GNN-Based Neural-GASh: Radiance Transfer Prediction for Real-Time Rendering using Neural Networks and Geometric Algebra

Efstratios Geronikolakis, Manos Kamarianakis, Antonis Protopsaltis and George Papagiannakis

Abstract. Precomputed Radiance Transfer (PRT) has long been a cornerstone of real-time rendering, enabling high-fidelity global illumination under complex lighting. However, traditional PRT suffers from prohibitive precomputation costs, limited scalability to high-resolution meshes, and impracticality for dynamic or animated scenes—severely constraining its use in interactive applications. Three key challenges arise in particular: (i) the expensive computation of per-vertex spherical harmonic (SH) coefficients, (ii) the lack of adaptability to changing geometry and lighting, and (iii) the memory and latency constraints that hinder deployment in real-time pipelines. Recent advances in neural rendering and neural radiance field (NeRF) research have demonstrated the potential of deep learning networks to model complex light transport efficiently. Yet, these methods are typically restricted to image-space rendering and remain poorly suited for SH-based lighting representations or integration within mesh-based pipelines. In parallel, Conformal Geometric Algebra (CGA) has gained attention in computer graphics for its ability to represent geometric entities and transformations in a unified algebraic framework, offering a mathematically consistent way to encode spatial and lighting relationships directly on meshes. To address these limitations, we introduce Neural-GASh, a neural rendering framework that reformulates PRT within a geometric deep learning paradigm. By encoding vertex–normal pairs as CGA multivectors and utilizing neural networks to predict radiance transfer coefficients, we bypass the costly precomputation while maintaining physical interpretability. The contributions of this work are threefold. First, we show that CGA-based encodings improve training stability and expressiveness by utilizing multivectors that capture spatial and angular relationships more richly than Euclidean inputs. Second, we show that graph neural networks (GNNs), leveraging mesh connectivity, capture self-occlusion and shadowing effects that MLP-based surrogates fail to reproduce. Third, we integrate the trained GI models into Unity for

Corresponding Author: kamarianakis@uoc.gr.

real-time rendering, for both static and animated assets. Experimental evaluation across diverse 3D meshes confirms that Neural-GASh reduces precomputation by orders of magnitude while preserving high-quality illumination, including complex shadowing. The framework scales to meshes with tens of thousands of vertices, and the Unity prototype demonstrates its feasibility in interactive applications. In summary, this work establishes GNN-based PRT as a scalable and effective alternative to classical precomputation, laying groundwork for future neural rendering pipelines in graphics and virtual reality.

Mathematics Subject Classification (2010). Primary 68U05.

Keywords. Precomputed Radiance Transfer, Neural Rendering, Graph Neural Networks, Conformal Geometric Algebra, Spherical Harmonics, Geometric Deep Learning, Unity Integration, Virtual Reality.

1. Introduction

Global Illumination (GI) simulates light transport to capture both direct and indirect illumination, encompassing phenomena such as reflection, refraction, and interreflections. Unlike local models [5], GI is essential for physical plausibility, as illustrated in Figure 1. The theoretical foundation is the rendering equation [18, 20], which describes the equilibrium of radiance but remains computationally intractable for most practical cases.

$$L_o(x, \hat{v}) = L_e(x, \hat{v}) + \int_{\Omega^+} f_r(x, \hat{\omega}, \hat{v})\, L_i(x, \hat{\omega})\, (\hat{\omega} \cdot \hat{n})\, d\hat{\omega}, \tag{1}$$

Equation 1 expresses outgoing radiance L_o as the sum of emitted radiance L_e and the integral of incident radiance L_i modulated by the BRDF f_r and the geometric term over the hemisphere Ω^+. While offline Monte Carlo methods—such as path tracing [11], bidirectional path tracing [22], and photon mapping [19]—can approximate this integral to produce photorealistic results, they are often too slow for real-time applications.

The inclusion of GI introduces indirect illumination, soft shadows, and interreflections, producing more realistic and visually coherent results [20, 7]. However, real-time rendering, particularly for VR/AR, demands strict frame-time budgets (30–120 FPS). This constraint has motivated research into efficient approximations such as Precomputed Radiance Transfer (PRT), which enables the rendering of complex light interactions in real time.

1.1. Precomputed Radiance Transfer (PRT)

To bridge the gap between physical accuracy and real-time performance, Sloan et al. introduced PRT [34]. PRT assumes a static scene and precomputes light transport, storing the transfer function in a basis representation, typically SH. This reduces the shading integral to a dot product (diffuse) or matrix

(A)

(B)

FIGURE 1. Comparison of a computer-generated living room scene rendered (a) with GI, and (b) with baked GI. In (a), Neural-GASh correctly shades the dynamic doctor model with self-shadows, unlike (b) Unity's static GI bake which relies on sparse light probes, leaving the character flatly lit.

operation (glossy), allowing for dynamic lighting with low runtime cost. The transfer function $T(\mathbf{x}, \omega_i)$ is projected onto SH basis functions Y_{lm}:

$$T(\mathbf{x}, \omega_i) \approx \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{lm}(\mathbf{x})\, Y_{lm}(\omega_i) \tag{2}$$

Here, $c_{lm}(\mathbf{x})$ are precomputed coefficients capturing visibility and material properties. During runtime, these are combined with the lighting environment coefficients L_{lm}:

$$L_o(\mathbf{x}, \omega_o) \approx \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{lm}(\mathbf{x})\, L_{lm} \tag{3}$$

1.2. Limitations and Motivation

Despite its efficiency, PRT suffers from significant limitations. First, it assumes static geometry; any mesh deformation or destruction invalidates the precomputed data. Second, the use of low-order SH limits high-frequency details like sharp shadows. Third, the offline precomputation phase is computationally expensive. For a simple geometry (672 vertices), precomputation takes ∼100 seconds, while a complex model (49k vertices) requires ∼3 hours [15]. These scalings render classic PRT impractical for large-scale or user-modifiable content. Furthermore, no complete, efficient PRT implementation currently exists within the Unity engine [1].

Recent neural approaches have attempted to address these gaps. Methods like NeuralPRT [28] and DeepPRT [23] utilize Multilayer Perceptrons (MLPs) to approximate radiance transfer. However, these often operate in image space or lack integration with game engines. Similarly, NeRFs [3] and 3D Gaussian Splatting [21] achieve high visual fidelity but typically remain static and image-centric, lacking the geometry-awareness required for traditional 3D pipelines.

1.3. Toward a Neural Formulation of PRT

To overcome the static nature and precomputation bottleneck of traditional PRT, we previously introduced a preliminary MLP-based approximation, called *Neural-GASh* [15], at the CGI 2025 ENGAGE workshop. While that study validated the utility of CGA for encoding vertex states, the MLP-based model treated vertices independently, lacking the topological awareness required for self-shadowing.

In this work, we present the **GNN-based Neural-GASh** framework, a new geometry-aware reformulation of the MPL-based iteration, that couples the CGA encoding with Graph Neural Networks (GNNs), to explicitly reason over mesh connectivity. Unlike the previous iteration, the new framework enables the model to analyze local neighborhoods and infer complex occlusion relationships.

This approach provides a scalable, real-time alternative to PRT that supports dynamic interactions and is deployable within the Unity game engine. By bridging physically-based rendering with geometric deep learning, the proposed framework establishes a path toward robust neural GI for interactive XR environments.

2. Problem Formulation

This section reformulates the task of GI as a learning-based problem, extending traditional PRT into a neural, geometry-aware framework. The objective is to preserve the theoretical rigor of PRT while eliminating its primary bottleneck: the expensive, per-mesh precomputation that restricts it to static scenes.

The defined the *Neural-GASh* framework, maps CGA encodings of surface geometry directly to SH coefficients. This approach bypasses the need

for runtime integration, enabling real-time radiance prediction consistent with diverse and dynamic topologies.

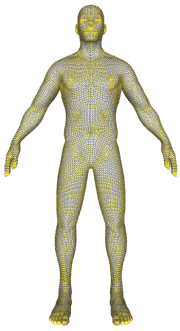
2.1. PRT Complexity and Problem Definition

For each vertex v with normal n, the baseline PRT algorithm computes a lighting-independent SH transfer vector $\mathbf{t}^\star(v) \in \mathbb{R}^9$. An HDR environment map is projected into SH form as $\mathbf{L} \in \mathbb{R}^{9\times 3}$. The diffuse RGB irradiance at vertex v is then given by:

$$\mathbf{C}(v) = \text{albedo}(v) \odot \big(\mathbf{t}(v)^\top \mathbf{L}\big).$$

The core challenge is that computing $\mathbf{t}^\star(v)$—specifically for shadowed transport—requires evaluating visibility integrals over the hemisphere for every vertex.

To quantify this cost, we benchmarked our Python-based PRT implementation across meshes of increasing complexity. Table 1 and Figure 2 illustrate the scaling behavior. While the unshadowed PRT scales linearly, the shadowed PRT incurs significantly higher costs due to ray–triangle intersection tests. Even with parallelization, calculating shadowed transfer for models with ∼70K vertices exceeds an hour, rendering classical methods intractable for real-time or dynamic applications. However, we leverage this high-fidelity offline simulation to generate the ground-truth training corpus, treating the slow computation as a one-time dataset generation cost.

Vertices	**Triangles**	**Complexity**	**Unshadowed (s)**	**Shadowed (s)**
2868	5862		25.50	116.97
24461	48918		216.87	1408.67

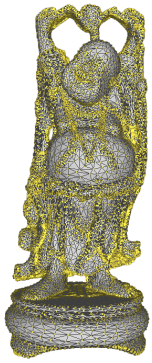
49990	100000		445.91	2963.69
72027	144046		639.18	5009.21

Table 1: Precomputation times of the Python PRT implementation for meshes of varying vertex counts. Shadowed PRT incurs significantly higher cost due to ray–triangle visibility queries.

