REFINE: Super-efficient 3D Gaussian Splatting Pruning via Rendering-Free Primitive Importance

Zhang Chen[1], Shuai Wan[1], Mengting Yu[1], Fuzheng Yang[2], and Junhui Hou[3]

[1] School of Electronics and Information, Northwestern Polytechnical University
[2] School of Telecommunication Engineering, Xidian University
[3] Department of Computer Science, City University of Hong Kong
https://github.com/ZhangChen2022/REFINE

Abstract. Existing pruning methods for 3D Gaussian splatting (3DGS) suffer from either severe quality degradation or prohibitive computational overhead. In this paper, we propose REFINE, a highly accelerated 3DGS pruning framework centered on a novel rendering-free primitive importance metric. Our approach leverages an analytically approximated, rendering-aware Hessian field to quantify the expected perceptual error induced by the removal of individual primitives. By modeling the joint modulation of visibility, projection geometry and the content adaptive hyperparameter, we entirely bypass costly forward rendering passes and derive an anisotropic perceptual weight field that serves as a high-fidelity proxy for primitive importance. Extensive experiments across multiple benchmark datasets demonstrate that REFINE maintains highly competitive rendering quality while achieving an unprecedented $3,000\times$ reduction in pruning-related computational complexity compared to state-of-the-art pruning methods.

Keywords: 3D Gaussian Splatting · Pruning · Primitive Importance · Rendering-free · Efficiency

1 Introduction

3D Gaussian Splatting (3DGS) [21] has emerged as a revolutionary representation for novel view synthesis (NVS) [3,9]. By explicitly representing scenes using 3D Gaussian primitives and utilizing an efficient tile-based rasterizer, 3DGS achieves real-time rendering rates exceeding 100 FPS at 1080p resolution [5,6].

Despite its impressive rendering performance, the explicit representation of 3DGS is notoriously storage-heavy [4,8,10]. To accurately capture high-frequency details and complex geometry, adaptive density control often generates millions of redundant Gaussian primitives for a single scene [5,12]. Since each primitive requires 59 parameters, storage requirements frequently reach the gigabyte (GB) level [19,32]. This massive redundancy severely limits the practical utility of 3DGS in resource-constrained environments, such as VR/AR headsets and mobile phones, and creates significant bottlenecks for network streaming [23,32–34].

Consequently, pruning redundant primitives while maintaining rendering quality has become a critical necessity [11]. Existing 3DGS pruning methods generally fall into two categories, both of which face an irreconcilable contradiction between computational efficiency and perceptual accuracy. The first category comprises parameter-based methods (e.g., LightGaussian [11]), which evaluate primitive importance directly in the parameter space. While computationally efficient, these methods ignore the physical non-linear mapping of the rendering pipeline, often leading to the erroneous pruning of critical high-frequency primitives and resulting in blurred rendering [39]. The second category includes render-aware methods (e.g., PUP 3D-GS [17]). While these preserve perceptual accuracy, they require forward rendering to accumulate sensitivity scores, trapping them in expensive rasterization loops and resulting in extremely high processing times.

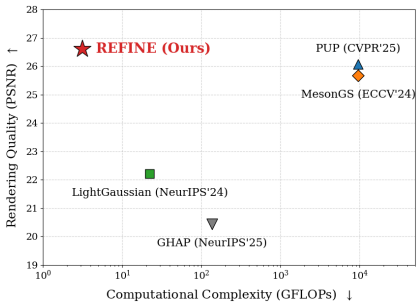


Fig. 1: Quality vs. Efficiency Trade-off on MipNeRF360 (Ratio = 0.5).

To address the challenge, we propose REFINE, a completely rendering-free, post-processing pruning framework that achieves state-of-the-art rendering quality with *exceptional* efficiency. To resolve the inconsistency between parameter space heuristics and image space visual degradation, we formalize primitive importance assessment through an analytically approximated Hessian field. Specifically, we quantify the expected visual error induced by primitive removal by decoupling the parameter gradients into two physically interpretable components: view-dependent visibility and geometric projection. Furthermore, we design the content-adaptive hyperparameter that dynamically calibrate the sensitivity of different physical attributes. By integrating these components, REFINE successfully obtains an importance score entirely without forward rendering. Extensive experiments demonstrate that our approach matches the visual fidelity of rendering-based methods while reducing the computational overhead by orders of magnitude, as shown in Fig. 1.

2 Related Work

Parameter-space Post-processing Pruning. The first category consists of methods that operate directly on the explicit geometric or appearance parameters of Gaussians. For instance, LightGaussian [11] evaluates the importance of a Gaussian primitive based on empirical scores (typically related to opacity and volume) and directly removes the lowest-scoring ones. Lee et al. [26] introduce a learnable mask parameter that considers both scale and opacity together, avoiding fixed thresholds to enable end-to-end learning of pruning decisions. OMG [25] extends significance scoring by factoring in Local Distinctiveness, which evaluates a Gaussian's importance by comparing its appearance features to its k-

nearest neighbors. Alternatively, GHAP [37] approaches compaction from the perspective of optimal transport, formulating it as a global reduction problem of Gaussian Mixture Models to mathematically merge redundant primitives into new representative centers. While these parameter-space methods are computationally efficient, they face inherent limitations in preserving rendering fidelity.

Rendering-aware Post-processing Pruning. The second category comprises render-aware pruning methods, which aim to link primitive removal directly to rendering error. For example, EAGLES [15] introduces a criterion to identify inefficient Gaussians by calculating their influence at a specific pixel based on alpha blending and transmittance values. MesonGS [39] proposes a render-aware metric that combines view-independent parameters with a view-dependent score obtained via forward rendering passes to accumulate physical pixel contributions. Going a step further into optimization quality, methods like PUP 3D-GS [17] utilize the Fisher approximation of the Hessian matrix to calculate the sensitivity of reconstruction error to each Gaussian spatial parameter. Speedy-Splat [16] reparameterizes this Hessian approximation to further reduce storage requirements during the pruning process. Additionally, Trimming [2] and ELMGS [1] propose Gradient-aware Pruning, which uses both opacity and gradient signals to prune inefficient Gaussians. Although these methods address the perceptual accuracy issues of parameter-space approaches, their reliance on the rendering pipeline introduces significant overhead. These render-aware methods typically require forward rendering, and in some cases backpropagation, resulting in extremely high computational complexity and processing time.

3DGS Compression. To address the massive memory demands of 3DGS [40], recent compression methods extend beyond pruning into parameter and restructuring compression [7]. Parameter compression minimizes storage via attribute quantization (e.g., CompGS [30], Niedermayr et al. [33]) and entropy coding (e.g., HAC [8]). Alternatively, restructuring methods fundamentally modify the 3DGS architecture for compactness, employing sparse anchors (e.g., Scaffold-GS [31]), neural MLPs (e.g., EAGLES [15]), or geometric structures like Octrees (e.g., Octree-GS [36]).

In summary, current post-processing pruning methods present a forced choice between the inaccuracy of parameter-space heuristics and the computational burden of image-space rendering evaluation. Our proposed REFINE bridges this gap by formulating primitive importance assessment directly on the parameter manifold, achieving the accuracy of render-aware methods with the efficiency of heuristic rules.

