EGGS: Exchangeable 2D/3D Gaussian Splatting for Geometry-Appearance Balanced Novel View Synthesis

Yancheng Zhang , Guangyu Sun , and Chen Chen

Institute of Artificial Intelligence, University of Central Florida
{yczhang, guangyu.sun, chen.chen}@ucf.edu

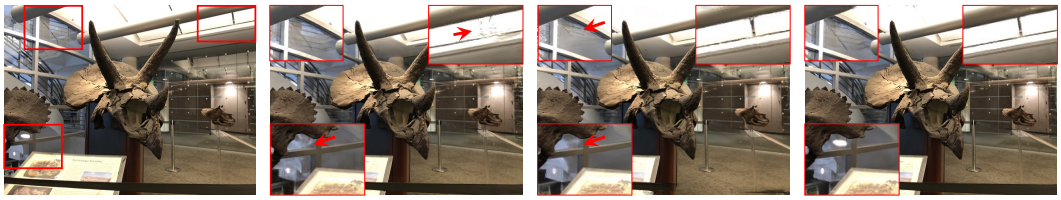


Figure 1: Comparison of 3DGS, 2DGS, and our EGGS. While 3DGS achieves high-fidelity appearance, it often produces inaccurate geometry, with imprecise surfaces and blurred edges. 2DGS improves geometric consistency across views but suffers from reduced appearance quality due to over-smoothed surfaces and loss of detail. In contrast, EGGS employs an exchangeable hybrid Gaussian representation that achieves both accurate geometry and high-quality appearance.

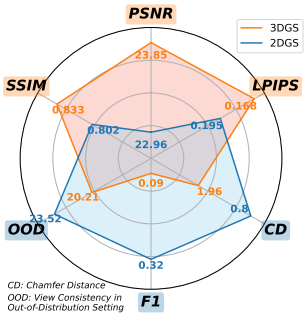
Abstract

Novel view synthesis (NVS) is crucial in computer vision and graphics, with wide applications in AR, VR, and autonomous driving. While 3D Gaussian Splatting (3DGS) enables real-time rendering with high appearance fidelity, it suffers from multi-view inconsistencies, limiting geometric accuracy. In contrast, 2D Gaussian Splatting (2DGS) enforces multi-view consistency but compromises texture details. To address these limitations, we propose Exchangeable Gaussian Splatting (EGGS), a hybrid representation that integrates 2D and 3D Gaussians to balance appearance and geometry. To achieve this, we introduce Hybrid Gaussian Rasterization for unified rendering, Adaptive Type Exchange for dynamic adaptation between 2D and 3D Gaussians, and Frequency-Decoupled Optimization that effectively exploits the strengths of each type of Gaussian representation. Our CUDA-accelerated implementation ensures efficient training and inference. Extensive experiments demonstrate that EGGS outperforms existing methods in rendering quality, geometric accuracy, and efficiency, providing a practical solution for high-quality NVS. Code and demo available at https://github.com/Fobow/EGGS.

1 Introduction

Novel view synthesis (NVS) is a fundamental task in computer graphics and computer vision, with broad applications in augmented reality (AR), virtual reality (VR), and autonomous driving [1, 2, 3]. Neural Radiance Fields (NeRF) [4] reconstruct implicit radiance fields via differentiable volume rendering. Despite achieving photorealistic appearance and accurate geometry, NeRF-based methods [5, 6, 7, 8, 9, 10] typically suffer from long training times and slow rendering speeds. 3D Gaussian Splatting (3DGS) [11] has emerged as an efficient alternative, leveraging anisotropic

39th Conference on Neural Information Processing Systems (NeurIPS 2025).



Method	Gaussian Type	Raster-izer	Type Exchange	Regular-ization	Setting
3DGS SIGGRAPH'23 [11]	3D	3D	✗	-	General
SuGaR CVPR'24 [14]	3D	3D	✗	Normal	General
GaussianPro ICML'24 [13]	3D	3D	✗	Normal	General
2DGS SIGGRAPH'24 [12]	2D	2D	✗	Depth & Normal	General
GS Surfels SIGGRAPH'24 [15]	2D	3D*	✗	Depth & Normal	General
TextureGS ECCV'24 [16]	2D	2D	✗	Depth & Normal	General
HybridGS CVPR'25 [17]	3D + 2D*	3D	✗	-	Transient
HorizonGS CVPR'25 [18]	3D / 2D	3D / 2D	✗	Depth & Normal	Varying-altitude
Ours	3D + 2D	Hybrid	✓	Frequency	General

Figure 2: Left: Comparison of 3DGS and 2DGS in **appearance** and **geometry** metrics. Right: Comparison between EGGS and related works. Prior works either use only single representation or do not explore complementary advantages of 3D and 2D Gaussians. ⋆ Gaussian Surfel [15] directly sets the z-scale of 3D Gaussian to zero and uses the rasterizer from 3DGS. * HybridGS [17] uses image-frame single-view 2D Gaussians [19, 20] instead of 2D Gaussians in the 3D space [12].

3D Gaussians for real-time, high-quality rendering. While 3DGS excels in appearance fidelity, its anisotropic nature often leads to multi-view inconsistencies, limiting geometric accuracy [12, 13]. As shown in Figure 1, this can lead to inaccurate edges and surfaces.

Following 3DGS, a line of work has focused on improving its geometric accuracy and reconstruction quality through additional regularization and novel representations, as shown in Figure 2 (right). SUGAR [14] and GaussianPro [13] introduce normal-based regularization, such as planar loss, to align Gaussian normals and encourage flatter shapes, thereby improving surface consistency. Gaussian Surfles [15] and GOF [21] incorporate additional geometry-aware constraints to enhance spatial coherence. 2D Gaussian Splatting (2DGS)[12] replaces 3D ellipsoids with 2D surfels, significantly improving multi-view consistency and geometric accuracy, as shown in Figure 2 (left). However, this comes at the cost of degraded appearance quality, as surfel-based representations struggle to preserve high-frequency details. TextureGS [16] attempts to decouple appearance and geometry within the 2DGS framework, but the single representation still limits overall rendering performance. Recently, HybridGS [17] combines 3DGS with image-space 2D Gaussians to address transient objects, but its radiance field remains fully represented by 3D Gaussians. HorizonGS [18], designed for varying-altitude scenes, decodes 2D Gaussians for surface reconstruction and 3D Gaussians for view synthesis separately via an MLP in ScaffoldGS [22]. While effective in their target domains, these methods do not explore a unified hybrid radiance representation. As a result, the complementary strengths of 2DGS and 3DGS in geometry and appearance remain underutilized.

Effectively combining 3D and 2D Gaussians to jointly improve appearance and geometry is non-trivial, as simply mixing the two representations does not necessarily improve reconstruction quality [18]. To start, the geometric accuracy of 2D Gaussians relies on a ray–splat–intersection-based rasterizer designed to enforce multi-view consistency. Using the projection-based 3DGS rasterizer to render 2D Gaussians can lead to suboptimal geometry [15]. Moreover, Gaussian parameters change significantly during training. For instance, 3D Gaussians may flatten to approximate surfaces, while 2D Gaussians may expand volumetrically to capture thin structures or translucent effects. Fixing the Gaussian type throughout optimization can limit the model's expressiveness. Finally, relying solely on photometric loss is insufficient to balance geometry and appearance. Additional regularization is required to guide the optimization of hybrid representations. Most importantly, the regularization strategy should account for the distinct characteristics of 3D and 2D Gaussians.

In response to these challenges, we introduce **E**xchan**g**eable **G**aussian **S**platting (EGGS), an adaptive hybrid representation that unifies 2D and 3D Gaussian splatting in a single framework. EGGS provides a practical and efficient solution for high-quality novel view synthesis and 3D reconstruction. Our main contributions are as follows:

- To preserve the complementary strengths of 3D and 2D Gaussians, we develop *Hybrid Gaussian Rasterization*, a unified rendering framework that supports both projection-based and ray–splat–intersection-based rasterization. We implement this framework with CUDA for efficient optimization, and ensure compatibility with existing 3DGS and 2DGS pipelines.
- We propose *Adaptive Type Exchange*, which enables an exchangeable hybrid of 2D and 3D Gaussians. We use effective rank as an auxiliary criterion to determine whether each Gaussian should dynamically switch its type during training, resulting in a more flexible and content-adaptive representation.

- To better balance geometry and appearance, we introduce *Frequency-Decoupled Optimization*, a regularization strategy in the frequency domain. Using the Discrete Wavelet Transform (DWT), we extract low-frequency components to guide scene geometry and high-frequency components to refine appearance. We supervise 3D and 2D Gaussians asymmetrically to exploit their distinct characteristics, where high-frequency signals guide 3D Gaussians toward detailed appearance, while low-frequency signals supervise 2D Gaussians for geometric consistency.
- We conduct extensive experiments demonstrating that EGGS significantly improves the trade-off between appearance fidelity and geometric accuracy. It outperforms both 3DGS and 2DGS in appearance quality, while achieving geometric accuracy and multi-view consistency comparable to 2DGS. Moreover, EGGS serves as a versatile representation that performs well in challenging scenarios such as few-shot and out-of-distribution view synthesis.

2 Related Works

Radiance Fields for Novel View Synthesis. Neural Radiance Fields (NeRF) [4] have emerged as a fundamental approach for novel view synthesis [23], representing scenes as continuous volumetric functions optimized via differentiable rendering. While NeRF achieves high-fidelity reconstruction, it requires dense sampling and significant computational resources. Subsequent works have improved either quality [24, 25] or efficiency [5, 7, 26, 6], but the excessive training and rendering time remains a major bottleneck. To address this, recent efforts have explored more efficient alternatives, such as 3D Gaussian Splatting (3DGS) [11], which represents scenes using a set of 3D Gaussians that can be efficiently rasterized and optimized for real-time rendering. To further improve the performance and efficiency of 3DGS, several extensions have been proposed. ScaffoldGS [22] introduces a voxel-based representation where an MLP is used to decode 3D Gaussians within each voxel. 3DGS-MCMC [27] formulates Gaussian densification as a Markov Chain Monte Carlo sampling process, enabling a more efficient and adaptive distribution of Gaussians across the scene.

