Eurographics Symposium on Rendering 2025
B. Wang and A. Wilkie
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 4

Reshadable Impostors with Level-of-Detail for Real-Time Distant Objects Rendering

X. Wu*[1], Z. Zeng*[2], J. Zhu†[1] and L. Wang‡[1]

[1]School of Software, Shandong University, China
[2]University of California, Santa Barbara, USA

Figure 1: *Our Reshadable Impostors with Level-of-Detail (RiLoD) provides efficient proxies for distant objects with complex geometry. In this* DEER *scene, rendered under environment lighting and a single area light, we demonstrate that RiLoD achieves higher geometric fidelity and more faithful material reproduction compared with multi-card billboards and traditional impostors. Additionally, our method supports material editing, which is typically infeasible with conventional image-based approximations.*

Abstract

We propose a new image-based representation for real-time distant objects rendering: Reshadable Impostors with Level-of-Detail (RiLoD). By storing compact geometric and material information captured from a few reference views, RiLoD enables reliable forward mapping to generate target views under dynamic lighting and edited material attributes. In addition, it supports seamless transitions across different levels of detail. To support reshading and LoD simultaneously while maintaining a minimal memory footprint and bandwidth requirement, our key design is a compact yet efficient representation that encodes and compresses the necessary material and geometric information in each reference view. To further improve the visual fidelity, we use a reliable forward mapping technique combined with a hole-filling filtering strategy to ensure geometric completeness and shading consistency. We demonstrate the practicality of RiLoD by integrating it into a modern real-time renderer. RiLoD delivers fast performance across a variety of test scenes, supports smooth transitions between levels of detail as the camera moves closer or farther, and avoids the typical artifacts of impostor techniques that result from neglecting the underlying geometry.

CCS Concepts

• ***Computing methodologies*** → *Rendering;* • ***Rendering*** → *Real-Time Rendering;*

1. Introduction

Photo-realism is becoming increasingly crucial for interactive applications such as video games, simulations, and visualizations.

† Corresponding author: zhujunqiu@mail.sdu.edu.cn.
‡ Corresponding author: luwang_hcivr@sdu.edu.cn.
* Dual First Authors.

One of the key aspects of achieving photo-realism is the use of highly detailed geometry, along with dynamic and sophisticated lighting. However, this pursuit of geometric and lighting complexity inherently conflicts with the demands of real-time performance in interactive applications.

The straightforward way to facilitate real-time performance while maintaining photo-realistic fidelity is to pre-compute and use simplified representations of distant objects with complex geometry. One popular solution is to pre-render the complex object from multiple reference views into image textures – an image-based representation commonly referred to as an *impostor* of the original object. During rendering, target views can then be generated efficiently through interpolation.

To support reshading under dynamic lighting, geometry buffers (G-buffers) such as depth, normals, and material attributes can also be baked alongside the image textures. This enables deferred shading in target views using the interpolated G-buffers. Furthermore, the level-of-detail (LoD) technique can be integrated by organizing multiple impostors of different resolutions into a hierarchical structure and switching between levels based on the object's distance to the camera. As a result, aliasing artifacts can be alleviated, and memory bandwidth consumption can be reduced.

While reshading and LoD each offer significant benefits individually, supporting both techniques simultaneously can be challenging, as each level of LoD requires storing a complete set of G-buffers for multiple reference views. This results in a large memory footprint and high bandwidth consumption, limiting scalability and efficiency of the impostors.

Moreover, when generating results of target views, simple interpolation can introduce noticeable visual artifacts such as dithering, blurring, or shading inconsistencies, especially when the target viewing angle deviates significantly from the reference views. These artifacts arise because the interpolation process inherently disrespect the actual geometry – it simply blends view-dependent attributes across reference images without reasoning about self-occlusions or geometry structure.

In this paper, we propose a new image-based representation for real-time distant objects rendering, *RiLoD – Reshadable Impostors with Level-of-Detail*.

To efficiently support both reshading and LoD, we draw inspiration from the recent trend of replacing full G-buffer storage with lightweight visibility buffers [BH13]. Specifically, instead of storing full material attributes, we propose to store only a compact material ID and UV per sample for each reference view at each LoD level. This significantly reduces the memory footprint and bandwidth requirements associated with supporting both reshading and LoD, while still enabling accurate shading through deferred material attributes lookup during rendering.

To reduce visual artifacts caused by neglecting the underlying geometry – as is common in methods that relies on interpolation, inspired by the forward mapping technique used in Image-Based Rendering (IBR) [SCK08], we propose a forward mapping strategy that projects all reference views directly into the target view. This geometry-aware process respects view-dependent visibility and avoids incorrect blending across surfaces. However, forward mapping may lead to coverage gaps due to limited sampling or occlusion. To address this, we propose a hybrid hole-filling strategy that combines a novel Joint Bilateral Filling algorithm with a Thin Detector [AMD22], which identifies under-covered regions and selectively applies robust inpainting to preserve shading and structure consistency.

We integrate RiLoD into a real-time renderer and evaluate it across a range of scenarios. It supports accurate reshading under dynamic lighting and target viewing conditions and enables smooth LoD transitions without noticeable popping or inconsistencies, while achieving real-time performance. Finally, both quantitative (MSE and LPIPS) and qualitative results show that our approach produces significantly fewer visual artifacts than prior impostor-based techniques.

In summary, our main contributions are as follows:

- a new impostor variant that supports both reshading and LoD for real-time distant objects rendering.
- a compact storage design for geometry and material to reduce the necessary memory footprint and bandwidth.
- a new forward mapping technique as well as a hole-filling strategy specially designed for impostors to generate target views with better visual fidelity.

