The Explicit Frontier: A Comparative Analysis of 3D Gaussian Splatting and the Quest for Real-Time Radiance Field Rendering

Likhitha H Y[1], Tejaswi H T[2], Thejaswini L Gowda[3], Muskan[4], Mrs. Megha H C[5]
[1,2,3,4,5] *Malnad College of Engineering*

Abstract- The advent of Neural Radiance Fields (NeRF) inaugurated a period of rapid advancement in photorealistic novel view synthesis, defining complex scenes as continuous volumetric functions. However, NeRF's computational demands—requiring hundreds of Multilayer Perceptron (MLP) queries per ray—rendered it fundamentally unsuitable for real-time applications. This report conducts a rigorous comparison of the three primary architectural movements developed to address this computational bottleneck: volumetric baking via the Sparse Neural Radiance Grid (SNeRG), implicit acceleration through multi-resolution hashing (Instant Neural Graphics Primitives, Instant-NGP), and the explicit representation paradigm shift, 3D Gaussian Splatting (3DGS). The analysis demonstrates that while Instant-NGP achieved breakthrough improvements in training speed and interactive rendering, 3DGS, through its utilization of optimized explicit primitives and fast differentiable rasterization, achieves superior fidelity, hyper-real-time rendering speeds, and enhanced practicality, thereby establishing the explicit representation as the current state-of-the-art architecture for deployment in interactive graphics pipelines.

I.NTRODUCTION

1.1. Context and Motivation: The Real-Time Imperative

Novel view synthesis, the process of reconstructing a comprehensive 3D scene representation from a sparse set of 2D input images, is a core objective in computer graphics and computer vision. Achieving photorealistic quality has driven the field, but the critical requirement for real-world applications in domains such as Augmented Reality (AR), Virtual Reality (VR), and advanced visualization systems is achieving high fidelity at interactive, real-time speeds. Traditional methods often struggled to balance geometric detail with photorealistic view-dependent appearance. Initial volumetric neural rendering methods, while solving the fidelity problem, failed to meet the necessary speed requirements, necessitating radical architectural innovation to overcome inherent structural limitations.

1.2. Foundational NeRF and the Computational Bottleneck

The breakthrough achieved by Neural Radiance Fields (NeRF) involved defining a scene as a continuous 5D function learned by a Multi-Layer Perceptron (MLP).[1] This implicit representation maps a 3D coordinate ($\mathbf{x}$) and a 2D viewing direction ($\mathbf{d}$) to a predicted volume density (σ) and emitted color ($\mathbf{c}$).[1] NeRF excels at implicitly encoding both intricate geometric detail and realistic view-dependent appearance.

The fundamental limitation of NeRF, however, resides in its rendering procedure. To accurately synthesize a ray's color, the MLP must be queried hundreds of times along the ray path to approximate the volumetric integral. This dependence on numerous sequential lookups makes the process computationally prohibitive. Rendering a frame at a standard resolution (e.g., 800 $\times$ 800) typically requires approximately a minute on a modern GPU. This sequential computation complexity, scaling with both the number of rays and the density of samples per ray, posed an intractable obstacle to real-time interaction, driving the necessity for wholesale architectural reform across the field.

1.3. Scope of Analysis

The subsequent research trajectory focused on resolving the NeRF latency issue by either modifying

the implicit function, hybridizing it with explicit data structures, or replacing the representation entirely. This report systematically evaluates the effectiveness of the three leading architectural responses:

1. Sparse Neural Radiance Grid (SNeRG): A volumetric baking technique that converts the continuous NeRF function into a compact, optimized explicit grid for faster inference.
2. Instant Neural Graphics Primitives (Instant-NGP): A hybrid method employing multi-resolution hash encoding to dramatically accelerate the MLP lookup process and improve training convergence.[5]
3. 3D Gaussian Splatting (3DGS) : A purely explicit representation optimized through differentiable rasterization, which bypasses the volumetric integration requirement altogether.[6]

The evolution of these techniques demonstrates a fundamental shift in the definition of success in neural rendering: the emphasis moved from purely photorealistic fidelity to interactive photorealism, where orders-of-magnitude gains in training time and rendering Frames Per Second (FPS) became the defining metrics of architectural superiority.

