r=1}^K n_r^{(t)}) + \sum_{r=1}^K \nu_r^{(t)} n_r^{(t)}, \end{aligned} \quad (8)$$

where  $\lambda^{(t)}$  denotes the Lagrange multiplier, and  $\boldsymbol{\nu}^{(t)} = \{\nu_r^{(t)}\}_{r=1}^K$  are the multipliers of the regional budgets. According to nonlinear programming theory [48], if  $(\mathbf{n}^*)^{(t)}$  represents the optimal budget configuration, it satisfies the KKT conditions:

- Stationarity:

$$\frac{\partial U_r((n_r^*)^{(t)})}{\partial n_r^{(t)}} - \lambda^{(t)} + \nu_r^{(t)} = 0, \quad (9)$$

- Primal Feasibility:

$$\sum_{r=1}^K (n_r^*)^{(t)} \leq B^{(t)}, \quad (n_r^*)^{(t)} \geq 0, \quad (10)$$

- Complementary Slackness:

$$\lambda^{(t)} \left( B^{(t)} - \sum_{r=1}^K (n_r^*)^{(t)} \right) = 0, \quad \nu_r^{(t)} (n_r^*)^{(t)} = 0. \quad (11)$$

During training, the budget constraint is binding, meaning  $\lambda^{(t)} > 0$  and  $\sum_{r=1}^K (n_r^*)^{(t)} = B^{(t)}$ . Furthermore, for any spatial region  $\Omega_r$  that actively participates in representing the scene, its optimal budget is strictly positive ( $(n_r^*)^{(t)} > 0$ ). According to the complementary slackness condition, this implies  $\nu_r^{(t)} = 0$ . Consequently, we derive the optimality criterion:

$$\frac{\partial U_r((n_r^*)^{(t)})}{\partial n_r^{(t)}} = \lambda^{(t)}, \quad \forall r \text{ where } (n_r^*)^{(t)} > 0. \quad (12)$$

Eq. (12) establishes that in the optimal state, the marginal utility yielded by a unit increase in the structural budget is uniformly equalized across all active regions, converging to the constant  $\lambda^{(t)}$ . The KKT condition provides a target equilibrium criterion for regional resource allocation. During the intermediate training phase at step  $t$ , the current structural



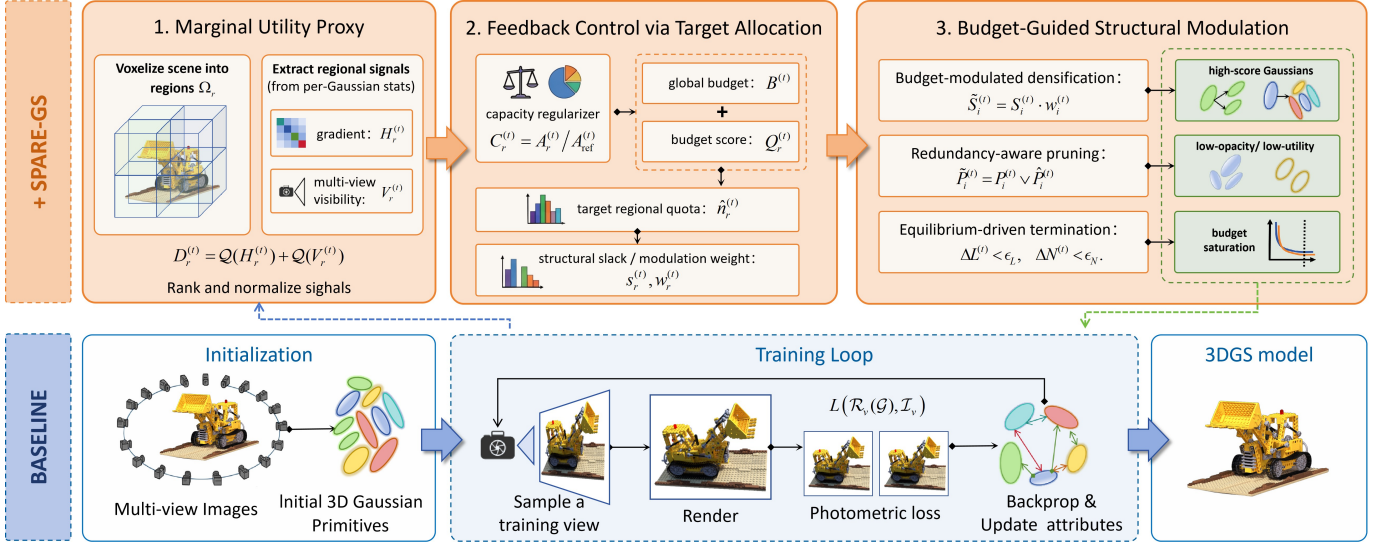


Fig. 2: **Overview of the proposed SPARE-GS framework.** To achieve global budgeted-constrained optimization, our method first establishes a *Marginal Utility Proxy* by voxelizing the scene and aggregating regional statistics. Then, we utilize *Feedback Control via Target Allocation* to dynamically compute capacity-regularized regional quotas. Finally, we apply the *Budget-Guided Structural Modulation* mechanism to guide densification, redundancy pruning, and equilibrium-driven termination. In the diagram, the standard 3DGS baseline pipeline (blue) is controlled by our proposed modules (orange), and the specific structural evaluation is guided by our mechanism (green).

allocation  $n_r^{(t)}$  is typically unbalanced. Their marginal utilities initially exhibit a skewed distribution:

$$\frac{\partial U_1(n_1^{(t)})}{\partial n_1^{(t)}} \neq \frac{\partial U_2(n_2^{(t)})}{\partial n_2^{(t)}} \neq \dots \neq \frac{\partial U_K(n_K^{(t)})}{\partial n_K^{(t)}}. \quad (13)$$

While unconstrained heuristics may inadvertently perpetuate this imbalance, the KKT optimality criterion dictates that progressively eliminating this gap is essential for an efficient structural evolution.

Therefore, although analytically computing the exact  $\lambda^{(t)}$  is intractable, this optimality condition provides a directional guide. The proposed method in Sec. IV leverages this principle as an online feedback mechanism. By measuring the relative structural demands, we modulate the densification and pruning operations to actively steer the marginal utilities across regions toward the KKT equilibrium during training:

$$\frac{\partial U_1(n_1^{(t+\delta t)})}{\partial n_1^{(t+\delta t)}} \approx \dots \approx \frac{\partial U_K(n_K^{(t+\delta t)})}{\partial n_K^{(t+\delta t)}} \rightarrow \lambda^{(t+\delta t)}, \quad (14)$$

where  $\delta t > 0$  represents the subsequent training steps. This targeted intervention smoothly steers the system toward the KKT equilibrium without requiring exact analytical solutions.

#### IV. PROPOSED METHOD

Based on the analysis in Sec. III, an efficient structural state is achieved by balancing marginal utility across different regions. To operationalize this, we first detail how to approximate the regional marginal utility by analyzing demand in Sec. IV-A. Then, we formulate the capacity-aware target budget in Sec. IV-B, and finally apply these targets to modulate the structural heuristics in Sec. IV-C. Fig. 2 illustrates the overall flowchart of our method.

##### A. Marginal Utility Proxy

To allocate resources effectively, we first identify which spatial regions lack structural details. We formulate a proxy to quantify marginal utility using native training statistics. Generally, the marginal utility of a spatial region reflects its local representational demand, as indicated by Eq. (7). A region exhibiting higher unfulfilled demand implies a greater potential for error reduction, thereby translating to a higher marginal utility when additional primitives are allocated. Therefore, we establish a proxy for the regional marginal utility using observable demand statistics natively tracked during the training process.

To evaluate these statistics, we uniformly partition the bounding box of active Gaussians into a  $32 \times 32 \times 32$  voxel grid with a small padding. Each voxel defines a local region  $\Omega_r$  containing  $n_r^{(t)}$  primitives. The regional demand is then computed exclusively across the  $K$  non-empty regions.

For each valid region  $\Omega_r$ , the proxy for marginal utility is formulated as a composite of training statistics:

$$\frac{\partial U_r(n_r^{(t)})}{\partial n_r^{(t)}} \approx D_r^{(t)} = Q(H_r^{(t)}) + Q(V_r^{(t)}), \quad (15)$$

where  $H_r^{(t)}$  denotes the regional gradient magnitude, indicating unresolved errors, and  $V_r^{(t)}$  captures multi-view visibility, indicating global influence of  $\Omega_r$ .  $Q(\cdot)$  is a rank-normalization function to map these extracted statistics into the interval  $[0, 1]$ , which is defined as  $Q(m_r^{(t)}) = \frac{\text{rank}(m_r^{(t)})}{K}$  with  $\text{rank}(m_r^{(t)})$  computing the ascending rank of an input metric  $m_r^{(t)}$  among all  $K$  valid regions. In the following, we detail the explicit derivation of each statistical component.

**Gradient Magnitude.** A large gradient indicates that the current primitives fail to accurately fit the local image, revealing a strong demand for additional structural details. To quantify this, we track the accumulated view-space positional gradient  $a_i^{(t)}$  and accumulated visibility count  $d_i^{(t)}$  during differentiable rasterization. Since large primitives accumulate higher gradients due to their broad screen coverage, we compute a footprint-normalized gradient to avoid size-induced bias. Averaging these normalized scores yields the regional gradient magnitude  $H_r^{(t)}$ , which evaluates the local convergence state of region  $\Omega_r$ :

$$H_r^{(t)} = \frac{1}{n_r^{(t)}} \sum_{i=1}^{N^{(t)}} \frac{a_i^{(t)}}{d_i^{(t)}} \mathbb{I}[k_i^{(t)} = r], \quad (16)$$

where  $N^{(t)}$  denotes the total number of active primitives at optimization step  $t$ ,  $k_i^{(t)}$  represents the assigned voxel index of the  $i$ -th primitive based on its spatial coordinate, and  $\mathbb{I}[\cdot]$  is the indicator function yielding 1 if the primitive resides in region  $\Omega_r$  (i.e.,  $k_i^{(t)} = r$ ) and 0 otherwise.

**Multi-view Visibility.** High multi-view visibility implies that a region strongly influences the rendering across multiple view-points, generally requiring greater structural detail to capture its complex appearance. Using the accumulated visibility count  $d_i^{(t)}$  tracked during the forward pass for visibility, the regional multi-view visibility  $V_r^{(t)}$  is computed as the regional average:

$$V_r^{(t)} = \frac{1}{n_r^{(t)}} \sum_{i=1}^{N^{(t)}} d_i^{(t)} \mathbb{I}[k_i^{(t)} = r]. \quad (17)$$

### B. Feedback Control via Target Allocation

**Target Quota Allocation.** To regulate structural evolution globally, we dynamically scale the global budget  $B^{(t)}$  according to the current number of active primitives  $N^{(t)}$ , without disrupting the native optimization trajectory of the base 3DGS pipeline.

Additionally, the representational capacity of Gaussians varies across regions: for the same photometric demand, regions containing smaller primitives generally require a larger numerical quota. We therefore introduce a geometric capacity regularizer  $C_r^{(t)}$  to calibrate the regional demand. For region  $\Omega_r$ , its average projected footprint is defined as

$$A_r^{(t)} = \frac{1}{n_r^{(t)}} \sum_{i=1}^{N^{(t)}} \left( \rho_i^{(t)} \right)^2 \mathbb{I}[k_i^{(t)} = r], \quad (18)$$

where  $\rho_i^{(t)}$  denotes the maximum projected 2D radius of primitive  $G_i^{(t)}$ . We then calculate the geometric capacity regularizer as  $C_r^{(t)} = A_r^{(t)} / A_{\text{ref}}^{(t)}$ , where  $A_{\text{ref}}^{(t)}$  denotes the median projected area across all active regions at step  $t$ . Accordingly, the capacity-adjusted budget score for region  $\Omega_r$  is formulated as  $Q_r^{(t)} = D_r^{(t)} / C_r^{(t)}$ .