Geometry-Appearance-Balanced Gaussian Splatting. While 3DGS achieves high appearance fidelity and is efficient in both training and rendering, the anisotropic nature of 3D Gaussians often exhibits multi-view inconsistency, resulting in limited geometric accuracy. To address this, several works propose geometry regularization techniques. SUGAR [14] and GaussianPro [13] introduce normal-based regularization to encourage flatter Gaussians that better align with scene surfaces. Gaussian Surfels [15] and GOF [28] further enforce depth accuracy and normal consistency to enhance geometric reconstruction. Instead of relying solely on regularization, 2DGS [12] adopts a 2D surfel representation with a specialized ray–splat–intersection rasterizer, ensuring multi-view consistency and significantly improving geometric accuracy compared to 3DGS. It also incorporates additional depth and normal regularization. However, this comes at the cost of reduced appearance quality, as 2D surfels struggle to preserve high-frequency detail. TextureGS [16] attempts to decouple geometry and appearance modeling within the 2DGS framework, but its appearance fidelity remains limited due to the inherent drawbacks of the 2D representation.

As demonstrated in Figure 2 (left), 3D Gaussians achieve better appearance quality in PSNR, SSIM, and LPIPS. In contrast, 2D Gaussians offer superior view consistency and geometric fidelity, resulting in more robust PSNR under out-of-distribution (OOD) conditions, improved point cloud accuracy in Chamfer Distance (CD), and higher depth accuracy in F1 score. As shown in Figure 2 (right), most existing methods [12, 11, 14, 13, 29, 30, 22, 31, 32, 33, 34] rely on a single Gaussian representation to reconstruct radiance fields, which limits their flexibility and adaptability. Although HybridGS [17] incorporates both 3D Gaussians and image-space 2D Gaussians to better handle transient content, its radiance field remains solely represented by 3D Gaussians. A radiance field that jointly leverages both 2D and 3D Gaussians remains largely unexplored. It is still unclear how 2D and 3D Gaussians can be made exchangeable during training and how to fully exploit their complementary strengths in appearance and geometry. We provide a more detailed discussion in Appendix B.

3 Method

We provide an overview of the EGGS framework in Figure 3. To enable the joint training of 2D and 3D Gaussians within a unified framework, we first introduce *Hybrid Gaussian Rasterization* in Section 3.1, which supports both ray–splat–intersection-based rendering for 2D Gaussians and projection-based rendering for 3D Gaussians. Next, we present *Adaptive Type Exchange* in Section 3.2,

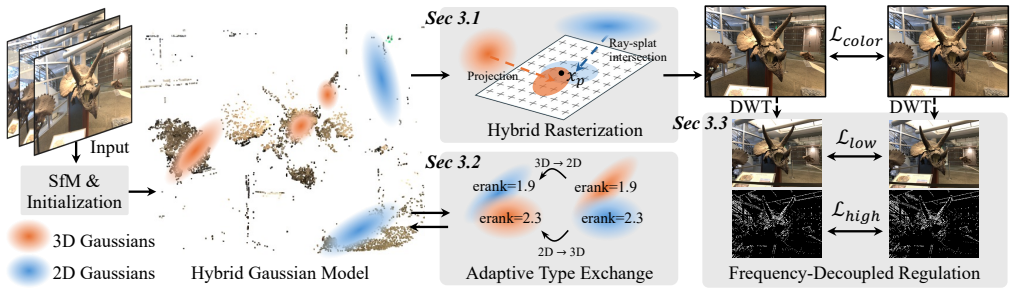


Figure 3: Overview of the EGGS framework. We initialize 2D and 3D Gaussians from sparse points obtained via structure-from-motion (SfM) [35, 36]. Their parameters are then jointly optimized using our CUDA-accelerated differentiable hybrid rasterization. To enhance the flexibility of the hybrid representation, Adaptive Type Exchange is introduced to allow each Gaussian to switch between 2D and 3D types during training. Finally, we apply Discrete Wavelet Transform (DWT) [37] and introduce Frequency-Decoupled Optimization to balance geometric accuracy and appearance fidelity.

which enables dynamic switching between 2D and 3D types during optimization. Finally, to optimize the hybrid model for balanced geometric consistency and appearance fidelity, we propose *Frequency-Decoupled Optimization* in Section 3.3, a supervision strategy that leverages the distinct frequency characteristics of 2D and 3D Gaussians.

3.1 Hybrid Gaussian Rasterization

Differentiable rasterization was introduced in 3DGS to enable gradient-based optimization of Gaussian parameters using a projection-based pipeline for real-time rendering. 2DGS later developed a ray–splat–intersection-based rasterizer tailored to 2D surfel representations, improving multi-view consistency and geometric accuracy. However, the architectural distinction between these two rasterization pipelines makes it non-trivial to render and optimize a hybrid model within a unified framework. While 2D Gaussians can be viewed as degenerate 3D Gaussians with zero scale along the z-axis, directly rendering them with the 3D rasterizer leads to geometric inaccuracies [12]. This is due to the affine projection approximation used in 3DGS, which introduces distortion at all points except the Gaussian center. We further analyze this issue in Section 4.2.

To leverage the complementary strengths of 3D and 2D Gaussians, it is necessary to render them within a unified framework. To this end, we propose *Hybrid Gaussian Rasterization*, which integrates both projection-based and ray–splat–intersection-based pipelines. In our rasterizer, each Gaussian primitive $\mathcal{G}$ is parameterized by a center $\boldsymbol{\mu} \in \mathbb{R}^3$, scale $\boldsymbol{s} \in \mathbb{R}^3$, rotation quaternion $\boldsymbol{r} \in \mathbb{R}^4$, opacity $\alpha \in \mathbb{R}$, and spherical harmonic (SH) color coefficients $\boldsymbol{f} \in \mathbb{R}^{3\times(l+1)^2}$, where l is the degree of view-dependent color. The view-dependent RGB color $\boldsymbol{c}$ is decoded from $\boldsymbol{f}$. The Gaussian shape is defined by the covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{RSS}^T\boldsymbol{R}^T$, where $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix derived from $\boldsymbol{r}$, and $\boldsymbol{S} = \text{diag}(s_x, s_y, s_z) \in \mathbb{R}^{3\times3}$ is the scaling matrix. We augment each Gaussian with a type specifier $t \in \{0, 1\}$ to indicate whether it is a 2D ($t = 0$) or 3D ($t = 1$) Gaussian. 2D Gaussians are initialized with $s_z = 0$, while the remaining parameters follow the initialization of 3DGS.

As shown in Figure 4, we rasterize Gaussians according to their types, where affine projection is used for 3D Gaussians and ray–splat–intersection is used for 2D Gaussians. The contribution of $\mathcal{G}_i^{3d}$ and $\mathcal{G}_i^{2d}$ is evaluated by computing the distance from the image-space pixel x_p to $\mathcal{G}_i^{proj}$ or $\mathcal{G}_i^{2d}$ in the 2D image plane or tangent frame, respectively:

$$d_i = \begin{cases} (\boldsymbol{u}_i(x_p)^2 + \boldsymbol{v}_i(x_p)^2 & \text{if } t_i = 0 \\ (x_p - \boldsymbol{\mu}'_{3d,i})^T \boldsymbol{\Sigma}_i'^{-1} (x_p - \boldsymbol{\mu}'_{3d,i}) & \text{otherwise} \end{cases} \quad (1)$$

where $\boldsymbol{\mu}'_{3d,i}$ and $\boldsymbol{\Sigma}_i$ are the projected center and covariance of the 3D Gaussian computed via affine projection, and $u_i(x_p)$ and $v_i(x_p)$ denote the coordinates of the intersection between the ray through x_p and the 2D Gaussian. The distance d_i can be computed simultaneously for both 3D and 2D Gaussians. The final contribution of each Gaussian

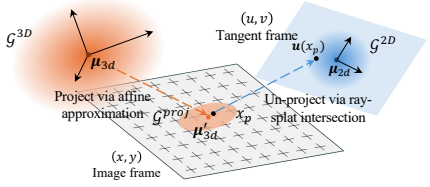


Figure 4: Illustration of Hybrid Gaussian Rasterization. The contribution of 3D Gaussians and 2D Gaussians is computed via affine projection and ray-splat-intersection, respectively.

is then computed uniformly as $\tilde{\alpha}_i = \alpha_i e^{-\frac{1}{2}d_i}$, where α_i is the opacity of the i-th Gaussian. With the above formulation, both 3D and 2D Gaussians can be rendered in a single α-blending pass:

$$C(x_p) = \sum_{i \in N} \boldsymbol{c}_i \tilde{\alpha}_i \prod_{j=1}^{i-1}(1 - \tilde{\alpha}_j) \tag{2}$$

where the final color at pixel x_p is computed from color $\boldsymbol{c}_i$ and contribution $\tilde{\alpha}_i$ of each Gaussian primitive. To support efficient and parallel rendering, we implement our hybrid rasterizer in CUDA. More details on initialization and densification are provided in Appendix A, and those on projection-based and ray–splat–intersection-based rasterization procedures are deferred to Appendix C.

3.2 Adaptive Type Exchange

While the type specifier introduced in Section 3.1 enables unified rendering of 2D and 3D Gaussians, each Gaussian primitive is initialized with a fixed type. Such fixed type assignment can limit the expressiveness of the model, as Gaussians may naturally deviate from their initial type during optimization. For example, 3D Gaussians may become increasingly flat to better model surfaces, while 2D Gaussians may take on more volumetric properties to capture semi-transparent regions. To fully exploit the flexibility of the hybrid model, the type of each Gaussian should dynamically adapt to its evolving geometric characteristics. To this end, we propose *Adaptive Type Exchange*, which allows each Gaussian to switch between 2D and 3D types during training.