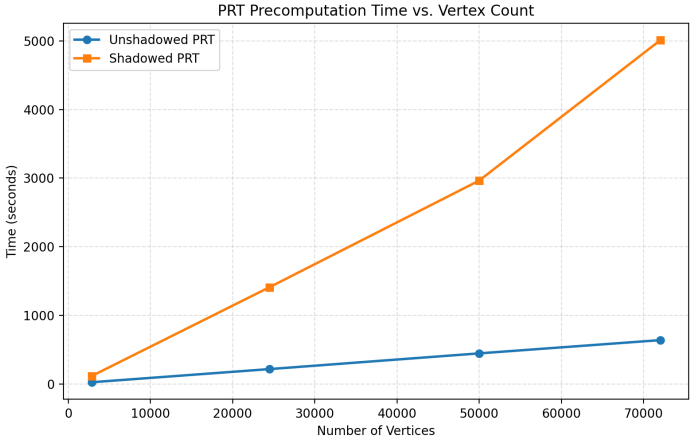


FIGURE 2. Scaling of unshadowed vs. shadowed PRT precomputation time with respect to vertex count. While both grow with complexity, shadowed PRT exhibits significantly steeper growth due to visibility tests.

Given this computational barrier, we formulate the problem as a regression task: approximating the mapping $(v, n) \mapsto \mathbf{t}(v)$ such that the predicted coefficients closely reproduce the ground truth $\mathbf{t}^\star(v)$ efficiently.

2.2. Geometric Formulation via CGA

To ensure the learned model generalizes across arbitrary geometries, we represent each surface element (v, n) as a 32-dimensional conformal multivector $\mathbf{m}(v, n) \in \mathbb{R}^{32}$ within the algebra $\mathbb{R}_{4,1}$. This embedding captures both spatial position and orientation in a unified algebraic structure, allowing geometric transformations to be expressed as single operations while preserving metric relationships.

Conformal Encoding of Geometry. To unify position and orientation, we construct a generic rigid body motion—a *Motor*—within the CGA $\mathbb{R}_{4,1}$. A motor M is the geometric product of a *Translator* T and a *Rotor* R, defined as $M = TR$.

Rotor (Orientation): We adopt the *half-angle* method for efficient rotor construction. Given a vertex normal $\mathbf{n}$ and the canonical up-vector $\mathbf{e}_y = (0, 1, 0)$, we define the half-angle vector $\mathbf{n}' = (\mathbf{n} + \mathbf{e}_y)/\|\mathbf{n} + \mathbf{e}_y\|$. The rotor R is obtained via the geometric product $R = \mathbf{n}'\mathbf{e}_y$. Decomposing R yields the coefficients:

$$R = n'_y + n'_x\, e_{12} - n'_z\, e_{23}. \tag{4}$$

This embeds the rotation naturally without trigonometric functions, where scalar n'_y corresponds to $\cos(\theta/2)$.

Translator (Position): The position of the vertex $\mathbf{p} = (x, y, z)$ is encoded into a translator T. In CGA, translations are represented using the point at infinity e_∞:

$$T = 1 - \frac{1}{2}(\mathbf{p}e_\infty) = 1 - \frac{1}{2}(xe_{1\infty} + ye_{2\infty} + ze_{3\infty}). \tag{5}$$

The Resulting Motor: The final input to the network is the motor $M = TR$. This multivector resides in the even subalgebra of CGA and uniquely characterizes the vertex's spatial state (location and facing) in a single algebraic entity. While this could mathematically be compressed to an 8-dimensional dual quaternion representation, the full 32-dimensional basis is retained, as richer geometric encodings often improve neural network convergence and stability, by providing a more expressive latent space [6, 36, 10]. Preliminary experiments [15] confirmed that the full multivector representation yields more stable learning behavior compared to lower-dimensional forms. Furthermore, this choice ensures extensibility for future operations involving other CGA entities (e.g., spheres or uniform scalings).

2.3. Neural Approximation Models

The *Neural-GASh* framework approximates the PRT operator using two complementary neural architectures, both trained on the CGA-encoded data described above.

1. Local Approximation (MLP): The first variant [15] employs a Multi-Layer Perceptron (MLP) for efficient per-vertex inference. This model treats

light transport as a local regression problem, making it highly suitable for real-time engines like Unity where computational budget is limited.

2. Topological Approximation (GNN): To capture non-local light interactions such as self-occlusion and shadowing, we introduce a Graph Neural Network (GNN) formulation (detailed in Section 4). This model incorporates mesh connectivity, propagating features across the surface topology to infer complex lighting relationships that local models cannot resolve.

2.4. Summary

This formulation establishes a coherent framework where diffuse PRT is treated as a learnable function over CGA space. By combining the half-angle rotor construction with neural regressors, we bridge physically-based rendering and geometric deep learning. The following sections detail the specific architecture and training of the GNN-based implementation.

3. State-of-the-art

The simulation of GI remains a central challenge in computer graphics, balancing physical accuracy with the computational constraints of real-time applications. Although offline methods, such as path tracing and photon mapping, provide ground-truth solutions to the rendering equation, they remain computationally prohibitive for real-time interactive simulations on consumer hardware. Consequently, real-time rendering has historically relied on approximations. Within this spectrum, PRT [34] emerged as a foundational technique, projecting light transport onto basis functions like SH to decouple lighting from geometry.

In recent years, the field has been transformed by neural rendering. Machine learning models, ranging from Multilayer Perceptrons (MLPs) to Convolutional Neural Networks (CNNs), are now used to approximate radiance transfer, denoise Monte Carlo integrals, or replace the rendering pipeline entirely via NeRFs and Gaussian Splatting (GS). This section reviews the trajectory from classical PRT to modern neural and explicit representations, identifying the specific geometric and topological limitations that necessitate the proposed Neural-GASh framework.

3.1. Precomputed Radiance Transfer (PRT)

PRT assumes a static scene to precompute the light transport operator $\mathcal{T}$. By representing both the incoming lighting L_{in} and the transfer function in a band-limited basis (typically SH), the shading integral is reduced to a simple dot product at runtime. Despite its efficiency, the classic formulation assumes that the geometry is immutable; any deformation or destruction invalidates the precomputed transfer coefficients.

3.1.1. Algebraic and Texture-Based Extensions. To address the limitations of PRT, several reformulation strategies have been proposed.

Geometric Algebra Integration: Papaefthymiou et al. [25] integrated CGA into the PRT framework. As detailed in recent surveys by Hitzer et al. [17], CGA offers a unified algebraic structure for geometric transformations, representing rotations via *rotors* rather than matrices. In the context of PRT, this allows for more efficient rotation of SH coefficients. However, this implementation inherits the static constraints of PRT. Specifically, SH coefficients are defined relative to a global coordinate frame; while rotors handle rotation efficiently, the framework does not support the translation or scaling of the transfer functions, rendering it unsuitable for freely moving or deforming objects.
PRT Textures (PRTT):. Dhawal et al. [12] proposed PRT Textures to decouple transfer data from mesh resolution. By storing SH coefficients in texture maps rather than vertex attributes, PRTT allows for high-frequency lighting details on low-polygon geometry. This approach enables effects like glossy interreflections to be resolved at the texel level. However, relying on textures introduces a baking constraint: interreflections are effectively "painted" into the texture space. If the object deforms or if the spatial relationship between surface patches changes, the baked transport is no longer valid, requiring a costly re-computation of the texture.
Estimation and Data-Driven Methods: Beyond direct rendering, PRT has been adapted for inverse problems. Thul et al. [37] utilized PRT to estimate reflectance and lighting from images, decomposing scenes into constituent material and illumination components. Similarly, Belcour et al. [4] proposed a data-driven paradigm to augment PRT with machine-learned priors. While these methods improve relighting capabilities, they often struggle with high-frequency specular effects due to the low-order nature of SH bases and remain computationally heavy for large-scale assets, such as open-world game levels.
Non-Neural Optimizations: MoMo-PRT [32] represents a significant algorithmic optimization, using moment-based data structures to approximate indirect illumination and soft shadows without neural inference. It achieves interactive performance on standard hardware but, like its predecessors, relies on static precomputation. It also faces challenges with memory consumption when scaling to large environments, that require high-frequency shadow details.

3.2. Neural Rendering and Radiance Fields

Neural rendering replaces or augments traditional pipelines with differentiable modules. These approaches leverage the universal approximation capabilities of neural networks to learn the mapping between scene configurations and radiometric outputs.

3.2.1. Neural Approximations of Radiance Transfer. Several works have attempted to learn the PRT operator directly to bypass the precomputation bottleneck.
DeepPRT:. Li et al. [23] introduced DeepPRT, a framework that predicts radiance transfer for deformable objects. The method employs a CNN that operates in texture space (UV space). The network takes a geometry representation as input and infers a texture map containing the SH transfer coefficients.

While this enables support for deformations, the reliance on 2D texture space is a significant limitation. UV mappings introduce discontinuities (seams) and distortions that do not exist on the 3D surface. Furthermore, the CNN processes the data as an image, lacking inherent awareness of the 3D proximity or angular relationships between surface points, which are critical for accurate self-shadowing.

NeuralPRT:. Rainer et al. [28] proposed NeuralPRT, which predicts GI effects in image space. The model takes G-buffer feature maps (depth, normal, albedo) as input and outputs a shaded image. While efficient, this approach is fundamentally view-dependent and decoupled from the scene geometry. It cannot simulate off-screen interactions (e.g., a shadow cast by an object behind the camera) and does not update the actual physical state of the mesh lighting, limiting its utility for physics-based downstream tasks or consistent multi-view rendering in XR.

3.2.2. Volumetric and Implicit Representations. NeRFs [3] represent scenes as continuous volumetric functions parameterized by MLPs. While capable of photorealistic view synthesis, standard NeRFs require expensive ray marching (hundreds of evaluations per pixel), resulting in high latency.