According to our KKT optimality analysis in Eq. (12), structural resources should be routed to regions with the high unfulfilled demand. Thus, we distribute the global budget  $B^{(t)}$

based on the capacity-adjusted demand  $Q_r^{(t)}$ . By normalizing  $Q_r^{(t)}$ , we calculate the target distribution ratio  $l_r^{(t)}$ :

$$l_r^{(t)} = \frac{Q_r^{(t)}}{\sum_{j=1}^K Q_j^{(t)}}. \quad (19)$$

Consequently, the target primitive quota for region  $\Omega_r$  is determined by:

$$\hat{n}_r^{(t)} = B^{(t)} l_r^{(t)}. \quad (20)$$

This formulation provides a global reference allocation that guides, rather than strictly enforces, structural growth and effective direction for resource allocation in 3DGS training.

**Feedback-driven Modulation.** With the target quota  $\hat{n}_r^{(t)}$  established in Eq. (20), we formulate a feedback control signal to modulate the structural evolution of 3DGS. Specifically, we quantify the relative structural deviation, for the region residing in the  $i$ -th primitive:

$$s_{k_i^{(t)}}^{(t)} = \frac{\hat{n}_{k_i^{(t)}}^{(t)} - n_{k_i^{(t)}}^{(t)}}{\hat{n}_{k_i^{(t)}}^{(t)} + 1}. \quad (21)$$

A positive slack ( $s_{k_i^{(t)}}^{(t)} > 0$ ) indicates that the region is currently under-budgeted, while a negative slack indicates that the current regional occupancy exceeds its target quota.

To translate the structural deviation into a stable control signal without imposing aggressive discontinuities, the final budget-aware modulation weight  $w_i^{(t)}$  assigned to the  $i$ -th primitive is formulated by coupling the regional slack with its representational demand:

$$w_i^{(t)} = \Phi \left( s_{k_i^{(t)}}^{(t)}, \tilde{l}_r^{(t)} \right), \quad (22)$$

where  $\Phi(\cdot)$  is a bounded sigmoid function that maps the structural deviation into a smooth gating signal, and  $\tilde{l}_r^{(t)}$  denotes the target distribution ratio  $l_r^{(t)}$  normalized against the maximum value across all valid regions.

### C. Budget-guided Structural Modulation

Finally, we apply the derived feedback weights in Eq. (22) to control the actual 3DGS optimization. They softly modulate densification, guide redundancy-aware pruning, and enable adaptive termination when equilibrium is reached.

**Budget-modulated Densification.** We incorporate the budget constraint by modulating the base densification score  $S_i^{(t)}$  with the effective control weight  $w_i^{(t)}$ . Specifically,  $S_i^{(t)}$  denotes the native densification priority assigned to Gaussian or candidate  $i$  by the underlying pipeline at iteration  $t$ , based on its original method-specific criterion. The resulting budget-aware score is computed as:

$$\tilde{S}_i^{(t)} = S_i^{(t)} \cdot w_i^{(t)}. \quad (23)$$

In implementation, the native densification threshold is kept unchanged, and this effective weight softly modulates the primitive-level gradient or importance score. Effectively, this modulated score accelerates densification in under-budgeted regions ( $w_i^{(t)} > 1$ ) and suppresses it in over-budgeted ones

( $w_i^{(t)} < 1$ ), continuously steering the structural growth toward the target allocation.

**Redundancy-aware Pruning.** To prevent the progressive accumulation of structural redundancy in over-budgeted and low-response regions, we formulate a conservative pruning constraint  $\hat{P}_i^{(t)}$  for each primitive  $G_i^{(t)}$ :

$$\hat{P}_i^{(t)} = \mathbb{I}[n_{k_i^{(t)}}^{(t)} > \hat{n}_{k_i^{(t)}}^{(t)}] \cdot \mathbb{I}[\alpha_i^{(t)} < \tau_\alpha] \cdot \mathbb{I}[\frac{a_i^{(t)}}{d_i^{(t)}} < \frac{\tau_{\text{org}}}{2}], \quad (24)$$

where  $\tau_\alpha = 0.05$  and  $\tau_{\text{org}}$  is the base gradient threshold. Specifically, a primitive is flagged for removal only if it resides in an over-budgeted region, has low opacity, and exhibits negligible gradient magnitude. We integrate this budget-aware constraint with the base pruning mask  $P_i^{(t)}$ :

$$\tilde{P}_i^{(t)} = P_i^{(t)} \vee \hat{P}_i^{(t)}. \quad (25)$$

To avoid removing low-opacity primitives immediately after an opacity reset, we disable this branch for a cooldown window equal to the standard pruning interval. In addition, newly generated primitives are protected from budget-aware pruning within the same structural update.

**Equilibrium-driven Termination.** As the optimization approaches a stable allocation state motivated by the KKT-inspired condition, continuing the fixed-iteration training schedule may yield negligible rendering improvements relative to the incurred computational cost. To detect such convergence, we monitor the exponential moving average of the photometric loss  $\bar{L}^{(t)}$  and the primitive count  $N^{(t)}$  at fixed intervals  $\Delta t$ . Their relative temporal variations are computed as

$$\begin{cases} \Delta L^{(t)} = 1 - \frac{\bar{L}^{(t)}}{\bar{L}^{(t-\Delta t)}} \\ \Delta N^{(t)} = \frac{N^{(t)}}{N^{(t-\Delta t)}} - 1 \end{cases}. \quad (26)$$

If both the photometric loss and primitive count remain stable, i.e.,  $0 \leq \Delta L^{(t)} < \epsilon_L$  and  $|\Delta N^{(t)}| < \epsilon_N$ , for  $\ell$  consecutive checking intervals, the optimization is considered stable and is adaptively terminated. In our implementation, we set  $\epsilon_L = 0.008$ ,  $\epsilon_N = 0.01$ , and  $\ell = 2$ . The complete optimization pipeline is summarized in Algorithm 1.

## V. EXPERIMENTS

To evaluate the effectiveness and plug-and-play capability of SPARE-GS, we conduct extensive experiments covering training-stage efficiency, downstream compression, and post-training pruning. We further provide in-depth analyses of the key components and hyperparameters, computational overhead, structural evolution, and representative failure cases to better understand the robustness of the proposed framework.

### A. Experiment Settings

**Datasets.** Following the evaluation of the original 3D-GS [1], we tested our SPARE-GS on the same challenging real-world scenes. Specifically, we used all nine scenes from the Mip-NeRF 360 dataset [49], which contains five outdoor and four indoor scenes, each featuring complex central objects or viewing areas and detailed backgrounds. Additionally, two

---

### Algorithm 1 Optimization Process of SPARE-GS

---

**Require:** A base 3DGS pipeline, training views  $\mathcal{V}$ , checking interval  $\Delta t$ , and patience  $\ell$

- 1: Initialize base 3D Gaussian primitives
- 2: **for** optimization step  $t = 1, \dots, T$  **do**
- 3:   Render view  $v \in \mathcal{V}$ , update parameters, and accumulate stats  $a_i^{(t)}, d_i^{(t)}, \rho_i^{(t)}$
- 4:   **if** structural evolution (densification/pruning) is triggered **then**
- 5:     % Part 1: Marginal Utility Proxy (Sec. IV-A)
- 6:     Discretize spatial volume into regions  $\Omega_r$  and map primitives to region indices  $k_i^{(t)}$
- 7:     Compute regional signals  $H_r^{(t)}, V_r^{(t)}$  and demand proxy  $D_r^{(t)}$  {Eqs. (15)–(18)}
- 8:     % Part 2: Feedback Control via Target Allocation (Sec. IV-B)
- 9:     Distribute the global budget according to the capacity-adjusted demand to obtain  $\hat{n}_r^{(t)}$  {Eqs. (19)–(20)}
- 10:    Compute structural slack  $s_{k_i^{(t)}}^{(t)}$  and budget-aware modulation weight  $w_i^{(t)}$  {Eqs. (21), (22)}
- 11:    % Part 3: Budget-Guided Structural Modulation (Sec. IV-C)
- 12:    Modulate the native densification score to obtain  $\tilde{S}_i^{(t)}$  {Eq. (23)}
- 13:    Execute densification operations based on the modulated score  $\tilde{S}_i^{(t)}$
- 14:    Execute redundancy-aware pruning based on budget constraints {Eqs. (24), (25)}
- 15:    **end if**
- 16:    **if**  $t \bmod \Delta t = 0$  **then**
- 17:     % Part 3 (Cont.): Equilibrium-Driven Termination (Sec. IV-C)
- 18:     Evaluate convergence indicators  $\Delta L^{(t)}$  and  $\Delta N^{(t)}$  {Eq. (26)}
- 19:     **if** the loss does not worsen and both indicators remain stable for  $\ell$  consecutive intervals **then**
- 20:       **break** {Adaptive termination}
- 21:     **end if**
- 22:    **end if**
- 23: **end for**
- 24: **return** Optimized structurally-parsimonious representation

---

outdoor scenes, *truck* and *train*, were taken from the Tanks & Temples dataset [50], and two indoor scenes, *drjohnson* and *playroom*, were taken from the Deep Blending dataset [51]. Furthermore, we supplemented our evaluation with eight urban scenes from the BungeeNeRF dataset [52] to comprehensively assess our method. For consistency, we used the COLMAP camera pose estimates provided in the original 3D-GS [1].

**Implementation Details.** Our SPARE-GS is a plug-and-play, purely online structural regulation framework that can be integrated into the existing 3DGS training pipeline. To highlight

the effectiveness of the budget-guided mechanism itself, all integrated comparisons regarding structural efficiency were conducted under a *drop-in* condition. That is, rather than altering the specific densification and pruning strategies of the underlying methods, we solely applied our modulation to their native heuristics, with no pipeline-specific hyperparameter tuning.

**Evaluation Metrics.** To ensure a fair comparison, all baselines and our adapted versions were executed on the same hardware setup (Intel i9 CPU, RTX 3090 GPU). We evaluated performance using three standard image quality metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) [53], and Learned Perceptual Image Patch Similarity (LPIPS) [54]. Beyond these, we validated our resource-constrained formulation by tracking the total training time and the active Gaussian primitive count. For downstream tasks, we further reported the Bjøntegaard Delta (BD) metrics to measure rate-distortion efficiency, alongside coding times to quantify the associated computational overhead.

### B. Evaluation on 3DGS Training Efficiency

In this section, we evaluated the efficacy of SPARE-GS in optimizing the efficiency-fidelity trade-off and constraining the final primitive footprint. To substantiate its generalizability, we integrated our framework into five representative pipelines by augmenting their default density control mechanisms with our budget-guided structural modulation. The evaluated pipelines include the original/vanilla 3DGS [1], a quality-enhanced variant (Mip-Splatting [35]), and three acceleration-oriented frameworks (TamingGS [32], DashGaussian [17], and FastGS [19]). Table I reports the training time, final Gaussian count, and rendering fidelity in terms of PSNR, SSIM, and LPIPS. Furthermore, qualitative comparisons demonstrating these structural benefits are provided in Figs. 3 and 4.

**Performance on Vanilla 3DGS [1].** The results on the original 3DGS support our analysis regarding structural redundancy. Relying on local gradient heuristics, the standard pipeline tends to over-allocate primitives in high-frequency regions. By applying our budget-guided structural modulation and adaptive termination, we reduce the final Gaussian primitive count by 35.93% to 54.26% and decrease the total training time by 29.23% to 38.06%. This reduction in structural complexity is achieved while maintaining comparable rendering quality. On the Deep Blending and Tanks & Temples datasets, the PSNR shows slight improvements (+0.14 and +0.18 dB, respectively), while on Mip-NeRF 360 and BungeeNeRF, the decreases are marginal (-0.08 and -0.09 dB, respectively). These results suggest that many suppressed primitives provide limited additional contribution to reconstruction quality, which is consistent with our budget-constrained motivation.

**Performance on Mip-Splatting [35].** Mip-Splatting improves alias-free rendering by augmenting 3DGS with scale-aware filtering mechanisms. Table I demonstrates a similar trend when SPARE-GS is integrated into this pipeline. Across all four datasets, including the large-scale BungeeNeRF, our method substantially reduces the Gaussian count by 40.23% to 49.98%

and accelerates training by 30.34% to 39.11%. Notably, the quality response varies across datasets. PSNR improves by 0.32 dB and 0.30 dB on Deep Blending and Tanks & Temples, respectively, while decreasing by 0.02 dB on Mip-NeRF 360 and 0.19 dB on BungeeNeRF.