The key to *Adaptive Type Exchange* is detecting discrepancies between a Gaussian's assigned type and its effective geometric dimensionality. Therefore, we introduce the effective rank (erank) [38, 39] as an indicator of this dimensionality, allowing the model to determine when type switching is needed during training. Given a Gaussian $\mathcal{G}$ with scaling $\boldsymbol{s} = (s_x, s_y, s_z)$, we define its effective rank as:

$$\text{erank}(\mathcal{G}) = \exp\left(-\sum_{i=0}^{2} \frac{q_i}{\|\boldsymbol{q}\|_1} \log \frac{q_i}{\|\boldsymbol{q}\|_1}\right), \quad \text{where } \boldsymbol{q} = (s_x^2, s_y^2, s_z^2). \tag{3}$$

As illustrated in Figure 5, erank provides a principled signal for deciding when to switch types. A perfectly isotropic 3D Gaussian has erank $= 3$, while a flattened Gaussian approaches erank $= 2$. If a Gaussian primitive $\mathcal{G}_i$ is assigned as 3D ($t_i = 1$) but its effective rank falls below a threshold θ_e, we mark it for conversion to 2D by setting $t_i' = 0$. Similarly, we update 2D Gaussians to 3D ($t_i' = 1$) when their effective rank exceeds the threshold. Yet, merely flipping the type specifier can lead to unstable parameter transitions, as the s_z scale is treated as least significant in 2D Gaussians. To ensure stable conversion, we reparameterize the covariance of 3D Gaussians during switching and adjust gradient flow to s_z for 2D Gaussians.

Reparameterization. ($3D \rightarrow 2D$) All three scales of a 3D Gaussian are initially optimized, whereas 2D Gaussians ignore the s_z scale during ray–splat–intersection-based rasterization. Accordingly, when converting a 3D Gaussian to 2D, only s_x and s_y are retained and s_z is discarded. However, directly discarding s_z can lead to instability during training when it is not the least significant scale. To prevent this, we reparameterize the 3D Gaussian before conversion so that s_z corresponds to the smallest axis. The key to stable conversion is aligning s_z with the least significant scale while preserving covariance $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T$. We first construct the converted scaling matrix $\boldsymbol{S}^*$ using a permutation matrix $\boldsymbol{P}$ that moves the least significant scale to the z-axis:

$$\boldsymbol{S}^* = \boldsymbol{P}\boldsymbol{S}\boldsymbol{P}^T \tag{4}$$

$\boldsymbol{P}$ is set to $\boldsymbol{P}_x$ or $\boldsymbol{P}_y$ if s_x or s_y is the least significant scale, respectively, where:

$$\boldsymbol{P}_x = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{P}_y = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{5}$$

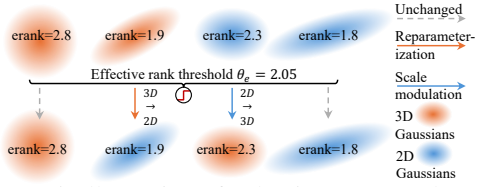


Figure 5: Illustration of Adaptive Type Exchange.

Then, to ensure the covariance $\boldsymbol{\Sigma}$ remain unchanged in $\boldsymbol{\Sigma}^* = \boldsymbol{R}^*\boldsymbol{S}^*\boldsymbol{S}^*\boldsymbol{R}^{*T}$, we set the rotation of 2D Gaussian as $\boldsymbol{R}^* = \boldsymbol{R}\boldsymbol{P}^T$. We note that $\boldsymbol{R}^*$ is converted to quaternions during optimization. To ensure a valid conversion, the rotation matrix $\boldsymbol{R}^*$ must be orthogonal with a positive determinant. $\boldsymbol{P}_x$ and $\boldsymbol{P}_y$ are designed to preserve these properties. Additional details are provided in Appendix D.

Scale Modulation. ($2D \rightarrow 3D$) When converting 2D Gaussians to 3D, all parameters are retained with the type specifier flipped. However, as mentioned above, the s_z scale of 2D Gaussians is not

optimized during rasterization. However, to allow 2D Gaussians to develop volumetric capacity and transition to 3D when needed, gradient flow must also be introduced along the z-axis. To support this, we incorporate s_z into the computation graph via a soft modulation based on opacity α, aligning geometric expressiveness with visual transparency. The intuition behind this design is that 2D-to-3D transitions often occur in regions with semi-transparent or volumetric effects that flat primitives cannot represent well. This modulation enables s_z to be optimized throughout training, while ensuring its updates remain stable and smoothly conditioned on opacity:

$$\alpha_i^* = \alpha_i e^{-\lambda_z \times s_z^*} \tag{6}$$

where s_z^* denotes the activated z-axis scale, computed via a soft gating function:

$$s_z^* = \text{sigmoid}\left(\frac{s_z - \theta_z}{T_z}\right) s_z \tag{7}$$

The soft scale modulation allows a 2D Gaussian to remain effectively two-dimensional when s_z is insignificant, in which case s_z^* approaches zero. Conversely, as a 2D Gaussian evolves toward a more volumetric form, an increasing s_z leads to reduced opacity, effectively enabling the representation of semi-transparent or volumetric effects. Additional details on the effective rank threshold, 3D Gaussian reparameterization and permutation, and 2D Gaussian scale modulation are provided in Appendix D.

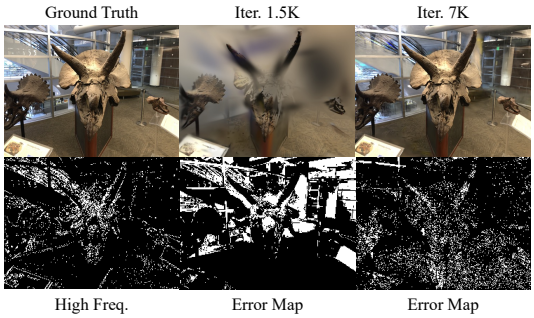


Figure 6: Illustration of reconstruction error during training. In early iterations, the model focuses on overall scene geometry, while high-frequency local details are progressively refined in later stages.

3.3 Frequency-Decoupled Optimization

With our hybrid rasterization and adaptive type exchange mechanism, 2D and 3D Gaussians can be jointly optimized within a unified and flexible framework. However, relying solely on photometric loss is insufficient to effectively optimize the hybrid model for balanced geometry and appearance. 2D and 3D Gaussians exhibit distinct characteristics during optimization and specialize in different aspects of the scene. 2D Gaussians are better suited for enforcing geometric consistency, while 3D Gaussians excel at capturing high-frequency appearance details. To fully leverage these complementary strengths, we introduce *Frequency-Decoupled Optimization*, a supervision strategy that decouples low- and high-frequency components and assigns them asymmetrically to 2D and 3D Gaussians, respectively.

Frequency Decoupling via Discrete Wavelet Transform. As shown in Figure 6, scene information can be effectively separated in the frequency domain. High-frequency components typically correspond to fine details that are refined in later training stages (e.g., 7K iterations), while low-frequency components capture overall scene geometry and are optimized earlier. This frequency-based separation aligns well with the complementary roles of 3D Gaussians in modeling appearance and 2D Gaussians in capturing geometry. Motivated by this, we introduce *Frequency-Decoupled Optimization* to supervise the hybrid model in the frequency domain. We apply DWT [37] to decompose the ground truth image $\mathcal{I}$ into low- and high-frequency components: $\mathcal{I}_l, \mathcal{I}_h = \text{DWT}(\mathcal{I})$. The same transformation is applied to the rendered image $\hat{\mathcal{I}}$ to obtain $\hat{\mathcal{I}}_l$ and $\hat{\mathcal{I}}_h$. The frequency loss is defined as $\mathcal{L}_i = \|\hat{\mathcal{I}}_i - \mathcal{I}_i\|_2^2$ for $i \in \{\text{low}, \text{high}\}$. We also include the standard appearance loss used in 3DGS: $\mathcal{L}_{\text{color}} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$. With access to frequency-specific losses, a naïve strategy is to combine all terms and apply them uniformly to all Gaussians:

$$\mathcal{L} = \mathcal{L}_{\text{color}} + \lambda_{\text{low}}\mathcal{L}_{\text{low}} + \lambda_{\text{high}}\mathcal{L}_{\text{high}}. \tag{8}$$

where $\mathcal{L}_{\text{low}}$ and $\mathcal{L}_{\text{high}}$ are applied equally to all 2D and 3D Gaussians. While supervision is decoupled in the frequency domain, this approach overlooks the distinctions of each representation.

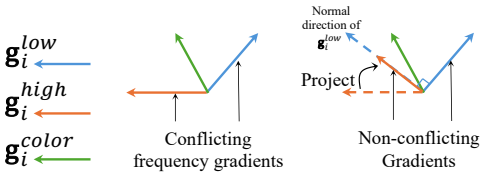


Figure 7: Illustration of frequency gradient projection. We project gradients from high-frequency loss onto the normal vector of gradients from low-frequency for 2D Gaussians.

Asymmetrical Gradient Update with Projected Conflicts. We denote the gradients to $\mathcal{G}_i$ from $\mathcal{L}_{\text{color}}$, $\mathcal{L}_{\text{low}}$ and $\mathcal{L}_{\text{high}}$ as $\mathbf{g}_i^{color}$, $\mathbf{g}_i^{low}$ and $\mathbf{g}_i^{high}$, respectively. As illustrated in Figure 7, conflicting gradients can arise when losses from different frequency components are directly applied to update Gaussian parameters (i.e., Eq.(8)), diminishing the effectiveness of frequency-based regularization. Such conflicts stem from the distinct characteristics of 2D and 3D Gaussians. 2D Gaussians are more effective at capturing overall geometry and ensuring multi-view consistency, where low-frequency signals offer more relevant guidance, while high-frequency gradients may counteract this by encouraging appearance-driven updates. Conversely, 3D Gaussians specialize in modeling fine-scale appearance and benefit more from high-frequency supervision, whereas low-frequency signals contribute less to their performance.