II.FOUNDATIONAL PRINCIPLES OF VOLUMETRIC NEURAL RENDERING

2.1. The 5D Continuous Volumetric Function

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$$

To enable the relatively small MLP to encode complex, high-frequency details, the input 3D coordinates are transformed using positional encoding.[7] Since fully-connected deep networks are naturally biased toward learning low frequencies faster, this mapping mitigates that bias.[7] However, this encoding increases the dimensionality of the MLP input, compounding the computational expense of hundreds of sequential queries required for each ray.

2.2. Differentiable Volume Rendering

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Here, $T(t)$ is the accumulated transmittance, which represents the probability that the light ray can travel unimpeded from the near bound t_n to the point t. Because the transmittance $T(t)$ is inherently differentiable, the entire rendering pipeline is end-to-end differentiable, allowing the MLP weights Θ to be optimized based purely on a set of input images and their known camera poses.7

2.3. Optimization Challenges and Speed Metrics

The efficacy of differentiable volume rendering necessitates dense sampling along the ray path to accurately approximate the integral. The critical observation is that the structural inefficiency of NeRF lies not in its conceptual representation, but in the repeated execution of the computationally expensive step: querying a large MLP hundreds of times per ray. This fundamental inefficiency created the architectural imperative for acceleration. Consequently, success in novel view synthesis quickly became defined by quantitative metrics focused on speed. The ultimate goal shifted from simply achieving photorealism to achieving interactive photorealism. This mandated new success criteria: not just marginal gains in image quality (PSNR, SSIM, LPIPS) but multi-order-of-magnitude reductions in training time and rendering latency, enabling a minimum of 30 FPS for interactive applications.

III.ACCELERATION WAVE I: IMPLICIT/EXPLICIT HYBRID APPROACHES

The first generation of solutions sought to maintain the implicit representation benefits of NeRF while mitigating its core latency problem through hybrid structures or deferred computation.

3.1. Sparse Neural Radiance Grids (SNeRG): Baking the Implicit Scene

The Sparse Neural Radiance Grid (SNeRG) approach, detailed in "Baking Neural Radiance Fields for Real-Time View Synthesis" , provided the first successful method for achieving real-time rendering from a trained NeRF. The core strategy is to pre-compute the trained continuous function and store it in a discrete, compact structure, termed the Sparse Neural Radiance Grid (SNeRG).

This required a reformulation of the standard NeRF architecture into a "Deferred NeRF." The MLP is restructured to separate view-dependent effects from

geometry and diffuse color. Instead of outputting the final view-dependent RGB color, the Deferred NeRF outputs the volume density (σ), a diffuse RGB color ($\mathbf{c}_d$), and a 4-dimensional feature vector ($\mathbf{v}_s$).

The SNeRG representation itself is a sparse 3D voxel grid containing these precomputed values (σ, $\mathbf{c}_d$, and $\mathbf{v}_s$) for only the *active voxels*. To enforce efficiency, an opacity regularizer (Cauchy loss) is added during training, penalizing predicted density and ensuring opacity is concentrated around surfaces. This sparsity is vital, leading to a compact representation averaging less than 90 MB per scene. Furthermore, the representation is compressed using techniques like 8-bit quantization and encoding the texture atlas using JPEG or H264, achieving compression rates up to $230\times$ for synthetic scenes.

The real-time performance comes from the rendering phase. Ray marching is accelerated through the sparse grid. Crucially, the accumulated feature vector ($\mathbf{v}_s$) along the ray is passed through a lightweight auxiliary MLP (MLP_{Φ}) only once per pixel to generate the view-dependent residual color. This deferred computation avoids the hundreds of MLP queries per ray typical of standard NeRF, enabling rendering speeds exceeding 30 frames per second on commodity hardware.

3.2. Instant Neural Graphics Primitives (Instant-NGP): The Hashed Acceleration

Instant-NGP significantly advanced the state of the art by tackling both the slow training and rendering issues simultaneously through a combination of algorithmic and hardware-specific optimizations.[5] Instant-NGP is built upon three core pillars: an improved ray marching scheme using occupancy grids, a smaller fully-fused neural network, and its main contribution, the Multi-Resolution Hash Encoding (MRHE).[5]