**Performance on TamingGS [32].** TamingGS aims to construct high-quality radiance fields under limited resources by introducing targeted structural allocation. As shown in Table I, despite this resource-aware baseline, integrating SPARE-GS further reduces the number of Gaussian primitives by 10.73% to 55.54% and decreases the training time by 7.91% to 38.94% across Mip-NeRF 360, Deep Blending, and Tanks & Temples. These resource reductions are achieved with only limited variations in rendering quality. The PSNR variation remains minimal, bounded within  $\pm 0.06$  dB on Mip-NeRF 360 and Tanks & Temples, while Deep Blending achieves a 0.14 dB improvement. Furthermore, on BungeeNeRF, the Gaussian count and training time remain virtually unchanged, with changes of -0.18% and +0.08%, respectively, while PSNR, SSIM, and LPIPS improve by 0.32 dB, 0.010, and 0.014, respectively. This result indicates that SPARE-GS does not enforce uniform compression; instead, it adapts the structural allocation to the demand of the underlying scene and pipeline.

**Performance on DashGaussian [17].** DashGaussian accelerates optimization through dynamic rendering resolutions and coordinated primitive growth. As shown in Table I, integrating our SPARE-GS further reduces the training time by 27.10% to 35.94% and decreases the final primitive count by 30.40% to 51.17%, while maintaining competitive reconstruction quality. Although DashGaussian controls the number of selected candidates at each densification step through global top-(k) ranking, it does not explicitly account for region-wise budget imbalance. SPARE-GS complements this mechanism by incorporating regional budget signals into the native ranking score, thereby increasing the priority of candidates located in under-budgeted regions without altering the original candidate-generation procedure.

**Performance on FastGS [19].** FastGS accelerates training by incorporating a multi-view consistency score, which functions as a structural regularizer and yields highly compact scene representations (e.g., producing  $\sim 558K$  primitives on Mip-NeRF 360). Table I demonstrates that our budget-guided framework successfully identifies and eliminates residual redundancies in this baseline. SPARE-GS further reduces the Gaussian count by 8.83% to 20.45% and the training time by 6.36% to 11.93%. Since FastGS already produces highly compact representations, the additional efficiency gains are comparatively smaller than those observed on other pipelines. The PSNR variations remain limited, ranging from -0.09 dB to -0.02 dB.

**Overall.** Across different pipeline configurations, SPARE-GS reduces the Gaussian count and training time by an average of 30.38% and 23.81%, respectively. These efficiency gains are accompanied by an average PSNR improvement of 0.01 dB and a negligible mean SSIM change, while LPIPS increases slightly by 0.005. The magnitude of the gain varies with



TABLE I: **Quantitative evaluation of our method in terms of 3DGS training efficiency across different base pipelines and datasets.**  $\Delta$  indicates the relative or absolute difference between the base method and our SPARE-GS. Bold values indicate improvements or resource reductions.  $\bar{\Delta}$  denotes the scene-count-weighted mean change.

| Base Pipeline      | Dataset              | Method            | Time (s) ↓     | Gaussians ↓    | PSNR ↑       | SSIM ↑        | LPIPS ↓       |
|--------------------|----------------------|-------------------|----------------|----------------|--------------|---------------|---------------|
| Vanilla 3DGS [1]   | Mip-NeRF 360 [49]    | Baseline          | 1327.08        | 2,640,494      | 27.52        | 0.813         | 0.221         |
|                    |                      | + SPARE-GS (Ours) | 939.23         | 1,691,678      | 27.44        | 0.809         | 0.234         |
|                    |                      | $\Delta$          | <b>-29.23%</b> | <b>-35.93%</b> | -0.08        | -0.004        | +0.013        |
|                    | Deep Blending [51]   | Baseline          | 1257.89        | 2,458,697      | 29.63        | 0.902         | 0.241         |
|                    |                      | + SPARE-GS (Ours) | 779.11         | 1,124,510      | 29.77        | 0.906         | 0.246         |
|                    |                      | $\Delta$          | <b>-38.06%</b> | <b>-54.26%</b> | <b>+0.14</b> | <b>+0.004</b> | +0.005        |
|                    | Tanks & Temples [50] | Baseline          | 718.15         | 1,572,798      | 23.83        | 0.850         | 0.171         |
|                    |                      | + SPARE-GS (Ours) | 485.66         | 973,619        | 24.01        | 0.849         | 0.181         |
|                    |                      | $\Delta$          | <b>-32.37%</b> | <b>-38.10%</b> | <b>+0.18</b> | -0.001        | +0.010        |
| Mip-Splatting [35] | Mip-NeRF 360 [49]    | Baseline          | 2364.79        | 6,506,221      | 27.83        | 0.915         | 0.098         |
|                    |                      | + SPARE-GS (Ours) | 1478.60        | 3,589,073      | 27.74        | 0.916         | 0.106         |
|                    |                      | $\Delta$          | <b>-37.47%</b> | <b>-44.84%</b> | -0.09        | <b>+0.001</b> | +0.008        |
|                    | Deep Blending [51]   | Baseline          | 2727.37        | 4,202,055      | 27.74        | 0.827         | 0.190         |
|                    |                      | + SPARE-GS (Ours) | 1899.76        | 2,511,604      | 27.72        | 0.826         | 0.201         |
|                    |                      | $\Delta$          | <b>-30.34%</b> | <b>-40.23%</b> | -0.02        | -0.001        | +0.011        |
|                    | Tanks & Temples [50] | Baseline          | 2246.00        | 3,503,352      | 29.43        | 0.903         | 0.239         |
|                    |                      | + SPARE-GS (Ours) | 1367.55        | 1,752,417      | 29.75        | 0.908         | 0.240         |
|                    |                      | $\Delta$          | <b>-39.11%</b> | <b>-49.98%</b> | <b>+0.32</b> | <b>+0.005</b> | +0.001        |
| TamingGS [32]      | Mip-NeRF 360 [49]    | Baseline          | 1415.70        | 2,358,386      | 23.80        | 0.860         | 0.157         |
|                    |                      | + SPARE-GS (Ours) | 946.15         | 1,371,594      | 24.10        | 0.860         | 0.165         |
|                    |                      | $\Delta$          | <b>-33.17%</b> | <b>-41.84%</b> | <b>+0.30</b> | 0.000         | +0.008        |
|                    | BungeeNeRF [52]      | Baseline          | 4015.86        | 7,675,877      | 28.25        | 0.924         | 0.088         |
|                    |                      | + SPARE-GS (Ours) | 2604.42        | 4,336,878      | 28.06        | 0.921         | 0.097         |
|                    |                      | $\Delta$          | <b>-35.15%</b> | <b>-43.50%</b> | -0.19        | -0.003        | +0.009        |
|                    | Deep Blending [51]   | Baseline          | 871.08         | 1,416,946      | 27.82        | 0.805         | 0.240         |
|                    |                      | + SPARE-GS (Ours) | 716.64         | 1,052,388      | 27.77        | 0.808         | 0.238         |
|                    |                      | $\Delta$          | <b>-17.73%</b> | <b>-25.73%</b> | -0.05        | <b>+0.003</b> | <b>-0.002</b> |
| DashGaussian [17]  | Mip-NeRF 360 [49]    | Baseline          | 597.56         | 1,178,605      | 29.97        | 0.907         | 0.246         |
|                    |                      | + SPARE-GS (Ours) | 550.27         | 1,052,175      | 30.11        | 0.911         | 0.243         |
|                    |                      | $\Delta$          | <b>-7.91%</b>  | <b>-10.73%</b> | <b>+0.14</b> | <b>+0.004</b> | <b>-0.003</b> |
|                    | Tanks & Temples [50] | Baseline          | 848.80         | 2,206,907      | 24.47        | 0.863         | 0.156         |
|                    |                      | + SPARE-GS (Ours) | 518.31         | 981,097        | 24.41        | 0.860         | 0.167         |
|                    |                      | $\Delta$          | <b>-38.94%</b> | <b>-55.54%</b> | -0.06        | -0.003        | +0.011        |
|                    | BungeeNeRF [52]      | Baseline          | 1209.50        | 2,426,465      | 27.39        | 0.903         | 0.127         |
|                    |                      | + SPARE-GS (Ours) | 1210.51        | 2,422,162      | 27.71        | 0.913         | 0.113         |
|                    |                      | $\Delta$          | +0.08%         | <b>-0.18%</b>  | <b>+0.32</b> | <b>+0.010</b> | <b>-0.014</b> |
| FastGS [19]        | Mip-NeRF 360 [49]    | Baseline          | 1102.76        | 2,532,764      | 27.73        | 0.815         | 0.220         |
|                    |                      | + SPARE-GS (Ours) | 803.87         | 1,762,803      | 27.65        | 0.814         | 0.227         |
|                    |                      | $\Delta$          | <b>-27.10%</b> | <b>-30.40%</b> | -0.08        | -0.001        | +0.007        |
|                    | Deep Blending [51]   | Baseline          | 858.09         | 2,313,205      | 29.60        | 0.906         | 0.237         |
|                    |                      | + SPARE-GS (Ours) | 579.23         | 1,129,622      | 29.78        | 0.907         | 0.241         |
|                    |                      | $\Delta$          | <b>-32.50%</b> | <b>-51.17%</b> | <b>+0.18</b> | <b>+0.001</b> | +0.004        |
|                    | Tanks & Temples [50] | Baseline          | 698.44         | 1,501,090      | 24.15        | 0.858         | 0.165         |
|                    |                      | + SPARE-GS (Ours) | 498.90         | 931,448        | 24.39        | 0.857         | 0.175         |
|                    |                      | $\Delta$          | <b>-28.57%</b> | <b>-37.95%</b> | <b>+0.24</b> | -0.001        | +0.010        |
| Average            | Mip-NeRF 360 [49]    | Baseline          | 2266.32        | 5,054,017      | 27.94        | 0.918         | 0.095         |
|                    |                      | + SPARE-GS (Ours) | 1451.74        | 2,994,782      | 28.10        | 0.924         | 0.095         |
|                    |                      | $\Delta$          | <b>-35.94%</b> | <b>-40.74%</b> | <b>+0.16</b> | <b>+0.006</b> | 0.000         |
|                    | Deep Blending [51]   | Baseline          | 269.08         | 558,202        | 27.29        | 0.800         | 0.254         |
|                    |                      | + SPARE-GS (Ours) | 248.83         | 508,898        | 27.26        | 0.797         | 0.259         |
|                    |                      | $\Delta$          | <b>-7.53%</b>  | <b>-8.83%</b>  | -0.03        | -0.003        | +0.005        |
|                    | Tanks & Temples [50] | Baseline          | 168.76         | 316,890        | 30.14        | 0.908         | 0.254         |
|                    |                      | + SPARE-GS (Ours) | 158.02         | 281,643        | 30.09        | 0.908         | 0.256         |
|                    |                      | $\Delta$          | <b>-6.36%</b>  | <b>-11.12%</b> | -0.05        | 0.000         | +0.002        |
| FastGS [19]        | Mip-NeRF 360 [49]    | Baseline          | 151.04         | 247,905        | 23.48        | 0.837         | 0.212         |
|                    |                      | + SPARE-GS (Ours) | 139.74         | 221,914        | 23.46        | 0.834         | 0.216         |
|                    |                      | $\Delta$          | <b>-7.48%</b>  | <b>-10.48%</b> | -0.02        | -0.003        | +0.004        |
|                    | BungeeNeRF [52]      | Baseline          | 453.17         | 1,173,619      | 27.30        | 0.903         | 0.140         |
|                    |                      | + SPARE-GS (Ours) | 399.10         | 933,641        | 27.21        | 0.895         | 0.148         |
|                    |                      | $\Delta$          | <b>-11.93%</b> | <b>-20.45%</b> | -0.09        | -0.008        | +0.008        |
|                    | Average              | $\bar{\Delta}$    | <b>-23.81%</b> | <b>-30.38%</b> | <b>+0.01</b> | 0.000         | +0.005        |