PlenOctrees: To accelerate inference, Yu et al. [40] introduced PlenOctrees, which bake the NeRF into an octree structure containing SH coefficients at leaf nodes. This converts the neural query into a fast tree traversal, achieving frame rates over 150 FPS. However, this speed comes at the cost of memory efficiency; uncompressed PlenOctrees can consume plenty of gigabytes of VRAM. Furthermore, the structure is rigid; the octree cannot easily adapt to dynamic, deforming geometry without complete re-generation.

Geometric Clifford Algebra Networks (GCANs): Ruhe et al. [31] proposed embedding geometric algebra directly into neural architectures. GCANs use Clifford algebra layers to enforce geometric equivariance (e.g., rotation invariance) mathematically. This approach is highly relevant to our work; however, GCANs are currently computationally dense. The complex algebraic operations require custom kernels for efficient execution, and current implementations are often slower than standard linear layers, making them difficult to deploy in strictly real-time render loops (30-90 FPS).

Deep Diminished Reality (DeepDR):. In the context of scene reconstruction, DeepDR [16] uses structure-aware networks to inpaint backgrounds when objects are removed. While primarily an inpainting technique, it highlights the difficulty of inferring lighting in occluded regions. DeepDR struggles with consistent shadow removal and boundary ambiguities, further emphasizing the need for topology-aware methods that understand occlusion explicitly rather than inferring it solely from pixel data.

3.3. Gaussian Splatting (GS)

Gaussian Splatting has recently emerged as a powerful alternative to both mesh-based and volumetric rendering.

3.3.1. The 3DGS Paradigm. Kerbl et al. [21] introduced 3D Gaussian Splatting (3DGS), which represents a scene as a collection of millions of 3D anisotropic Gaussians. Each Gaussian carries attributes: position, covariance (scale/rotation), opacity, and SH color coefficients. Unlike NeRFs, 3DGS utilizes a rasterization-based pipeline. The Gaussians are sorted and "splatted" onto the screen, allowing for real-time performance (100+ FPS) with quality comparable to offline rendering.

3.3.2. Limitations for Dynamic GI. Despite its visual fidelity, 3DGS shares the "static world" assumption of classic PRT.

1. **Staticity:** The Gaussians are optimized for a fixed scene state. Moving or deforming an object requires updating the position and covariance of thousands of primitives, which is non-trivial to perform coherently without artifacts.
2. **Memory Footprint:** High-fidelity scenes require millions of Gaussians, leading often to excessive usage of VRAM (exceeding 2-4 GB per scene) [9].
3. **Lack of Topology:** As 3DGS is essentially a point cloud, it lacks connectivity information (edges/faces). As such the implementation of physics interactions, collisions, or logic is difficult as it requires surface continuity—features standard in mesh-based XR applications.

3.3.3. Recent Extensions. Recent works have attempted to address these gaps. Liang et al. [24] introduced temporal components to model dynamic scenes, effectively playing back 4D recordings. Zhang et al. [43] proposed combining 3DGS with relightable SH coefficients. While promising, these methods typically use low-order SH (limiting shadow sharpness) or are restricted to specific material types (diffuse only). Crucially, they do not offer a generalized solution for shading standard skinned meshes used in game engines.

3.4. Real-Time Rendering in XR and VR Environments

The requirements for Extended Reality (XR) differ significantly from standard screen-based rendering. VR/AR applications demand stereoscopic rendering at high frame rates (72-120 Hz) to prevent motion sickness, imposing strict budgets on shading complexity (typically <8ms per frame).

3.4.1. Applied XR Frameworks. Previous work in our research group has focused on optimizing physically-based rendering for these constrained environments. The MAGES framework (v3.0 [27] and v4.0 [46]) introduced modular authoring tools for VR medical training simulations. These systems require high-fidelity rendering of anatomy (e.g., organ surface properties) while maintaining interactivity for cutting and deformation operations. Similarly, applications in cultural heritage [13, 26] and pandemic response training (Covid-19 VR Strikes Back [44, 45]) demonstrated the need for scalable rendering solutions that run on standalone headsets. These projects established that while standard PBR is feasible, full Global Illumination remains a bottleneck,

often forcing developers to use static lightmaps that break immersion when objects move.

3.4.2. Neural XR Solutions. Recent efforts have begun porting neural rendering to VR. VR-NeRF [39] optimizes the NeRF pipeline for dual-view rendering, while VRSplat [38] adapts Gaussian Splatting for VR interaction. Re-ReND [30] attempts to bake neural fields into standard meshes to leverage hardware rasterization. However, these methods are primarily focused on *view synthesis* (passively viewing the scene) rather than *interactive relighting* (interactively changing the light or geometry).

3.5. Identified Research Gaps

Table 2 summarizes the landscape of current GI methods. The analysis reveals a clear dichotomy: methods are either fast but light a static scenes (PRT, 3DGS), or light dynamic scenes but computationally heavy/geometry-unaware (NeRF, Image-space Neural GI).

Method	Input Domain	Model	Deformations	Topology	Engine Support	Limitations
PRT [34]	3D Mesh	SH Basis	No	No	Partial	Precomputation; static only
PRT+CGA [25]	3D Mesh	SH+CGA Rotors	No	No	Partial	No translation/scaling
PRTT [12]	3D Mesh	Texture-SH	No	No	Yes	Baking artifacts
DeepPRT [23]	3D Mesh	CNN (Textures)	Yes	No	No	UV-dependent; offline training
NeuralPRT [28]	2D Image	MLP/CNN	No	No	No	Image-space; no 3D logic
PlenOctrees [40]	Volumetric	Octree+SH	No	No	No	High memory; bounded scenes
3DGS [21]	Point Cloud	3D Gaussians	No	No	Partial	High memory; static scenes
Neural-GASh [15]	3D Mesh	SH Basis+MLP	Yes	No	Yes	No self-shadows

TABLE 2. Comparative summary of GI and neural rendering methods. Existing methods generally lack the combination of topological awareness and engine integration required for dynamic XR.

We identify specific gaps that the proposed Neural-GASh framework aims to fill:

Gap 1: Absence of Geometry-Aware Learning. Current neural GI methods do not learn on the mesh manifold. Image-space methods (NeuralPRT) lack depth coherence, while texture-space methods (DeepPRT) introduce UV distortion and discontinuities. There is a lack of methods that utilize raw geometric data—vertex positions and normals—directly as input features for GI prediction, preventing the model from understanding the actual 3D shape of the object.

Gap 2: Lack of Topological Inference. Light transport is inherently non-local (shadows are cast by distant vertices) but topologically coherent (light leaks do not occur across watertight surfaces). Point-based methods (3DGS) and implicit volumes (NeRF) ignore the mesh topology. Graph Neural Networks (GNNs) offer a mechanism to propagate information across the mesh surface, effectively modeling local interreflections and occlusion. However, no existing GI framework combines CGA (for efficient geometric encoding) with GNNs (for topological reasoning) to solve the radiance transfer problem.
Gap 3: Integration with Standard XR Pipelines. Many state-of-the-art neural rendering papers utilize custom Python renderers that cannot be easily integrated into commercial game engines like Unity or Unreal. For practical adoption in fields like medical training or cultural heritage, the neural model must output standard rendering data (e.g., SH coefficients) that fit into existing shader pipelines.

The Neural-GASh framework is designed to bridge these specific gaps, providing a geometry-aware, topology-sensitive, and engine-compatible solution for real-time GI.

4. Neural-GASh (GNN-Based)

While the initial MLP-based Neural-GASh [15] demonstrated that conformal geometric features could drive radiance regression, it exhibited a critical limitation: the absence of self-shadowing. As illustrated in Figure 3, concave regions (e.g., the area beneath the arm of a humanoid) lacked the necessary occlusion cues. This deficiency stems from the inductive bias of the MLP, which processes vertices independently, remaining blind to the topological connectivity required to infer light blocking. To address this, we transitioned to a GNN architecture capable of exploiting mesh topology to model non-local light transport.

4.1. Architecture Design and Rationale

To recover self-occlusion, we systematically evaluated four architectural strategies: (i) deeper MLPs with enriched datasets, (ii) a single end-to-end GNN, (iii) Teacher–Student distillation, and (iv) a Two-Stage GNN. A summary of these strategies along with a list of conclusions drawn are presented below.

4.1.1. Limitations of Single-Stage and Distillation Approaches. Initially, we attempted to retain the deployment simplicity of the MLP by employing a **Teacher–Student** paradigm. A large, visibility-aware GNN (Teacher) supervised a lightweight MLP (Student) trained solely on local CGA features. However, this approach failed to produce plausible shadows. We identified two theoretical reasons for this failure:

1. **Information Mismatch:** The teacher network had access to 57 input features (32 CGA + 25 explicit visibility samples), whereas the student had access only to 32. Visibility is not a deterministic function of local

(A) Traditional PRT (B) MLP Result

FIGURE 3. Comparison showing the lack of self-shadowing in the MLP baseline (b) compared to ground truth (a). The red box highlights the missing contact shadow.

curvature; it depends on global geometry. The student was effectively asked to solve an ill-posed inverse problem without sufficient data.

2. **Inductive Bias Gap:** Even when the student was upgraded to a lightweight GNN, it failed to approximate the teacher's high-frequency occlusion signals. The regression loss Mean Squared Error (MSE) drove the student to predict the mean illumination, resulting in "muddy" or overly smooth shading that lacked distinct shadow boundaries.

Similarly, a **single end-to-end GNN** trained to map CGA features directly to SH coefficients (without an explicit visibility intermediate representation) underperformed. While it captured low-frequency gradients, it struggled to localize sharp shadow lines. These findings necessitated a **Two-Stage Architecture** that explicitly decouples geometric understanding (visibility prediction) from photometric reconstruction (radiance transfer).