2. Related Work

Traditional and Optimization-Based Representation Scene representations in computer graphics are generally categorized into explicit and implicit types. Explicit methods, such as point clouds [LPC*00], polygonal meshes, and voxel grids [LC98], directly describe surfaces, making them easy to render and edit. However, they are storage-intensive when representing complex scenes. In contrast, implicit representations, like metaballs [Bli82] and signed distance fields (SDFs) [FPRJ00], define geometry through functions. These enable flexible topologies and efficient ray intersection, although they necessitate sampling to extract surfaces.

With the rise of deep learning and optimization techniques, representation methods have significantly improved in both efficiency and expressiveness. PointNet and PointNet++ [QSMG17; QYSG17] process point clouds using neural networks, while DeepSDF [PFS*19] leverages latent codes to compactly encode SDFs. Neural Radiance Fields (NeRF) [MST*21] models radiance fields for high-quality novel view synthesis, which has been further refined by [MRS*21; BMT*21]. 3D Gaussian Splatting (3DGS) [KKLD23] accelerates rendering with explicit global representations. However, obtaining full radiance fields remains costly, resulting in trade-offs between quality and speed.

Level-of-Detail for Representation Level-of-Detail is crucial for rendering complex scenes efficiently. Early work focused on Shader LoD [HFTF15; Con17], with systems automatically generating simpler shader programs by reducing computations or texture lookups, allowing rendering to scale from detailed close-ups to cheaper distant views. Olano et al. [OB10] introduces LEAN Mapping, filtering specular highlights in bump and normal maps, enabling the use of standard MIP and anisotropic filtering hardware. Extending this, Dupuy et al. [DHI*13] introduced LEADR

Mapping, a reflectance filtering technique that achieves level-of-detail on displacement-mapped surfaces by utilizing mipmapped displacement gradients to compute an anisotropic Beckmann distribution for physically-based microfacet BRDFs.

Proxy-Based Representation The radiance of a light field can be captured from various positions and directions under different lighting and temporal conditions. However, acquiring a full light field representation is expensive. Several methods have been proposed to transform this global problem into a local one by segmenting the light field into multiple proxies.

Image-Based Rendering (IBR)[SCK08] represents the light field by establishing associations among reference images and utilizing image groups. Pulli et al.[PHC*97] employed a stereo camera system to represent a scene and proposed a multi-view interpolation method to integrate information from the stereo system for novel view rendering. Pfister et al.[PZVG00] introduced a surfel-based rendering approach that leverages hierarchical forward warping and visibility splatting to achieve efficient interactive rendering of complex geometries. Mildenhall et al.[MSO*19] utilized a multiplane image (MPI) representation of the scene and employed neural networks to integrate local light field information, producing high-quality results efficiently.

Billboards [MB05], derived from IBR, serve as a texture-based representation commonly used in rendering pipelines to approximate complex geometry. In real-time rendering, they are frequently employed for rendering complex objects such as flames, smoke, explosions, clouds, and distant trees [Guy00; Fer*04; DKY*00; KL16; DDSD03]. Kawasaki et al. [KS02] proposed "microfacet billboarding" to address the challenge of rendering intricately shaped objects. Impostors [For01; DSSD99], a variant of billboards, are generated by rendering complex geometry into an image texture from a specific viewpoint, often replacing distant static geometry to optimize performance. Szirmay-Kalos et al. [SALP05] introduced "distance impostors" to determine approximate ray intersections, significantly accelerating rendering speeds while maintaining the quality of global illumination effects. Ryan [Bru18] proposed multi-card billboards and octahedral impostors, which store directional information of the replaced objects to enable rapid rendering from arbitrary viewpoints. Additionally, several impostor techniques [Ris06; PMDS06; VK06] incorporate ray-casting methods to enhance rendering quality, though this comes with increased computational overhead. Apart from IBR, other approaches employ proxies to achieve localized light field representations. Zhu et al. [ZBX*21] transformed complex lighting fixtures into proxies, effectively decoupling intricate illumination effects from the scene.

3. Problem Analysis

Impostors [For01] are a type of geometric proxy technique designed to simplify scene rendering by replacing objects that are far from the viewpoint with image textures. This technique significantly reduces rendering overhead by avoiding the expensive computation and storage costs associated with the high geometric complexity of distant objects.

A key limitation of traditional impostors is that they discard most of the original geometric and material information of objects, preventing accurate re-shading under novel lighting conditions. This limits their flexibility in dynamic environments. To support re-shading, geometry buffers (G-buffers) can also be baked into image textures for each reference view, allowing the interpolated G-buffers to be used in subsequent deferred shading pipelines. However, this often comes with increased memory footprint and bandwidth consumption. This also means the specific material parameters of objects (such as reflectivity, roughness, etc.) are baked in the image textures and cannot be flexibly edited during rendering. This representation limits the ability to make real-time adjustments to materials in dynamic environments.

Traditional impostors are also prone to noticeable aliasing artifacts, particularly when viewed up close or at oblique angles. Also, although they reduce geometric complexity, they still incur relatively high memory and bandwidth overhead due to the need to store image textures from multiple reference views. A natural extension to address these issues is to incorporate level-of-detail (LoD) representations. By smoothly transitioning between impostors at different resolutions or distances, LoD techniques not only mitigate aliasing artifacts but also further reduce memory and bandwidth consumption. This makes LoD techniques particularly effective for handling large-scale scenes with many distant objects or rapidly changing viewpoints.

While both reshading and LoD offer clear advantages, combining the two within impostor-based systems introduces significant challenges. Specifically, supporting reshading across multiple LoD levels typically requires maintaining full G-buffer data for each reference view and each level of detail. This quickly leads to a substantial increase in memory and bandwidth consumption. As the number of LoD levels and reference views grows, the scalability of such representations becomes increasingly constrained, making them less practical for large-scale, real-time applications.