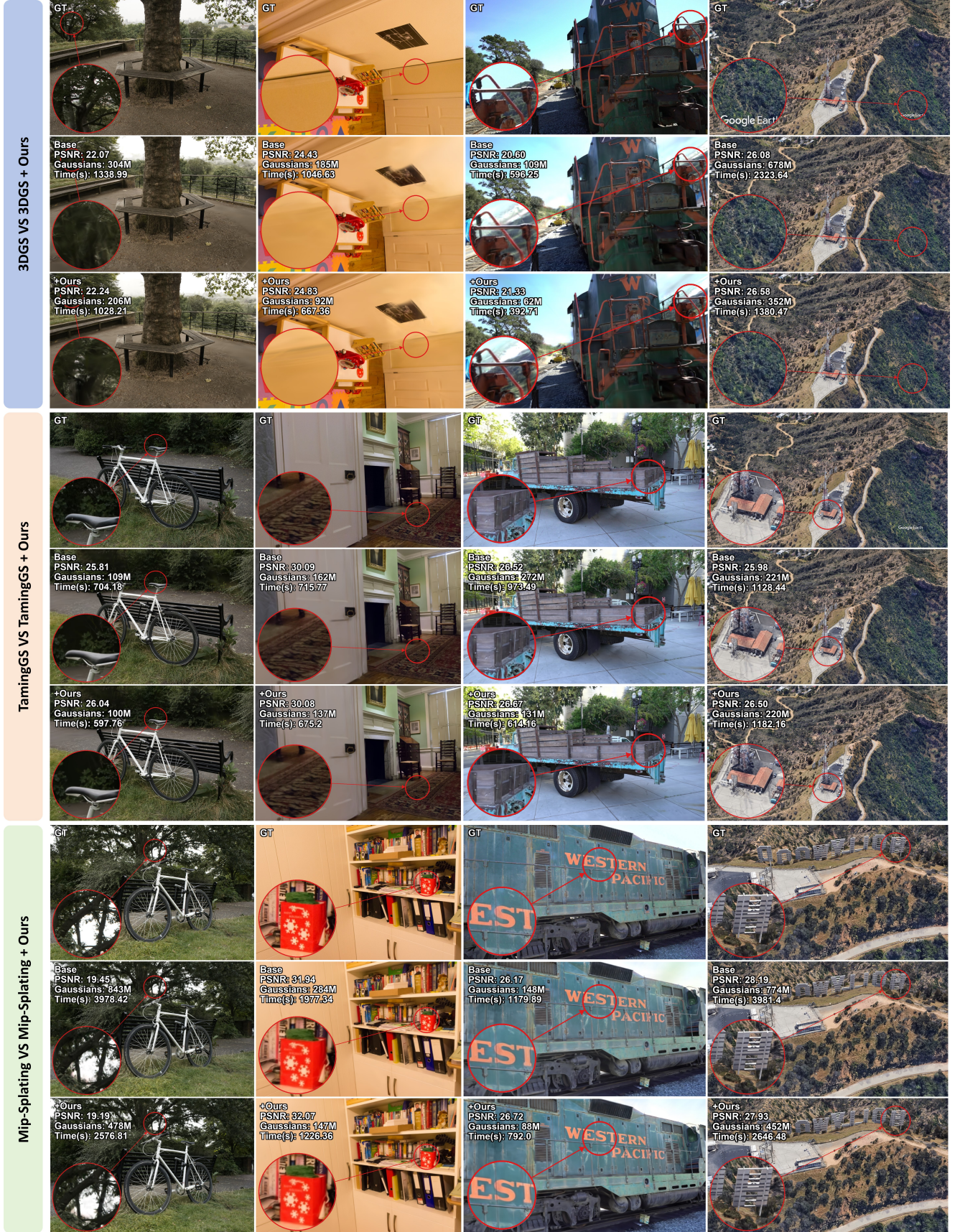


Fig. 3: **Comparison of rendering quality.** Rows (Top to Bottom): Grouped by base methods into 3DGS, TamingGS, and Mip-Splatting. Within each three-row block, we sequentially show the Ground Truth (GT), the baseline method (Base), and the baseline enhanced by our method (+ Ours). Columns (Left to Right): Representative scenes from Mip-NeRF 360, Deep Blending, Tanks & Temples, and BungeeNeRF. Please zoom in for details.





Fig. 4: **Comparison of rendering quality.** Rows (Top to Bottom): Grouped by base methods into DashGaussian and FastGS. Within each three-row block, we sequentially show the Ground Truth (GT), the baseline method (Base), and the baseline enhanced by our method (+ **Ours**). Columns (Left to Right): Representative scenes from Mip-NeRF 360, Deep Blending, Tanks & Temples, and BungeeNeRF. Please zoom in for details.

the structural characteristics of each base pipeline: substantial reductions are observed on Vanilla 3DGS, Mip-Splatting, and DashGaussian, whereas the improvements are more moderate on the already compact FastGS pipeline. On TamingGS with BungeeNeRF, SPARE-GS preserves essentially the same resource consumption while noticeably improving rendering metrics. Overall, these results demonstrate that the proposed budget-guided regulation adapts to different structural regimes and generally provides a more favorable efficiency-quality trade-off.

### C. Evaluation on 3DGS Post-Processing Performance

In this section, we evaluated the effectiveness of SPARE-GS for downstream post-processing tasks. We first tested post-training compression, a task aimed at minimizing the

storage footprint of 3D scenes. For this purpose, we evaluated five representative post-training compression frameworks: LightGaussian [25], SOG [24], MesonGS [26], FCGS [15], and GHAP [27]. Subsequently, we examined post-training pruning, which typically relies on specific criteria to eliminate primitives for further memory reduction. we evaluated three pruning techniques (PUP [28], REFINE [29], and RAP [30]) without fine-tuning. All post-processing frameworks take the 3D Gaussian representations optimized by the original 3DGS pipeline with our SPARE-GS integrated as their inputs.

**Performance on 3DGS Compression.** Table II shows that using SPARE-GS representations generally reduces both the resulting model size and post-processing time. For LightGaussian and SOG, the BD-Rate reductions range from 27.78% to 55.38% and from 35.60% to 55.99%, respectively. MesonGS

TABLE II: **Quantitative evaluation of our method in terms of 3DGS post-processing compression.** The rightmost columns report relative processing time reductions ( $\Delta$ Time) and Bjøntegaard Delta (BD) metrics. ‘N/A’ indicates insufficient overlap between the corresponding R-D curves to compute the reported BD metric.