4.2. The Two-Stage GNN Pipeline

The adopted framework (Figure 4) consists of two specialized networks executed sequentially. The first infers the topology-dependent visibility field, while the second uses that field to regress the final lighting coefficients.

4.2.1. Stage 1: Visibility GNN. The objective of this network is to approximate the expensive ray-casting step of PRT. It maps the 32-dimensional CGA multivector of a vertex to 25 binary visibility indicators (corresponding to sampled directions on the hemisphere).

Architecture: As shown in Figure 5, the network begins with a linear projection to a 64-dimensional latent space. The core backbone consists of three stacked **TransformerConv** blocks. Unlike standard graph convolutions, TransformerConv utilizes multi-head attention to assign importance weights to

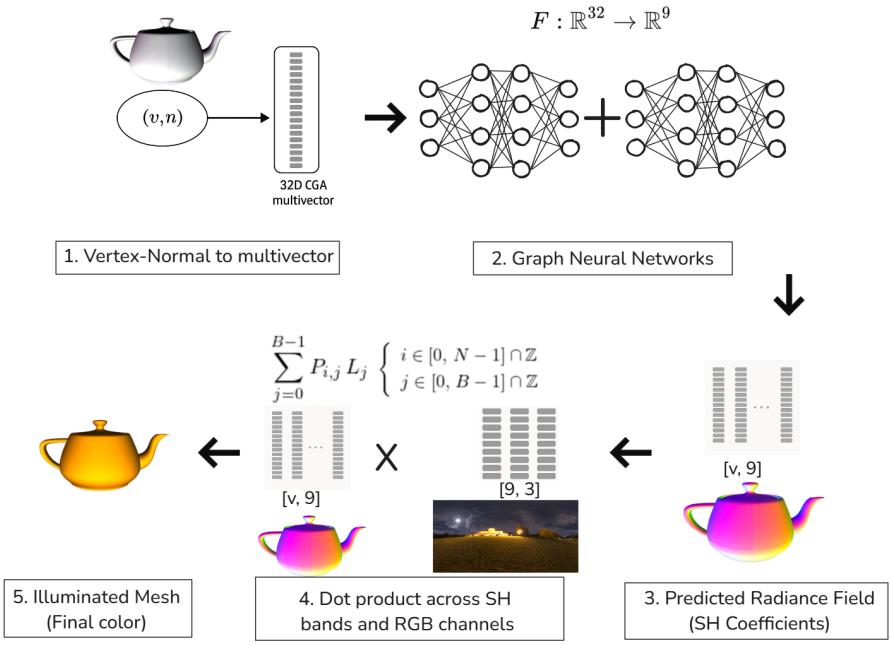


FIGURE 4. Conceptual illustration of the two-stage GNN-based Neural-GASh framework. Stage 1 predicts visibility; Stage 2 predicts radiance.

neighbors. This allows the vertex to "attend" to geometrically salient neighbors (e.g., those creating a cavity) while ignoring others. Each block includes Layer Normalization to stabilize gradients and a ReLU activation. A residual connection preserves high-frequency features deep into the network. The final head is a linear projection predicting the 25 logits.

Training Protocol: The network is trained on a dataset of meshes processed with offline ray-tracing. Since visibility is sparse (many directions are fully visible or fully occluded), we employ a **Class-Balanced Binary Cross-Entropy Loss**. This ensures that rare occlusion events contribute equally to the gradient. Optimization uses the AdamW optimizer with a cosine annealing schedule to refine convergence.

4.2.2. Stage 2: Coefficients GNN. The second network regresses the 9 SH transfer coefficients. It receives a concatenated input vector of dimension 57 (32 CGA features + 25 predicted visibility features).

Architecture: Illustrated in Figure 6, this model is deeper, utilizing five blocks of **GATv2** (Graph Attention Network v2) layers. GATv2 improves upon standard attention by making the attention weights dynamic with respect to the query node, allowing the network to model complex interreflections where the influence of a neighbor depends on the local surface orientation.

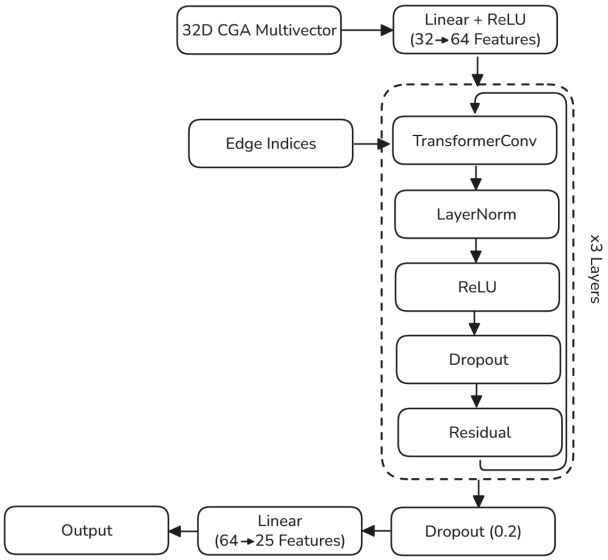


FIGURE 5. Flowchart of the Visibility GNN architecture.

We employ a latent dimension of 96. To prevent over-smoothing (a common issue in deep GNNs where features become indistinguishable), we apply strict **Dropout (p=0.2)** and Identity Skip Connections at every block.
Training Protocol: We minimize the Mean Squared Error (MSE) between the predicted and ground-truth SH coefficients. To handle the wide dynamic range of radiance values, we employ **Mixed Precision Training** (FP16), which not only reduces memory usage but also acts as a regularizer. Gradients are clipped to a fixed norm to prevent exploding updates during the initial training phases. We also monitor an auxiliary metric, "Shadow MSE," which calculates error strictly on vertices with low luminance, ensuring the model does not neglect dark regions.

4.3. Real-Time Integration via Microservice

Integrating GNNs with complex attention mechanisms into a game engine like Unity presents significant engineering challenges. Unity's native inference engine (Barracuda) is optimized for standard CNNs and MLPs but lacks support for the sparse matrix operations and scatter-gather primitives required by GATv2 and TransformerConv layers.

4.3.1. The Flask-Unity Bridge. To bypass these limitations without sacrificing architectural complexity, we implemented a microservice architecture using a local **Flask server**. This design decouples the inference backend (PyTorch) from the rendering frontend (Unity).

The NeuralGASh pipeline (Figure 7) executes as follows:

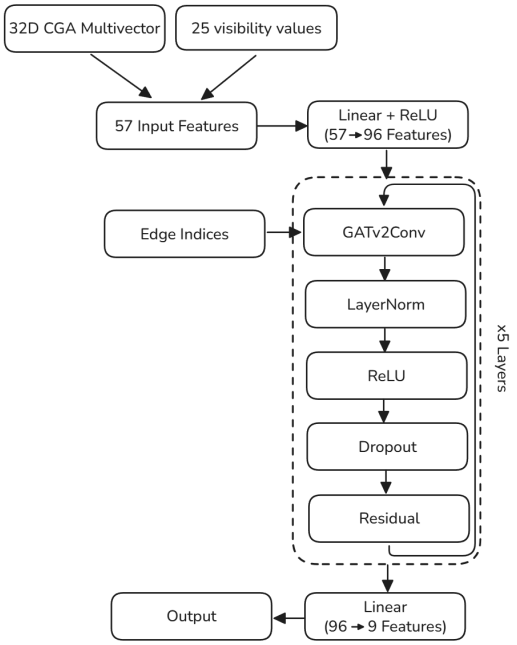


FIGURE 6. Flowchart of the Coefficients GNN architecture.

1. **Data Extraction (Unity):** A C# script extracts raw vertex and normal buffers.
2. **CGA Encoding (Compute Shader):** To offload the CPU, a DirectCompute shader runs on the GPU to convert Euclidean vectors into 32-D CGA multivectors in parallel.
3. **Transmission:** The binary data is serialized and sent via HTTP POST to the local server.
4. **Graph Construction (Python):** The server constructs a radius-based graph (k-NN) during processing.
5. **Inference:** The Visibility GNN and Coefficients GNN run sequentially.
6. **Post-Processing:** An optional Ambient Occlusion (AO) pass refines the coefficients (detailed below).
7. **Rendering:** The coefficients are returned to Unity, where a custom shader computes the dot product $C_{final} = \sum c_{lm} L_{lm}$.

4.3.2. Ambient Occlusion (AO) Refinement. To further enhance the perceptual realism of contact shadows, we integrated a geometric Ambient Occlusion pass [7] that runs on the server after inference. The algorithm iterates over the graph neighborhood. For a vertex v and neighbor n, we compute the projection of vector $v\vec{n}$ onto the normal $\vec{N}_v$. If the neighbor lies above the tangent plane (positive projection), it contributes to occlusion. This geometric

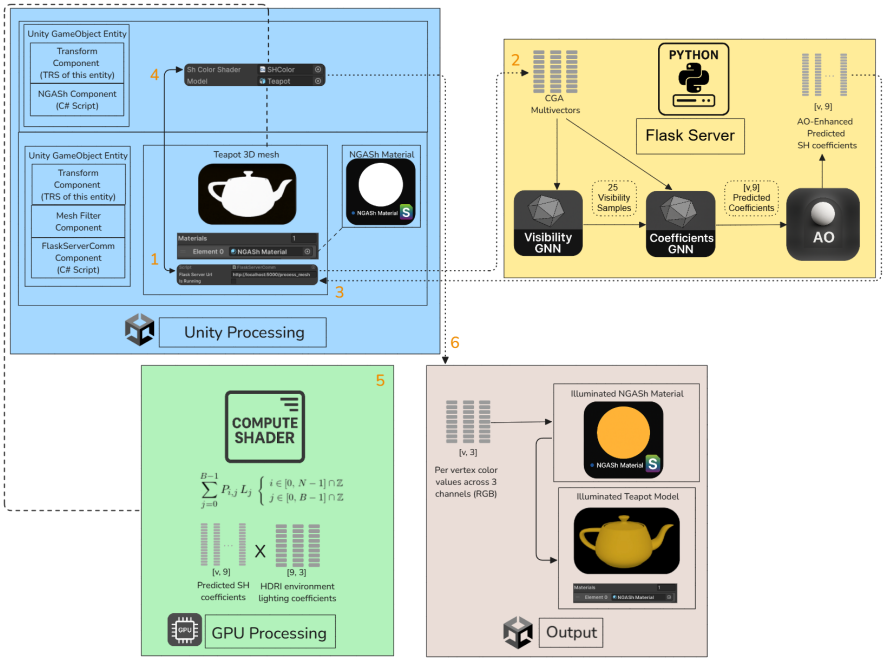


FIGURE 7. Detailed integration pipeline showing the data flow between Unity, the Flask Server, and the GPU.

heuristic allows us to darken cavities that the neural network might have slightly under-predicted, ensuring "watertight" shadowing in deep crevices.