3 Proposed Method

Our goal is to quantify the visual importance of each Gaussian primitive *without* additional, costly forward rendering, which is then used to guide pruning. To achieve this, we first analyze the parameter space Hessian matrix to establish an importance weight field (Section 3.1). Next, we analytically model these Hessian

weights and calculate primitive-wise importance scores (Section 3.2). Finally, we execute highly efficient pruning using the importance scores (Section 3.3).

3.1 Preliminary

3D Gaussian Splatting. 3DGS [21] is a point-based NVS technique that uses 3D Gaussians to model the scene. Formally, we represent a scene as a set of N Gaussian primitives and define the set as:

$$\mathcal{G} = \{G_i = (\mu_i, s_i, q_i, c_i, \alpha_i)\}_{i=1}^{N}, \tag{1}$$

where $\mu_i \in \mathbb{R}^3$ denotes the center position (mean), $s_i \in \mathbb{R}^3$ represents the scaling factors, and $q_i \in \mathbb{R}^4$ is the rotation quaternion. To capture view-dependent appearance, $c_i \in \mathbb{R}^{3\times 16}$ denotes the Spherical Harmonic (SH) coefficients, while $\alpha_i \in \mathbb{R}$ represents the opacity of the primitive [20, 27].

Given a camera pose, a differentiable rasterizer renders a 2D image of resolution $h \times w$, denoted as $\mathcal{I} \in \mathbb{R}^{h\times w\times 3}$, by projecting the 3D Gaussian primitives. Denote by $\mathbf{I} \in \mathbb{R}^{h\times w\times 3}$ the flattened rendered image. We formalize the 3DGS rendering process as a differentiable mapping $R(\cdot) : \mathcal{M} \to \mathcal{I}$, such that $\mathbf{I} = R(\mathcal{G})$. This operator represents a complex, non-linear transformation from the high-dimensional Gaussian parameter space $\mathcal{M}$ to the rendered 2D image space $\mathcal{I}$.

Importance Modeling via Scaled Fisher Information. When the primitives in $\mathcal{G}$ are subject to a perturbation $\Delta\mathcal{G}$ (via primitive pruning), we analytically approximate the resulting image space variation using a first-order Taylor expansion:

$$\Delta\mathbf{I} \approx \mathbf{J}_R \cdot \Delta\mathcal{G}, \tag{2}$$

where $\mathbf{J}_R = \frac{\partial R}{\partial \mathcal{G}}$ denotes the Jacobian matrix of the rendering function $R(\cdot)$ with respect to the primitives $\mathcal{G}$. Given that quality assessment in image space is commonly characterized by the L_2 norm, we project this metric into the parameter space via $Q(\cdot)$

$$Q(\Delta\mathcal{G}) = \|\Delta\mathbf{I}\|_2^2 \approx \Delta\mathcal{G}^\top (\underbrace{\mathbf{J}_R^\top \mathbf{J}_R}_{\mathbf{H}}) \Delta\mathcal{G}, \tag{3}$$

where Q quantifies the image quality degradation caused by pruning, directly reflecting the importance of the removed primitives. The term $\mathbf{H} = \mathbf{J}_R^\top \mathbf{J}_R$ constitutes the Gauss-Newton approximation of the Hessian matrix [13], defining a metric tensor that characterizes the importance of the parameter space. This formulation possesses a rigorous statistical interpretation: assuming the observed image $\mathbf{I}_{obs}$ is corrupted by independent and identically distributed Gaussian white noise $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, i.e., $\mathbf{I}_{obs} = R(\mathcal{G}) + \epsilon$, where σ is the standard deviation of the noise, the negative log-likelihood of the rendering is given by [14]:

$$-\log \phi(\mathbf{I}_{obs}|\mathcal{G}) \propto \frac{1}{2\sigma^2}\|\mathbf{I}_{obs} - R(\mathcal{G})\|_2^2, \tag{4}$$

where ϕ denotes the likelihood function. The Fisher Information Matrix (FIM) $\mathbf{F}$, defined as the covariance of the gradient of the log-likelihood [14,29], simplifies under this Gaussian assumption to:

$$\mathbf{F} = \mathbb{E}\left[\left(\frac{\partial \log \phi}{\partial \mathcal{G}}\right)\left(\frac{\partial \log \phi}{\partial \mathcal{G}}\right)^{\top}\right] \approx \frac{1}{\sigma^2}\mathbf{J}_R^{\top}\mathbf{J}_R. \tag{5}$$

Consequently, $\mathbf{H}$ in Eq. (3) is essentially a scaled FIM. It directly quantifies the information contribution of each primitive to the final rendered image.

3.2 Rendering-free Primitive Importance Metric

The 3D Gaussian representation of a scene $\mathcal{G}$ typically contains over a million primitives, making the calculation and storage of the dense matrix $\mathbf{H}$ infeasible. To this end, we introduce two structural assumptions: (**1**) *Primitive Independence*: ignoring interaction between different Gaussian primitives; and (**2**) *Attribute Orthogonality*: ignoring second-order couplings between different attributes within the same primitive. See the experiments in Section 4.3 for the rationality verification.

Based on these assumptions, $\mathbf{H}$ can be further approximated as a diagonal matrix, i.e., $\mathbf{W} = \mathrm{diag}(\mathbf{H})$ with the i-th diagonal element being w_i. By treating the perturbation $\Delta\mathcal{G}$ as the removal of a primitive G_i and expanding this variation across its different attributes, we can express the importance of the i-th primitive as:

$$D(G_i) = \sum_{k \in \{gem, col, opa\}} w_i^k \cdot \tilde{G}_i^k, \tag{6}$$

where $\tilde{G}_i^k$ denotes the k-th attribute subset (geometry area *gem*, color *col*, and opacity *opa*) of the i-th Gaussian primitive G_i, and w_i^k represents the corresponding Hessian weight of G_i^k.

Recall that w_i^k is essentially the squared magnitude of the corresponding column vector in the rendering Jacobian matrix. Consequently, w_i^k can be calculated by the expected squared L_2-norm of the image error gradient induced by this parameter across a set of sampled camera viewpoints $\mathcal{V}$:

$$w_i^k = \mathbb{E}_{\mathbf{v} \in \mathcal{V}}\left[\left\|\frac{\partial R^{\mathbf{v}}}{\partial G_i^k}\right\|_2^2\right], \tag{7}$$

where $\mathbb{E}[\cdot]$ denotes the mathematical expectation. Intuitively, the rasterization pipeline of 3DGS consists of two sequential physical operations: mapping a 3D Gaussian onto the 2D space, and alpha-blending it along the ray. The Hessian field/matrix effectively maps the rendering processes, assigning high weights to proximal, highly salient primitives, and low weights to distant or occluded ones. Inspired by this forward rendering mechanism, we approximately decompose $\frac{\partial R^{\mathbf{v}}}{\partial G_i^k}$