| Dataset                                | Input: Vanilla 3DGS (Base) |          |       |       |       | Input: SPARE-GS (Ours) |          |       |       |       | Ours vs. Base              |                      |                    |
|----------------------------------------|----------------------------|----------|-------|-------|-------|------------------------|----------|-------|-------|-------|----------------------------|----------------------|--------------------|
|                                        | Size (MB)                  | Time (s) | PSNR  | SSIM  | LPIPS | Size (MB)              | Time (s) | PSNR  | SSIM  | LPIPS | $\Delta$ Time $\downarrow$ | BD-Rate $\downarrow$ | BD-PSNR $\uparrow$ |
| Compression Method: LightGaussian [25] |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                           | 407.56                     | 101.25   | 27.19 | 0.807 | 0.228 | 244.08                 | 85.89    | 27.05 | 0.802 | 0.244 | -13.89%                    | -27.78%              | N/A                |
|                                        | 321.67                     | 97.40    | 26.98 | 0.802 | 0.233 | 192.69                 | 83.72    | 26.83 | 0.798 | 0.248 |                            |                      |                    |
|                                        | 235.90                     | 92.23    | 26.76 | 0.797 | 0.239 | 141.19                 | 80.87    | 26.58 | 0.792 | 0.254 |                            |                      |                    |
| Deep Blending                          | 365.53                     | 98.21    | 29.66 | 0.905 | 0.242 | 150.59                 | 77.36    | 29.63 | 0.908 | 0.250 | -19.16%                    | -55.38%              | N/A                |
|                                        | 289.13                     | 94.27    | 29.54 | 0.903 | 0.244 | 119.31                 | 76.77    | 29.48 | 0.906 | 0.252 |                            |                      |                    |
|                                        | 212.78                     | 90.75    | 29.31 | 0.900 | 0.247 | 88.09                  | 74.82    | 29.30 | 0.904 | 0.255 |                            |                      |                    |
| Tanks & Temples                        | 238.77                     | 86.05    | 23.56 | 0.849 | 0.174 | 137.43                 | 76.25    | 23.69 | 0.841 | 0.194 | -10.42%                    | -51.75%              | N/A                |
|                                        | 188.82                     | 83.51    | 23.44 | 0.845 | 0.178 | 108.81                 | 75.20    | 23.53 | 0.837 | 0.198 |                            |                      |                    |
|                                        | 138.66                     | 81.57    | 23.29 | 0.842 | 0.182 | 79.85                  | 73.51    | 23.40 | 0.833 | 0.202 |                            |                      |                    |
| BungeeNeRF                             | 1020.73                    | 158.37   | 27.68 | 0.911 | 0.103 | 510.44                 | 112.34   | 27.56 | 0.910 | 0.115 | -28.23%                    | -37.50%              | N/A                |
|                                        | 804.53                     | 149.28   | 27.51 | 0.908 | 0.108 | 402.31                 | 106.59   | 27.37 | 0.906 | 0.120 |                            |                      |                    |
|                                        | 587.86                     | 136.87   | 27.36 | 0.905 | 0.112 | 294.05                 | 100.08   | 27.18 | 0.902 | 0.125 |                            |                      |                    |
| Compression Method: SOG [24]           |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                           | 45.92                      | 39.45    | 26.44 | 0.782 | 0.255 | 28.13                  | 26.63    | 26.25 | 0.777 | 0.271 | -32.65%                    | -35.60%              | N/A                |
|                                        | 39.28                      | 38.64    | 25.98 | 0.772 | 0.267 | 24.13                  | 26.01    | 25.80 | 0.766 | 0.283 |                            |                      |                    |
|                                        | 29.56                      | 37.46    | 24.71 | 0.742 | 0.298 | 18.18                  | 25.18    | 24.49 | 0.736 | 0.314 |                            |                      |                    |
| Deep Blending                          | 45.20                      | 36.81    | 29.35 | 0.900 | 0.246 | 19.27                  | 19.88    | 29.28 | 0.902 | 0.256 | -46.77%                    | -55.99%              | N/A                |
|                                        | 38.64                      | 35.77    | 29.05 | 0.897 | 0.251 | 16.54                  | 18.87    | 28.99 | 0.899 | 0.261 |                            |                      |                    |
|                                        | 28.54                      | 34.42    | 28.02 | 0.883 | 0.270 | 12.27                  | 18.21    | 27.81 | 0.884 | 0.280 |                            |                      |                    |
| Tanks & Temples                        | 25.84                      | 26.97    | 23.10 | 0.830 | 0.194 | 15.21                  | 18.23    | 23.17 | 0.821 | 0.216 | -32.89%                    | -39.88%              | N/A                |
|                                        | 22.02                      | 26.64    | 22.80 | 0.821 | 0.205 | 13.03                  | 17.74    | 22.79 | 0.811 | 0.228 |                            |                      |                    |
|                                        | 16.57                      | 25.66    | 21.85 | 0.794 | 0.237 | 9.87                   | 17.23    | 21.74 | 0.783 | 0.260 |                            |                      |                    |
| BungeeNeRF                             | 128.33                     | 80.67    | 27.40 | 0.903 | 0.116 | 66.24                  | 45.68    | 27.15 | 0.898 | 0.132 | -43.74%                    | -44.17%              | N/A                |
|                                        | 110.47                     | 77.24    | 27.00 | 0.895 | 0.127 | 57.22                  | 43.65    | 26.72 | 0.889 | 0.144 |                            |                      |                    |
|                                        | 82.65                      | 74.05    | 25.71 | 0.868 | 0.163 | 42.91                  | 41.18    | 25.43 | 0.859 | 0.181 |                            |                      |                    |
| Compression Method: MesonGS [26]       |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                           | 24.41                      | 5840.30  | 26.61 | 0.796 | 0.245 | 16.43                  | 4539.65  | 26.49 | 0.792 | 0.257 | -19.77%                    | -4.30%               | +0.07 dB           |
|                                        | 18.05                      | 4925.81  | 26.54 | 0.794 | 0.247 | 12.16                  | 3890.42  | 26.28 | 0.788 | 0.263 |                            |                      |                    |
|                                        | 11.38                      | 3973.40  | 26.06 | 0.781 | 0.265 | 7.72                   | 3395.97  | 25.56 | 0.768 | 0.289 |                            |                      |                    |
| Deep Blending                          | 23.57                      | 5553.81  | 29.82 | 0.903 | 0.250 | 11.57                  | 3492.71  | 29.82 | 0.906 | 0.258 | -30.92%                    | -21.35%              | +0.25 dB           |
|                                        | 17.50                      | 4664.81  | 29.82 | 0.903 | 0.254 | 8.58                   | 3280.84  | 29.64 | 0.901 | 0.269 |                            |                      |                    |
|                                        | 11.02                      | 3642.29  | 29.54 | 0.896 | 0.269 | 5.50                   | 2801.20  | 29.03 | 0.889 | 0.292 |                            |                      |                    |
| Tanks & Temples                        | 12.56                      | 4248.56  | 23.31 | 0.838 | 0.197 | 10.52                  | 3419.86  | 23.36 | 0.836 | 0.203 | -19.81%                    | -14.38%              | +0.13 dB           |
|                                        | 9.35                       | 3673.83  | 23.19 | 0.830 | 0.209 | 7.83                   | 2916.13  | 23.12 | 0.824 | 0.219 |                            |                      |                    |
|                                        | 7.54                       | 3164.61  | 22.81 | 0.814 | 0.224 | 5.12                   | 2555.14  | 22.64 | 0.800 | 0.248 |                            |                      |                    |
| BungeeNeRF                             | 65.79                      | 11658.14 | 27.18 | 0.901 | 0.118 | 36.57                  | 7109.88  | 27.09 | 0.899 | 0.129 | -35.69%                    | N/A                  | N/A                |
|                                        | 47.99                      | 9018.83  | 27.35 | 0.903 | 0.119 | 26.67                  | 5871.53  | 27.12 | 0.897 | 0.135 |                            |                      |                    |
|                                        | 29.62                      | 6374.47  | 27.21 | 0.894 | 0.135 | 16.53                  | 4415.33  | 26.67 | 0.878 | 0.162 |                            |                      |                    |
| Compression Method: FCGS [15]          |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                           | 41.68                      | 386.81   | 27.35 | 0.804 | 0.233 | 34.08                  | 181.79   | 27.28 | 0.802 | 0.239 | -77.72%                    | -24.34%              | +0.17 dB           |
|                                        | 34.75                      | 907.27   | 27.19 | 0.801 | 0.231 | 24.71                  | 165.20   | 27.16 | 0.799 | 0.243 |                            |                      |                    |
|                                        | 27.75                      | 763.06   | 27.01 | 0.797 | 0.238 | 20.72                  | 111.44   | 27.01 | 0.796 | 0.247 |                            |                      |                    |
| Deep Blending                          | 37.23                      | 554.19   | 29.67 | 0.901 | 0.244 | 20.59                  | 73.47    | 29.71 | 0.905 | 0.249 | -86.41%                    | -53.99%              | N/A                |
|                                        | 27.62                      | 675.78   | 29.49 | 0.899 | 0.247 | 14.75                  | 62.63    | 29.60 | 0.903 | 0.251 |                            |                      |                    |
|                                        | 24.31                      | 224.61   | 29.24 | 0.896 | 0.252 | 12.56                  | 61.55    | 29.47 | 0.902 | 0.254 |                            |                      |                    |
| Tanks & Temples                        | 24.71                      | 461.62   | 23.47 | 0.841 | 0.181 | 17.73                  | 89.69    | 23.82 | 0.843 | 0.188 | -55.54%                    | N/A                  | +0.47 dB           |
|                                        | 18.40                      | 435.02   | 23.39 | 0.838 | 0.184 | 13.06                  | 193.63   | 23.77 | 0.840 | 0.191 |                            |                      |                    |
|                                        | 16.00                      | 186.84   | 23.32 | 0.835 | 0.188 | 11.18                  | 198.37   | 23.67 | 0.837 | 0.195 |                            |                      |                    |
| BungeeNeRF                             | -                          | -        | -     | -     | -     | -                      | -        | -     | -     | -     | -                          | -                    | -                  |
| Compression Method: GHAP [27]          |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                           | 436.52                     | 129.57   | 27.43 | 0.808 | 0.231 | 281.14                 | 127.93   | 27.31 | 0.802 | 0.247 | -19.56%                    | -9.29%               | +0.06 dB           |
|                                        | 311.69                     | 121.37   | 27.29 | 0.802 | 0.245 | 200.73                 | 86.35    | 27.10 | 0.794 | 0.263 |                            |                      |                    |
|                                        | 186.95                     | 112.02   | 27.00 | 0.790 | 0.269 | 120.39                 | 77.70    | 26.73 | 0.778 | 0.291 |                            |                      |                    |
| Deep Blending                          | 410.42                     | 108.53   | 29.81 | 0.903 | 0.245 | 186.02                 | 64.26    | 29.75 | 0.905 | 0.251 | -27.84%                    | N/A                  | -0.05 dB           |
|                                        | 293.08                     | 94.18    | 29.82 | 0.903 | 0.249 | 132.83                 | 76.99    | 29.70 | 0.904 | 0.256 |                            |                      |                    |
|                                        | 175.83                     | 91.61    | 29.79 | 0.902 | 0.257 | 79.67                  | 71.14    | 29.66 | 0.903 | 0.264 |                            |                      |                    |
| Tanks & Temples                        | 258.91                     | 116.62   | 23.79 | 0.846 | 0.180 | 162.22                 | 60.88    | 23.92 | 0.843 | 0.193 | -42.11%                    | -39.07%              | +0.14 dB           |
|                                        | 184.86                     | 92.72    | 23.76 | 0.842 | 0.192 | 115.82                 | 49.98    | 23.80 | 0.836 | 0.207 |                            |                      |                    |
|                                        | 110.87                     | 52.65    | 23.67 | 0.832 | 0.212 | 69.46                  | 40.80    | 23.64 | 0.823 | 0.231 |                            |                      |                    |
| BungeeNeRF                             | 1114.36                    | 452.21   | 27.63 | 0.910 | 0.106 | 594.80                 | 319.47   | 27.46 | 0.907 | 0.119 | -23.63%                    | -26.10%              | +0.30 dB           |
|                                        | 795.87                     | 316.60   | 27.41 | 0.903 | 0.119 | 424.80                 | 251.97   | 27.12 | 0.897 | 0.138 |                            |                      |                    |
|                                        | 477.50                     | 260.45   | 26.94 | 0.889 | 0.146 | 254.86                 | 214.57   | 26.47 | 0.876 | 0.174 |                            |                      |                    |



TABLE III: **Quantitative evaluation of our method in terms of 3DGS pruning.** Various pruning techniques are evaluated using the representations of vanilla 3DGS and SPARE-GS as inputs.

| Dataset                     | Input: Vanilla 3DGS (Base) |          |       |       |       | Input: SPARE-GS (Ours) |          |       |       |       | Ours vs. Base              |                      |                    |
|-----------------------------|----------------------------|----------|-------|-------|-------|------------------------|----------|-------|-------|-------|----------------------------|----------------------|--------------------|
|                             | Size (MB)                  | Time (s) | PSNR  | SSIM  | LPIPS | Size (MB)              | Time (s) | PSNR  | SSIM  | LPIPS | $\Delta$ Time $\downarrow$ | BD-Rate $\downarrow$ | BD-PSNR $\uparrow$ |
| Pruning Method: PUP [28]    |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                | 436.61                     | 137.39   | 26.94 | 0.807 | 0.227 | 281.21                 | 108.93   | 26.39 | 0.798 | 0.243 | -18.42%                    | -22.18%              | +2.04 dB           |
|                             | 311.87                     | 132.38   | 24.66 | 0.781 | 0.248 | 200.87                 | 108.40   | 23.05 | 0.753 | 0.275 |                            |                      |                    |
|                             | 187.12                     | 130.16   | 20.62 | 0.703 | 0.305 | 120.52                 | 108.92   | 18.87 | 0.651 | 0.343 |                            |                      |                    |
| Deep Blending               | 410.46                     | 110.27   | 29.77 | 0.906 | 0.239 | 186.05                 | 73.00    | 29.61 | 0.906 | 0.248 | -33.00%                    | -39.48%              | +3.57 dB           |
|                             | 293.19                     | 107.79   | 29.08 | 0.898 | 0.251 | 132.90                 | 72.25    | 27.65 | 0.884 | 0.275 |                            |                      |                    |
|                             | 175.91                     | 105.55   | 25.67 | 0.860 | 0.296 | 79.74                  | 71.58    | 21.73 | 0.812 | 0.344 |                            |                      |                    |
| Tanks & Temples             | 258.98                     | 65.04    | 23.06 | 0.840 | 0.182 | 162.26                 | 49.02    | 22.96 | 0.834 | 0.196 | -24.06%                    | -26.16%              | +2.03 dB           |
|                             | 184.98                     | 67.11    | 21.38 | 0.813 | 0.207 | 115.90                 | 50.01    | 20.24 | 0.795 | 0.231 |                            |                      |                    |
|                             | 110.99                     | 62.05    | 18.06 | 0.742 | 0.265 | 69.54                  | 48.44    | 16.54 | 0.708 | 0.295 |                            |                      |                    |
| BungeeNeRF                  | 1114.41                    | 212.16   | 27.68 | 0.912 | 0.099 | 594.84                 | 109.42   | 27.38 | 0.907 | 0.112 | -45.84%                    | -35.97%              | +2.65 dB           |
|                             | 796.01                     | 186.75   | 26.75 | 0.894 | 0.114 | 424.89                 | 99.66    | 25.98 | 0.879 | 0.136 |                            |                      |                    |
|                             | 477.61                     | 164.91   | 23.49 | 0.831 | 0.166 | 254.93                 | 96.29    | 21.67 | 0.792 | 0.206 |                            |                      |                    |
| Pruning Method: REFINE [29] |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                | 436.61                     | 5.21     | 27.25 | 0.807 | 0.227 | 281.06                 | 3.46     | 26.65 | 0.798 | 0.244 | -32.63%                    | -18.55%              | +1.07 dB           |
|                             | 311.87                     | 4.44     | 26.21 | 0.789 | 0.247 | 200.91                 | 3.73     | 25.20 | 0.770 | 0.272 |                            |                      |                    |
|                             | 187.12                     | 4.48     | 23.55 | 0.733 | 0.299 | 120.57                 | 2.33     | 21.89 | 0.680 | 0.340 |                            |                      |                    |
| Deep Blending               | 410.46                     | 3.79     | 29.59 | 0.900 | 0.245 | 186.05                 | 2.52     | 29.40 | 0.900 | 0.255 | -43.08%                    | -40.77%              | +1.59 dB           |
|                             | 293.19                     | 3.71     | 29.09 | 0.893 | 0.255 | 132.90                 | 1.83     | 28.44 | 0.886 | 0.274 |                            |                      |                    |
|                             | 175.91                     | 2.90     | 27.64 | 0.870 | 0.283 | 79.74                  | 1.57     | 26.29 | 0.849 | 0.315 |                            |                      |                    |
| Tanks & Temples             | 258.98                     | 3.24     | 23.35 | 0.839 | 0.182 | 162.26                 | 1.66     | 23.27 | 0.835 | 0.194 | -33.11%                    | -28.60%              | +1.43 dB           |
|                             | 184.98                     | 2.59     | 22.35 | 0.816 | 0.203 | 115.90                 | 2.00     | 21.83 | 0.804 | 0.222 |                            |                      |                    |
|                             | 110.99                     | 1.69     | 20.15 | 0.758 | 0.256 | 69.54                  | 1.37     | 19.33 | 0.737 | 0.282 |                            |                      |                    |
| BungeeNeRF                  | 1114.41                    | 14.20    | 26.95 | 0.904 | 0.109 | 594.84                 | 6.24     | 26.45 | 0.893 | 0.127 | -49.08%                    | -36.41%              | +2.63 dB           |
|                             | 796.01                     | 10.48    | 25.45 | 0.874 | 0.136 | 424.89                 | 4.83     | 24.52 | 0.845 | 0.171 |                            |                      |                    |
|                             | 477.61                     | 7.31     | 22.53 | 0.783 | 0.214 | 254.93                 | 5.22     | 21.08 | 0.718 | 0.276 |                            |                      |                    |
| Pruning Method: RAP [30]    |                            |          |       |       |       |                        |          |       |       |       |                            |                      |                    |
| Mip-NeRF 360                | 436.61                     | 78.81    | 27.37 | 0.810 | 0.230 | 281.21                 | 47.62    | 27.02 | 0.800 | 0.250 | -34.05%                    | -21.54%              | +1.08 dB           |
|                             | 311.87                     | 74.15    | 26.47 | 0.790 | 0.250 | 200.87                 | 59.70    | 25.70 | 0.770 | 0.280 |                            |                      |                    |
|                             | 187.12                     | 71.81    | 24.08 | 0.730 | 0.304 | 120.52                 | 40.91    | 23.12 | 0.697 | 0.342 |                            |                      |                    |
| Deep Blending               | 410.46                     | 79.44    | 29.72 | 0.905 | 0.243 | 186.05                 | 30.97    | 29.48 | 0.904 | 0.254 | -57.70%                    | -39.69%              | +1.46 dB           |
|                             | 293.19                     | 73.17    | 29.22 | 0.898 | 0.256 | 132.90                 | 37.25    | 28.54 | 0.889 | 0.277 |                            |                      |                    |
|                             | 175.91                     | 70.76    | 27.86 | 0.877 | 0.287 | 79.74                  | 26.27    | 26.69 | 0.857 | 0.320 |                            |                      |                    |
| Tanks & Temples             | 258.98                     | 42.74    | 23.61 | 0.847 | 0.176 | 162.26                 | 25.09    | 23.63 | 0.840 | 0.191 | -37.22%                    | -29.98%              | +1.16 dB           |
|                             | 184.98                     | 41.19    | 22.93 | 0.825 | 0.198 | 115.90                 | 30.19    | 22.62 | 0.808 | 0.223 |                            |                      |                    |
|                             | 110.99                     | 39.10    | 21.18 | 0.765 | 0.254 | 69.54                  | 21.96    | 20.60 | 0.734 | 0.291 |                            |                      |                    |
| BungeeNeRF                  | -                          | -        | -     | -     | -     | -                      | -        | -     | -     | -     | -                          | -                    | -                  |