The MRHE is designed to offload the high-frequency modeling burden from the MLP to a highly parallel explicit data structure.[8] For any given 3D coordinate, the algorithm finds the surrounding voxels at multiple resolution levels (L) and hashes their vertices.[5] These hashed vertices serve as keys to look up trainable feature vectors. The feature vectors are then linearly interpolated based on the coordinate's position, concatenated with the viewing direction (encoded via spherical harmonics), and passed to a very small, four-layer, 64-neuron MLP.[5] The resulting architecture effectively shifts the signal representation capability from the sequentially processed MLP to the parallelized hash table lookups, achieving training convergence in a matter of seconds.[9]

Instant-NGP's speed is facilitated by implementing the entire system using fully-fused CUDA kernels, which minimize wasted bandwidth and maximize parallelism on modern GPUs.[9] The improved ray marching scheme also skips sampling over empty space or behind high-density areas using multiscale occupancy grids, speeding up sampling by 10x to 100x compared to naive approaches.[5] Instant-NGP achieved impressive fidelity, registering average PSNRs of 25.51 dB and SSIMs of 0.684 on benchmark scenes.[11]

However, this speed comes with trade-offs. The learned 3D hash feature embeddings, while crucial for high-quality fitting, account for over 99% of the total storage size, leading to a non-negligible memory footprint. Furthermore, the hash table structure introduces complexity in managing hash collisions, though the multi-resolution nature helps disambiguate these errors.[8] The reliance on a large set of hyperparameters for effective tuning also stems from the addition of this grid structure.[8]

Table 1 provides a structural comparison of these early acceleration methods against the fundamental NeRF architecture.

Table 1: Architectural Comparison of Real-Time Radiance Field Techniques

Feature	Classic NeRF (Baseline)	Sparse Neural Radiance Grid (SNeRG)	Instant-NGP	**3D Gaussian Splatting (3DGS) **
Representation Type	Implicit (MLP	Pre-computed	Hybrid (Hashed	Explicit (Optimized

	weights)	Implicit (Baked Grid)	Features + MLP)	Primitives)
Core Structure	Deep Fully Connected MLP	Sparse Voxel Grid + Deferred MLP_{Φ}	Multi-Resolution Hash Encoding	Set of Anisotropic 3D Gaussians
View Dependence Handling	Per-Sample MLP Query	Single MLP Query per Ray (Deferred)	Per-Sample MLP Query (Faster)	Spherical Harmonics (per Gaussian) [6]
Training Time	Hours/Days	Hours	Seconds/Minutes [10]	Minutes (Fastest Convergence) [12]
Rendering Process	Differentiable Volume Rendering	Accelerated Ray Marching (Grid)	Ray Marching (Occupancy Grid) [5]	Differentiable Rasterization [6]

The acceleration wave initiated by SNeRG and Instant-NGP represented two distinct strategies: SNeRG focused on optimizing *rendering* efficiency post-training by shifting computation from numerous 3D samples to a single 2D pixel evaluation, while Instant-NGP focused on both *training and rendering* efficiency by shifting the representation complexity from the sequential MLP to highly parallelized, hardware-native lookups. Crucially, both methods retained the core structure of volumetric sampling and integration, which remained the fundamental impediment to achieving truly hyper-real-time performance.

IV.THE EXPLICIT REVOLUTION: 3D GAUSSIAN SPLATTING

The development of 3D Gaussian Splatting (3DGS) marked a definitive break from the volumetric integration model that constrained NeRF and its derivatives. By adopting a representation built upon explicit, discrete primitives, 3DGS enabled high fidelity while transitioning the rendering task to highly efficient, GPU-native differentiable rasterization.[6]

4.1. Core Representation: Anisotropic 3D Gaussians

3DGS represents a scene not as a continuous density field, but as a set of optimized, discrete 3D Gaussian distributions.[6] Each individual Gaussian i is parameterized by seven primary components: its 3D position ($\mathbf{p}_i$); its volume opacity (α_i); its view-dependent color, encoded using spherical harmonic (SH) coefficients; and a 3x3 anisotropic covariance matrix (Σ_i) that defines the primitive's shape and orientation in 3D space.[6] The ability to optimize this anisotropic covariance matrix is instrumental in accurately capturing high-frequency scene details and geometric complexity.