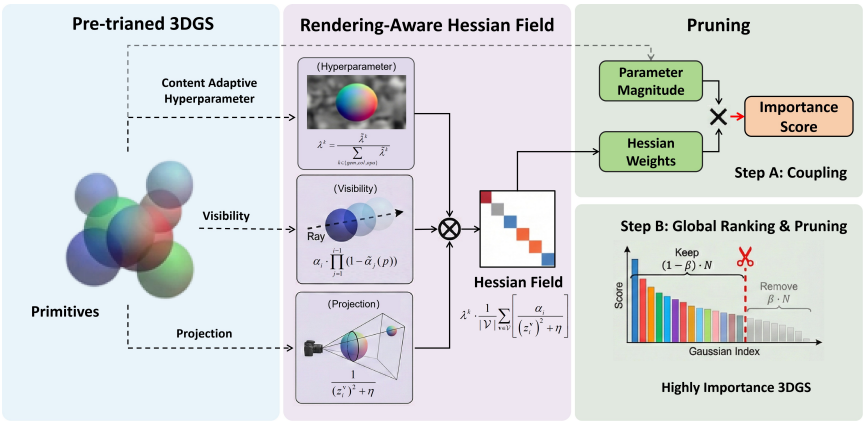


Fig. 2: Overview of our **REFINE**, a super-efficient and effective pruning method for 3DGS. **(Middle)** The rendering-aware Hessian field is computed by decomposing into visibility and projection *without* performing feedforward rendering. **(Right)** The importance score is computed by coupling parameter magnitudes with Hessian weights. And primitives are globally ranked by this score and pruned by the ratio value, achieving super-efficient pruning without any rendering passes.

into two corresponding independent components: view-dependent visibility $V_i^{\mathbf{v}}$ and geometric projection $P_i^{\mathbf{v}}$:

$$\frac{\partial R^{\mathbf{v}}}{\partial G_i^k} \approx V_i^{\mathbf{v}} \cdot P_i^{\mathbf{v}}, \tag{8}$$

where $V_i^{\mathbf{v}}$ describes how the attributes are influenced by the alpha-blending; and $P_i^{\mathbf{v}}$ defines how the attributes are effected by the projection from $\mathcal{M}$ to $\mathcal{I}$. In the following, we will explicitly model these two items.

Modeling of Visibility. Rasterization in 3DGS is a view-dependent, sort-based blending process [2, 15, 24]. The final color of each pixel p is derived by alpha-blending the ordered set of overlapping Gaussians $\mathcal{N}$:

$$C(p) = \sum_{i \in \mathcal{N}} \left[\tilde{c}_i \cdot \tilde{\alpha}_i(p) \cdot T_i(p) \right], \tag{9}$$

where $\tilde{\alpha}_i(p)$ is the evaluated density at pixel p multiplied by the standalone opacity α_i, and $T_i(p) = \prod_{j=1}^{i-1}(1-\tilde{\alpha}_j(p))$ denotes the accumulated transmittance. When the i-th Gaussian primitive is pruned, its gradient contribution to the final image quality is directly limited by T_i and α_i. Ignoring second-order occlusion differentials, the visibility $V_i^{\mathbf{v}}$ can be modeled as:

$$V_i^{\mathbf{v}} \approx \alpha_i \cdot T_i(p). \tag{10}$$

However, calculating the exact accumulated transmittance requires complex depth sorting and ray marching, which is equivalent to a forward rendering

pass. So we introduce a conservative zero-occlusion approximation (i.e., assuming $T_i \approx 1.0$). This models the worst-case scenario where the primitive is completely unoccluded, allowing its standalone opacity α_i to exclusively govern its visibility score.

Modeling of Geometric Projection $P_i^{\mathbf{v}}$. Beyond view-dependent visibility, the geometric mapping transformation changes the visual impact of the primitive. To quantify this influence, we formulate its perspective projection onto the 2D image space $\mathcal{I}$. Define the projection function $\Pi(\cdot) : \mathbb{R}^3 \to \mathbb{R}^2$ as [25,26,28]:

$$\Pi(x_i^{\mathbf{v}}, y_i^{\mathbf{v}}, z_i^{\mathbf{v}}) = \left(f_x^{\mathbf{v}} \frac{x_i^{\mathbf{v}}}{z_i^{\mathbf{v}}} + u_x^{\mathbf{v}}, f_y^{\mathbf{v}} \frac{y_i^{\mathbf{v}}}{z_i^{\mathbf{v}}} + u_y^{\mathbf{v}} \right), \tag{11}$$

where $f^{\mathbf{v}}$ represents the focal length of the camera at viewpoint $\mathbf{v}$, which is specifically parameterized as $f_x^{\mathbf{v}}$ and $f_y^{\mathbf{v}}$ along the x and y axes of the image plane, respectively; The tuple $(u_x^{\mathbf{v}}, u_y^{\mathbf{v}})$ represents the principal point coordinates; Additionally, the vector $(x_i^{\mathbf{v}}, y_i^{\mathbf{v}}, z_i^{\mathbf{v}})^\top$ denotes the 3D center coordinate μ_i of the i-th Gaussian primitive, transformed into the local camera coordinate system of viewpoint $\mathbf{v}$.

Following the local affine approximation introduced in Elliptical Weighted Average (EWA) volume splatting [42], the perspective projection can be locally linearized using its Jacobian matrix $\mathbf{J}_\Pi \in \mathbb{R}^{2\times 3}$. The image space visual displacement, also induced by a 3D spatial perturbation, is computed by this Jacobian. Consequently, the projection effect $P_i^{\mathbf{v}}$ is proportional to the energy of the projection matrix, which can be measured by

$$P_i^{\mathbf{v}} \approx \|\mathbf{J}_\Pi\|_F^2 = \mathrm{Tr}(\mathbf{J}_\Pi^\top \mathbf{J}_\Pi), \tag{12}$$

where $\mathrm{Tr}(\cdot)$ denotes the trace operator. Assuming a scaled orthographic projection as a local approximation of the perspective projection—which effectively ignores the eccentric view-dependent terms (i.e., assuming $x_i^{\mathbf{v}} \approx y_i^{\mathbf{v}} \approx 0$ locally) and assumes an isotropic focal length $f_x^{\mathbf{v}} \approx f_y^{\mathbf{v}} \approx f^{\mathbf{v}}$—the Jacobian energy simplifies to:

$$\mathrm{Tr}(\mathbf{J}_\Pi^\top \mathbf{J}_\Pi) \approx 2 \left(\frac{f^{\mathbf{v}}}{z_i^{\mathbf{v}}} \right)^2. \tag{13}$$

Omitting the constant factor $2(f^{\mathbf{v}})^2$, the geometric influence fundamentally scales with the inverse square of the depth. To ensure numerical stability and prevent singularities when a primitive is exceptionally close to the camera plane, we introduce a stability constant $\eta = 0.05$, yielding:

$$P_i^{\mathbf{v}} \propto \frac{1}{(z_i^{\mathbf{v}})^2 + \eta}. \tag{14}$$

where $z_i^{\mathbf{v}}$ is the depth of the primitive relative to camera $\mathbf{v}$. Physically, Eq. (14) demonstrates that the importance of a primitive is highly depth-dependent: primitives closer to the camera induce a larger visual impact when pruned, whereas distant ones exert smaller influence on the image space.