4.4. Results and Analysis

Experiments were conducted on a workstation with an AMD Ryzen 7 7700 and NVIDIA RTX 4060. We evaluated the method on a set of unseen medical meshes ranging from 3K to 34K vertices (Table 3).

Mesh	Wireframe	Vertices	PRT (s)	Ours (s)
Coolseal		9673	417.54	0.14
Bipolar		3808	151.90	0.05

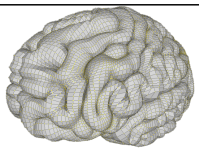
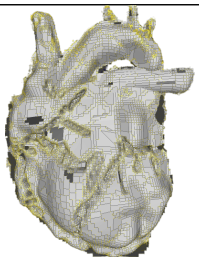
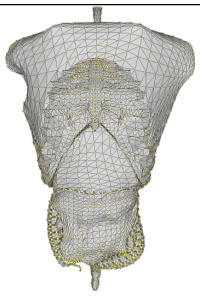
Focus		8534	369.27	0.09
Brain		21966	1051.54	0.29
Heart		29661	1619.16	2.69
Anatomy		33995	1908.58	4.13

Table 3: Comparison of precomputation times. Our method reduces setup time from roughly 30 minutes to seconds, a speedup factor of $> 3000\times$.

4.4.1. Qualitative Evaluation. We evaluated visual fidelity under three distinct illumination conditions to test robustness: Dark (low contrast), Bright (high saturation), and Normal.

Normal Environment: Table 4 presents the results for the standard lighting scenario. The GNN-based Neural-GASh (Column 3) produces shading that is nearly indistinguishable from Traditional PRT (Column 1). Unlike the MLP baseline (Column 2), which renders the deep fissures of the brain as flat surfaces, the GNN correctly darkens these regions. The difference maps (Column 5) confirm that the error is minimal and stochastic, rather than structural.

Extreme Conditions: In the **Dark Environment** (omitted for brevity), the method demonstrated stability, avoiding the noise artifacts often seen in neural rendering when signal-to-noise ratios are low. In the **Bright Environment**, the high-intensity light typically exposes shading errors. However, our AO-enhanced pipeline maintained strong contrast in shadow penumbras, successfully anchoring the objects to the ground plane—an effect completely absent in the MLP results.

FIGURE 8. HDRI map used for the Normal environment results.

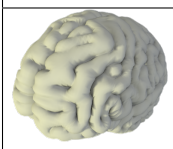
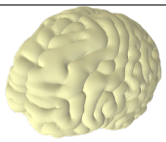
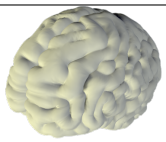
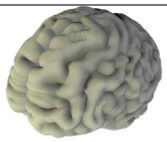
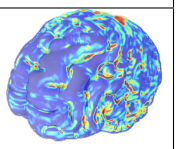
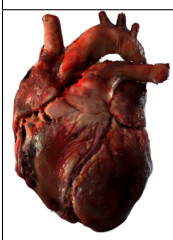
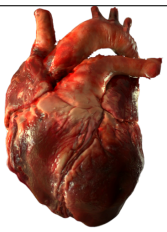
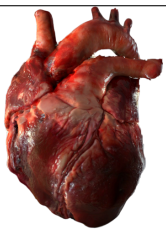
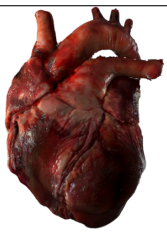
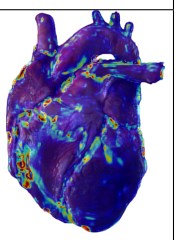
Trad. PRT	MLP-GASh	GNN-GASh	AO-Enhanced	Difference
				
				

TABLE 4. Visual comparison in a Normal Lit environment. Note the improved self-shadowing in the GNN version compared to the MLP, specifically in the crevices of the Brain model.

4.4.2. Quantitative Error Analysis. Figures 9 and 10 analyze the error distribution per SH coefficient. The MSE remains consistently low (< 0.015).
Analysis of Zonal Harmonics: A distinct pattern emerges in the error plots: peaks are observed at indices 3 and 7. In our SH basis enumeration, these correspond to the $m = 0$ modes of the $l = 1$ and $l = 2$ bands, known as **Zonal Harmonics**. These terms encode the azimuthally symmetric component of the lighting field—essentially the "vertical" gradient of light. Since GNNs learn primarily from local neighborhood differences, they excel at high-frequency details (shadows) but can sometimes drift on global, low-frequency gradients [29]. The higher error in zonal terms reflects this locality bias, though the magnitude is small enough to be perceptually negligible.

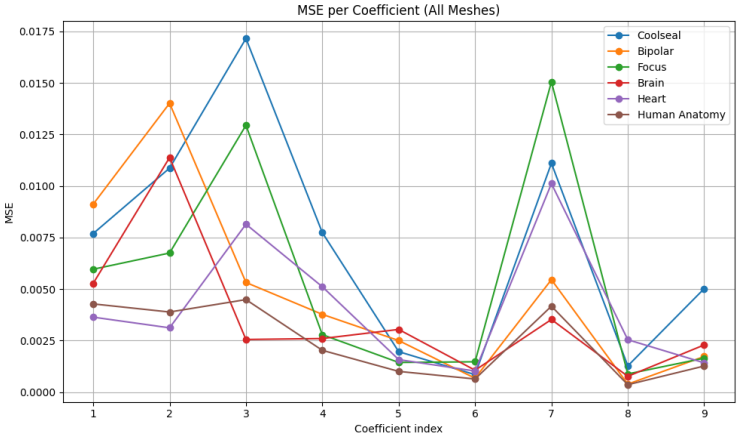


FIGURE 9. MSE across the 9 SH coefficients. Peaks at indices 3 and 7 correspond to Zonal Harmonics.

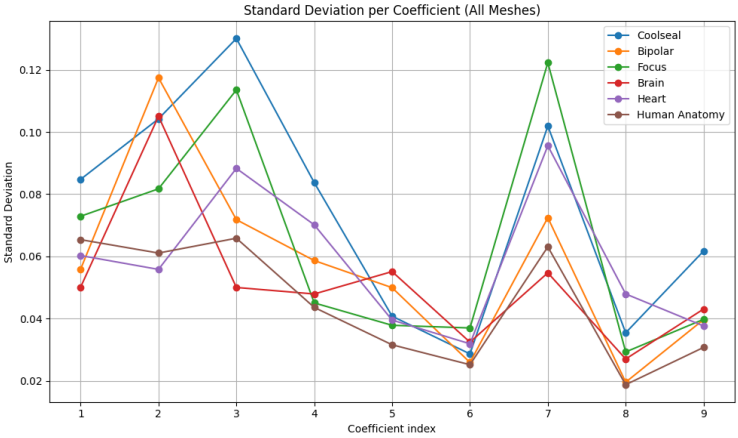


FIGURE 10. Standard Deviation (STD) of the predicted coefficients.

Visibility Prediction Accuracy: Figure 11 breaks down the accuracy of the Visibility GNN (Stage 1). We observe a clear correlation between geometric smoothness and accuracy.

- **High Accuracy (Brain, Anatomy):** These organic shapes have continuous curvature. The network achieves $> 80\%$ accuracy because the local neighborhood is a strong predictor of occlusion.
- **Lower Accuracy (Bipolar, Coolseal):** These mechanical tools feature sharp disjoint parts (e.g., scissor jaws). The visibility function here is discontinuous; a vertex might be occluded by a jaw that is spatially distant in the graph (few hops away).

Despite lower raw accuracy on mechanical parts, the subsequent radiance regression remains robust, suggesting the network learns to be resilient to noisy visibility inputs.

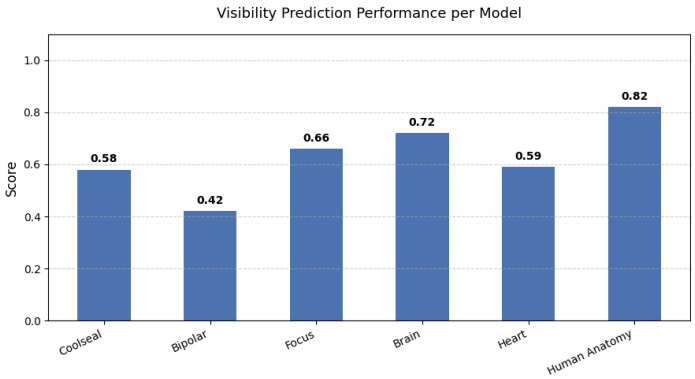


FIGURE 11. Prediction accuracy of the Visibility GNN. Organic meshes yield higher accuracy than mechanical ones with disjoint parts.