Algorithm 1: Frequency-Decoupled Optimization

Require : Gaussians $\{\mathcal{G}_i\}_{i=0}^{N-1}$, appearance loss $\mathcal{L}_{color}$, frequency loss $\mathcal{L}_{low}$ and $\mathcal{L}_{high}$.

$\mathbf{g}^{color}, \mathbf{g}^{low}, \mathbf{g}^{high} \leftarrow \nabla_{\mathcal{G}}\mathcal{L}_{color}, \nabla_{\mathcal{G}}\mathcal{L}_{low}, \nabla_{\mathcal{G}}\mathcal{L}_{high}$;

// Process per Gaussian gradient conflict

for $i \leftarrow 0$ **to** $N-1$ **do**

  // There are conflict Gradients in different frequencies

  if $\mathbf{g}_i^{low} \cdot \mathbf{g}_i^{high} < 0$ **then**

    if $t_i == 0$ **then**

      $\mathbf{g}_i^{high} \leftarrow \mathbf{g}_i^{high} - \frac{\mathbf{g}^{high} \cdot \mathbf{g}^{low}}{||\mathbf{g}^{low}||^2}\mathbf{g}^{low}$; (9)

        // Type is 2D, project $\mathbf{g}_i^{high}$ onto normal of $\mathbf{g}_i^{low}$

    else

      $\mathbf{g}_i^{low} \leftarrow \mathbf{g}_i^{low} - \frac{\mathbf{g}^{high} \cdot \mathbf{g}^{low}}{||\mathbf{g}^{high}||^2}\mathbf{g}^{high}$; (10)

        // Type is 3D, project $\mathbf{g}_i^{low}$ onto normal of $\mathbf{g}_i^{high}$

  $\Delta\mathcal{G}_i = \mathbf{g}_i^{color} + \mathbf{g}_i^{low} + \mathbf{g}_i^{high}$

return Update $\Delta\mathcal{G}$

To address this issue, we propose an asymmetrical update strategy that applies frequency supervision based on Gaussian type, as shown in Algorithm 1. For each Gaussian, we check for potential gradient conflicts by computing the inner product between $\mathbf{g}_i^{\text{low}}$ and $\mathbf{g}_i^{\text{high}}$, where a negative value indicates divergent update directions [40]. When such conflict is detected, we retain the frequency component most relevant to the Gaussian type and project the other. Specifically, for 2D Gaussians, we preserve supervision from low-frequency and remove the conflicting component of high-frequency by projecting $\mathbf{g}_i^{\text{high}}$ onto the normal vector of $\mathbf{g}_i^{\text{low}}$, as shown in Eq.(9). Similarly, for 3D Gaussians, we retain $\mathbf{g}_i^{\text{high}}$ and project $\mathbf{g}_i^{\text{low}}$ as indicated in Eq.(10). This asymmetrical supervision ensures each Gaussian is updated along its most informative direction while minimizing interference from less relevant frequency signals. More details on DWT and the gradient projection strategy are provided in Appendix E and Appendix F, respectively.

4 Experiments

Datasets and Metrics. We evaluate EGGS on several widely used benchmarks. For appearance evaluation, we use Mip-NeRF360 [25], LLFF [41], Tanks&Temples [42], and DTU [43]. For geometry evaluation, we use DTU, which provides ground-truth point clouds, and Tanks&Temples, which offers ground-truth depth maps. Additional dataset details are provided in Appendix A. Following prior work [25, 11, 13, 12], we report PSNR, SSIM [44], and LPIPS [45] to evaluate the appearance quality of synthesized novel views. For geometry, we follow [12, 39] and report Chamfer Distance [46] on DTU to assess reconstruction accuracy.

Baselines. To demonstrate the effectiveness of EGGS, we compare against several single-representation methods that use either 3D or 2D Gaussians. For 3D Gaussian-based methods, we include vanilla 3DGS [11], GaussianPro [13] and GOF [28], which incorporate geometric regularization, and FreGS [47], which introduces frequency-based supervision. For 2D Gaussian-based methods, we consider vanilla 2DGS [12] and TextureGS [16], which improves the appearance fidelity of 2D Gaussians. Additional discussion of related methods is provided in Appendix B.

Implementation. We implement our hybrid rasterizer based on the CUDA rasterization code of 3DGS [11]. We used the Haar filter for the DWT [37, 48]. For Frequency-Decoupled Optimization, we set the weight for the frequency components as $\lambda_{low} = 0.2$ and $\lambda_{low} = 0.4$. For Adaptive Type Switch, we set the erank threshold as 2.05. We offer more details about our training pipeline and parameter setting in Appendix A. All experiments are conducted on an A5000 GPU.

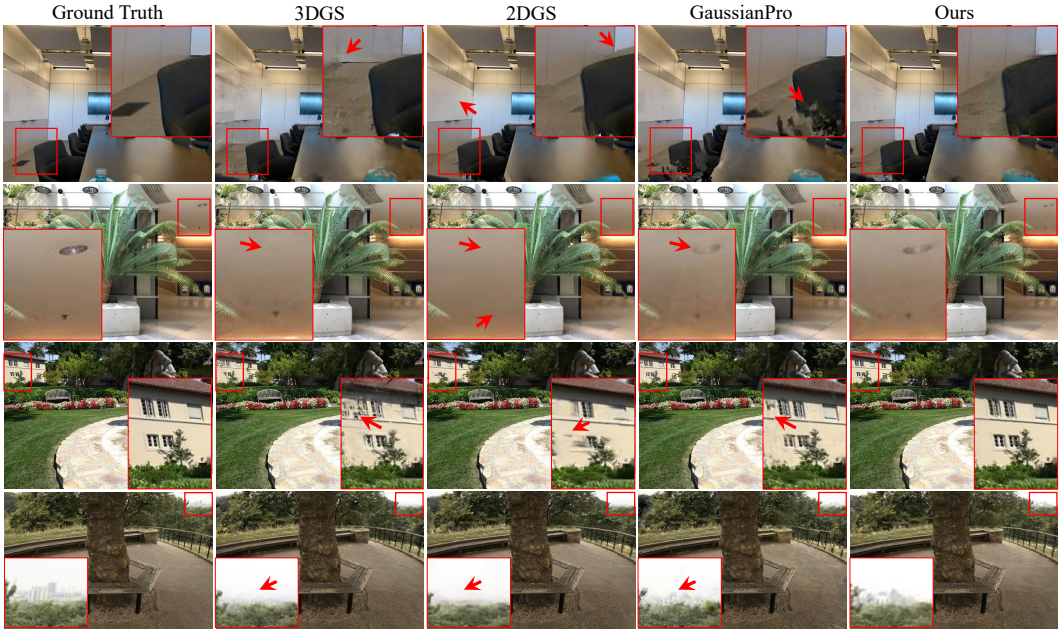


Figure 8: Qualitative comparison on LLFF, Tanks&Temples, and Mip-NeRF360. 3DGS suffers from inaccurate scene geometry. While 2DGS improves geometric fidelity, it overlooks texture and local details. EGGS recovers more accurate geometry while preserving high-frequency details. Additional visual results and videos are available in the supplementary material and project website.

Table 1: Quantitative comparison on Mip-NeRF360, LLFF and Tanks&Temples datasets. The best, second-best, and third-best entries are marked in red, orange, and yellow, respectively.

Method	Mip-NeRF360			LLFF			Tanks&Temples		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
2DGS SIGGRAPH'24 [12]	26.81	0.796	0.297	24.93	0.815	0.147	22.96	0.802	0.195
TextureGS ECCV'24	27.14	0.803	0.285	25.58	0.837	0.117	22.43	0.811	0.189
3DGS SIGGRAPH'23 [11]	27.43	0.814	0.257	26.12	0.865	0.099	23.85	0.833	0.168
GOF TOG'24 [28]	27.42	0.826	0.234	25.57	0.854	0.121	22.41	0.831	0.172
GaussianPro ICML'24 [13]	27.92	0.825	0.208	26.53	0.867	0.105	23.92	0.855	0.162
FreGS CVPR'24 [47]	27.85	0.826	0.209	26.11	0.860	0.102	23.96	0.849	0.178
Ours	27.96	0.851	0.192	27.34	0.895	0.083	24.41	0.923	0.153

4.1 Results

Appearance. Figure 8 and Table 1 show qualitative and quantitative comparisons on Mip-NeRF360, LLFF, and Tanks&Temples. 3D Gaussian-based methods generally achieve better PSNR, SSIM, and LPIPS but often produce blurred geometry due to anisotropic Gaussians. GaussianPro and FreGS improve reconstruction via geometric or frequency regularization but still lack geometric accuracy. 2DGS produces cleaner edges and better geometry, yet oversmooths details and underperforms in appearance. TextureGS enhances 2D appearance but remains inferior to 3D-based methods. In contrast, EGGS outperforms all baselines by combining the strengths of 2D and 3D Gaussians. It recovers more accurate geometry while preserving high-frequency visual details.

Geometry. We evaluate geometry reconstruction quality on Tanks&Temples and DTU. As shown in Figure 9, both 2DGS and EGGS produce more accurate depth maps than 3DGS, with sharper surfaces and clearer edges. However, 2DGS sacrifices appearance fidelity due to the lack of high-frequency detail. In contrast, EGGS improves geometry over 3DGS while also preserving appearance quality. Table 2 reports Chamfer Distance on the DTU dataset, where EGGS outperforms 3DGS and SUGAR. Note that SUGAR, 2DGS, and GOF prioritize surface reconstruction and mesh extraction, often at the cost of appearance. Although 2DGS is slightly more accurate geometrically, EGGS achieves a better trade-off, offering stronger appearance quality alongside competitive geometry.

Efficiency. Table 3 compares the model size and training time of EGGS with 3DGS, 2DGS, and GaussianPro on LLFF and Tanks&Temples. While 2DGS uses the fewest Gaussians, its training time exceeds that of 3DGS. GaussianPro enhances appearance quality over 3DGS but incurs significantly higher training cost. In contrast, EGGS strikes a favorable balance, requiring fewer Gaussians than both 3DGS and GaussianPro, while achieving the shortest training time among all methods.