By substituting Eqs. (10) and (14) into Eq. (8), and introducing the hyperparameter λ^k to absorb the omitted constants and calibrate the varying sensitivities of attributes, we have

$$\frac{\partial R^{\mathbf{v}}}{\partial \tilde{G}_i^k} \approx \lambda^k \cdot \frac{\alpha_i}{(z_i^{\mathbf{v}})^2 + \eta}. \tag{15}$$

Content-Adaptive Hyperparameters $\{\lambda^k\}$. The relative sensitivities of attributes vary between scenes. For example, opacity is critical for semi-transparent elements (e.g., foliage), while geometry dominates in rigid structures. A static λ^k is hard to capture this variation. Therefore, we propose a content-adaptive mechanism to dynamically determine attribute sensitivity λ^k.

Specifically, we extract three statistical features from the Gaussian parameters to characterize the scene content: color variance F_{col}, opacity ambiguity F_{opa}, and scale anisotropy F_{gem}:

$$F_{col} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \bar{Y})^2, F_{opa} = \frac{1}{N}\sum_{i=1}^{N}\alpha_i(1-\alpha_i), F_{gem} = \frac{1}{N}\sum_{i=1}^{N}\frac{\max(s_i)}{\min(s_i)+\eta}. \tag{16}$$

where Y_i is the perceptual luma computed by c_i, and $\bar{Y}$ is the mean luma across the scene. Physically, F_{col} quantifies the richness of high-frequency textures; F_{opa} is a concave function to evaluate the proportion of semi-transparent regions; and F_{gem} measures the average structural stretch of the primitives. And the content-adaptive hyperparameter λ^k is obtained:

$$\lambda^k = \frac{\tilde{\lambda}^k}{\sum_{k\in\{gem,col,opa\}} \tilde{\lambda}^k}, \tag{17}$$

where the unnormalized $\tilde{\lambda}^k$ are calculated as:

$$\tilde{\lambda}^{gem} = \frac{\ln(F_{gem}+\eta)}{\mathbb{E}[\ln(F_{gem})]}, \quad \tilde{\lambda}^{col} = \frac{F_{col}}{\mathbb{E}[F_{col}]}, \quad \tilde{\lambda}^{opa} = \frac{F_{opa}}{\mathbb{E}[F_{opa}]}. \tag{18}$$

We calibrate these features using their expectations $\mathbb{E}[\cdot]$ to balance their magnitude discrepancies, applying logarithmic smoothing to F_{gem} to suppress extreme stretches.

Finally, the rendering-aware Hessian weight w_i^k in Eq. (7) is derived as:

$$w_i^k = \lambda^k \cdot \frac{1}{|\mathcal{V}|}\sum_{\mathbf{v}\in\mathcal{V}}\left[\frac{\alpha_i}{(z_i^{\mathbf{v}})^2+\eta}\right]. \tag{19}$$

3.3 Super-efficient Pruning Process

During the pruning execution, for a pre-trained 3DGS scene containing N primitives, we couple the estimated Hessian field with parameter values following our prior derivations to calculate the final rendering-free importance score for each

individual Gaussian primitive. Subsequently, all primitives are globally ranked based on these evaluated scores. Given a target pruning ratio $\beta \in (0, 1)$, we permanently remove the bottom $\beta \cdot N$ primitives associated with the lowest importance scores, as shown in Fig. 2. By entirely bypassing actual rendering passes, our method achieves a highly efficient pruning process.

4 Experiments

4.1 Experiment Settings

Datasets. We evaluated our method on the same challenging real-world scenes as 3D-GS. We used all nine scenes from the Mip-NeRF 360 dataset [3], which contains five outdoor and four indoor scenes, each featuring complex central objects or viewing areas and detailed backgrounds. Additionally, two outdoor scenes, *truck* and *train*, were taken from the Tanks & Temples dataset [22], and two indoor scenes, *drjohnson* and *playroom*, were taken from the Deep Blending dataset [18]. For consistency, we used the COLMAP camera pose estimates provided in the original 3D-GS creator's pre-experiments [21].

Implementation Details. Our REFINE is a plug-and-play, purely post-processing pruning pipeline that can be seamlessly applied to any pre-trained 3DGS model [21]. To highlight the effectiveness of the evaluation metric itself, all comparisons regarding pruning effects were conducted under a *zero-shot* condition. That is, after removing primitives based on the algorithm, strictly no subsequent fine-tuning rendering optimization was performed.

Baseline Methods. We compared our REFINE with four representative and state-of-the-art 3DGS pruning methods, including GHAP [37], LightGaussian [11], MesonGS [39], and PUP 3D-GS [17].

4.2 Experimental Results

We comprehensively tested the performance of pruning various methods in terms of rendering fidelity and computational complexity across pruning ratios from 10% to 70%.

Comparisons of Rendering Fidelity. We adopted three widely accepted image quality evaluation metrics: Peak Signal-to-Noise Ratio (PSNR, higher is better), Structural Similarity Index (SSIM, higher is better), and Learned Perceptual Image Patch Similarity (LPIPS, lower is better) [41]. From Table 1, among the methods evaluated, there are significant differences in performance. LightGaussian, relying on heuristics, exhibits fragile robustness at high pruning rates; for instance, on the MipNeRF 360 dataset, its PSNR drops sharply to 22.21 when the pruning ratio reaches 50%. This validates our assertion that only relying on the value of parameters leads to the erroneous removal of critical high-frequency primitives. Meanwhile, GHAP yields a lower PSNR without fine-tuning, as its optimal transport formulation is designed for iterative cluster reconstruction.

Table 1: Quantitative Comparison of Different Pruning Methods. The best and second-best are highlighted in red and blue, respectively.

Dataset	Method	Ratio = 0.1			Ratio = 0.3			Ratio = 0.5			Ratio = 0.7		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MipNeRF 360	original 3D GS [21]	27.35	0.814	0.217	27.35	0.814	0.217	27.35	0.814	0.217	27.35	0.814	0.217
	GHAP [37]	20.65	0.607	0.445	20.62	0.594	0.455	20.44	0.571	0.470	19.93	0.530	0.494
	LightGaussian [11]	25.48	0.793	0.234	23.61	0.775	0.246	22.21	0.753	0.260	19.54	0.674	0.315
	MesonGS [39]	25.87	0.800	0.227	25.86	0.800	0.228	25.68	0.795	0.232	24.35	0.760	0.260
	PUP [17]	27.34	0.814	0.217	27.24	0.812	0.219	26.07	0.790	0.241	25.12	0.786	0.259
	REFINE (**Ours**)	27.34	0.814	0.217	27.29	0.813	0.218	26.61	0.800	0.233	24.43	0.745	0.285
Tanks & Temples	original 3D GS [21]	23.39	0.842	0.183	23.39	0.842	0.183	23.39	0.842	0.183	23.39	0.842	0.183
	GHAP [37]	16.84	0.592	0.467	17.00	0.587	0.473	16.97	0.576	0.482	16.82	0.553	0.501
	LightGaussian [11]	22.78	0.838	0.186	22.62	0.836	0.187	21.37	0.816	0.202	18.09	0.729	0.271
	MesonGS [39]	22.08	0.817	0.202	22.05	0.816	0.203	21.98	0.812	0.207	21.41	0.786	0.232
	PUP [17]	23.39	0.842	0.183	23.38	0.841	0.184	23.18	0.832	0.193	21.39	0.787	0.233
	REFINE (**Ours**)	23.38	0.841	0.183	23.33	0.839	0.185	22.97	0.828	0.196	20.98	0.776	0.243
Deep Blending	original 3D GS [21]	29.52	0.903	0.242	29.52	0.903	0.242	29.52	0.903	0.242	29.52	0.903	0.242
	GHAP [37]	23.12	0.768	0.456	23.18	0.767	0.459	23.16	0.763	0.464	22.92	0.752	0.475
	LightGaussian [11]	28.91	0.899	0.245	28.79	0.896	0.247	27.14	0.874	0.260	21.61	0.778	0.323
	MesonGS [39]	29.08	0.900	0.246	29.06	0.899	0.246	29.00	0.899	0.248	28.49	0.890	0.261
	PUP [17]	29.52	0.903	0.242	29.51	0.903	0.243	29.28	0.899	0.248	29.14	0.894	0.256
	REFINE (**Ours**)	29.52	0.903	0.242	29.49	0.903	0.243	29.28	0.899	0.249	27.93	0.871	0.282

Table 2: Computational Efficiency Comparison of Different Pruning Methods. The best and second-best are highlighted in red and blue, respectively.