Besides, traditional impostor techniques typically rely on simple interpolation to generate target views. Due to the lack of underlying geometric awareness, these methods often fail to reproduce the full complexity of object shapes, especially in regions with self-occlusion, sharp silhouettes, or high-frequency surface details.

Our goal is to support both reshading and LOD simultaneously, while maintaining minimal memory footprint and bandwidth requirements. In addition, we aim to improve visual quality by incorporating geometric awareness into the process, thereby reducing visual artifacts.

4. Method

In this section, we present our Reshadable Impostors with Level-of-Detail (RiLoD) for real-time distant objects rendering.

By encoding lightweight geometric and material information from a limited number of reference views, RiLoD facilitates reliable forward mapping to synthesize target views under dynamic lighting and edited materials. In addition, it supports seamless transitions across different levels of detail. As illustrated in Figure 2, our pipeline consists of two main stages: 1) The input object is processed to generate memory-efficient impostors, each containing

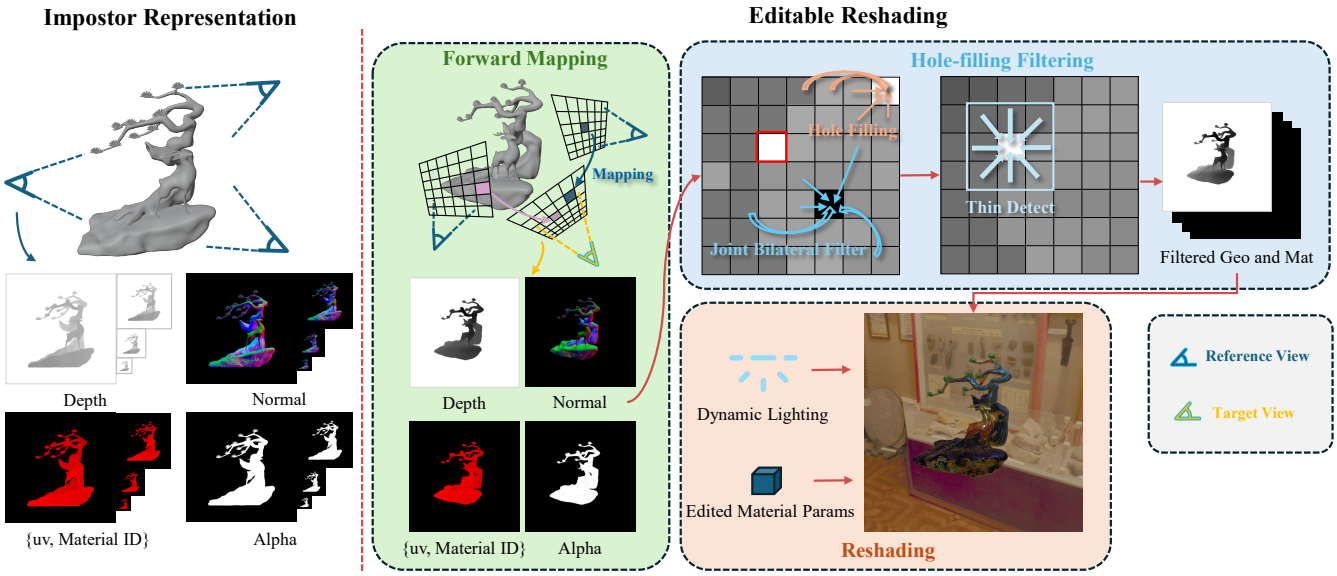

Figure 2: *Pipeline overview. Left: We use orthogonal projections from multiple reference views around the input object to bake its geometric and material properties into the impostor representation, thereby constructing a corresponding LoD hierarchy. Right: The projection transformation between the impostors and target views is applied to map the impostor data onto the target view. Hole-filling filtering is then applied to refine the mapped result, and the filtered information enables re-shading with dynamic editing of lighting and materials.*

both geometry and material properties; and 2) The impostors are used as simplified proxies for the object, contributing to the subsequent rendering pipeline.

In Section 4.1, we detail the process of quickly "baking" the impostors and describe the types of data stored within them to support re-shading. Section 4.2 discusses the rendering approach using impostors and explains how our method achieves artifact-free, real-time rendering results.

4.1. Generating Impostors

To fully reproduce the complicity of the original objects, we store the following information: 1) Geometric information: Including normal vectors ($\vec{n}$), depth (d), and transparency α. These parameters capture the spatial position, orientation, and coverage of objects, enabling accurate representation at different LoD levels. 2) Material information: Including UV coordinates (uv) and material IDs, allowing for the accurate retrieval of the object's albedo, roughness, and metallic properties during shading. By combining this geometric and material data, our method can reconstruct G-buffer information, thus supporting high-quality shading. 3) Reference view information: Including view position (C_i) and the transformation matrix (P_i) from impostor to world space. With these view parameters, a correspondence between the impostors and the target view can be established.

As shown in Figure 2 (left), our baking process is similar to traditional impostor methods. For each object, we first select a set of viewpoints based on the bounding sphere of the object, then cache the geometric and material data into textures set via orthogonal projection. The next challenge is to construct the LoD structure for the impostor. To achieve this, we use a multi-resolution rendering strategy, where each resolution corresponds to a different LoD level. However, reducing resolution leads to a lower sampling rate, causing data loss and generating artifacts.

To address this issue, we propose a heuristic data compression method aimed at reducing storage overhead while maintaining visual fidelity. Specific measures include:

- Depth: We retain the minimum depth value of four neighboring pixels to ensure consistency in depth information.
- Normal vectors and UV coordinates: To eliminate erroneous data, we filter the normal vectors using the von Mises-Fisher distribution [Kar] while preserving depth information, and use averaging for UV coordinates to reduce inaccuracies.
- Material IDs: We compress the material IDs of four neighboring pixels into a single channel, thereby reducing storage requirements.
- Transparency: We average the transparency values of neighboring pixels to smooth out transparency variations.