4.4.3. Latency Breakdown and Scalability. Figure 12 provides a logarithmic breakdown of the inference latency. The total time is dominated by graph construction and the GNN forward passes.

Asymptotic Scaling Analysis: Let V be the vertex count, k the neighborhood size, H the hidden width, and L the number of GNN layers. With a fixed k, the inference time T_{GNN} scales linearly:

$$T_{\text{GNN}} = \mathcal{O}(L \cdot V \cdot k \cdot H).$$

This linear complexity is critical for scalability. However, graph construction (kNN search) typically scales as $\mathcal{O}(V \log V)$ using tree-based methods. In our current implementation, the most expensive step is the CPU-based Ambient Occlusion pass, which uses a dense search scaling super-linearly. As shown in Table 5, this causes a spike for the "Anatomy" mesh (34k vertices). To support larger assets in production, we recommend implementing the kNN and AO steps via GPU compute shaders, which would restore near-linear scaling behavior.

4.4.4. Dynamic Scenes. A major advantage of Neural-GASh is the support for animation. As shown in Figure 13, coefficients update dynamically as the mesh deforms. In multi-object scenes (Figure 1), our method correctly shades skinned characters (e.g., the Doctor), whereas Unity's static light-mapping pipeline fails to account for dynamic actors, leaving them flatly lit. This confirms Neural-GASh as a viable alternative for interactive XR applications where light-baking is insufficient.

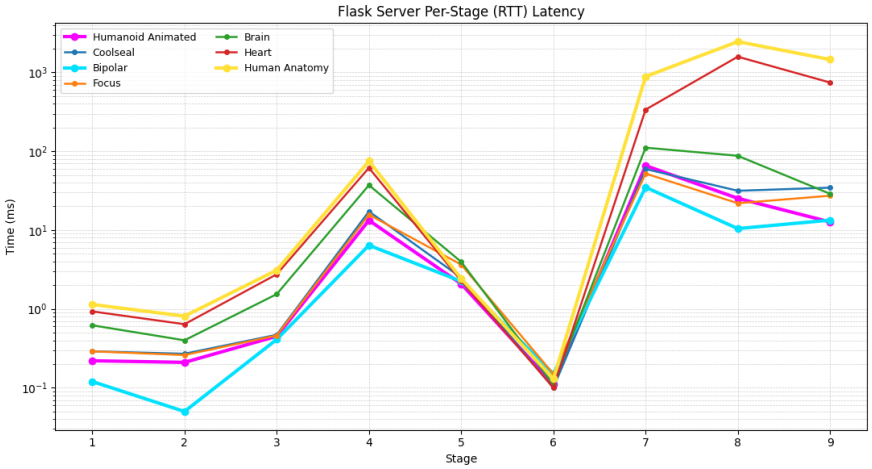


FIGURE 12. Latency breakdown per stage. Note the linear scaling of GNN inference and superlinear scaling of the optional AO pass.

Mesh	Vertices	Min (ms)	Total (ms)
Coolseal	9673	0.10	146.47
Bipolar	3808	0.05	68.10
Focus	8534	0.15	121.85
Brain	21966	0.11	271.30
Heart	29661	0.10	2730.27
Anatomy	33995	0.13	4886.75

TABLE 5. End-to-end latency. Large meshes incur high costs primarily due to the CPU-based AO pass.

5. Discussion & Future Work

The results of this work demonstrate that the proposed Neural-GASh framework provides a viable and efficient alternative to traditional PRT. By replacing the heavy precomputation stage with a learned model, we achieved significant reductions in runtime, making real-time rendering feasible even in dynamic scenes. The adoption of CGA multivectors as the input representation proved particularly effective: while early experiments with simple Euclidean vertex-normal pairs often led to unstable training and artifacts, the 32-dimensional multivector encoding offered a richer and more stable input space. This allowed both MLP and GNN architectures to converge more reliably and to yield sharper reconstructions of radiance transfer coefficients.

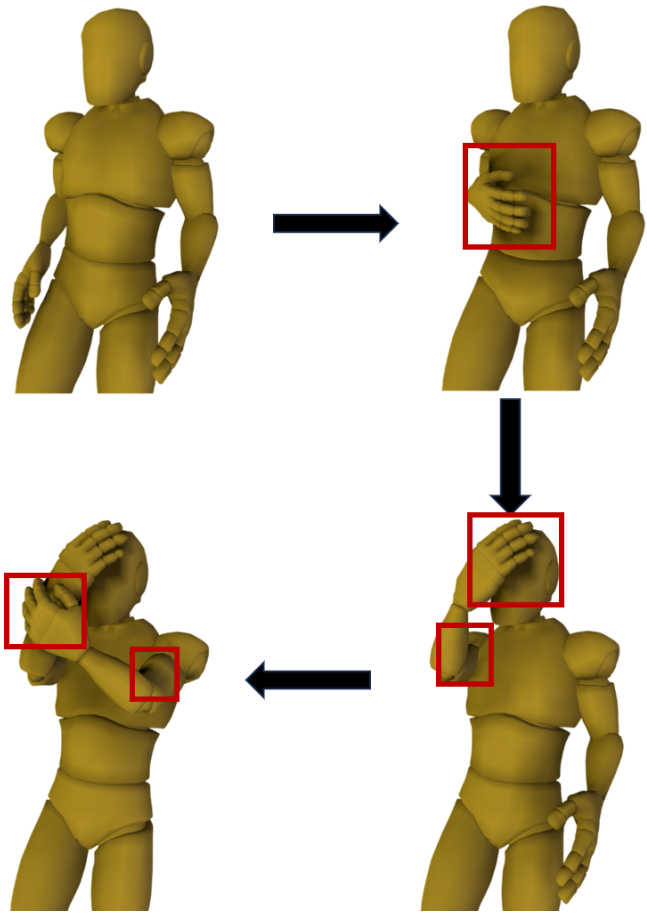


FIGURE 13. Real-time shading of animated geometry. The shadows update frame-by-frame as the mesh deforms.

The comparison between MLP- and GNN-based formulations revealed that, although both benefit from the CGA embedding, GNNs are uniquely capable of leveraging mesh connectivity and local neighborhood information. This property enabled them to capture self-occlusion effects and complex shadowing patterns that remained elusive in purely feed-forward architectures. Nevertheless, errors were observed to concentrate in specific SH coefficients, notably those associated with higher-frequency angular content, which had a disproportionate impact on visual quality. The teacher–student paradigm further highlighted the limitations of MLPs for this task, while experiments with dedicated visibility-prediction networks confirmed that topology-aware models can serve as effective providers of visibility cues. At the same time, this direction introduced additional architectural complexity that must be carefully managed in practical deployments.

It is important to emphasize that the current implementation is based exclusively on the *diffuse component* of the traditional PRT formulation. As a result, the method is achromatic and does not account for interreflections or specular transport. These phenomena, while critical for photorealistic rendering, introduce additional complexity in both precomputation and learning, and were therefore excluded from the present study. Extending Neural-GASh to support specular reflections, glossy materials, and interreflections constitutes a natural and necessary direction for future work.

Another important observation relates to the behavior of the SH coefficients predicted by the neural networks. Although the overall accuracy was high, errors were not uniformly distributed across all coefficients. In particular, indices 2, 3, and 7 consistently exhibited higher deviations across several meshes, as illustrated in Figures 9 and 10. This pattern can be explained by the role of SH in approximating lighting. For a given band index L, the number of basis functions is $B = L^2$. In our implementation, we used three bands ($L = 2$), yielding nine coefficients in total. The lower-order coefficients (such as index 0 and 1) primarily encode the low-frequency, smooth components of lighting, whereas higher-order coefficients (e.g., indices 2, 3, and 7) capture more directional and oscillatory components that vary rapidly over the sphere. These higher-frequency terms are inherently more sensitive to small perturbations in geometry, sampling noise, or approximation errors introduced by the neural network. As a result, both traditional PRT and neural surrogates face greater challenges in predicting them accurately [41]. Nevertheless, even in these more difficult cases, the errors remained modest, and the GNN-based Neural-GASh was able to reproduce the dominant illumination structure with high fidelity, indicating that the essential radiance transfer signal is robustly captured.

5.1. Limitations

Despite its strengths, several limitations remain. First, while visibility samples are not required at inference, they are still necessary during training, and their extraction constitutes a bottleneck when generating large-scale datasets. Second, although the approach generalized well to the tested medical meshes, broader applicability across domains such as natural scenes, game assets, or animated characters has yet to be demonstrated. The scalability of the method is another concern: although real-time performance was achieved for meshes of approximately up to 35,000 vertices, it is unclear how inference pipelines will behave for models containing millions of vertices. Furthermore, the current work was restricted to three SH bands (nine coefficients). While sufficient for diffuse global illumination, higher-order expansions would be needed to represent sharper lighting effects, at the cost of greater computational demand. Finally, the Python-based prototype provided flexibility for experimentation, but it remains less optimized than a production-level C++ or CUDA implementation.

Additional limitations relate to the architectural and system-level design. In the two-stage GNN pipeline, prediction errors from the visibility network

can propagate to the coefficient network, occasionally resulting in blurred or misplaced shadow boundaries. Moreover, while Unity–Flask communication was found to be acceptable on a local workstation (tens of milliseconds for mid-sized meshes), the quantitative latency budget leaves little margin for high-FPS applications and could worsen in remote or distributed setups. Memory usage is another constraint: although the current evaluation covered meshes up to ∼34K vertices, scaling to models with millions of vertices would stress GPU RAM due to the quadratic growth of edge tensors. Careful engineering (e.g., tiling, sparsification, compression) will be necessary to make the method practical at that scale.