Dataset	Method	Ratio = 0.1		Ratio = 0.3		Ratio = 0.5		Ratio = 0.7	
		Time (s) ↓	GFLOPs↓	Time (s)↓	GFLOPs↓	Time (s)↓	GFLOPs↓	Time (s)↓	GFLOPs↓
MipNeRF 360	GHAP [37]	14.56	244.30	12.42	190.26	10.70	136.23	8.96	82.19
	LightGaussian [11]	48.99	22.29	44.80	22.29	41.43	22.29	6.75	22.29
	MesonGS [39]	17.74	9539.19	13.59	9539.19	9.95	9539.19	33.70	9539.19
	PUP [17]	44.01	9582.81	40.28	9582.81	37.02	9582.81	38.15	9582.81
	REFINE (**Ours**)	5.90	3.14	5.46	3.14	3.85	3.14	2.55	3.14
Tanks & Temples	GHAP [37]	7.67	129.59	6.62	100.93	5.71	72.26	4.84	43.60
	LightGaussian [11]	21.99	16.59	20.27	16.59	18.47	16.59	3.85	16.59
	MesonGS [39]	9.36	7104.05	7.46	7104.05	5.58	7104.05	14.11	7104.05
	PUP [17]	18.90	7136.18	17.75	7136.18	15.92	7136.18	16.75	7136.18
	REFINE (**Ours**)	3.34	1.67	2.70	1.67	1.99	1.67	1.36	1.67
Deep Blending	GHAP [37]	13.04	216.19	10.93	168.37	9.37	120.55	7.88	72.74
	LightGaussian [11]	24.41	25.73	21.09	25.73	18.00	25.73	6.15	25.73
	MesonGS [39]	15.09	11017.66	11.94	11017.66	9.26	11017.66	12.83	11017.66
	PUP [17]	19.94	11067.59	16.92	11067.59	13.84	11067.59	15.15	11067.59
	REFINE (**Ours**)	5.37	1.77	4.31	1.77	3.33	1.77	1.42	1.77

Across the 10% to 70% pruning, our REFINE demonstrates exceptional efficacy. At a 10% removal ratio on MipNeRF 360, REFINE achieves a PSNR of 27.34, an SSIM of 0.814, and an LPIPS of 0.217, performing fully on par with the state-of-the-art rendering-based method, PUP. Notably, on the Tanks & Temples dataset, REFINE remains highly competitive with PUP across all pruning ratios (e.g., 22.97 vs. 23.18 at 50% pruning). Even when pushed to a 50% pruning ratio on MipNeRF 360, REFINE maintains strong visual fidelity (26.61 vs. 26.07). To achieve extreme rendering-free speed, REFINE explicitly approximates and omits second-order coupling effects. However, we note a performance drop at an extreme pruning ratio (Ratio = 0.7) on the Deep Blending dataset (e.g., PSNR drops to 27.93). This behavior is physically intuitive. Deep Blending scenes are characterized by extremely dense primitive distributions and severe occlusions. At extreme pruning ratios, the cross-primitive coupling effect, which is intentionally omitted in our rendering-free analytical formulation for the sake of super-efficiency, becomes non-negligible.

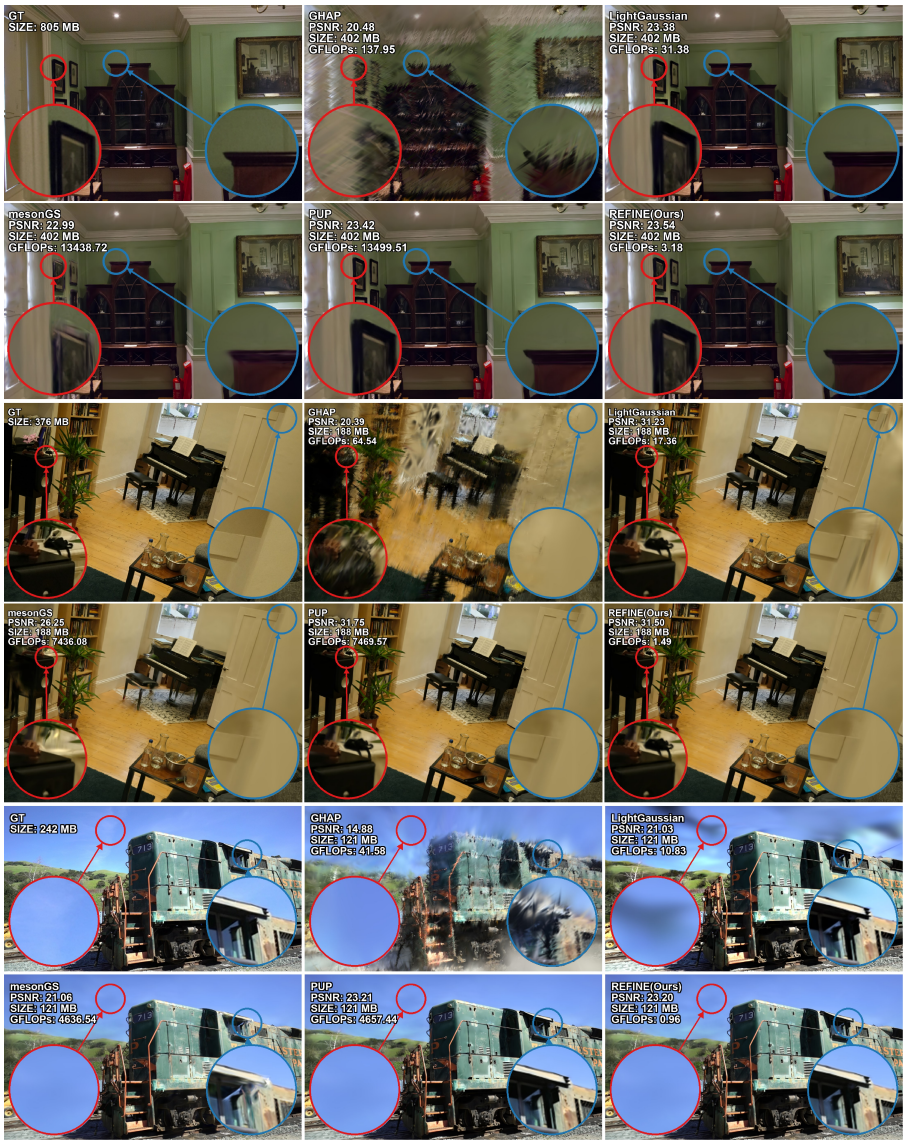


Fig. 3: Visual comparison after 50% pruning using REFINE and other methods. Top: *drjohnson* from Mip-NeRF 360. Middle: *room* from Mip-NeRF 360. Bottom: *train* from Tanks & Temples. Zooming in for details.