Finally, the geometric and material properties (such as normal vectors, depth, transparency, UV coordinates, and material IDs) are stored in two separate textures. One texture encodes geometric information $\mathcal{G}$ $\{\vec{n}, d, \alpha\}$, while the other stores material information $\mathcal{M}$ $\{uv$, material IDs$\}$. This method results in comparable storage overhead to traditional impostor techniques, but with significant improvements in accuracy and shading performance.

4.2. Editable Reshading

We have introduced how to construct impostors for the input objects, which will be used for editable reshading – i.e., reshading

under dynamic lighting and modified material attributes – in the target view. As shown on the right side of Figure 2, the information from the impostors is passed to the target view through the forward mapping pass, and the final rendering result is generated through three stages: Forward Mapping, Hole-filling filtering and Reshading.

Forward Mapping. In this stage, our goal is to obtain the geometric and material information of the target view from the impostors. To ensure the accuracy of the mapping, we employ the forward mapping technique from Image-Based Rendering (IBR) [SCK08], rather than traditional interpolation methods. The key to this process is correctly projecting the information from the impostor to the target view, ensuring data accuracy.

The forward mapping is calculated by the following formula:

$$\{\bar{\mathcal{G}}, \bar{\mathcal{M}}\} = P_t * P_i^{-1} * \{\mathcal{G}, \mathcal{M}\}$$

where $\bar{\mathcal{G}}, \bar{\mathcal{M}}$ represent the mapped information in the target view, P_t is the transformation matrix from world space to target view. This process reconstructs the world space information of the reference using d and P_i, and then completes the mapping according to P_t.

A common problem in the forward mapping pass is that the contents of multiple impostors might be projected to the same position in the target view, causing data conflicts. To solve this, we apply a shallow depth pruning strategy. Specifically, we calculate the depth of each impostor projected onto the target view and select the closest impostor data to the viewpoint. This step ensures that the most accurate impostor is selected during projection. Once the relevant information from the impostor is determined, we directly map $\mathcal{M}$ to $\bar{\mathcal{M}}$, replacing the depth d in $\mathcal{G}$ with the depth $\bar{d}$ in the target view to construct $\bar{\mathcal{G}}$.

Through forward mapping based on camera and viewpoint matrix transformations, the impostors can provide more accurate geometric and material information compared to traditional interpolation methods.

Hole-filling Filtering. Although the forward mapping pass retrieves a large amount of geometric and material information from the impostors, there might still be holes or inaccuracies in the target view, especially when the deviation between the target and reference views is significant. To fill these holes and filter out inaccurate information, we employ a two-stage filtering method.

- **Stage 1:** First, we divide the target view into multiple blocks. Within each block, we filter and repair the existing information. We start by removing invalid back-facing information (determined by the normal vector $\vec{n}$ and the target view direction). Next, we apply joint bilateral filtering to the geometric and material information within each block to ensure the reliability of the information. This process is described by the following equation:

$$\{\hat{\mathcal{G}}, \hat{\mathcal{M}}\} = \frac{\sum_{p_s \in N(p_c)} w(p_s, p_c) * \{\bar{\mathcal{G}}, \bar{\mathcal{M}}\}}{\sum_{p_s \in N(p_c)} w(p_s, p_c)}$$

  where $\hat{\mathcal{G}}$, $\hat{\mathcal{M}}$ represent the filtered information, p_c is the position of the filtered information, and $N(p_c)$ is the block containing p_c. This stage of filtering is quite conservative. Although a large neighborhood of information is considered, some information still fails to be correctly filtered.
- **Stage 2:** To further repair holes and inaccuracies in the target view, we introduce thin object detection method, inspired by FSR 2 [AMD22]. This method analyzes the depth information to identify thin object regions and fills in the missing details based on surrounding context. In this stage, only the normal $\vec{n}$ and depth information d are filtered and repaired. This method effectively detects missing information and ensures the completeness of the target view.

Reshading. After the gather filtering stage, we obtain accurate and complete geometric and material information in the target view. In this stage, these details are used for the final reshading calculation. The reshading formula is as follows:

$$L(o, \vec{w}) = F(o, \vec{w}, \hat{\mathcal{G}}, \hat{\mathcal{M}}, \mathcal{B}_m, \mathcal{C}_t)$$

where o represents the object, $\vec{w}$ represents the light direction, $\hat{\mathcal{G}}$ is the filtered geometric information, $\hat{\mathcal{M}}$ is the material information, $\mathcal{B}_m$ is the material parameter cache containing albedo, metallicity, and roughness, and $\mathcal{C}_t$ represents the target view. Similar to conventional visibility buffer [BH13] shading processes, we first reconstruct the G-buffer based on $\hat{\mathcal{G}}$ and $\hat{\mathcal{M}}$ before performing final shading.

During reshading, we traverse the compressed material IDs in the impostors and compute a weighted average based on their weight ratios in the target view, ultimately synthesizing the rendered result for the target view. This process ensures that the material information between different impostors is correctly combined, generating a high-quality target view. Because we have stored the material IDs, we can perform editable reshading and modify the material information as needed.

4.3. Editable Reshading with LoD

To accelerate the forward mapping process and ensure consistent rendering effects at different view distances, we select the impostor's LoD based on the target view distance and the input object's radius. According to the Nyquist sampling theorem, the LoD is calculated as follows:

$$L = \log_2(\frac{l}{r})$$

where L represents the LoD of the impostor, l is the view distance, and r is the radius of the input object. This method allows us to choose the appropriate impostor LoD under different viewing conditions, ensuring rendering quality.