5.2. Future Work

Several avenues emerge for future research. The most immediate priority is the expansion of the training dataset, both in mesh diversity and lighting conditions. Larger and more heterogeneous data would strengthen generalization and mitigate overfitting, moving the method closer to robust deployment across varied applications. A second major direction is the elimination of the Unity–Flask communication bottleneck. Deploying GNN inference directly inside Unity—through native runtimes or dedicated plugins—would eliminate round-trip latency and make the pipeline viable for production-level interactive rendering. A third critical step is to extend Neural-GASh beyond diffuse transfer by incorporating specular and interreflection components. Training on richer transport phenomena would enable the framework to reproduce glossy, reflective, and more photorealistic effects.

Other promising directions include developing sample- and band-independent architectures, as well as exploring transformer-based GNNs or implicit neural representations. These models are particularly appealing because they can capture long-range dependencies and non-local visibility cues more effectively than standard message-passing GNNs, thereby reducing the required depth and alleviating issues of over-smoothing and scalability [8]. On the systems side, further optimizations such as C++/Vulkan backends, pruning, and quantization could substantially improve runtime efficiency. Finally, packaging the approach into developer-friendly APIs for Unity or Unreal would facilitate adoption in practical game and visualization pipelines.

5.3. Directions for further research

This section outlines several promising directions in which the present framework can be extended. These avenues highlight both theoretical opportunities and practical enhancements enabled by geometric algebra and its integration with modern neural rendering pipelines.

5.3.1. Quaternionic and Octonionic Color Processing. A natural next step concerns the integration of holistic color processing within the GA-based lighting pipeline. Current experiments treat radiance and surface color largely independently, while recent advances in quaternionic color theory [14] demonstrate that color can be represented and manipulated as a single structured algebraic entity. Quaternionic treatments—especially formulations that decompose color

into luminance and chromatic subspaces using the grey–scale axis—offer an elegant and physically consistent way to encode shading, reflectance, and color interactions simultaneously. Embedding such representations into our framework could enable a unified model where illumination, geometry, and color coexist within a single multivector pipeline. This direction may further strengthen the expressive power of learned radiance transfer, allowing neural networks to predict full-spectrum appearance effects rather than greyscale or achromatic approximations.

5.3.2. Topology-Aware Learning With Zeon and Grassmann Algebras. A promising direction for future research is the explicit integration of algebraic topology using specialized Clifford algebras. Stacey Staples' *zeon algebra*—a modification of geometric algebra tailored to graph-theoretic operations [33, 35]—provides a rigorous algebraic mechanism for expressing adjacency, incidence, connectivity, and higher order combinatorial interactions. Incorporating zeon algebraic operators into the learning pipeline may enable the network to reason explicitly about mesh topology, potentially improving visibility prediction, occlusion reasoning, and structure-sensitive shading.

Similarly, Grassmann algebra (the exterior algebra), which is a subalgebra of GA, has been shown to encode topological and geometric relationships relevant to visibility computation [2]. The use of Grassmann representations could provide a mathematically sound foundation for training networks to capture view-dependent or occlusion-sensitive phenomena through algebraic primitives instead of purely numerical descriptors.

Moreover, work by the Nanjing GIS research group demonstrates how multivector representations can encode topology and spatial transformations with high precision [42]. Adapting such techniques to real-time rendering might lead to visibility or radiance transfer models that are intrinsically aware of global and local mesh structure.

5.3.3. Algebraic Extensions and Unified Multivector Frameworks. The current system uses a 32-dimensional CGA embedding primarily to encode motors. However, CGA supports a broad class of geometric primitives beyond points, directions, and transformations—including spheres, circles, lines, and conformal transformations. Leveraging these representations in the deep learning pipeline could enable future work on global illumination effects such as soft shadows, penumbra regions, or area-light interactions by treating them as algebraic objects instead of discrete samples.

An especially promising direction is the incorporation of higher-grade operators and mixed-grade multivector learning, which could unify radiance transfer, visibility, topology, and color within a single consistent algebraic framework. Such an approach could serve as a foundation for a "universal geometric neural renderer" capable of reasoning jointly about geometry, appearance, topology, and lighting.

Overall, geometric algebra continues to offer a diverse mathematical landscape, and the synergy between GA and deep learning is still largely

unexplored. The extensions outlined above have the potential to significantly expand the expressiveness, generalizability, and practicality of real-time neural radiance transfer.

5.4. Conclusion

PRT has long been valued for its ability to capture global illumination effects with high fidelity, but its reliance on expensive offline sampling and coefficient precomputation makes it impractical for dynamic or interactive scenes. This work addressed that gap by introducing *Neural-GASh*, a framework that reformulates PRT within a neural rendering paradigm. By encoding vertex–normal pairs as CGA multivectors and training modern neural architectures, we demonstrated that radiance transfer can be learned and inferred in real time without traditional precomputation.

The key contributions of this work are threefold. First, we established that CGA-based representations improve stability and expressiveness compared to Euclidean inputs, enabling neural models to learn light transport more effectively. Second, we showed that GNNs, leveraging mesh connectivity, capture self-shadowing and occlusion effects that MLPs fail to reproduce. Third, we integrated the resulting GI models into Unity via a server-based pipeline, demonstrating practical deployment in interactive environments with dynamic and animated deformable geometry. Together, these contributions transform PRT from a static precomputation technique into a neural pipeline compatible with real-time applications.

Several challenges remain: reliance on visibility samples during training, limited scalability to million-vertex meshes, the restriction to diffuse transfer, and Python–Unity communication overhead. These limitations, however, point directly toward future work in dataset expansion, optimized GPU/C++ implementations, and extensions to specular and higher-order lighting.

In conclusion, this work establishes that combining geometric algebra with GNN-based neural models provides a viable path toward real-time radiance transfer. By overcoming the fundamental precomputation bottleneck of traditional PRT, Neural-GASh lays groundwork for the next generation of neural rendering pipelines, enabling interactive global illumination in applications ranging from gaming to medical visualization and virtual reality.

6. Declarations

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

Funding. This work was partially funded by the Innosuisse Swiss Accelerator (2155012933-OMEN-E), and the Horizon Europe Project INDUX-R (GA 101135556).

Data Availability. The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

[1] AKGWSB: CasualPRT (2022). URL https://github.com/AKGWSB/CasualPRT. Accessed: 2025-03-03

[2] Aveneau, L., Charneau, S., Fuchs, L., Mora, F.: A Framework for n-Dimensional Visibility Computations, pp. 273–294. Springer London, London (2011). DOI 10.1007/978-0-85729-811-9_14. URL https://doi.org/10.1007/978-0-85729-811-9_14

[3] Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5460–5469 (2022). DOI 10.1109/CVPR52688.2022.00539

[4] Belcour, L., Deliot, T., Barbier, W., Soler, C.: A Data-Driven Paradigm for Precomputed Radiance Transfer. Proc. ACM Comput. Graph. Interact. Tech. **5**(3) (2022). DOI 10.1145/3543864. URL https://doi.org/10.1145/3543864

[5] Blinn, J.F.: Models of light reflection for computer synthesized pictures. SIGGRAPH Comput. Graph. **11**(2), 192–198 (1977). DOI 10.1145/965141.563893. URL https://doi.org/10.1145/965141.563893

[6] Bronstein, M.M., Bruna, J., Cohen, T., Veličković, P.: Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478 (2021)

[7] Bunnell, M.: Chapter 14 Dynamic Ambient Occlusion and Indirect Lighting, vol. 2, pp. 223–233 (2005)

[8] Chen, D., O'Bray, L., Borgwardt, K.: Structure-aware transformer for graph representation learning. In: International conference on machine learning, pp. 3469–3489. PMLR (2022)

[9] Chen, G., Wang, W.: A Survey on 3D Gaussian Splatting. CoRR **abs/2401.03890** (2024). URL https://doi.org/10.48550/arXiv.2401.03890

[10] Cohen, T., Geiger, M., Köhler, J., Welling, M.: Spherical cnns. ArXiv **abs/1801.10130** (2018). URL https://api.semanticscholar.org/CorpusID:3525710

[11] Cook, R.L., Porter, T., Carpenter, L.: Distributed ray tracing. SIGGRAPH Comput. Graph. **18**(3), 137–145 (1984). DOI 10.1145/964965.808590. URL https://doi.org/10.1145/964965.808590

[12] Dhawal, S., Kt, A., Narayanan, P.J.: Transfer Textures for Fast Precomputed Radiance Transfer. In: B. Sauvage, J. Hasic-Telalovic (eds.) Eurographics 2022 - Posters. The Eurographics Association (2022). DOI 10.2312/egp.20221012

[13] Efstratios, G., Michael, T., Stephanie, B., Athanasios, L., Paul, Z., George, P.: New Cross/Augmented Reality Experiences for the Virtual Museums of the Future. In: M. Ioannides, E. Fink, R. Brumana, P. Patias, A. Doulamis, J. Martins, M. Wallace (eds.) Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection, pp. 518–527. Springer International Publishing, Cham (2018)

[14] Ell, T.A., Bihan, N.L., Sangwine, S.J.: Signal and Image Processing, chap. 4, pp. 67–116. John Wiley & Sons, Ltd (2014). DOI https://doi.org/10.1002/9781118930908.ch4. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118930908.ch4

[15] Geronikolakis, E., Kamarianakis, M., Protopsaltis, A., Papagiannakis, G.: Neural-GASh: A CGA-based neural radiance prediction pipeline for real-time shading. In: Computer Graphics International 2025 (CGI 2025) – ENGAGE 2025 Workshop, Lecture Notes in Computer Science. Springer (2025). (Best GA Computing Paper Award)

[16] Gsaxner, C., Mori, S., Schmalstieg, D., Egger, J., Paar, G., Bailer, W., Kalkofen, D.: DeepDR: Deep Structure-Aware RGB-D Inpainting for Diminished Reality . In: 2024 International Conference on 3D Vision (3DV), pp. 750–760. IEEE Computer Society, Los Alamitos, CA, USA (2024). DOI 10.1109/3DV62453.2024.00037. URL https://doi.ieeecomputersociety.org/10.1109/3DV62453.2024.00037