Comparisons of Computational Complexity. Computational complexity is the key to applying pruning algorithms in practice. We recorded the total processing time (in seconds) and the total computational cost (GFLOPs) required to evaluate primitive importance for each method [35] [38]. As reported in Table 2, rendering-based methods (e.g., PUP and MesonGS) incur massive computational burdens, demanding up to ∼11,000 GFLOPs to evaluate a single scene. On the contrary, REFINE's computational cost remains remarkably low, requiring merely 3.14, 1.67, and 1.77 GFLOPs on the MipNeRF 360, Tanks & Temples, and Deep Blending datasets, respectively. This represents a staggering reduction in computational cost by over 3000× compared to SOTA rendering-based methods. Correspondingly, REFINE achieves the fastest processing speed across all scenarios, completing the entire pruning execution in as little as 1.36 seconds. This efficiency stems from our closed-form solution Eq. (19). By bypassing rendering entirely, REFINE relies solely on $\mathcal{O}(N)$ parameter space algebraic operations. Consequently, it serves as an super-efficient, plug-and-play module ideal for resource-constrained devices, visual comparison as shown in Fig. 3. Note that while GFLOPs are reduced by $> 3000\times$, the latency reduction is relatively smaller ($< 20\times$). This is because rendering-based methods fully saturate the highly parallelized GPU during rasterization, masking their massive computational burden, whereas REFINE achieves its speed with minimal hardware utilization. Furthermore, the total latency includes fixed I/O overhead (e.g., sorting and saving primitives). A detailed FLOPs breakdown per primitive is provided in the Supplementary Material.

4.3 Ablation Study

To better understand our proposed REFINE method, we conducted thorough ablation studies at a 50% pruning ratio. Table 3 reports the average performance across the evaluated datasets (*bicycle*, *bonsai*, and *kitchen*) for different attribute configurations. Fig. 4 presents a visual comparison of the ablation study.

Effectiveness of Components. We first evaluated extreme configurations where primitive importance is dictated solely by a single component (e.g., *w/o* $V_i^{\mathbf{v}}$ relying exclusively on geometric projection, or *w/o* $P_i^{\mathbf{v}}$ relying purely on view-dependent visibility). Table 3 shows that single component strategies perform poorly (e.g., PSNR 27.24 for *w/o* $P_i^{\mathbf{v}}$), confirming

Table 3: Ablation study on attribute modulation at a 50% pruning ratio.

Conf.	PSNR ↑	SSIM ↑	LPIPS ↓
w/o $V_i^{\mathbf{v}}$	27.74	0.849	0.212
w/o $P_i^{\mathbf{v}}$	27.24	0.842	0.214
Equal λ^k	27.82	0.852	0.209
Ours	**28.35**	**0.862**	**0.198**

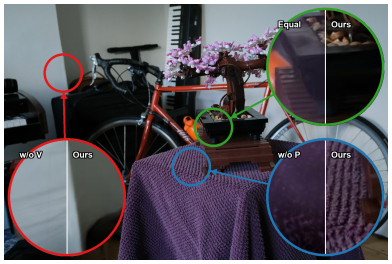


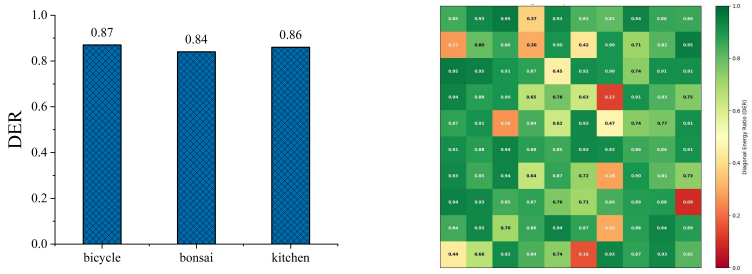
Fig. 4: Visual ablation study after 50% pruning.

the necessity of jointly modeling all components.

Superiority of the Content Adaptive Hyperparameter. A static weight distribution (*Equal* λ^k) yields a suboptimal mean PSNR of 27.82. By dynamically adjusting to scene statistics, our adaptive method (**Ours**) boosts this to 28.35.

Validation of Structured Approximation. We quantitatively verified the rationality of the two assumptions on primitive independence and attribute orthogonality in parameter space construction by analyzing the energy distribution of the Hessian matrix.

Verification of Primitive Independence: Although the 3DGS rendering process involves alpha blending induced occlusion coupling, we assume the Hessian matrix $\mathbf{H}$ has a significant block-diagonal structure. To verify this, we randomly selected 100 Gaussian primitives in evaluated datasets, used the 30-Nearest Neighbors to sample high-density local regions to construct a coupling test set, and calculated the exact Hessian matrix regarding the rendering loss. We used the Diagonal Energy Ratio (DER) as a quantitative metric $\text{DER} = \sum_{i=1}^{N} \|\mathbf{H}_{i,i}\|_F^2 \,/\, \|\mathbf{H}\|_F^2$ represents the autocorrelation block of the i-th Gaussian in the sampled Hessian matrix, and $\mathbf{H}$ represents the sampled Hessian matrix. A DER value closer to 1 indicates that energy is primarily distributed on the diagonal, implying lower coupling energy.



(a) Average DER of Hessian matrices **(b)** Details of DER on *bonsai*

Fig. 5: DER of Hessian matrices for different Gaussian primitives.

Experimental results are shown in Fig. 5 (a). Even in regions with dense spatial overlap, the DER remains as high as 86%. This means that nearly 90% of the energy is concentrated in the diagonal blocks, and gradient interactions between Gaussian primitives account for only 14%. The heatmap in Fig. 5 (b) further intuitively displays the DER energy values of 100 randomly sampled Gaussian primitives on *bonsai*. This finding provides strong statistical support for ignoring the off-diagonal blocks.

Verification of Attribute Orthogonality. Regarding the parameter space within a single Gaussian, we verified the orthogonality of its parameter subspaces. Com-

bining the specific implementation process, we categorized attributes into Geometry, Color, and Opacity. We similarly used the DER as the quantitative metric.

As shown in Table 4, the average DER reaches 0.79, indicating that the majority of the second-order optimization energy is highly concentrated on the diagonal blocks. Specifically, the *bicycle* scene exhibits the highest DER of 0.91, which perfectly aligns with our theoretical assumption that cross-primitive coupling effects can be safely decoupled. Notably, *kitchen* scene exhibits a lower DER (0.67). This is primarily because *kitchen* is a complex indoor environment characterized by dense, overlapping surfaces and severe occlusions. Consequently, a massive number of Gaussian primitives heavily overlap along the same rendering rays, leading to stronger gradient dependencies and a relatively lower diagonal energy concentration.

Table 4: Average DER of Hessian matrices across different scenes.