The shift to this representation inherently leads to improved memory efficiency compared to NeRF's volumetric grids.[13]

4.2. Optimization and Density Control

The 3DGS optimization pipeline begins with a sparse point cloud, typically derived from the initial Structure-from-Motion (SfM) camera calibration process.[6] The key to achieving both speed and quality is the interleaved optimization and density control strategy applied to the Gaussians.[6]

This strategy actively regulates the number of primitives, preventing computational redundancy while maximizing fidelity. It encompasses two main operations: Densification, which adds new Gaussians in regions where the reconstruction error gradient is high, and Pruning, which removes overly transparent or highly redundant Gaussians to improve efficiency.[6]

Further research, such as FastGS, has demonstrated that regulating the number of Gaussians based on multi-view consistency provides superior control over the optimization process, leading to substantial training acceleration—up to a $15.45\times$ speedup compared to vanilla 3DGS on specific datasets, while maintaining comparable rendering quality.[12]

4.3. Differentiable Rasterization

The revolutionary speed of 3DGS stems directly from its rendering strategy, which substitutes complex ray marching and volumetric integration with a fast, visibility-aware, differentiable Gaussian splatting process.[6]

In this process, each 3D Gaussian is first projected onto the 2D image plane, resulting in a 2D ellipse. These projected Gaussians are then sorted by depth, and the final pixel color is determined by blending the overlapping primitives using α-blending.[6] This approach is inherently compatible with existing GPU rasterization pipelines.

By framing the rendering problem in this manner, 3DGS successfully transfers the computational load to highly parallelizable graphics hardware operations, allowing for speeds often exceeding 100 FPS. This fundamental change from calculating continuous volumetric integration to efficiently compositing discrete, optimized primitives is what enables 3DGS to achieve hyper-real-time performance and overcome the inherent latency of volumetric neural rendering.[13]

V.METHODOLOGY FOR QUANTITATIVE PERFORMANCE EVALUATION

The comprehensive evaluation of novel view synthesis techniques must utilize standardized metrics that quantify both the fidelity of the reconstructed scene and the efficiency of the underlying architecture.

5.1. Fidelity Metrics: PSNR, SSIM, and LPIPS

Fidelity assessment relies on comparisons between the synthesized image and the ground truth images:

- Peak Signal-to-Noise Ratio (PSNR): PSNR is a traditional, color-wise metric used to assess the quality of reconstructed images, evaluating the difference based on the Mean Squared Error (MSE).[14] A higher PSNR score correlates with better image quality.[14]
- Structural Similarity Index Measure (SSIM): SSIM is designed to better align with the Human Visual System (HVS). It considers perceived changes in structural information, luminance, and contrast.[14] SSIM is computed across image patches, which offers robustness to slight misalignments between the predicted and reference images, a frequent issue due to calibration differences.[14] An increase in SSIM typically reflects a superior capability in reconstructing fine details.[15]
- Learned Perceptual Image Patch Similarity (LPIPS): LPIPS is a modern perceptual metric that uses features extracted by a Convolutional Neural Network (CNN) (often pretrained on image classification tasks) to quantify human-perceived similarity.[14] It provides a more accurate measure of visual quality than traditional pixel-based methods. Lower LPIPS scores denote higher perceived similarity.

5.2. Efficiency Metrics: Training Time, Storage, and Rendering Speed

Efficiency defines the practicality of a method for real-world deployment:

- Training Time: This is the duration required for the model to reach acceptable convergence, typically measured in seconds, minutes, or hours on standard hardware. The reduction from hours (NeRF) to minutes (3DGS) represents a crucial advance.[10]
- Rendering Speed (FPS): Frames Per Second (FPS) measures the rate at which novel views can be generated. Real-time performance is generally established at ≥ 30 FPS, with speeds exceeding 100 FPS being classified as hyper-real-time.[2]
- Model Compactness/Storage: Measured in megabytes (MB), this metric assesses the storage footprint of the final scene representation. Compactness is essential for deployment and streaming applications.

VI.COMPARATIVE ANALYSIS AND EXPERIMENTAL RESULTS

6.1. Fidelity and Perceptual Quality Benchmarks

The progression of techniques shows a continuous pursuit of higher fidelity, evidenced by improvements in perceptual metrics. Instant-NGP, leveraging its high-frequency hash encoding, achieved competitive results, demonstrated by scores such as 25.51 dB PSNR and 0.684 SSIM on standard scene benchmarks.[11]

However, the state-of-the-art results are now primarily captured by 3DGS. Methods employing similar advanced density control and explicit representations have demonstrated superior performance across the board, notably achieving higher SSIM scores than the most recent few-shot NeRF techniques.[15] The noted increase in SSIM is a strong indicator of the superior capability of explicit and highly-optimized hybrid methods in reconstructing fine geometric and textural details compared to purely implicit volumetric approaches.[15]

6.2. Efficiency Benchmarks: Speed and Training Convergence

The most profound differences among the methods lie in their efficiency metrics, particularly the transformation of training and rendering speed.