exhibits more moderate but consistent gains on the datasets where BD-Rate can be computed, with reductions ranging from 4.30% to 21.35%, accompanied by processing-time reductions ranging from 19.77% to 35.69%. FCGS achieves BD-Rate reductions of 24.34% and 53.99% on Mip-NeRF 360 and Deep Blending, respectively, together with a BD-PSNR improvement of 0.47 dB on Tanks & Temples. GHAP also benefits from the compact input representations, although the magnitude of its rate-distortion improvement varies across datasets. Specifically, it achieves BD-Rate reductions ranging from 9.29% to 39.07% where the metric is available, while its BD-PSNR changes range from -0.05 dB to +0.30 dB.

The processing-time results follow a similar trend. SOG reduces latency by up to 46.77%, while FCGS achieves a reduction of up to 86.41%. These improvements can be attributed to the smaller number of input primitives and the correspondingly reduced processing workload for downstream clustering, quantization, and coding operations.

**Performance on 3DGS Pruning.** As shown in Table III, PUP achieves BD-Rate reductions ranging from 22.18% to 39.48%, BD-PSNR improvements ranging from 2.03 dB to

3.57 dB, and processing-time reductions ranging from 18.42% to 45.84%. REFINE obtains BD-Rate reductions ranging from 18.55% to 40.77%, together with BD-PSNR improvements ranging from 1.07 dB to 2.63 dB. RAP similarly achieves BD-Rate reductions ranging from 21.54% to 39.69%, BD-PSNR improvements ranging from 1.08 dB to 1.46 dB, and processing-time reductions ranging from 34.05% to 57.70%. These consistent trends across different pruning strategies suggest that regulating structural growth during training provides a more compact input representation for subsequent post-training simplification.

**Summary.** Across both compression and pruning evaluations, SPARE-GS generally reduces post-processing time and improves downstream rate-distortion efficiency. Although individual operating points may exhibit modest variations in PSNR, SSIM, or LPIPS, the reported BD metrics indicate more favorable quality-size trade-offs for most combinations of downstream methods and datasets where sufficient curve overlap is available. The magnitude of the improvement varies across downstream methods, but the overall results support SPARE-GS as an effective pre-conditioning strategy for sub-

TABLE IV: **Ablation Study on Core Components.** The individual contributions of Budget-modulated Densification (*BMD*), Redundancy-aware Pruning (*RAP*), and Equilibrium-driven Termination (*EDT*) are evaluated by removing each module from the complete pipeline.

| Method         | <i>BMD</i> | <i>RAP</i> | <i>EDT</i> | Gaussians↓ | Time(s)↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
|----------------|------------|------------|------------|------------|----------|-------|-------|--------|
| Baseline       | ×          | ×          | ×          | 2.64M      | 1314     | 27.51 | 0.813 | 0.221  |
| w/o <i>BMD</i> | ×          | ✓          | ✓          | 1.88M      | 1091     | 27.50 | 0.813 | 0.227  |
| w/o <i>RAP</i> | ✓          | ×          | ✓          | 2.32M      | 1161     | 27.48 | 0.809 | 0.228  |
| w/o <i>EDT</i> | ✓          | ✓          | ×          | 1.68M      | 964      | 27.45 | 0.809 | 0.233  |
| <b>Full</b>    | ✓          | ✓          | ✓          | 1.69M      | 928      | 27.46 | 0.809 | 0.234  |



Fig. 5: **Visual comparison of ablation study.** Zooming in for details.

sequent 3DGS post-processing.

#### D. Ablation Study

In this section, we conducted ablation studies on the Mip-NeRF 360 dataset to evaluate the individual contributions of the proposed modules and analyze the sensitivity of the main hyperparameters. To ensure a fair comparison, all ablation variants, including the complete model, were re-evaluated under the same experimental settings.

1) *Effectiveness of Proposed Modules:* As detailed in Sec. IV-C, SPARE-GS comprises three components: Budget-modulated Densification, *BMD*, Redundancy-aware Pruning, *RAP*, and Equilibrium-driven Termination, *EDT*. We conducted a leave-one-out ablation study by removing one component at a time and comparing each resulting variant with both the baseline and the complete model. The results are summarized in Table IV.

**Impact of Budget-modulated Densification.** *BMD* acts as the stage regulator of structural growth. Instead of allowing the base densification rule to expand all high-gradient primitives uniformly, it reweights the densification score according to the regional budget deviation. Removing *BMD* increases the primitive count from 1.69M to 1.88M and the training time from 928.40 s to 1090.97 s. This corresponds to around 11.1% more Gaussians and 17.5% longer training compared with the full model. The larger increase in training time suggests that unconstrained densification not only leaves more primitives in the final representation but also introduces additional optimization overhead throughout subsequent iterations. Although w/o *BMD* achieves a slightly higher PSNR, this improvement is only 0.04 dB over the full model and comes with a clear increase in structural costs. This confirms that

*BMD* provides an effective front-end control mechanism that suppresses unnecessary growth while preserving comparable rendering quality.

**Impact of Redundancy-aware Pruning.** *RAP* serves as the complementary back-end regulator to *BMD*. While *BMD* suppresses unnecessary growth before new primitives are created, *RAP* removes redundant primitives that have already accumulated in over-budgeted regions. This role is clearly reflected by the w/o *RAP* variant, where the number of Gaussians increases to 2.32M, which is 36.8% higher than the full model. The training time also rises from 928.40 s to 1160.72 s, indicating that redundant primitives not only enlarge the final representation but also continuously increase rendering and optimization costs during training. Meanwhile, the rendering quality remains nearly unchanged, with only a 0.02 dB PSNR difference between w/o *RAP* and the full model. This suggests that most primitives removed by *RAP* are indeed low-contribution structural redundancies rather than essential high-frequency details. Therefore, *RAP* is critical for converting the budget feedback signal into actual model compactness.

**Impact of Equilibrium-driven Termination.** *EDT* mainly targets the temporal redundancy of training rather than directly reducing the final primitive count. Compared with the full model, the w/o *EDT* variant continues optimization for a longer time, increasing the training cost from 928.40 s to 964.34 s. However, this additional optimization does not translate into meaningful quality improvements: PSNR changes from 27.46 dB to 27.45 dB, while SSIM remains the same and LPIPS differs only marginally. The primitive count of w/o *EDT* is also very close to that of the full model, indicating that the structural state has already become saturated when *EDT* triggers termination. Therefore, *EDT* effectively identifies the low-return stage of training, where continuing the fixed schedule mostly incurs computational costs without providing noticeable reconstruction benefits.

Overall, the ablation results reveal a clear division of labor among the three modules. *BMD* prevents excessive growth by regulating densification before new primitives are introduced, *RAP* removes low-contribution primitives accumulated in over-budgeted regions, and *EDT* avoids redundant optimization after the structural state becomes saturated. Their combination yields a favorable efficiency-quality trade-off: compared with the baseline, the full model removes nearly one million Gaussians and reduces training time by 29.3%, while maintaining comparable rendering fidelity.

2) *Hyperparameter Sensitivity Analysis:* We further analyzed the sensitivity of representative hyperparameters introduced in our framework, including the spatial grid resolution, the pruning opacity threshold, and the thresholds governing equilibrium-driven termination. The quantitative results are presented in Table V.

**Spatial Grid Size  $K$ .** The spatial grid resolution controls the granularity at which regional statistics are aggregated and target budgets are assigned. The three resolutions produce similar Gaussian counts and perceptual metrics, indicating



TABLE V: **Sensitivity analysis of key hyperparameters.** Default settings used in our full model are marked in gray.

| Hyperparameters   | Gaussians ↓      | Time (s) ↓    | PSNR ↑       | SSIM ↑       | LPIPS ↓      |
|-------------------|------------------|---------------|--------------|--------------|--------------|
| $K$               |                  |               |              |              |              |
| $16^3$            | <b>1,685,685</b> | <b>888.92</b> | 27.43        | <b>0.809</b> | <b>0.234</b> |
| $32^3$ (Default)  | 1,693,989        | 928.40        | <b>27.46</b> | <b>0.809</b> | <b>0.234</b> |
| $64^3$            | 1,688,937        | 894.68        | 27.42        | <b>0.809</b> | <b>0.234</b> |
| $\tau_\alpha$     |                  |               |              |              |              |
| 0.01              | 2,230,887        | 1136.96       | <b>27.48</b> | <b>0.809</b> | <b>0.228</b> |
| 0.05 (Default)    | 1,693,989        | 928.40        | 27.46        | <b>0.809</b> | 0.234        |
| 0.10              | <b>1,313,693</b> | <b>744.71</b> | 27.34        | 0.804        | 0.245        |
| $\ell$            |                  |               |              |              |              |
| 1 step            | <b>1,688,295</b> | <b>788.26</b> | 27.31        | 0.807        | 0.236        |
| 2 steps (Default) | 1,693,989        | 928.40        | <b>27.46</b> | <b>0.809</b> | <b>0.234</b> |
| 3 steps           | 1,694,566        | 937.14        | <b>27.46</b> | <b>0.809</b> | <b>0.234</b> |
| $\epsilon_L$      |                  |               |              |              |              |
| 0.016             | <b>1,692,243</b> | <b>868.78</b> | 27.43        | 0.808        | 0.235        |
| 0.008 (Default)   | 1,693,989        | 928.40        | 27.46        | <b>0.809</b> | <b>0.234</b> |
| 0.004             | 1,692,740        | 962.89        | <b>27.49</b> | <b>0.809</b> | <b>0.234</b> |
| $\epsilon_N$      |                  |               |              |              |              |
| 0.05              | 1,694,729        | <b>898.40</b> | 27.37        | 0.808        | <b>0.234</b> |
| 0.01 (Default)    | <b>1,693,989</b> | 928.40        | <b>27.46</b> | <b>0.809</b> | <b>0.234</b> |
| 0.001             | 1,694,289        | 941.40        | <b>27.46</b> | <b>0.809</b> | <b>0.234</b> |

limited sensitivity to the grid granularity. Although  $16^3$  is slightly faster,  $32^3$  achieves the highest PSNR while providing finer spatial localization than the coarse grid. We therefore adopt  $32^3$  as a conservative middle-resolution setting.