5. Result and Analysis

5.1. Experimental Setup

We evaluate our method on an Intel i9-12900K 16-core processor (3.2GHz) with 128GB of RAM and a single NVIDIA GeForce

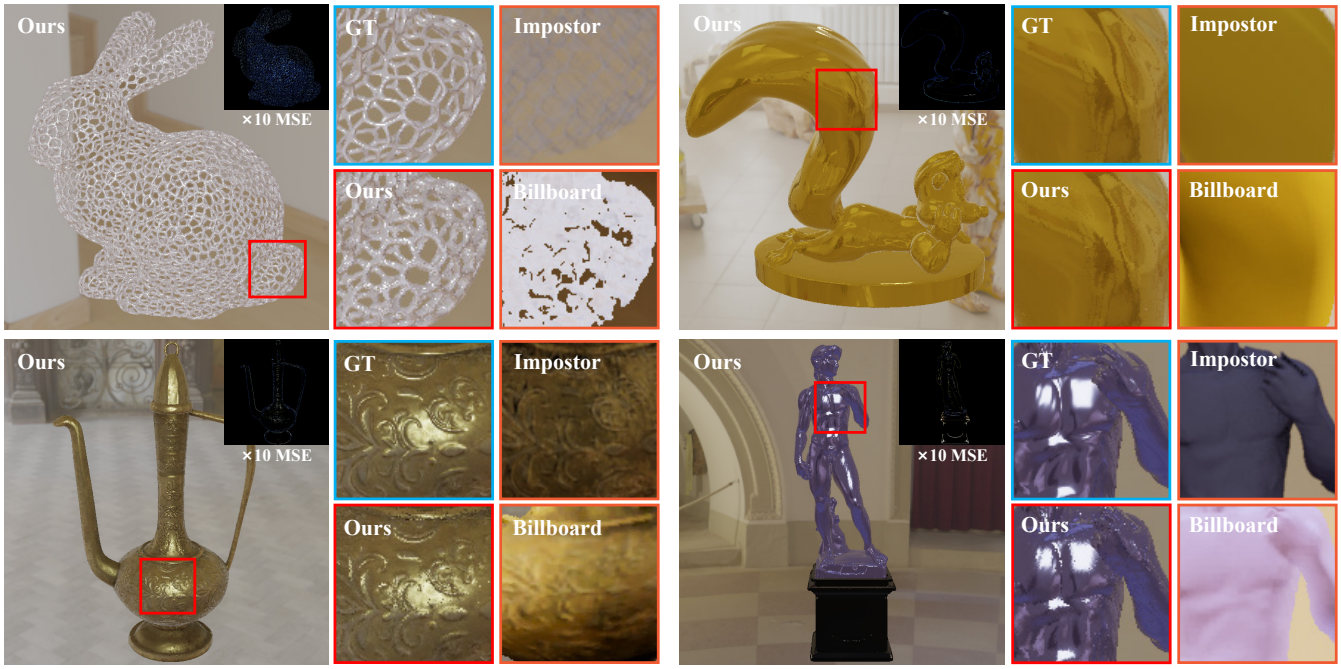

Figure 3: *Quality Comparison. The top row presents the* RABBIT *scene and* PEPE LE *scene, while the bottom row shows the* TEAPOT *scene and* DAVID *scene. Each scene is illuminated by an environment light and a single area light. The Billboard results were rendered using Unreal Engine (UE), while the Impostor results were generated in Blender. Ours utilizes 18 impostors. Consistent lighting conditions maintained across all methods, including Ours and the GT. The MSE indicates the rendering error of Ours compared to the GT.*

RTX 4090 GPU with 24GB of video memory. Our implementation is based on Vulkan. For comparison, we use UE's Impostor Baker [Epi18] to generate multi-card billboards, and Blender's plugin Instant Impostor Baker [Ale23] for the traditional impostor method. The ground-truth is generated via rasterization and traditional deferred shading pipeline. Since our focus lies in evaluating the accuracy of geometric and material representation in RiLoD, we render only direct lighting without any shadow effects. All images are rendered at a resolution of 1024×1024.

As discussed in Section 4.1, RiLoD is constructed from multiple viewpoints and stores geometric and material information in two separate textures. In our implementation, we typically employ 6 to 18 views to capture object details. These views are selected based on regular polyhedral configurations to ensure uniform coverage. For compat storage, $\vec{n}$ are encoded using 2D octahedral coordinates and stored together with depth (d) and transparency (α) in a single R32G32B32A32 texture. The uv are compressed into a 32-bit float, and the material IDs are similarly compressed into another 32-bit float, both of which are stored together in an R32G32 texture. In practice, at higher LoD levels, we encode the material count per impostor group (assuming $\leq$8 materials) using a single 32-bit UINT, where each 4-bit segment represents the count of a distinct material per pixel.

In Section 4.2, the bilateral filter weight $w(p_s, p_c)$ used in Gather Filtering is computed as the product of several weighting terms:

$$w(p_s, p_c) = w_{\text{spatial}}(p_s, p_c) \cdot w_{\text{geometry}}(p_s, p_c) \cdot w_{\text{normal}}(p_s, p_c) \cdot w_{\text{depth}}(p_s, p_c)$$

Here, w_{spatial} and w_{geometry} represent the screen-space and world-space distances between the neighboring pixel p_s and the center pixel p_c, respectively. w_{normal} and w_{depth} measure the consistency of surface orientation and depth, respectively, and are designed to suppress contributions from pixels with significantly different geometric properties.