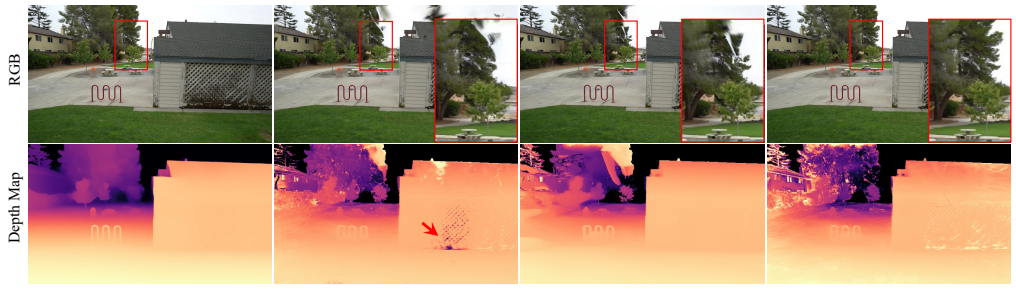


Figure 9: Qualitative comparison on Tanks&Temples. EGGS achieves better overall reconstruction quality, producing more accurate depth maps than 3DGS and recovering high-frequency details better than 2DGS.

Table 3: Comparison on efficiency. # Gaussians is the average number of Gaussians.

Dataset	Method	PSNR	#Gaussians	Training
LLFF	3DGS	26.12	919K	10min
	2DGS	24.93	343K	11min
	GaussainPro	26.53	933K	20min
	EGGS	27.34	581K	9min
Tanks& Temples	3DGS	23.85	1502K	13min
	2DGS	22.96	416K	15min
	GaussainPro	23.92	1381K	35min
	EGGS	24.41	754K	11min

Table 4: Ablation study. Repr. stands for the Gaussian type, Hyb. for hybrid rasterization, Ex. for type exchange, and Freq. for frequency regularization.

ID	Repr.	Hyb.	Ex.	Freq.	PSNR↑	SSIM ↑	LPIPS ↓
i	3D	✗	✗	✗	26.12	0.865	0.099
ii	3D+2D	✗	✗	✗	26.01	0.859	0.105
iii	3D+2D	✓	✗	✗	26.23	0.867	0.097
iv	3D+2D	✓	✓	✗	26.58	0.874	0.093
v	3D	✗	✗	✓	26.19	0.867	0.101
vi	3D+2D	✓	✗	✓	26.41	0.871	0.096
vii	3D+2D	✓	✓	✓	27.34	0.895	0.083

4.2 Ablation and Generalization Analysis

Ablation Study. We evaluate the effectiveness of each component in EGGS in Table 4. Row *i* is the vanilla 3DGS baseline. In row *ii*, we adopt a hybrid 2D/3D representation but rasterize all Gaussians using the 3DGS rasterizer, which leads to performance degradation. Row *iii* incorporates our hybrid rasterizer, which renders Gaussians according to their type. However, this setting still lacks flexibility and regularization. Row *iv* incorporates adaptive type exchange to enhance the flexibility. Rows *v*–*vii* study frequency-based supervision, which provides only limited gains for non-hybrid 3DGS (row *v*) but is more effective in the hybrid setting. The full model in row *vii* achieves the best performance, indicating that decoupled frequencies more effectively exploit the strengths of the exchangeable hybrid representation. We provide more ablation and analysis in Appendix F.

Generalization Analysis. We evaluate the robustness of EGGS in challenging scenarios, including few-shot and out-of-distribution (OOD) settings. Following prior work, we use LLFF [41] for few-shot evaluation [49, 50] and OOD-NVS [51] for OOD evaluation [52]. More details are provided in Appendix A. As shown in Table 11 and Figure 10, EGGS achieves robust performance in both settings, benefiting from its balanced multi-view consistency and appearance fidelity. This indicates that the hybrid representation generalizes better than single-type baselines.

We also emphasize that EGGS serves as a general underlying representation and is compatible with various optimization strategies developed for specialized settings [49, 50, 53, 54, 55, 51]. We discuss these orthogonal techniques in Appendix B, and provide further remarks on limitations and broader impacts in Appendix H.

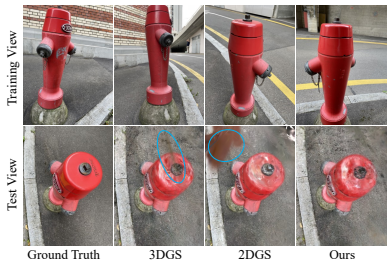


Figure 10: Comparison in the OOD setting.

Table 5: Generalization performance.

Setting	Method	PSNR ↑	SSIM ↑	LPIPS ↓
Few-shot	3DGS	19.52	0.719	0.279
	2DGS	18.50	0.661	0.321
	EGGS	**20.13**	**0.735**	**0.258**
OOD	3DGS	20.21	0.763	0.242
	2DGS	23.52	0.863	0.188
	EGGS	**24.17**	**0.907**	**0.151**

Table 2: Quantitative Geometry Comparison on DTU. Chamfer Distance (CD) is reported per scene. mCD and PSNR denote the mean Chamfer Distance and mean PSNR across all scenes, respectively.

Method	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	mCD	PSNR
3DGS	2.14	1.53	2.08	1.68	3.49	2.21	1.43	2.07	2.22	1.75	1.79	2.55	1.53	1.52	1.50	1.96	32.82
SUGAR	1.47	1.33	1.13	0.61	2.25	1.71	1.15	1.63	1.62	1.07	0.79	2.45	0.98	0.88	0.79	1.33	31.59
2DGS	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80	32.43
GOF	0.50	0.82	0.37	0.37	1.12	0.78	0.73	1.18	1.29	0.71	0.77	0.90	0.44	0.69	0.49	0.74	32.58
Ours	0.65	0.77	0.58	0.53	1.08	1.01	0.96	1.31	1.45	0.72	0.88	1.53	0.67	0.83	0.66	0.91	33.65

5 Conclusion

This paper presents EGGS, a hybrid Gaussian Splatting framework that combines the appearance fidelity of 3D Gaussians with the geometric accuracy of 2D Gaussians. The design integrates Hybrid Gaussian Rasterization for unified rendering, Adaptive Type Exchange for flexible representation, and Frequency-Decoupled Optimization to balance geometry and appearance. EGGS outperforms both 2D- and 3D-only baselines across multiple benchmarks. Future work includes extending the hybrid representation to more diverse and challenging scenarios.

Acknowledgments: This work was supported by Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number 140D0423C0074. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.

References

[1] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21401–21412, 2024.

[2] Pengzhi Li, Chengshuai Tang, Qinxuan Huang, and Zhiheng Li. Art3d: 3d gaussian splatting for text-guided artistic scenes generation. *arXiv preprint arXiv:2405.10508*, 2024.

[3] Linhan Wang, Kai Cheng, Shuo Lei, Shengkun Wang, Wei Yin, Chenyang Lei, Xiaoxiao Long, and Chang-Tien Lu. Dc-gaussian: Improving 3d gaussian splatting for reflective dash cam videos. *arXiv preprint arXiv:2405.17705*, 2024.

[4] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, pages 333–350. Springer, 2022.

[6] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8248–8258, 2022.

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510, 2022.

[8] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18458–18469, 2023.

[9] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022.

[10] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–12, 2023.

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023.

[12] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[13] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. In *Forty-first International Conference on Machine Learning*, 2024.

[14] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024.

[15] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[16] Tian-Xing Xu, Wenbo Hu, Yu-Kun Lai, Ying Shan, and Song-Hai Zhang. Texture-gs: Disentangling the geometry and texture for 3d gaussian splatting editing. In *European Conference on Computer Vision*, pages 37–53. Springer, 2024.

[17] Jingyu Lin, Jiaqi Gu, Lubin Fan, Bojian Wu, Yujing Lou, Renjie Chen, Ligang Liu, and Jieping Ye. Hybridgs: Decoupling transients and statics with 2d and 3d gaussian splatting. *arXiv preprint arXiv:2412.03844*, 2024.

[18] Lihan Jiang, Kerui Ren, Mulin Yu, Linning Xu, Junting Dong, Tao Lu, Feng Zhao, Dahua Lin, and Bo Dai. Horizon-gs: Unified 3d gaussian splatting for large-scale aerial-to-ground scenes. *arXiv preprint arXiv:2412.01745*, 2024.

[19] Yunxiang Zhang, Alexandr Kuznetsov, Akshay Jindal, Kenneth Chen, Anton Sochenov, Anton Kaplanyan, and Qi Sun. Image-gs: Content-adaptive image representation via 2d gaussians. *arXiv preprint arXiv:2407.01866*, 2024.

[20] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *European Conference on Computer Vision*, pages 327–345. Springer, 2024.

[21] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024.

[22] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024.

[23] Shai Avidan and Amnon Shashua. Novel view synthesis in tensor space. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1034–1040. IEEE, 1997.

[24] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021.

[25] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.

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

[27] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024.

[28] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024.

[29] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024.

[30] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024.

[31] Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks. *arXiv preprint arXiv:2412.20522*, 2024.

[32] Rui Xu, Wenyue Chen, Jiepeng Wang, Yuan Liu, Peng Wang, Lin Gao, Shiqing Xin, Taku Komura, Xin Li, and Wenping Wang. Supergaussians: Enhancing gaussian splatting using primitives with spatially varying colors. *arXiv preprint arXiv:2411.18966*, 2024.

[33] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

[34] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024.

[35] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[36] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016.

[37] Christopher E Heil and David F Walnut. Continuous and discrete wavelet transforms. *SIAM review*, 31(4):628–666, 1989.

[38] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.

[39] Junha Hyung, Susung Hong, Sungwon Hwang, Jaeseong Lee, Jaegul Choo, and Jin-Hwa Kim. Effective rank analysis and regularization for enhanced 3d gaussian splatting. *arXiv preprint arXiv:2406.11672*, 2024.

[40] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

[41] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019.