Sequences	Average DER
bicycle	0.91
bonsai	0.78
kitchen	0.67
Average	**0.79**

5 Conclusion and Discussion

In this paper, we presented REFINE, an super-efficient, rendering-free pruning framework for 3DGS. By analytically approximating a rendering-aware Hessian field to evaluate primitive importance, REFINE successfully breaks the trade-off between visual fidelity and computational cost, achieving unprecedented acceleration over existing rendering-based methods. Beyond post-training pruning, ours method also offers promising avenues for broader 3DGS optimization. It can be seamlessly integrated into rate-distortion optimization to guide efficient compression, or serve as a dynamic regularizer to accelerate the standard 3DGS training process.

Although our method achieves impressive performance, our primitive independence assumption may lead to quality degradation at *extreme* pruning ratios where cross-primitive coupling becomes severe. Future work could explore incorporating lightweight, localized re-rendering passes or low-rank off-diagonal approximations to mitigate this while preserving real-time efficiency.

References

1. Ali, M.S., Bae, S.H., Tartaglione, E.: ElmGS: Enhancing memory and computation scalability through compression for 3D Gaussian splatting. In: Winter Conf. Appl. Comput. Vis. pp. 2591–2600 (2025)
2. Ali, M.S., Qamar, M., Bae, S.H., Tartaglione, E.: Trimming the fat: Efficient compression of 3D Gaussian splats through pruning. In: BMVC (2024)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In: CVPR. pp. 5470–5479 (2022)

4. Chen, J., Li, Z., Cai, Y., Jiang, H., Qian, C., Kang, J., Gao, S., Zhao, H., Mao, T., Zhang, Y.: HAIF-GS: Hierarchical and induced flow-guided Gaussian splatting for dynamic scene. In: NeurIPS. vol. 38, pp. 125539–125563 (2026)
5. Chen, Y., Li, M., Wu, Q., Lin, W., Harandi, M., Cai, J.: PCGS: Progressive compression of 3D Gaussian splatting. In: AAAI. vol. 40, pp. 3111–3119 (2026)
6. Chen, Y., Wu, Q., Li, M., Lin, W., Harandi, M., Cai, J.: Fast feedforward 3D Gaussian splatting compression. In: ICLR. vol. 2025, pp. 74859–74872 (2025)
7. Chen, Y., Wu, Q., Li, M., Lin, W., Hou, J., Harandi, M., Cai, J.: Feedforward compression of static and streamable 3D Gaussian splatting. IEEE TCSVT (2026)
8. Chen, Y., Wu, Q., Lin, W., Harandi, M., Cai, J.: HAC: Hash-grid assisted context for 3D Gaussian splatting compression. In: ECCV. pp. 422–438 (2024)
9. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised NeRF: Fewer views and faster training for free. In: CVPR. pp. 12882–12891 (2022)
10. Duisterhof, B.P., Mandi, Z., Yao, Y., Liu, J.W., Seidenschwarz, J., Shou, M.Z., Ramanan, D., Song, S., Birchfield, S., Wen, B., Ichnowski, J.: DeformGS: Scene flow in highly deformable scenes for deformable object manipulation. In: International Workshop on the Algorithmic Foundations of Robotics (WAFR). pp. 263–282 (2024)
11. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. In: NeurIPS. vol. 37, pp. 140138–140158 (2024)
12. Fang, G., Wang, B.: Mini-splatting: Representing scenes with a constrained number of Gaussians. In: ECCV. pp. 165–181 (2024)
13. Foresee, F.D., Hagan, M.T.: Gauss-Newton approximation to Bayesian learning. In: Int. Conf. Neural Networks. vol. 3, pp. 1930–1935 (1997)
14. Fujita, K., Okada, K., Katahira, K.: The Fisher information matrix: A tutorial for calculation for decision making models. PsyArXiv preprint (2022)
15. Girish, S., Gupta, K., Shrivastava, A.: Eagles: Efficient accelerated 3D Gaussians with lightweight encodings. In: ECCV. pp. 54–71 (2024)
16. Hanson, A., Tu, A., Lin, G., Singla, V., Zwicker, M., Goldstein, T.: Speedy-splat: Fast 3D Gaussian splatting with sparse pixels and sparse primitives. In: CVPR. pp. 21537–21546 (2025)
17. Hanson, A., Tu, A., Singla, V., Jayawardhana, M., Zwicker, M., Goldstein, T.: PUP 3D-GS: Principled uncertainty pruning for 3D Gaussian splatting. In: CVPR. pp. 5949–5958 (2025)
18. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM TOG **37**(6), 1–15 (2018)
19. Huang, H., Huang, W., Yang, Q., Xu, Y., Li, Z.: A hierarchical compression technique for 3D Gaussian splatting compression. In: ICASSP. pp. 1–5 (2025)
20. Jiang, Y., Shen, Z., Wang, P., Su, Z., Hong, Y., Zhang, Y., Yu, J., Xu, L.: HiFi4G: High-fidelity human performance rendering via compact Gaussian splatting. In: CVPR. pp. 19734–19745 (2024)
21. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian splatting for real-time radiance field rendering. ACM TOG **42**(4), 139:1–139:14 (2023)
22. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and Temples: Benchmarking large-scale scene reconstruction. ACM TOG **36**(4), 1–13 (2017)
23. Kong, H., Yang, X., Wang, X.: Efficient Gaussian splatting for monocular dynamic scene rendering via sparse time-variant attribute modeling. In: AAAI. vol. 39, pp. 4374–4382 (2025)