Scene	DEER	BUNNY	PEPE LE	TEAPOT	DAVID
Tri Count	33053	536406	570992	500738	31317506
Storage (MB)	35.53	44.78	30.68	28.585	22.935

Table 1: *Test Scenes Scale and RiLoD Storage. The storage is compressed using Block Compression [Khr25].*

We evaluate our method across five representative scenes, each designed to highlight common challenges in rasterization: complex textures (the DEER scene), discontinuous depth (the BUNNY scene), high-frequency reflections (the PEPE LE scene), complex normal maps (the TEAPOT scene), and large-scale geometry (the DAVID scene). The scale and characteristics of each scene are summarized in Table 1.

Scene	MSE			LPIPS		
	Ours	Billboard	Impostor	Ours	Billboard	Impostor
DEER	**0.003**	0.040	0.007	**0.005**	0.150	0.064
BUNNY	**0.013**	0.080	0.034	**0.014**	0.205	0.184
PEPE LE	**0.002**	0.056	0.018	**0.001**	0.144	0.063
TEAPOT	**0.002**	0.025	0.008	**0.002**	0.119	0.047
DAVID	**0.001**	0.033	0.009	**0.012**	0.080	0.057

Table 2: *Objective Quality Comparison. We calculated the quality and perceptual metrics between different methods and their GT. Specifically, the Billboard and Impostor methods were compared with the GT generated in UE and Blender, respectively. The best quality is marked in* ***bold****.*

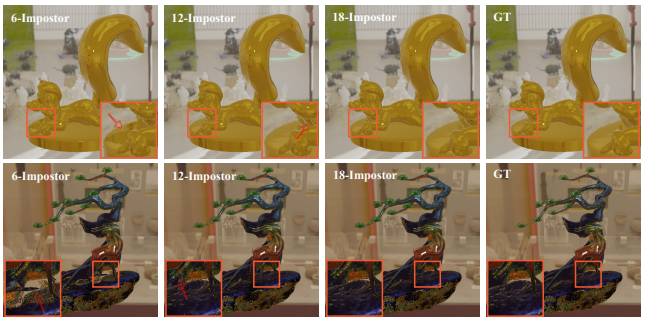

Figure 4: *Multi scale impostors Comparison. The selection of LoD levels is dependent on the complexity of the object. Only 6 impostors are available, some information is missing. With 12 impostors, the information in simpler scenes is nearly complete, though some details are still missing. When 18 impostors are employed, RiLoD can robustly represent complex scenes while significantly reducing rendering costs.*

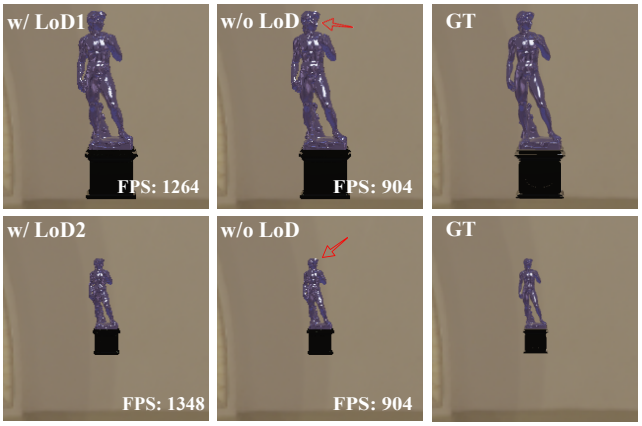

Figure 5: *Lod Comparison. As the distance between the target view and the object increases, the absence of LOD queries leads to flickering of high-frequency information in the results. The GT is obtained using* 2× *SSAA.*

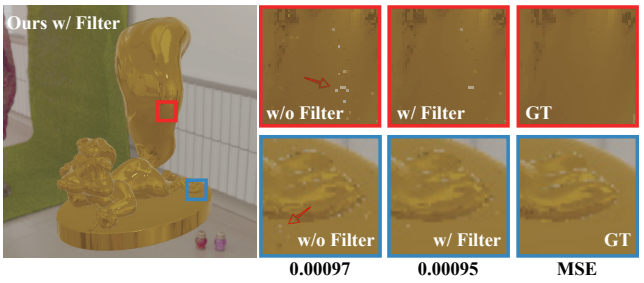

Figure 6: *Hole-filling Filtering effect. By incorporating filtering, we compensate for the inaccuracies introduced by imperfect forward mapping, thereby improving the quality of the rendered results.*

5.2. Quality Analysis

Our method method was compared with the multi-card billboard of UE and the tranditional impostor in Blender. Figure 1 and Figure 3 show the direct lighting results obtained by rendering with RiLoD in five different scenes, while cropped regions further highlight the visual differences among the compared techniques. Traditional impostor and billboard struggle to maintain accuracy when rendering complex geometry, ofen leading to incorrect filling in discontinuous regions (the BUNNY scene). In areas with high-frequency details (the TEAPOT scene), these methods tend to produce excessive blurring. Furthermore, since they only store albedo and normal information, they are unable to capture reflective effects resulting from low roughness and high metallic (the DAVID scene and PEPE LE scene). Table 2 shows that we can achieve better quality compared to Impostor and Billboard. Our outperforms both multi-card billboard and traditional impostor, both in terms of subjective visual quality and objective metrics.

The representation capability of RiLoD at different scales was also evaluated, as shown in Figure 4. We demonstrate the results on a simple purely reflective scene (the PEPE LE scene) and a more complex scene with intricate occlusion and materials (the DEER scene). When fewer impostors are used, our method still provides accurate geometric and material information, though some details are missing. As the number of impostors increases, the missing information is filled in, and the geometric and material details become more accurate.

Figure 5 shows the results of our level-of-details module effect. When using RiLoD, as the target view moves farther from the camera, our method automatically switches to a higher-level LoD, resulting in more stable outcomes and reduced rendering overhead.

Our filter module effectively fills holes and corrects inaccuracies introduced by the forward mapping process, as shown in Figure 6. With the application of this module, we obtain more precise material and geometric information, which subsequently enhances the rendering quality.