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

[43] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[44] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[46] Harry Barrow, J Tenenbaum, A Hanson, and E Riseman. Recovering intrinsic scene characteristics. *Comput. vis. syst*, 2(3-26):2, 1978.

[47] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21424–21433, 2024.

[48] Gilbert Strang and Truong Nguyen. *Wavelets and filter banks*. SIAM, 1996.

[49] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20775–20785, 2024.

[50] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *European conference on computer vision*, pages 145–163. Springer, 2024.

[51] Yutong Chen, Marko Mihajlovic, Xiyi Chen, Yiming Wang, Sergey Prokudin, and Siyu Tang. Splatformer: Point transformer for robust 3d gaussian splatting. In *The Thirteenth International Conference on Learning Representations*, 2025.

[52] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024.

[53] Ruihong Yin, Vladimir Yugay, Yue Li, Sezer Karaoglu, and Theo Gevers. Fewviewgs: Gaussian splatting with few view matching and multi-stage training. *arXiv preprint arXiv:2411.02229*, 2024.

[54] Wangze Xu, Huachen Gao, Shihe Shen, Rui Peng, Jianbo Jiao, and Ronggang Wang. Mvpgs: Excavating multi-view priors for gaussian splatting from sparse input views. In *European Conference on Computer Vision*, pages 203–220. Springer, 2024.

[55] Zhuowen Shen, Yuan Liu, Zhang Chen, Zhong Li, Jiepeng Wang, Yongqing Liang, Zhengming Yu, Jingdong Zhang, Yi Xu, Scott Schaefer, et al. Solidgs: Consolidating gaussian surfel splatting for sparse-view surface reconstruction. *arXiv preprint arXiv:2412.15400*, 2024.

[56] Wenyan Cong, Yiqing Liang, Yancheng Zhang, Ziyi Yang, Yan Wang, Boris Ivanovic, Marco Pavone, Chen Chen, Zhangyang Wang, and Zhiwen Fan. E3d-bench: A benchmark for end-to-end 3d geometric foundation models. *arXiv preprint arXiv:2506.01933*, 2025.

[57] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024.

[58] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024.

[59] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

Appendix

Table of contents:

A Implementation Details

Training Pipeline and Parameter Setting. Our training setup closely follows 3DGS [11]. We assume camera poses are provided or can be estimated using structure-from-motion (SfM) [35]. Initial sparse point clouds are generated via COLMAP [35, 36]. All methods, including EGGS and baselines, are trained for 30K iterations. Learning rates for Gaussian parameters follow the default configurations from 3DGS and 2DGS. We adopt the densification strategy from 3DGS, which refines Gaussian distributions by pruning or duplicating them in under- or over-reconstructed regions. Densification begins at iteration 500, ends at iteration 15K, and is performed every 100 iterations.

In EGGS, we employ Hybrid Gaussian Rasterization, where each Gaussian is rendered according to its assigned type. Gaussian types are randomly initialized. To enable flexible representation, we introduce Adaptive Type Exchange, which is performed every 500 iterations from step 500 to 30K. During type switching, we set the effective rank threshold θ_e to 2.05. For scale modulation, which allows 2D Gaussians to evolve into 3D Gaussians as the s_z scale becomes more significant, we use $\theta_z = 1.05$ and temperature $T = 0.001$ in Eq. 7. Additionally, we incorporate frequency-based supervision using the Discrete Wavelet Transform (DWT). More details on Frequency-Decoupled Optimization are provided in Appendix F.

Datasets and Evaluation Protocols. We evaluate the performance of EGGS and baselines on several widely used datasets, including Mip-NeRF360 [25], LLFF [41], Tanks&Temples [42], DTU [43], and OOD-NVS [51]. We follow the standard train/test splits used in prior work [25], with additional dataset statistics provided in Table 6. To ensure fair comparison with baselines [11, 12, 28], we downsample input images using the same factors as in [52]. All datasets provide RGB images for evaluating appearance quality. DTU additionally offers ground-truth point clouds for computing Chamfer Distance, while Tanks&Temples provides ground-truth depth maps for depth accuracy evaluation. Beyond standard dense-view evaluation, we also perform evaluation under few-shot settings using 3 views from LLFF, and out-of-distribution (OOD) setting using OOD-NVS.

B Related Works

Following 3DGS [11], a line of work has been proposed to enhance the geometry accuracy and reconstruction quality. As discussed in the related work section, most existing methods are based on

Table 6: Details on the datasets used for evaluation of appearance and geometry.

Dataset	Ground Truth	Evaluation Metric	Evaluation Protocol	Factor
Mip-NeRF 360 (outdoor) [25]	RGB image	Appearance	Standard	4
Mip-NeRF 360 (indoor) [25]	RGB image	Appearance	Standard	2
LLFF [41]	RGB image	Appearance	Standard and Few-shot	8
Tanks&Temples [42]	RGB image; Depth	Appearance and Geometry (F1)	Standard	2
DTU [43]	RGB image; Point Cloud	Geometry (chamfer distance)	Standard	2
OOD-NVS [51]	RGB image	Appearance	OOD	1

a single representation [14, 13, 12, 15, 28, 16, 47]. Here, we further expand on several recent efforts that aim to exploit the advantages of 3D and 2D Gaussians, as summarized in Table 7. We consider a method to be general if the training pipeline follows the standard 3DGS.

HybridGS [17] proposes to combine 3D Gaussians and image-space 2D Gaussians to remove transient objects during reconstruction. However, the 2D Gaussians in HybridGS are defined in the image frame, lacking the multi-view consistency provided by 2DGS [12]. Additionally, HybridGS employs a three-stage training pipeline specifically designed for transient object removal, making it less general than pipelines based on 3DGS. Moreover, since the training code of HybridGS is not publicly available, a direct comparison with EGGS is not feasible.

HorizonGS [18], on the other hand, is built upon ScaffoldGS, where voxel-based MLPs are used to decode Gaussian primitives. HorizonGS generates 3D Gaussians from MLPs for novel view synthesis and 2D Gaussians for surface reconstruction. Thus, it still follows a single-representation scheme. Furthermore, HorizonGS is specifically designed for aerial-ground scenarios and introduces a two-stage training pipeline to address conflicts from varying altitudes. Similar to HybridGS, this design is task-specific and less generalizable.

In contrast to HybridGS and HorizonGS, we note that several recent optimization techniques are more general and adhere to the 3DGS training pipeline, such as ScaffoldGS [22] and 3DGS-MCMC [27]. ScaffoldGS improves efficiency by introducing voxel-based MLPs, while 3DGS-MCMC enhances the densification process by reformulating 3D Gaussians as Markov Chain Monte Carlo (MCMC) samples. These methods are orthogonal to EGGS. Since EGGS serves as a general underlying representation, we believe such orthogonal optimizations can be incorporated to further enhance our framework. Exploring the potential of integrating these techniques with our exchangeable hybrid representation is a promising direction for future work.

In this work, we assume that camera poses are either available or can be estimated using structure-from-motion (SfM) [35], and that initial sparse point clouds can be generated using COLMAP [35, 36]. However, in practice, recovering geometric information such as camera poses and point clouds remains challenging. Recently, 3D Geometric Foundation Models (GFMs) have emerged as a promising approach to improve the generalizability of 3D reconstruction [56]. Feed-forward models such as DUSt3R [57], MASt3R [58], and VGGT [59] can predict robust geometric attributes in a single forward pass, even when the input multi-view images exhibit minimal or no overlap. Incorporating GFMs can potentially enhance the reconstruction quality of our method in open-world scenarios.

Table 7: Comparison of the setting with related works.

Method	Gaussian Type	Setting	Training Pineline
3DGS [11]	3D	General	Single-stage
2DGS [12]	2D	General	Single-stage
HybridGS [17]	3D+2D	Transient	Three-stage
HorizonGS [18]	3D/2D	Varying-altitude	Two-stage
ScaffoldGS [22]	3D	General	Single-stage
3DGS-MCMC [27]	3D	General	Single-stage
EGGS	3D+2D	General	Single-stage

C Differentiable Rasterization for 2D/3D Gaussians Splatting

In this section, we detail the differentiable rasterization procedures used in 3DGS [11] and 2DGS [12], focusing on how the intermediate coordinates in Eq.(1) are computed. In both methods, images are rendered by computing the color at each screen-space pixel x_p from a set of N Gaussians $\{\mathcal{G}_i\}_{i=0}^{N-1}$. The final pixel color is obtained by rasterizing the Gaussians onto the image plane, but the rasterization procedures differ significantly between 3DGS and 2DGS.

3DGS employs an affine approximation to the projective transformation for rasterization. For a 3D Gaussian $\mathcal{G}_i^{3d}$ with type $t_i = 1$, it is first projected onto the image plane via affine projection [11]. The resulting projected Gaussian is denoted as $\mathcal{G}_i^{\text{proj}}$, with 2D center $\boldsymbol{\mu}'_{3d,i}$ and covariance $\boldsymbol{\Sigma}'_i$ in the image plane. The projected covariance is computed as:

$$\boldsymbol{\Sigma}'_i = \boldsymbol{J}\boldsymbol{W}\boldsymbol{\Sigma}_i\boldsymbol{W}^T\boldsymbol{J}^T, \tag{11}$$

where $\boldsymbol{J}$ is the Jacobian matrix of the affine projection, and $\boldsymbol{W}$ accounts for the world-to-camera transformation [11]. Once $\boldsymbol{\Sigma}'_i$ is obtained, the projected center $\boldsymbol{\mu}'_{3d,i}$ can be computed accordingly.