The rendering time for Classic NeRF was prohibitive, requiring roughly a minute per frame. SNeRG successfully crossed the real-time threshold by achieving over 30 FPS through deferred computation. Instant-NGP achieved a massive leap in training efficiency, reducing convergence time to seconds or less than a minute for some datasets, while achieving high real-time FPS.[9]

3D Gaussian Splatting further accelerates this trend. By replacing ray integration with differentiable rasterization, 3DGS achieves hyper-real-time rendering speeds, often exceeding 100 FPS. Furthermore, optimized 3DGS variants, utilizing mechanisms like multi-view consistency for densification, report significant training acceleration, confirming that the explicit representation paradigm allows for rapid convergence alongside superior rendering performance.[12]

6.3. Memory and Compactness Trade-offs

The memory footprint reveals key trade-offs in the accelerated architectures. SNeRG prioritizes compactness, achieving scene sizes under 90 MB through explicit sparsity enforcement, 8-bit quantization, and advanced compression techniques. This compactness enhances its deployability.

Instant-NGP, conversely, sacrifices compactness for speed. While the MLP is small, the learned 3D hash feature embeddings, which hold the scene information, account for over 99% of the total storage size. This results in a non-negligible memory footprint that limits deployment on resource-constrained devices. 3DGS , as an explicit primitive-based representation, scales memory according to geometric complexity, generally offering a more scalable solution than dense volumetric grids.

Table 2 synthesizes the comparative performance across these generations of architectures.

Table 2: Comparative Benchmarks: Fidelity and Efficiency

Method	PSNR (dB, Avg.)	SSIM (Avg.)	LPIPS (Avg.)	Training Time (Typical)	Rendering Speed (FPS)
Classic NeRF	25-27	0.85-0.90	0.10-0.20	Hours to Days	Very Low (~0.1 - 1 FPS)
SNeRG	High (Comparable to NeRF)	High	Low	Hours	Real-Time (>30 FPS)
Instant-NGP	High (e.g., 25.51) [11]	High (e.g., 0.684) [11]	Medium (e.g., 0.398) [11]	Seconds to Minutes [10]	High Real-Time (Tens to Hundreds)
**3DGS **	Highest (State-of-the-Art)	Highest (Superior Detail) [15]	Lowest (Best Perceptual Quality)	Minutes (Rapid Convergence) [12]	Hyper Real-Time (>100 FPS)

VII.DISCUSSION AND FUTURE DIRECTIONS

7.1. Critical Analysis of Architectural Trade-offs

The evolution examined in this report demonstrates a clear technological progression driven by the need to externalize and parallelize the bottleneck computation of NeRF. The inadequacy of NeRF's speed was a direct result of its structural reliance on iterative volumetric sampling and complex MLP lookups.

SNeRG provided the first viable escape route by showing that computation could be deferred from hundreds of 3D queries per ray to a single 2D query per pixel. Instant-NGP provided dramatic speedup by showing that the representation itself could be highly parallelized through hash encoding, shifting the learning burden from the slow, sequential MLP to fast, explicit feature lookups.[9]

However, the major advancement of 3DGS is the realization that volumetric integration itself is the core constraint that must be removed. By successfully transforming the scene representation from a mathematically complex continuous field into a set of discrete, manipulable primitives, 3DGS enabled the adoption of differentiable rasterization.[6] This approach represents a highly effective technological crossover, moving the computation from the general-purpose deep learning pipeline into the domain of highly optimized, specialized graphics processing units (GPUs). Consequently, the speed scales efficiently with screen space complexity rather than volumetrically with ray density, which is why 3DGS achieves unparalleled rendering performance.[13]

7.2. Remaining Challenges and Outlook

While 3DGS represents a peak in static scene reconstruction, several critical challenges persist that define the trajectory of future research:

The challenge of handling dynamic scenes remains significant. NeRF variants require retraining for changes [13], and 3DGS must continually manage a massive, changing set of Gaussians in real time, necessitating ongoing optimization of density control mechanisms.[12] Similarly, scaling these representations to truly large, unbounded environments is complex, demanding highly efficient data management for both explicit primitives (3DGS) and sparse data structures (Instant-NGP).