**Pruning Opacity Threshold  $\tau_\alpha$ .** This parameter, defined in Eq. (24), controls the aggressiveness of redundancy removal. Setting a high threshold of  $\tau_\alpha = 0.10$  aggressively removes Gaussians, achieving a highly compact model of 1.31 M primitives. However, this leads to a noticeable degradation in rendering quality, where PSNR drops to 27.34 dB and LPIPS spikes to 0.245, indicating lost perceptual details. Conversely, a very conservative threshold of  $\tau_\alpha = 0.01$  retains 2.23 M primitives, which achieves a better LPIPS of 0.228 at the cost of limiting the compression capability. Therefore, setting  $\tau_\alpha = 0.05$  strikes the balance between representational capacity and structural parsimony.

**Termination Patience  $\ell$ .** This parameter dictates the number of consecutive intervals required to trigger the Equilibrium-driven Termination established in Eq. (26). A minimal patience of  $\ell = 1$  causes the optimization to halt prematurely before full convergence, degrading the final rendering fidelity with PSNR dropping to 27.31 dB and LPIPS increasing to 0.236. Our default setting of  $\ell = 2$  mitigates such premature termination and serves as a stable indicator of structural saturation.

**Loss Saturation Threshold  $\epsilon_L$ .** The loss tolerance  $\epsilon_L$  determines the sensitivity of the photometric convergence check. A loose tolerance of  $\epsilon_L = 0.016$  can trigger the halting condition prematurely, leading to sub-optimal rendering quality. Alternatively, setting a tight tolerance of  $\epsilon_L = 0.004$  prolongs the training to 962 s without yielding meaningful perceptual quality gains over the default configuration. We set  $\epsilon_L = 0.008$  to maintain a practical balance between training efficiency and convergence quality.

**Count Saturation Threshold  $\epsilon_N$ .** We additionally evaluate

the threshold for primitive count variation,  $\epsilon_N$ , which acts as an explicit indicator of structural convergence. A loose count threshold of  $\epsilon_N = 0.05$  halts the structural evolution before full optimization, reducing the PSNR to 27.37 dB. Conversely, a strict threshold of  $\epsilon_N = 0.001$  fails to detect saturation in time, extending the training duration to 941 s without improving rendering quality or LPIPS. Our default setting of  $\epsilon_N = 0.01$  preserves photometric accuracy while preventing redundant computations.

Overall, the default configuration consistently provides a balanced trade-off between compactness, training efficiency, and rendering fidelity, while more aggressive settings mainly trade quality for additional compression or speed.

### E. Further Analysis

To examine the effectiveness of the marginal utility proxy, we conducted a leave-one-out ablation study by removing the regional gradient, multi-view visibility, or geometric capacity normalization while retaining the remaining components.

TABLE VI: **Ablation Study on the Marginal Utility Proxy.**

| Method         | $H_r^{(t)}$ | $V_r^{(t)}$ | $C_r^{(t)}$ | Gaussians ↓ | Time(s) ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|----------------|-------------|-------------|-------------|-------------|-----------|--------|--------|---------|
| w/o Gradient   | ×           | ✓           | ✓           | 1.72M       | 972       | 27.45  | 0.809  | 0.234   |
| w/o Visibility | ✓           | ×           | ✓           | 1.73M       | 930       | 27.46  | 0.808  | 0.235   |
| w/o Capacity   | ✓           | ✓           | ×           | 1.73M       | 961       | 27.46  | 0.808  | 0.234   |
| <b>Full</b>    | ✓           | ✓           | ✓           | 1.69M       | 928       | 27.46  | 0.809  | 0.234   |

As shown in Table VI, removing any component increases the final Gaussian count while preserving comparable rendering quality. Gradient magnitude and capacity normalization have clearer effects on training efficiency, increasing the training time by 4.67% and 3.56%, respectively, when removed. Multi-view visibility mainly contributes to structural compactness, with its removal increasing the Gaussian count by 1.90%, although its effect on training time is limited. These results indicate that the three signals provide complementary, rather than individually dominant, guidance for regional allocation.

**Computational Time Breakdown.** To verify the computational efficiency of SPARE-GS, we profiled the runtime breakdown across the nine Mip-NeRF 360 scenes. As illustrated in Fig. 6, the overhead introduced by our online budget module is practically negligible. For each scene, the accumulated budget-estimation time ranges from 0.33 s to 0.93 s over the entire training process. This constitutes a mere 0.073% of the total training time on average, occupying just 0.87% of the structural update phase. In contrast, standard operations (i.e., backward propagation, optimizer updates, and forward rendering) dominate the runtime. This near-zero overhead stems from two design choices. First, SPARE-GS directly reuses statistics natively maintained during standard 3DGS training (e.g., accumulated gradients, visibility, and opacity), avoiding redundant metric computations. Second, our lightweight operations, including voxel assignment, regional aggregation, and budget allocation, are executed exclusively at structural update intervals rather than at every rendering step. Consequently, the acceleration reported in Table I primarily

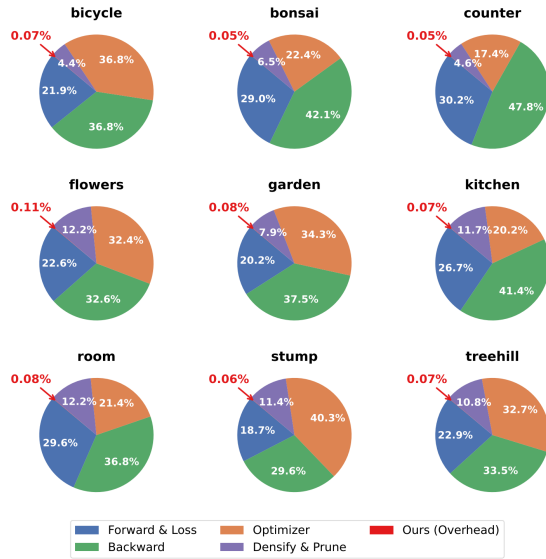


Fig. 6: **Computational time breakdown.** The pie charts illustrate the temporal distribution of different training components. The overhead introduced by SPARE-GS is highlighted in red.

TABLE VII: **Representative Failure cases on Mip-NeRF 360.**

| Scene   | Efficiency gains ( $\downarrow$ ) |               | Quality degradation |               |                |
|---------|-----------------------------------|---------------|---------------------|---------------|----------------|
|         | $\Delta$ Gaussians                | $\Delta$ Time | $\Delta$ PSNR       | $\Delta$ SSIM | $\Delta$ LPIPS |
| bonsai  | -33.71%                           | -19.13%       | -0.28               | -0.002        | +0.007         |
| flowers | -32.73%                           | -24.97%       | -0.20               | -0.009        | +0.016         |
| garden  | -33.24%                           | -44.77%       | -0.24               | -0.008        | +0.016         |

stems from the reduced structural redundancies and adaptive termination.

**Analysis of Structural Regulation.** To understand how SPARE-GS regulates structural growth, we analyzed the active densification phase across the Mip-NeRF 360 scenes. As optimization progresses, SPARE-GS consistently suppresses excessive Gaussian proliferation. The average reduction in primitive count widens steadily from 8.7% at 1.5K iterations to 26.5% at 6K, ultimately reaching 35.8% by the end of the densification stage (15K). Scene-wise reductions at 15K iterations remain highly consistent, ranging from 29.7% to 48.0%. This steady divergence validates that SPARE-GS is not merely a post-hoc pruning filter for an already overgrown model. Instead, it proactively reshapes the evolutionary trajectory by continuously penalizing redundant growth in over-budgeted regions and guiding new primitives toward a more compact allocation. These results confirm that the proposed budget-guided regulation effectively controls Gaussian proliferation during the critical structural growth phase across diverse scenes.

**Failure Cases and Limitations.** Table VII presents representative cases where the efficiency gains of SPARE-GS are accompanied by more noticeable quality degradation. Across the three scenes, SPARE-GS reduces the Gaussian count

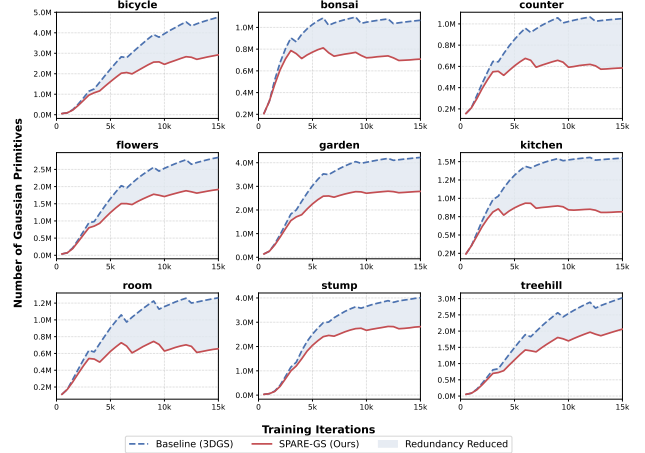


Fig. 7: **Evolution of Gaussian count during the densification phase (0–15k iterations).** The shaded area represents the computational and memory footprint saved by our region-budget-aware regulation over time.

by approximately 33% and the training time by 19.13% to 44.77%, while PSNR decreases by 0.20 dB to 0.28 dB. These cases may reflect limitations of the current heuristic demand estimation and capacity calibration in scenes with complex occlusions, fine details, or uneven view coverage. Future work could improve robustness by incorporating uncertainty-aware, occlusion-aware, or learned demand estimation, together with more conservative budget control in low-confidence regions.

## VI. CONCLUSION

In this work, we revisited the efficiency problem of 3DGS from the perspective of global structural resource allocation. Instead of treating primitive growth as a collection of independent local decisions, we formulated the structural evolution of 3DGS as a budget-constrained optimization problem and introduced SPARE-GS to dynamically align primitive allocation with regional representational demand. Through KKT-inspired budget assignment and feedback regulation, SPARE-GS provides a unified mechanism for coordinating structural growth, redundancy removal, and adaptive termination, thereby guiding the representation toward a more balanced efficiency-quality trade-off. Extensive experiments across standard standard, accelerated, and structure-enhanced 3DGS pipelines, as well as downstream compression and pruning methods, demonstrate the broad applicability of this formulation. Overall, SPARE-GS establishes a general framework for reasoning about and regulating finite structural resources in Gaussian-based scene representations. Future work will explore more accurate uncertainty-aware or learned demand estimation to improve robustness in structurally complex scenes.

## REFERENCES

- [1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139:1–139:14, 2023.