On the other hand, 2DGS applies ray–splat–intersection to rasterize 2D Gaussians. For a 2D Gaussian $\mathcal{G}_i^{2d}$ with $t_i = 0$, it is not directly projected onto the image plane. Instead, to preserve the geometric accuracy of $\mathcal{G}_i^{2d}$, the pixel x_p is unprojected into the local tangent frame defined by the Gaussian [12]. This is done by computing the intersection between the ray passing through x_p and the tangent plane of $\mathcal{G}_i^{2d}$, resulting in the local coordinates:

$$u(x_p) = \frac{h_u^2 h_v^4 - h_u^4 h_v^2}{h_u^1 h_v^2 - h_u^2 h_v^1}, \quad v(x_p) = \frac{h_u^4 h_v^1 - h_u^1 h_v^4}{h_u^1 h_v^2 - h_u^2 h_v^1}, \tag{12}$$

where h_u and h_v are derived from the homogeneous plane equations associated with the pixel $x_p = (x, y)$ as:

$$h_u = (\boldsymbol{W}\boldsymbol{H})^T h_x, \quad h_v = (\boldsymbol{W}\boldsymbol{H})^T h_y. \tag{13}$$

More details about the homogeneous transformation matrices $\boldsymbol{H}$ and $\boldsymbol{W}$ can be found in 2DGS [12]. By solving Eq. (12), we obtain the 2D position of x_p in the tangent frame, denoted as $\mathbf{u}_i(x_p) = (\boldsymbol{u}_i(x_p), \boldsymbol{v}_i(x_p))$. Note that the center $\boldsymbol{\mu}_{2d,i}$ of $\mathcal{G}_i^{2d}$ is defined as the origin of this tangent frame.

With the projected center $\boldsymbol{\mu}'_{3d,i}$ and covariance $\boldsymbol{\Sigma}_i$ for 3D Gaussians, and the intersection coordinates $\boldsymbol{u}_i(x_p)$ and $\boldsymbol{v}_i(x_p)$ for 2D Gaussians, we can perform hybrid rasterization as described in Section 3.1.

D Effective Rank and Type Switch

In this section, we provide additional details about Adaptive Type Exchange. We focus on the effective rank threshold and the design choices behind 3D Gaussian reparameterization and 2D Gaussian scale modulation. We also present statistics on the distribution of Gaussian types over training iterations.

Table 8: Ablation on different erank thresholds. We evaluate the appearance of EGGS on the LLFF dataset with different erank thresholds.

erank threshold	PSNR	SSIM	LPIPS
1.9	26.15	0.871	0.103
1.95	26.23	0.868	0.101
2	26.49	0.874	0.097
2.05	27.24	0.885	0.086
2.1	26.37	0.844	0.113
2.15	25.78	0.831	0.126

Effective Rank Threshold. As described in Section 3.2, we use the effective rank (erank) to assess the mismatch between a Gaussian's assigned type and its actual geometric dimensionality. We set the erank threshold to 2.05 in our experiments and study its effect in Table 8. When a 3D Gaussian becomes increasingly flat and its erank drops below the threshold, it is converted to a 2D Gaussian. Conversely, if a 2D Gaussian's erank exceeds the threshold, it is switched to 3D. A higher threshold causes more 3D Gaussians to be converted to 2D, as more will fall below the threshold. This can degrade performance when the threshold is set too high. In contrast, a lower threshold (e.g., 1.9) results in fewer 3D Gaussians being converted to 2D, causing the model to behave more like the 3DGS baseline and limiting the benefits of hybrid representation.

While the erank metric has been previously introduced [38, 39], our contribution lies in its integration into a dynamic type exchange mechanism for hybrid Gaussian representation. We acknowledge that

erank is a heuristic measure of effective dimensionality and may not behave monotonically in all scenarios. When a Gaussian has scales $(1, 1, s_z)$ with the first two scales fixed, the erank increases with s_z initially but may drop as s_z becomes dominant (e.g., erank returns to 2 when $s_z \approx 2.6$). In such cases, a volumetric Gaussian could technically fall below the threshold θ_e and be converted to 2D. However, we note that such configurations are rare in practice. All three scales are updated jointly during training, and our method includes reparameterization and soft modulation to ensure that type transitions remain stable and consistent with the evolving shape. Although the erank threshold does not come with theoretical guarantees for all edge cases, it demonstrates empirical effectiveness across diverse datasets.

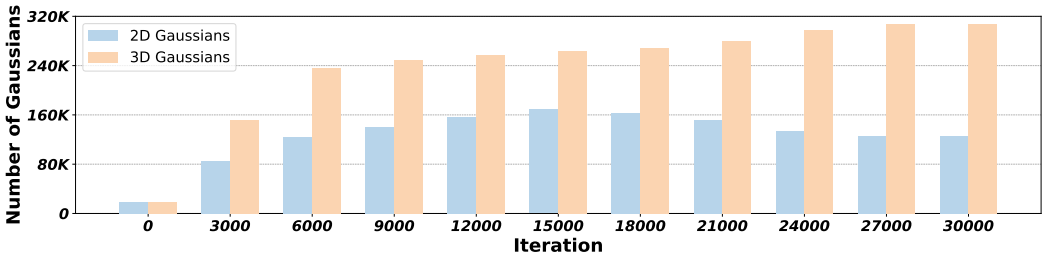


Figure 11: The number 3D and 2D Gaussians in different iterations during training.

Permutation-Based Reparameterization. The key to reparameterization is preserving the covariance during type switching. We achieve this using permutation matrices $\boldsymbol{P}_x$ and $\boldsymbol{P}_y$, designed to be orthogonal with a positive determinant. Specifically, we define:

$$\boldsymbol{P}_x = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{P}_y = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{14}$$

where $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$ and $\boldsymbol{I}$ is the identity matrix, and $\det(\boldsymbol{P}) = 1$. During conversion, the new $\boldsymbol{S}^*$ scaling and rotation $\boldsymbol{R}^*$ are computed using these permutation matrices to ensure that the transformed covariance $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}\boldsymbol{R}^T$ remains unchanged:

$$\boldsymbol{S}^* = \boldsymbol{P}\boldsymbol{S}\boldsymbol{P}^T \text{ and } \boldsymbol{R}^* = \boldsymbol{R}\boldsymbol{P}^T \tag{15}$$

It is easy to see the converted covariance is unchanged:

$$\begin{aligned} \boldsymbol{\Sigma}^* &= \boldsymbol{R}^*\boldsymbol{S}^*\boldsymbol{S}^{*T}\boldsymbol{R}^{*T} \\ &= \boldsymbol{R}\boldsymbol{P}^T \cdot \boldsymbol{P}\boldsymbol{S}\boldsymbol{P}^T \cdot \boldsymbol{P}\boldsymbol{S}^T\boldsymbol{P}^T \cdot \boldsymbol{P}\boldsymbol{R}^T \\ &= \boldsymbol{R} \cdot (\boldsymbol{P}^T\boldsymbol{P}) \cdot \boldsymbol{S} \cdot (\boldsymbol{P}^T\boldsymbol{P}) \cdot \boldsymbol{S}^T \cdot (\boldsymbol{P}^T\boldsymbol{P}) \cdot \boldsymbol{R}^T \\ &= \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T \\ &= \boldsymbol{\Sigma} \end{aligned} \tag{16}$$

In addition to preserving the covariance, the permutation must ensure that the converted rotation matrix $\boldsymbol{R}^*$ has a positive determinant. This is important because $\boldsymbol{R}^*$ is converted into a quaternion during optimization, and most 3D graphics frameworks assume right-handed coordinate systems [46]. A negative determinant implies a reflection, which cannot be represented by a unit quaternion. Given that the original rotation matrix $\boldsymbol{R}$ satisfies $\det(\boldsymbol{R}) > 0$, and the permutation matrices $\boldsymbol{P}_x$ and $\boldsymbol{P}_y$ are orthogonal with $\det(\boldsymbol{P}) = 1$, the determinant of the converted rotation $\boldsymbol{R}^* = \boldsymbol{R}\boldsymbol{P}^T$ is given by: $\det(\boldsymbol{R}^*) = \det(\boldsymbol{R}) \cdot \det(\boldsymbol{P}^T) = \det(\boldsymbol{R}) \cdot \det(\boldsymbol{P}) = \det(\boldsymbol{R}) > 0$. Therefore, $\boldsymbol{P}_x$ and $\boldsymbol{P}_y$ preserve both the covariance and the positive determinant of the rotation matrix, ensuring compatibility with quaternion-based optimization.

Gaussian-Type Distribution. Figure 11 shows the distribution of 2D and 3D Gaussians throughout training. As described in Appendix A, we randomly initialize Gaussian types, resulting in roughly equal numbers of 2D and 3D Gaussians at the start. During optimization, both types increase due to densification, with a more significant rise in 3D Gaussians. This trend can be attributed to the SfM-initialized points already containing geometric structure, allowing 2D Gaussians to capture coarse geometry in early iterations. Notably, the number of 2D Gaussians gradually decreases in

the later stages of training, especially after densification ends at 15K iterations. This suggests that many 2D Gaussians are being converted to 3D Gaussians, likely to better recover under-reconstructed regions and capture finer scene details.

Gaussian-Type Initialization. A key feature of EGGS is its exchangeable representation, which allows each Gaussian primitive to change its type as needed, regardless of its initial type. To verify this capability, we conduct a simple experiment by investigating three initialization scenarios: we initialize all Gaussians as 2D, all as 3D, or with random type assignments, and observe the distribution of Gaussian types throughout training. Below, we show the percentage of 3D Gaussians at different iterations. As shown in Table 9, even when the model is initialized entirely with 2D Gaussians, part of the Gaussians are converted to 3D Gaussians during training, leading to a hybrid model in the final stage. This demonstrates that 2D Gaussians can indeed transition to 3D types during training, despite potentially incorrect initialization, and vice versa.