[17] Hitzer, E., Kamarianakis, M., Papagiannakis, G., Vašík, P.: Survey of new applications of geometric algebra. Mathematical Methods in the Applied Sciences **47**(14), 11368–11384 (2024)

[18] Immel, D.S., Cohen, M.F., Greenberg, D.P.: A radiosity method for non-diffuse environments. SIGGRAPH Comput. Graph. **20**(4), 133–142 (1986). DOI 10.1145/15886.15901. URL https://doi.org/10.1145/15886.15901

[19] Jensen, H.W.: Global illumination using photon maps. In: Proceedings of the Eurographics Workshop on Rendering Techniques '96, p. 21–30. Springer-Verlag, Berlin, Heidelberg (1996)

[20] Kajiya, J.T.: The rendering equation. SIGGRAPH Comput. Graph. **20**(4), 143–150 (1986). DOI 10.1145/15886.15902. URL https://doi.org/10.1145/15886.15902

[21] Kerbl, B., Kopanas, G., Leimkuehler, T., Drettakis, G.: 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Trans. Graph. **42**(4) (2023). DOI 10.1145/3592433. URL https://doi.org/10.1145/3592433

[22] Lafortune, E., Willems, Y.: Bi-Directional Path Tracing. Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics) **93** (1998)

[23] Li, Y., Wiedemann, P., Mitchell, K.: Deep Precomputed Radiance Transfer for Deformable Objects. Proc. ACM Comput. Graph. Interact. Tech. **2**(1) (2019). DOI 10.1145/3320284. URL https://doi.org/10.1145/3320284

[24] Liang, Z., Zhang, Q., Hu, W., Zhu, L., Feng, Y., Jia, K.: Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration. In: A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, G. Varol (eds.) Computer Vision – ECCV 2024, pp. 281–297. Springer Nature Switzerland, Cham (2025)

[25] Papaefthymiou, M., Papagiannakis, G.: Real-time rendering under distant illumination with Conformal Geometric Algebra. Mathematical Methods in the Applied Sciences **41**(11), 4131–4147 (2018)

[26] Papagiannakis, G., Geronikolakis, E., Pateraki, M., Bendicho, V.M., Tsioumas, M., Sylaiou, S., Liarokapis, F., Grammatikopoulou, A., Dimitropoulos, K., Nikos, G., Partarakis, N., Margetis, G., Drossis, G., Vassiliadi, M., Chalmers, A., Stephanidis, C., Thalmann, N.: Mixed Reality Gamified Presence and Storytelling for Virtual Museums (2018). DOI 10.1007/978-3-319-08234-9_249-2

[27] Papagiannakis, G., Zikas, P., Lydatakis, N., Kateros, S., Kentros, M., Geronikolakis, E., Kamarianakis, M., Kartsonaki, I., Evangelou, G.: MAGES 3.0: Tying the knot of medical VR. In: ACM SIGGRAPH 2020 Immersive Pavilion, SIGGRAPH '20. Association for Computing Machinery, New York, NY, USA (2020). DOI 10.1145/3388536.3407888. URL https://doi.org/10.1145/3388536.3407888

[28] Rainer, G., Bousseau, A., Ritschel, T., Drettakis, G.: Neural Precomputed Radiance Transfer. Computer Graphics Forum (Proceedings of the Eurographics conference) **41**(2) (2022). URL http://www-sop.inria.fr/reves/Basilic/2022/RBRD22

[29] Ramamoorthi, R., Hanrahan, P.: An efficient representation for irradiance environment maps. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, p. 497–500. Association for Computing Machinery, New York, NY, USA (2001). DOI 10.1145/383259.383317. URL https://doi.org/10.1145/383259.383317

[30] Rojas, S., Zarzar, J., Pérez, J.C., Sanakoyeu, A., Thabet, A., Pumarola, A., Ghanem, B.: Re-ReND: Real-time Rendering of NeRFs across Devices. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3609–3618 (2023). DOI 10.1109/ICCV51070.2023.00336

[31] Ruhe, D., Gupta, J.K., De Keninck, S., Welling, M., Brandstetter, J.: Geometric clifford algebra networks. In: Proceedings of the 40th International Conference on Machine Learning, ICML'23. JMLR.org (2023)

[32] Schneider, A., Schönborn, S., Egger, B., Frobeen, L., Vetter, T.: Efficient Global Illumination for Morphable Models. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 3885–3893 (2017). DOI 10.1109/ICCV.2017.417

[33] Schott, R., Stacey Staples, G.: Operator Calculus on Graphs. IMPERIAL COLLEGE PRESS (2012). DOI 10.1142/p843. URL https://www.worldscientific.com/doi/abs/10.1142/p843

[34] Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Trans. Graph. **21**(3), 527–536 (2002). DOI 10.1145/566654.566612. URL https://doi.org/10.1145/566654.566612

[35] Staples, G.S.: Clifford Algebras and Zeons. WORLD SCIENTIFIC (2019). DOI 10.1142/11340. URL https://www.worldscientific.com/doi/abs/10.1142/11340

[36] Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., Riley, P.: Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds (2018). DOI 10.48550/arXiv.1802.08219

[37] Thul, D., Tsiminaki, V., Ladický, L., Pollefeys, M.: Precomputed Radiance Transfer for Reflectance and Lighting Estimation. In: 2020 International Conference on 3D Vision (3DV), pp. 1147–1156 (2020). DOI 10.1109/3DV50981.2020.00125

[38] Tu, X., Radl, L., Steiner, M., Steinberger, M., Kerbl, B., de la Torre, F.: VRSplat: Fast and Robust Gaussian Splatting for Virtual Reality. Proc. ACM Comput. Graph. Interact. Tech. **8**(1) (2025). DOI 10.1145/3728311. URL https://doi.org/10.1145/3728311

[39] Xu, L., Agrawal, V., Laney, W., Garcia, T., Bansal, A., Kim, C., Rota Bulò, S., Porzi, L., Kontschieder, P., Božič, A., Lin, D., Zollhöfer, M., Richardt, C.:

VR-NeRF: High-Fidelity Virtualized Walkable Spaces. In: SIGGRAPH Asia 2023 Conference Papers, SA '23. Association for Computing Machinery, New York, NY, USA (2023). DOI 10.1145/3610548.3618139. URL https://doi.org/10.1145/3610548.3618139

[40] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: PlenOctrees for Real-time Rendering of Neural Radiance Fields. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5732–5741 (2021). DOI 10.1109/ICCV48922.2021.00570

[41] Yu, A., Yang, Y., Townsend, A.: Tuning Frequency Bias in Neural Network Training with Nonuniform Data. In: International Conference on Learning Representations (2023). URL https://api.semanticscholar.org/CorpusID:252531626

[42] Yuan, L., Yu, Z., Luo, W., Yi, L., Lü, G.: Multidimensional-unified topological relations computation: a hierarchical geometric algebra-based approach. International Journal of Geographical Information Science **28**(12), 2435–2455 (2014). DOI 10.1080/13658816.2014.929136. URL https://doi.org/10.1080/13658816.2014.929136

[43] Zhang, L., Han, Y., Lin, W., Ling, J., Xu, F.: PRTGaussian: Efficient Relighting Using 3D Gaussians with Precomputed Radiance Transfer. In: 2024 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pp. 1–6. IEEE (2024)

[44] Zikas, P., Kamarianakis, M., Kartsonaki, I., Lydatakis, N., Kateros, S., Kentros, M., Geronikolakis, E., Evangelou, G., Apostolou, A., Catilo, P.A.A., Papagiannakis, G.: Covid-19 - vr strikes back: innovative medical vr training. In: ACM SIGGRAPH 2021 Immersive Pavilion, SIGGRAPH '21. Association for Computing Machinery, New York, NY, USA (2021). DOI 10.1145/3450615.3464546. URL https://doi.org/10.1145/3450615.3464546

[45] Zikas, P., Kateros, S., Lydatakis, N., Kentros, M., Geronikolakis, E., Kamarianakis, M., Evangelou, G., Kartsonaki, I., Apostolou, A., Birrenbach, T., et al.: Virtual reality medical training for covid-19 swab testing and proper handling of personal protective equipment: Development and usability. Frontiers in virtual reality **2**, 740197 (2022)

[46] Zikas, P., Protopsaltis, A., Lydatakis, N., Kentros, M., Geronikolakis, S., Kateros, S., Kamarianakis, M., Evangelou, G., Filippidis, A., Grigoriou, E., Angelis, D., Tamiolakis, M., Dodis, M., Kokiadis, G., Petropoulos, J., Pateraki, M., Papagiannakis, G.: MAGES 4.0: Accelerating the World's Transition to VR Training and Democratizing the Authoring of the Medical Metaverse. IEEE Comput. Graph. Appl. **43**(2), 43–56 (2023). DOI 10.1109/MCG.2023.3242686. URL https://doi.org/10.1109/MCG.2023.3242686

Efstratios Geronikolakis
Department of Computer Science,
University of Crete,
Voutes Campus, 70013 Heraklion, Greece
Orchid ID: 0000-0002-3974-2816
e-mail: stratosg@csd.uoc.gr

Manos Kamarianakis
Department of Mathematics & Applied Mathematics,
University of Crete,
Voutes Campus, 70013 Heraklion, Greece
Orchid ID: 0000-0001-6577-0354
e-mail: kamarianakis@uoc.gr

Antonis Protopsaltis
Department of Electical & Computer Engineering,
University of Western Macedonia,
Kozani, Greece
Orchid ID: 0000-0002-5670-1151
e-mail: aprotopsaltis@uowm.gr

George Papagiannakis
Department of Computer Science,
University of Crete,
Voutes Campus, 70013 Heraklion, Greece
Orchid ID: 0000-0002-2977-9850
e-mail: papagian@ics.forth.gr