- [2] Q. Shen, Z. Wu, X. Yi, P. Zhou, H. Zhang, S. Yan, and X. Wang, "Gamba: Marry gaussian splatting with mamba for single-view 3d reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2025.
- [3] B. Zhou, S. Zheng, H. Tu, R. Shao, B. Liu, S. Zhang, L. Nie, and Y. Liu, "GPS-Gaussian+: Generalizable pixel-wise 3d gaussian splatting for real-time human-scene rendering from sparse views," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2025.
- [4] T. Wu, J.-M. Sun, Y.-K. Lai, Y. Ma, L. Kobbelt, and L. Gao, "DeferredGS: Decoupled and relightable gaussian splatting with deferred shading," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 8, pp. 6307–6319, 2025.
- [5] X. Lei, M. Wang, W. Zhou, and H. Li, "GaussNav: Gaussian splatting for visual navigation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 5, pp. 4108–4121, 2025.
- [6] G. Fang and B. Wang, "Efficient scene modeling via structure-aware and region-prioritized 3d gaussians," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 48, no. 4, pp. 4623–4641, 2026.
- [7] Y. Wang, T. Huang, H. Chen, and G. H. Lee, "FreeSplat++: Generalizable 3d gaussian splatting for efficient indoor scene reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 48, no. 7, pp. 7749–7765, 2026.
- [8] T. Bai, Z. Qiu, H. Chen, and Z. Dai, "Plug-and-play optimization for 3d gaussian splatting compression: Distribution regularization, probabilistic pruning and detail compensation," in *AAAI*, vol. 40, no. 4, 2026, pp. 2372–2380.
- [9] S. Niedermayr, J. Stumpfegger, and R. Westermann, "Compressed 3D Gaussian splatting for accelerated novel view synthesis," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024, pp. 10349–10358.
- [10] X. Du, Y. Wang, K. Zhan, and X. Yu, "Mobile-gs: Real-time gaussian splatting for mobile devices," in *Int. Conf. Learn. Represent.*, 2026.
- [11] J. Chen, Z. Li, Y. Cai, H. Jiang, C. Qian, J. Kang, S. Gao, H. Zhao, T. Mao, and Y. Zhang, "HAIF-GS: Hierarchical and induced flow-guided Gaussian splatting for dynamic scene," in *Adv. Neural Inform. Process. Syst.*, vol. 38, 2025, pp. 125539–125563.
- [12] Y. Chen, Q. Wu, M. Li, W. Lin, J. Hou, M. Harandi, and J. Cai, "Feed-forward compression of static and streamable 3D Gaussian splatting," *IEEE Trans. Circuit Syst. Video Technol.*, 2026.
- [13] M. Niemeyer, F. Manhardt, M.-J. Rakotosaona, M. Oechsle, D. Duckworth, R. Gosula, K. Tateno, J. Bates, D. Kaeser, and F. Tombari, "RadSplat: Radiance field-informed Gaussian splatting for robust real-time rendering with 900+ FPS," in *International Conference on 3D Vision (3DV)*, 2025, pp. 134–144.
- [14] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai, "Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2025.
- [15] Y. Chen, Q. Wu, M. Li, W. Lin, M. Harandi, and J. Cai, "Fast feedforward 3D Gaussian splatting compression," in *Int. Conf. Learn. Represent.*, vol. 2025, 2025, pp. 74859–74872.
- [16] G. Fang and B. Wang, "Mini-splatting: Representing scenes with a constrained number of Gaussians," in *Eur. Conf. Comput. Vis.*, 2024, pp. 165–181.
- [17] Y. Chen, J. Jiang, K. Jiang, X. Tang, Z. Li, X. Liu, and Y. Nie, "Dashgaussian: Optimizing 3d gaussian splatting in 200 seconds," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2025, pp. 11146–11155.
- [18] A. Hanson, A. Tu, G. Lin, V. Singla, M. Zwicker, and T. Goldstein, "Speedy-splat: Fast 3D Gaussian splatting with sparse pixels and sparse primitives," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2025, pp. 21537–21546.
- [19] S. Ren, T. Wen, Y. Fang, and B. Lu, "Fastgs: Training 3d gaussian splatting in 100 seconds," pp. 26094–26103, 2026.
- [20] S. Rota Bulò, L. Porzi, and P. Kotschieder, "Revising densification in gaussian splatting," in *Eur. Conf. Comput. Vis.*, 2024, pp. 347–362.
- [21] Z. Zhang, W. Hu, Y. Lao, T. He, and H. Zhao, "Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting," in *Eur. Conf. Comput. Vis.*, 2024, pp. 326–342.
- [22] K. Cheng, X. Long, K. Yang, Y. Yao, W. Yin, Y. Ma, W. Wang, and X. Chen, "Gaussianpro: 3d gaussian splatting with progressive propagation," in *Int. Conf. Mach. Learn.*, 2024, pp. 8123–8140.
- [23] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "HAC: Hash-grid assisted context for 3D Gaussian splatting compression," in *Eur. Conf. Comput. Vis.*, 2024, pp. 422–438.
- [24] W. Morgenstern, F. Barthel, A. Hilsman, and P. Eisert, "Compact 3D scene representation via self-organizing Gaussian grids," in *Eur. Conf. Comput. Vis.*, 2024, pp. 18–34.
- [25] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, "LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS," in *Adv. Neural Inform. Process. Syst.*, vol. 37, 2024, pp. 140138–140158.
- [26] S. Xie, W. Zhang, C. Tang, Y. Bai, R. Lu, S. Ge, and Z. Wang, "MesonGS: Post-training compression of 3D Gaussians via efficient attribute transformation," in *Eur. Conf. Comput. Vis.*, 2024, pp. 434–452.
- [27] T. Wang, M. Li, G. Zeng, C. Meng, and Q. Zhang, "Gaussian herding across pens: An optimal transport perspective on global gaussian reduction for 3DGS," in *Adv. Neural Inform. Process. Syst.*, vol. 38, 2025, pp. 157898–157923.
- [28] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein, "PUP 3D-GS: Principled uncertainty pruning for 3D Gaussian splatting," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2025, pp. 5949–5958.
- [29] Z. Chen, S. Wan, M. Yu, F. Yang, and J. Hou, "Refine: Super-efficient 3d gaussian splatting pruning via rendering-free primitive importance," *arXiv preprint arXiv:2606.09074*, 2026.
- [30] K. Yang, Q. Yang, Y. Xu, and Z. Li, "Rap: Fast feedforward rendering-free attribute-guided primitive importance score prediction for efficient 3d gaussian splatting processing," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2026, pp. 33323–33332.
- [31] Z. Ye, W. Li, S. Liu, P. Qiao, and Y. Dou, "Absgs: Recovering fine details in 3d gaussian splatting," in *ACM Int. Conf. Multimedia*, 2024, pp. 1053–1061.
- [32] S. S. Mallick, R. Goel, B. Kerbl, M. Steinberger, F. V. Carrasco, and F. De La Torre, "Taming 3dgs: High-quality radiance fields with limited resources," 2024, pp. 1–11.
- [33] S. Kheradmand, D. Rebain, G. Sharma, W. Sun, Y.-C. Tseng, H. Isack, A. Kar, A. Tagliasacchi, and K. M. Yi, "3d gaussian splatting as markov chain monte carlo," *Adv. Neural Inform. Process. Syst.*, vol. 37, pp. 80965–80986, 2024.
- [34] P. Wang, Y. Wang, D. Wang, S. Mohan, Z. Fan, L. Wu, R. Cai, Y.-Y. Yeh, Z. Wang, Q. Liu *et al.*, "Steepest descent density control for compact 3d gaussian splatting," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2025, pp. 26663–26672.
- [35] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, "Mip-splatting: Alias-free 3d gaussian splatting," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024, pp. 19447–19456.
- [36] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, "Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024, pp. 20654–20664.
- [37] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024*, 2024, pp. 1–11.
- [38] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "HAC++: Towards 100x compression of 3d gaussian splatting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 11, pp. 10210–10226, 2025.
- [39] J. Chen, Y. Chen, Y. Zou, Y. Huang, P. Wang, Y. Liu, Y. Sun, and W. Wang, "MEGS<sup>2</sup>: Memory-efficient gaussian splatting via spherical gaussians and unified pruning," in *Int. Conf. Learn. Represent.*, 2026.
- [40] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, "Compact 3D Gaussian representation for radiance field," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024, pp. 21719–21728.
- [41] X. Liu, X. Wu, P. Zhang, S. Wang, Z. Li, and S. Kwong, "CompGS: Efficient 3D scene representation via compressed Gaussian splatting," in *ACM Int. Conf. Multimedia*, 2024, pp. 2936–2944.
- [42] M. S. Ali, M. Qamar, S. H. Bae, and E. Tartaglione, "Trimming the fat: Efficient compression of 3D Gaussian splats through pruning," in *Brit. Mach. Vis. Conf.*, 2024.
- [43] M. S. Ali, S.-H. Bae, and E. Tartaglione, "ElmGS: Enhancing memory and computation scalability through compression for 3D Gaussian splatting," in *Winter Conf. Appl. Comput. Vis.*, 2025, pp. 2591–2600.
- [44] S. Girish, K. Gupta, and A. Shrivastava, "Eagles: Efficient accelerated 3D Gaussians with lightweight encodings," in *Eur. Conf. Comput. Vis.*, 2024, pp. 54–71.
- [45] G. Gordon and R. Tibshirani, "Karush-kuhn-tucker conditions," *Optimization*, vol. 10, no. 725/36, p. 725, 2012.
- [46] G. Kopanas, J. Philip, T. Leimkühler, and G. Drettakis, "Point-based neural rendering with per-view optimization," *Comput. Graph. Forum*, vol. 40, no. 4, pp. 29–43, 2021.
- [47] K. M. Ormazabal, "The law of diminishing marginal utility in alfred marshall's principles of economics," *Journal of the History of Economic Thought*, vol. 2, no. 1, pp. 91–126, 1995.
- [48] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer.
- [49] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 5470–5479.



- [50] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and Temples: Benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, 2017.
- [51] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–15, 2018.
- [52] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, "Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering," in *Eur. Conf. Comput. Vis.*, 2022, pp. 106–122.
- [53] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.
- [54] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 586–595.

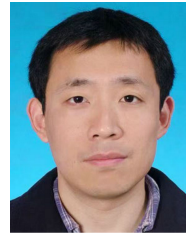


**Jiazhi Xia** is a Professor in the School of Computer Science and Engineering, Central South University. His research interests include visualization, geometry processing, and 3D vision.



**Zhang Chen** received the B.E. degree in Electronics and Information Engineering and the M.E. degree in Signal and Information Processing from Northwestern Polytechnical University, Xi'an, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree in Information and Communication Engineering at the same university. Since 2026, he has been a Research Assistant with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China. His current research interests include 3DGS pruning, 3DGS optimization, multimedia quality assessment, and 3D reconstruction.

multimedia quality assessment, and 3D reconstruction.



**Wei Yao Lin** (Senior Member, IEEE) received the B.E. and M.E. degrees from Shanghai Jiao Tong University, Shanghai, China, in 2003 and 2005, respectively, and the Ph.D. degree from the University of Washington, Seattle, WA, USA, in 2010, all in electrical engineering. He is currently a Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. He has authored or co-authored more than 100 technical papers in top journals/conferences, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, International Journal of Computer Vision, IEEE TRANSACTIONS ON IMAGE PROCESSING, CVPR, and ICCV. He holds more than 20 patents. His research interests include video/image analysis, computer vision, and video/image processing applications.

CHINE INTELLIGENCE, International Journal of Computer Vision, IEEE TRANSACTIONS ON IMAGE PROCESSING, CVPR, and ICCV. He holds more than 20 patents. His research interests include video/image analysis, computer vision, and video/image processing applications.



**Shuai Wan** (Member, IEEE) received the B.E. degree in Telecommunication Engineering and the M.E. degree in Communication and Information System from Xidian University, Xi'an, China, in 2001 and 2004, respectively, and obtained the Ph.D. in Electronic Engineering from Queen Mary, University of London in 2007. She is currently a Professor at Northwestern Polytechnical University in Xi'an, China. Previously, she served as a Professor at the Royal Melbourne Institute of Technology in Melbourne, Australia, from 2016 to 2025. Her research

interests include scalable/multiview video coding, video quality assessment and hyperspectral image compression.



**Fuzheng Yang** (Member, IEEE) received the B.E. degree in Telecommunication Engineering, the M.E. degree and the Ph.D. in Communication and Information System from Xidian University, Xi'an, China, in 2000, 2003 and 2005, respectively. He became a lecturer and an Associate Professor in Xidian University in 2005 and 2006, respectively. He has been a professor of communications engineering with Xidian University since 2012. He is also an Adjunct Professor of School of Engineering in RMIT University. During 2006–2007, he served

as a visiting scholar and postdoctoral researcher in Department of Electronic Engineering in Queen Mary, University of London. His research interests include video quality assessment, video coding and multimedia communication.



**Junhui Hou** (Senior Member, IEEE) is a Professor with the Department of Computer Science, City University of Hong Kong. His research interests include multidimensional visual computing, such as light field, hyperspectral, geometry, and event data. He received the Early Career Award from the Hong Kong Research Grants Council in 2018, IEEE Multimedia Rising Star Award in 2023, the Excellent Young Scientists Fund from NSFC in 2024, and the IEEE TIP Best Paper Award in 2025. He is serving as a Senior Area Editor for IEEE TIP and

an Associate Editor for IEEE TVCG and TMM. He served as an Associate Editor for IEEE TIP and TCSVT.