E Discrete Wavalet Transformation

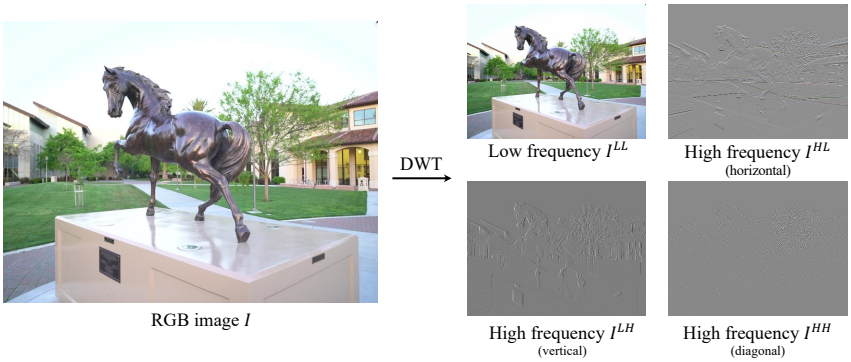


Figure 12: Illustration of the level-1 Discrete Wavalet Transformation.

In this section, we provide additional details on the Discrete Wavelet Transform (DWT) used in our Frequency-Decoupled Optimization. DWT is a widely adopted technique for frequency-domain analysis. Given an image $\mathcal{I}$, DWT decomposes it into four sub-bands: one low-frequency component and three high-frequency components corresponding to horizontal, vertical, and diagonal directions. Formally, we define the low-pass filter matrix $\boldsymbol{L}$ as:

$$L = \begin{bmatrix} \cdots & \cdots & \cdots & & & \\ \cdots & \ell_{-1} & \ell_0 & \ell_1 & \cdots & \\ \cdots & \cdots & \ell_{-1} & \ell_0 & \ell_1 & \cdots \\ & & & \cdots & \cdots & \cdots \end{bmatrix} \tag{17}$$

where $\boldsymbol{\ell}$ is the 1D low-pass wavelet filter. Similarly, the high-pass matrix $\boldsymbol{H}$ is derived from the 1D high-pass wavelet filter $\boldsymbol{h}$. We use orthogonal 1D wavelet filters such that $\boldsymbol{\ell}$ and $\boldsymbol{h}$ are the same [37, 48]. The four sub-bands are computed as:

$$\begin{aligned} \mathcal{I}^{LL} &= L\mathcal{I}L^T; \\ \mathcal{I}^{LH} &= H\mathcal{I}L^T; \\ \mathcal{I}^{HL} &= L\mathcal{I}H^T; \\ \mathcal{I}^{HH} &= H\mathcal{I}H^T; \end{aligned} \tag{18}$$

We provide illustrative example in Figure 12. We extract the low-frequency feature as $\mathcal{I}_l = \mathcal{I}^{LL}$, and the high-frequency component $\mathcal{I}_h$ as the composition of directional details: $\mathcal{I}^{LH}$ (horizontal), $\mathcal{I}^{HL}$ (vertical), and $\mathcal{I}^{HH}$ (diagonal). In our implementation, we use a level-1 Haar filter for the DWT.

Table 9: Effect of 3D Gaussian percentage on reconstruction quality. PSNR denotes the peak signal-to-noise ratio at each iteration step under different initialization strategies.

Iter.	0	3000	6000	9000	12000	15000	18000	21000	24000	27000	30000	PSNR
All 2D initialization	0.0%	19.0%	29.1%	33.7%	35.0%	38.9%	39.8%	41.4%	43.2%	45.7%	47.8%	27.25
All 3D initialization	100.0%	86.2%	75.3%	69.0%	63.9%	59.7%	58.7%	58.0%	57.4%	57.0%	57.0%	27.51
Random initialization	49.9%	58.1%	59.9%	57.3%	55.1%	52.3%	52.5%	52.9%	52.9%	53.1%	54.2%	27.86

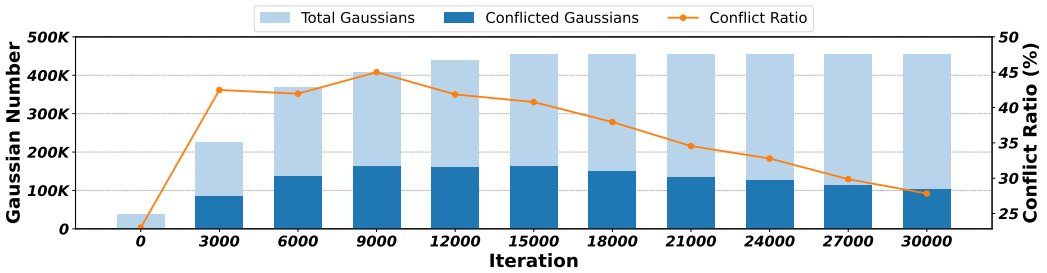


Figure 13: The number of Gaussians with conflicted gradients in different iterations during training.

F Gradient Conflict Analysis in Frequency-Decoupled Optimization.

We provide additional details on Frequency-Decoupled Optimization and Algorithm 1. After applying DWT, both 3D and 2D Gaussians receive gradients from the high-frequency and low-frequency losses. As discussed in Section 3.3, the distinct roles of 3D and 2D Gaussians—where 3D Gaussians prioritize fine detail and 2D Gaussians emphasize geometric structure—can lead to conflicting gradient directions.

We empirically analyze this in Figure 13, where "conflicted Gaussians" are defined as those with negative inner product between low- and high-frequency gradients, i.e., $\mathbf{g}_i^{\text{low}} \cdot \mathbf{g}_i^{\text{high}} < 0$ in Algorithm 1. In early training (e.g., before 15K iterations), about 45% of Gaussians exhibit such conflicts. Although the conflict ratio gradually decreases as training progresses, around 20% of Gaussians still experience conflicts at convergence. This supports our motivation that naively combining $\mathcal{L}_{\text{low}}$ and $\mathcal{L}_{\text{high}}$ as Eq.(8) for all Gaussians provides suboptimal supervision, and underscores the need for the proposed asymmetrical update strategy. In Table 10, we compare different strategies for applying frequency-based supervision. As baselines, we include vanilla 3DGS and EGGS without Frequency-Decoupled Optimization (EGGS w/o FDO), which includes hybrid rasterization and type exchange but no frequency regularization. EGGS w/ DWT applies frequency losses directly as in Eq.(8), yielding only marginal gains due to unresolved gradient conflicts (Figure13). A simple alternative is to mask out conflicting frequency gradients—for example, ignoring high-frequency gradients for 2D Gaussians. We denote this variant as EGGS w/ DWT + mask. While masking helps reduce conflicts, it may discard useful gradient signals. In contrast, our full method achieves the best performance by leveraging gradient projection to suppress only the conflicting components while retaining informative gradients. For theoretical background on gradient projection and conflict resolution, we refer readers to [40].

Table 10: Ablation on different use of the frequency loss. FDO stands for Frequency-Decoupled Optimization.

Method	**PSNR**	**SSIM**	**LPIPS**
3DGS	26.12	0.865	0.099
EGGS w/o FDO	26.58	0.874	0.093
EGGS w/ DWT	26.82	0.877	0.092
EGGS w/ DWT + mask	27.07	0.881	0.089
EGGS	27.34	0.895	0.083

Table 11: Comparison of inference efficiency. We report the average FPS in each dataset.

Method	LLFF	Tanks&Temples	Mip-NeRF360.
3DGS	323	158	145
2DGS	187	59	76
GaussianPro	308	166	121
EGGS	268	125	104

G Inference Efficiency

As shown in Table 11, we compare the rendering efficiency of different methods in terms of frames per second (FPS). During training, the number of parameters significantly impacts performance, as backpropagation and parameter updates are computationally expensive. In contrast, inference

efficiency is primarily determined by the rasterization strategy. 3DGS-based methods, including GaussianPro, employ affine projection-based rasterization, which is efficient but less accurate, resulting in higher FPS at inference. Since both 3DGS and GaussianPro use the same projection-based rasterization pipeline, the difference in their inference speed mainly arises from model size—that is, the number of Gaussians used. While the number of primitives affects performance, its influence remains moderate given the similar scale of models.

In contrast, 2DGS adopts a ray–splat–intersection rasterization pipeline, which provides improved geometric accuracy but is more computationally intensive, resulting in slower rendering. EGGS integrates both 2DGS and 3DGS rasterization strategies in a hybrid manner, achieving a favorable balance between accuracy and efficiency. While EGGS 's FPS is slightly lower than that of 3DGS, it remains significantly faster than 2DGS. Additionally, EGGS benefits from a shorter training time than 3DGS, owing to its reduced model size and more effective optimization dynamics.

H Discussion

Broader Impact. This work introduces an exchangeable hybrid Gaussian splatting framework that improves the trade-off between geometry accuracy and appearance fidelity in neural rendering. By enabling flexible type adaptation and frequency-aware supervision, our method can enhance 3D reconstruction quality in both synthetic and real-world scenarios. Potential applications include autonomous driving, augmented reality, and robotics, where accurate scene geometry and photorealism are both essential. While our approach primarily targets academic benchmarks, it may inform future developments in real-time perception systems.

Limitations. The current initialization of Gaussian types is random and does not incorporate semantic or structural cues from the sparse point cloud, which may limit early-stage optimization. Incorporating semantic priors could improve convergence and final quality. Additionally, as a general-purpose representation, our method has not been explicitly tested under extreme conditions such as low-light environments, highly reflective surfaces, or scenes with significant transient content. Evaluating and adapting the framework to such challenging scenarios may further demonstrate the robustness and versatility of the hybrid representation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction clearly align with the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are explicitly discussed in the discussion section, providing a clear understanding of the work's boundaries.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The assumptions for each theoretical result are clearly stated, and the correctness is analyzed in detail in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method and algorithms are clearly explained, with parameters and experimental settings provided to ensure reproducibility of the main results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The evaluation primarily uses public datasets cited and explained in the experimental setting section, and the code is available in the anonymous repository with sufficient instructions for reproduction.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer:[Yes]

Justification: The experimental setting is clearly explained, including all necessary training and test details to understand the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper reports results as averages across scenes and independent runs, following standard practices in the field and providing appropriate statistical reliability.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computational resources are discussed in the Implementation section, providing sufficient information to reproduce the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research fully conforms with the NeurIPS Code of Ethics in all respects.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The discussion section addresses both potential societal impacts of the work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and models used in the paper are properly cited, with licenses and terms of use respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code is provided in the anonymous URL and includes accompanying documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.