Additionally, we demonstrate in Figure 1 that RiLoD allows for the editing of material properties of input objects and supports rendering under arbitrary lighting conditions.

5.3. Performance Analysis

Table 1 presents the storage overhead of RiLoD after compression using Block Compress [Khr25]. We applied BC5 and BC7 for

Scene	Proxy Gen			Forward Mapping			Surfel Filtering			Shading	Total		
	6-Imp.	12-Imp.	18-Imp.	6-Imp.	12-Imp.	18-Imp.	6-Imp.	12-Imp.	18-Imp.	\	6-Imp.	12-Imp.	18-Imp.
Deer	4	6	29	0.08	0.18	0.30	0.09	0.10	0.10	0.05	0.22 (1230 fps)	0.35 (1090 fps)	0.45 (950 fps)
Bunny	8	15	171	0.08	0.2	0.37	0.12	0.12	0.12	0.06	0.26 (1217 fps)	0.38 (1030 fps)	0.55 (850 fps)
Pepe Le	8	15	193	0.08	0.2	0.33	0.09	0.09	0.09	0.05	0.22 (1266 fps)	0.34 (1087 fps)	0.47 (929 fps)
Teapot	5	13	155	0.06	0.17	0.26	0.09	0.09	0.09	0.05	0.20 (1324 fps)	0.31 (1090 fps)	0.40 (1038 fps)
David	182	491	8610	0.06	0.17	0.26	0.10	0.11	0.11	0.05	0.21 (1283 fps)	0.33 (1100 fps)	0.42 (1010 fps)

Table 3: *Render time (in ms) using RiLoD. We recorded the rendering times required by RiLoD at three scales: 6-Impostor, 12-Impostor, and 18-Impostor. Additionally, we measured the actual frame rates achieved during rendering.*

2x compression. The storage overhead of RiLoD is fully acceptable under current hardware conditions. Compared to the multi-card Billboard method for storing identical material parameters, our method, utilizing a compact storage design, maintains at least a comparable storage footprint for materials with simple parameterizations and reduces storage overhead by approximately 38% to 50% for materials characterized by complex textures.

Table 3 presents the rendering time costs using RiLoD at different scales. The method consistently achieves high performance on all scales. By incorporating our LoD strategy, which further reduce memory bandwidth consumption and accelerate forward mapping, replacing interpolation between impostors with more accurate forward mapping is fully feasible. Note that the total rendering time may exceed the actual sum of rendering pass time due to additional overhead, such as CPU-side state management.

5.4. Limitation and Discussion

Although RiLoD has significantly improved in expressing distant objects and supports efficient rendering, our method still has some limitations.

High-frequency geometry structures. RiLoD has difficulty accurately representing high-frequency structures, such as hair or leaf veins. Although forward mapping is precise, small changes in perspective can cause large variations in the mapped results when objects are very thin, especially when information from different impostors conflicts at the object boundaries.

Morphing objects. RiLoD handles rigid transformations well, but real-time updates are needed when the proxied object deform, leading to high computational costs. However, frame-based updates can be used in practical applications to support dynamic objects. It is important to note that our method imposes no limitations on rigid animation.

Editing of geometric information. RiLoD cannot directly edit geometric information. The impostors must maintain geometric consistency, and any change in geometry requires updating all related information in RiLoD.

Refraction. RiLoD supports the rendering of translucent and transparent objects by accounting for intrinsic transparency in the alpha computation; however, it does not currently handle cases involving refraction.

6. Conclusions And Future Work

In this paper, we introduce a new image-based representation designed for real-time distant objects rendering. By enabling both reshading and LoD, our method achieves high rendering efficiency while maintaining geometric and material fidelity under dynamic lighting and varying material conditions. The compact storage design of geometric and material information significantly reduces memory overhead, enabling seamless integration into modern real-time rendering pipelines. Our forward mapping strategy, combined with a hybrid hole-filling approach, mitigates the artifacts commonly introduced by simple view interpolation in prior methods, resulting in rendered outputs that more closely align with the ground-truth. Quantitative evaluations and qualitative comparisons demonstrate that RiLoD not only outperforms existing impostor and billboard techniques in rendering quality, but also maintains low overhead in both storage and performance, making it a practical solution for rendering large-scale scenes with complex distant objects.

Future work may explore integrating our algorithm with virtual reality (VR) systems to enable efficient rendering of distant objects in large-scale immersive environments. In addition, while our current method focuses on local lighting, an important direction is to extend the algorithm to support global illumination, including multiple light bounces. Although we have proposed a compact storage design, further optimization is desirable for handling large-scale scenes. In particular, exploring more advanced compression techniques could help reduce storage requirements and improve runtime performance – especially on devices with limited memory and bandwidth.

Acknowledgments. We thank the reviewers for their valuable comments. This work has been partially supported by the National Natural Science Foundation of China (No. 62272275), the Taishan Scholars Program (No. tsqn202312231), Qilu University of Technology (Shandong Academy of Sciences) Faculty of Computer Science and Technology Pairing Program (No. 2024JDJH13).

References

[Ale23] ALEX. *Instant Impostors v1.3 | One-click Impostor Generation (EEVEE) - Blender Market*. [Online; accessed 2025-04-07]. 2023. URL: https://blendermarket.com/products/instant-impostors--one-click-impostor-generation-eevee 6.

[AMD22] AMD. *AMD FidelityFX™ Super Resolution 2 (FSR 2) - AMD GPUOpen*. [Online; accessed 2025-03-31]. 2022. URL: https://gpuopen.com/fidelityfx-superresolution-2/ 2, 5.

[BH13] BURNS, CHRISTOPHER A and HUNT, WARREN A. "The visibility buffer: a cache-friendly approach to deferred shading". *Journal of Computer Graphics Techniques (JCGT)* 2.2 (2013), 55–69 2, 5.