Furthermore, while Instant-NGP provided unprecedented training speed, it and related methods can struggle with robust reconstruction when input images are extremely sparse (e.g., ten views), a domain where some specialized volumetric NeRF variants show stronger generalization.[10] Lastly, model compactness remains a concern. The speed of both Instant-NGP and 3DGS is predicated on large underlying data structures (hash embeddings or the Gaussian point cloud). Just as compression is required for Instant-NGP's substantial feature embeddings , advanced compression and streaming techniques will be essential for the widespread deployment of 3DGS assets in consumer-grade devices.

7.3. Conclusion

The rigorous comparative analysis of volumetric and explicit neural rendering techniques confirms a decisive victory for the explicit paradigm in the context of real-time performance. The original NeRF, while achieving photorealism, was structurally too slow due to its requirement for dense volumetric sampling and iterative MLP querying.

The first generation of accelerators, SNeRG and Instant-NGP, introduced powerful hybrid solutions: SNeRG demonstrated efficient rendering via deferred computation , and Instant-NGP revolutionized training speed via multi-resolution hash encoding.[9]

However, 3D Gaussian Splatting fundamentally solved the core bottleneck by dispensing entirely with volumetric integration. By leveraging optimized explicit 3D primitives and highly parallel differentiable rasterization, 3DGS simultaneously achieved superior visual fidelity (evidenced by high SSIM metrics) and hyper-real-time rendering speeds, positioning it as the leading architectural solution for integrating neural scene representations into contemporary interactive graphics pipelines.

REFERENCES

[1] NeRF: representing scenes as neural radiance fields for view synthesis - ResearchGate, accessed on November 9, 2025, https://www.researchgate.net/publication/3574987

45_NeRF_representing_scenes_as_neural_radiance_fields_for_view_synthesis
[2] Baking Neural Radiance Fields for Real-Time ... - CVF Open Access, accessed on November 9, 2025, https://openaccess.thecvf.com/content/ICCV2021/papers/Hedman_Baking_Neural_Radiance_Fields_for_Real-Time_View_Synthesis_ICCV_2021_paper.pdf
[3] Spatial Computing 101: NeRFs vs. Gaussian Splatting - Vidya Technology, accessed on November 9, 2025, https://vidyatec.com/blog/spatial-computing-101-nerfs-vs-gaussian-splatting/
[4] Instant-NGP - nerfstudio, accessed on November 9, 2025, https://docs.nerf.studio/nerfology/methods/instant_ngp.html
[5] 3D Gaussian Splatting for Real-Time Radiance Field Rendering - Inria, accessed on November 9, 2025, https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/
[6] NeRF: Neural Radiance Fields - Matthew Tancik, accessed on November 9, 2025, https://www.matthewtancik.com/nerf
[7] A New Perspective To Understanding Multi-resolution Hash Encoding For Neural Fields, accessed on November 9, 2025, https://arxiv.org/html/2505.03042v1
[8] Instant Neural Graphics Primitives with a Multiresolution Hash ..., accessed on November 9, 2025, https://nvlabs.github.io/instant-ngp/
[9] A Critical Analysis of NeRF-Based 3D Reconstruction, accessed on November 9, 2025, https://air.uniud.it/retrieve/69effe8f-d971-4eaa-88f8-14d8100fc549/remotesensing-15-03585-v2.pdf
[10] NerfBaselines, accessed on November 9, 2025, https://nerfbaselines.github.io/
[11] [2511.04283] FastGS: Training 3D Gaussian Splatting in 100 Seconds - arXiv, accessed on November 9, 2025, https://arxiv.org/abs/2511.04283
[12] The Battle For Realism in 3D Rendering: A Brief Overview of NeRFs vs. Gaussian Splatting | by Aahana | Antaeus AR | Medium, accessed on November 9, 2025, https://medium.com/antaeus-ar/the-battle-for-realism-in-3d-rendering-a-brief-overview-of-nerfs-vs-gaussian-splatting-580cff4d8801
[13] What are the NeRF Metrics? - Radiance Fields, accessed on November 9, 2025, https://radiancefields.com/what-are-the-nerf-metrics
[14] SGCNeRF: Few-Shot Neural Rendering via Sparse Geometric Consistency Guidance, accessed on November 9, 2025, https://arxiv.org/html/2404.00992v3