24. Kwak, S., Kim, J., Jeong, J.Y., Cheong, W.S., Oh, J., Kim, M.: MoDec-GS: Global-to-local motion decomposition and temporal interval adjustment for compact dynamic 3D Gaussian splatting. In: CVPR. pp. 11338–11348 (2025)
25. Lee, J.C., Ko, J.H., Park, E.: Optimized minimal 3D Gaussian splatting. In: NeurIPS. vol. 38, pp. 135864–135888 (2026)
26. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3D Gaussian representation for radiance field. In: CVPR. pp. 21719–21728 (2024)
27. Lei, J., Weng, Y., Harley, A.W., Guibas, L., Daniilidis, K.: MoSca: Dynamic Gaussian fusion from casual videos via 4D motion scaffolds. In: CVPR. pp. 6165–6177 (2025)
28. Li, H., Liu, J., Sznaier, M., Camps, O.: 3D-HGS: 3D half-gaussian splatting. In: CVPR. pp. 10996–11005 (2025)
29. Liu, J., Yuan, H., Lu, X.M., Wang, X.: Quantum Fisher information matrix and multiparameter estimation. Journal of Physics A: Mathematical and Theoretical **53**(2), 023001 (2020)
30. Liu, X., Wu, X., Zhang, P., Wang, S., Li, Z., Kwong, S.: CompGS: Efficient 3D scene representation via compressed Gaussian splatting. In: ACM MM. pp. 2936–2944 (2024)
31. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering. In: CVPR. pp. 20654–20664 (2024)
32. Navaneet, K., Pourahmadi Meibodi, K., Abbasi Koohpayegani, S., Pirsiavash, H.: CompGS: Smaller and faster Gaussian splatting with vector quantization. In: ECCV. pp. 330–349 (2024)
33. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3D Gaussian splatting for accelerated novel view synthesis. In: CVPR. pp. 10349–10358 (2024)
34. Papantonakis, P., Kopanas, G., Kerbl, B., Lanvin, A., Drettakis, G.: Reducing the memory footprint of 3D Gaussian splatting. In: ACM SIGGRAPH. vol. 7, pp. 1–17 (2024)
35. Qiao, L., Chuprat, S.: MEAA-Net: Memory-efficient asymmetric attention for resource-constrained lung nodule classification. IEEE Access (2026)
36. Ren, K., Jiang, L., Lu, T., Yu, M., Xu, L., Ni, Z., Dai, B.: Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians. IEEE TPAMI pp. 1–15 (2025)
37. Wang, T., Li, M., Zeng, G., Meng, C., Zhang, Q.: Gaussian herding across pens: An optimal transport perspective on global gaussian reduction for 3DGS. In: NeurIPS. vol. 38, pp. 157898–157923 (2026)
38. Weloday, F., Su, J.: LWMSCNN-SE: A lightweight multi-scale network for efficient maize disease classification on edge devices. arXiv preprint arXiv:2601.07957 (2026)
39. Xie, S., Zhang, W., Tang, C., Bai, Y., Lu, R., Ge, S., Wang, Z.: MesonGS: Post-training compression of 3D Gaussians via efficient attribute transformation. In: ECCV. pp. 434–452 (2024)
40. Youn, S., Lee, S., Kim, G., Kwon, W., Bae, S.H., Oh, J.: SUCCESS-GS: Survey of compactness and compression for efficient static and dynamic Gaussian splatting. arXiv preprint arXiv:2512.07197 (2025)
41. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR. pp. 586–595 (2018)
42. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: EWA splatting. IEEE TVCG **8**(3), 223–238 (2002)

A Supplementary Material

GFLOPs vs. Latency & Computation Methodology. In Sec. 4.2, we highlight that REFINE achieves a staggering reduction in GFLOPs ($> 3000\times$) compared to rendering-based state-of-the-art methods like PUP. However, the observed wall-clock latency reduction on a high-end GPU (e.g., RTX 3090) is relatively smaller ($< 20\times$).

This discrepancy fundamentally arises from the varying hardware utilization paradigms. Baseline render-aware methods heavily rely on the highly parallelized differentiable rasterization pipeline, which fully saturates the GPU's capacity. This extreme parallelization masks their massive computational burden. In contrast, our REFINE bypasses the rendering pipeline entirely and achieves its high processing speed with minimal hardware utilization, relying purely on lightweight algebraic operations. Therefore, wall-clock time on workstation-grade GPUs conceals the true computational cost. GFLOPs serve as a much more accurate reflection of the actual hardware burden, which is especially critical for deploying 3DGS on resource-constrained edge devices. To precisely quantify the

Table B.1: Computation Breakdown for the Entire Scene.

Method	Modules & Operations	Total FLOPs	GFLOPs
REFINE	· Adaptive Features (∼16) · Visibility Subsampling (∼14) · Intrinsic Coupling (∼23)	$16 \times N$ $14 \times N \times 64$ $23 \times N$	**∼1.96**
Light-Gaussian	· Volume & Opacity (∼11) · Frustum Projection (∼35)	$11 \times N$ $35 \times N \times \lvert\mathcal{V}\rvert$	∼22.0
GHAP	· Covariance Prep. (∼120) · Iterative Clustering (∼40) · Parameter Decomp. (∼350)	$120 \times N$ $40 \times N \times \mathbf{K}$ $350 \times N$	∼85.0
MesonGS	· Forward Pass (∼5,000) · Backward Pass (∼10,000) · Grad. Accumulation (∼11)	$5,000 \times N \times \lvert\mathcal{V}\rvert$ $10,000 \times N \times \lvert\mathcal{V}\rvert$ $11 \times N$	∼9,450
PUP	· Diff. Rasterization (∼15,000) · Fisher Matrix (∼72) · SVD Decomposition (∼500)	$15,000 \times N \times \lvert\mathcal{V}\rvert$ $72 \times N \times \lvert\mathcal{V}\rvert$ $500 \times N$	∼9,492

**Note: Based on a standard scene with $N \approx 2.1 \times 10^6$ primitives and $|\mathcal{V}| \approx 300$ views. K is GHAP's target block size (≈ 1000).*

efficiency gains, we provide a complexity breakdown in Table B.1. The total computational cost is an aggregate of primitive-wise operations scaled by the number of primitives N and the number of training views $|\mathcal{V}|$. As reported, rendering-based methods (PUP and MesonGS) incur massive computational burdens, as their complexity scales linearly with $|\mathcal{V}| \times N$ due to repetitive forward and backward rasterization passes for every view. In contrast, REFINE remains remarkably efficient; by bypassing rendering and leveraging a rendering-free Hessian.

Post-Pruning Fine-Tuning. Although we evaluated pruning under a strict zero-shot, post-processing setting to directly compare the inherent effectiveness of the importance metrics, REFINE also serves as an exceptionally robust and efficient initialization for subsequent optimization pipelines.

To demonstrate this, we compared REFINE with PUP under a fine-tuning setting at an extreme pruning ratio of 70%. As shown in Table B.2, after removing the redundant primitives, both methods underwent fine-tuning optimization. REFINE achieves consistently higher PSNR across the evaluated scenes, while maintaining highly competitive SSIM. This validates that our rendering-free metric correctly identifies and retains the most structurally significant primitives, providing an optimal starting point for fine-tuning while saving massive computational overhead during the pruning phase.

Table B.2: Pruning Comparisons with Fine-tuning (ratio = 70%)

Dataset	PUP			REFINE (Ours)		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
bicycle	25.01	0.745	0.244	**25.15**	**0.748**	0.248
bonsai	31.50	0.938	0.197	**31.60**	0.934	0.202
kitchen	29.11	0.921	0.137	**30.55**	0.913	0.152

Fig. B.1: Visual comparisons of a failure case on a very dense scene (Room) with an extreme 70% pruning ratio. Ignoring cumulative interactions leads to local blurring.

Analysis of Failure Cases. To achieve extreme rendering-free speed, our theoretical formulation introduces two structural approximations: primitive independence and attribute orthogonality. While our empirical results in the main text demonstrate that these assumptions hold remarkably well in general cases, they naturally present limitations under extreme conditions. Specifically, since our post-processing pruning lacks access to original multi-view images, all current methods rely on non-strict approximations. In highly dense or heavily occluded scenes, and pushed to extremely aggressive pruning ratios (e.g., 70%), the cumulative cross-primitive interactions, such as alpha blending along a dense ray,

become too strong to ignore. As shown in Fig. B.1, removing primitives based on isolated parameter scores in such densely overlapping areas can amplify the disruption of these interactions, leading to localized blurring and visual artifacts. In these failure cases, our method's PSNR drops slightly below that of fully render-aware methods like PUP.

Therefore, while REFINE provides a highly effective heuristic metric for ultra-fast pruning, it trades off marginal rendering fidelity in exceptionally dense regions for a 3000× speedup. Future work could explore incorporating lightweight, localized re-rendering passes or low-rank off-diagonal approximations to mitigate this while preserving real-time efficiency.