[Bli82] BLINN, JAMES F. "A generalization of algebraic surface drawing". *ACM transactions on graphics (TOG)* 1.3 (1982), 235–256 2.

[BMT*21] BARRON, JONATHAN T, MILDENHALL, BEN, TANCIK, MATTHEW, et al. "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields". *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, 5855–5864 2.

[Bru18] BRUCKS, RYAN. *Octahedral Impostors*. [Online; accessed 2025-03-31]. 2018. URL: https://shaderbits.com/blog/octahedral-impostors 3.

[Con17] CONFERENCE, GAME DEVELOPERS. *Automated Level of Detail Generation for Halo Reach - YouTube*. [Online; accessed 2025-05-27]. 2017. URL: https://www.youtube.com/watch?v=tnzcuJOTlek 2.

[DDSD03] DÉCORET, XAVIER, DURAND, FRÉDO, SILLION, FRANÇOIS X, and DORSEY, JULIE. "Billboard clouds for extreme model simplification". *ACM SIGGRAPH 2003 Papers*. 2003, 689–696 3.

[DHI*13] DUPUY, JONATHAN, HEITZ, ERIC, IEHL, JEAN-CLAUDE, et al. "Linear efficient antialiased displacement and reflectance mapping". *ACM Transactions on Graphics (TOG)* 32.6 (2013), 1–11 2.

[DKY*00] DOBASHI, YOSHINORI, KANEDA, KAZUFUMI, YAMASHITA, HIDEO, et al. "A simple, efficient method for realistic animation of clouds". *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, 19–28 3.

[DSSD99] DECORET, XAVIER, SILLION, FRANÇOIS, SCHAUFLER, GERNOT, and DORSEY, JULIE. "Multi-layered impostors for accelerated rendering". *Computer Graphics Forum*. Vol. 18. 3. Wiley Online Library. 1999, 61–73 3.

[Epi18] EPIC. *Impostor Baker Plugin in Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community*. [Online; accessed 2025-04-07]. 2018. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/impostor-baker-plugin-in-unreal-engine 6.

[Fer*04] FERNANDO, RANDIMA et al. *GPU gems: programming techniques, tips, and tricks for real-time graphics*. Vol. 590. Addison-Wesley Reading, 2004 3.

[For01] FORSYTH, TOM. "Impostors: adding clutter". *Game programming gems* 2 (2001), 488–496 3.

[FPRJ00] FRISKEN, SARAH F, PERRY, RONALD N, ROCKWOOD, ALYN P, and JONES, THOUIS R. "Adaptively sampled distance fields: A general representation of shape for computer graphics". *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, 249–254 2.

[Guy00] GUYMON, MEL. "Pyro-techniques: Playing with fire". *Game Developer* 7.2 (2000), 23–27 3.

[HFTF15] HE, YONG, FOLEY, THERESA, TATARCHUK, NATALYA, and FATAHALIAN, KAYVON. "A system for rapid, automatic shader level-of-detail". *ACM Transactions on Graphics (TOG)* 34.6 (2015), 1–12 2.

[Kar] KARIS, BRIAN. *Graphic Rants: Normal map filtering using vMF (part 3)*. [Online; accessed 2025-03-28]. URL: https://graphicrants.blogspot.com/2018/05/normal-map-filtering-using-vmf-part-3.html 4.

[Khr25] KHRONOS. *Compressed Image Formats :: Vulkan Documentation Project*. [Online; accessed 2025-04-07]. 2025. URL: https://docs.vulkan.org/spec/latest/appendices/compressedtex.html 6, 7.

[KKLD23] KERBL, BERNHARD, KOPANAS, GEORGIOS, LEIMKÜHLER, THOMAS, and DRETTAKIS, GEORGE. "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.* 42.4 (2023), 139–1 2.

[KL16] KLINT, JOSH and LENGYEL, ERIC. *Vegetation Management in Leadwerks Game Engine 4*. CRC Press, 2016 3.

[KS02] KAWASAKI, SHUNTARO YAMAZAKI RYUSUKE SAGAWA HIROSHI and SAKAUCHI, KATSUSHI IKEUCHI MASAO. "Microfacet billboarding". *13th Eurographics Workshop on Rendering: Pisa, Italy*. 2002, 169 3.

[LC98] LORENSEN, WILLIAM E and CLINE, HARVEY E. "Marching cubes: A high resolution 3D surface construction algorithm". *Seminal graphics: pioneering efforts that shaped the field*. 1998, 347–353 2.

[LPC*00] LEVOY, MARC, PULLI, KARI, CURLESS, BRIAN, et al. "The digital Michelangelo project: 3D scanning of large statues". *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, 131–144 2.

[MB05] MCREYNOLDS, TOM and BLYTHE, DAVID. *Advanced graphics programming using OpenGL*. Elsevier, 2005 3.

[MRS*21] MARTIN-BRUALLA, RICARDO, RADWAN, NOHA, SAJJADI, MEHDI SM, et al. "Nerf in the wild: Neural radiance fields for unconstrained photo collections". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, 7210–7219 2.

[MSO*19] MILDENHALL, BEN, SRINIVASAN, PRATUL P, ORTIZ-CAYON, RODRIGO, et al. "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines". *ACM Transactions on Graphics (ToG)* 38.4 (2019), 1–14 3.

[MST*21] MILDENHALL, BEN, SRINIVASAN, PRATUL P, TANCIK, MATTHEW, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". *Communications of the ACM* 65.1 (2021), 99–106 2.

[OB10] OLANO, MARC and BAKER, DAN. "Lean mapping". *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 2010, 181–188 2.

[PFS*19] PARK, JEONG JOON, FLORENCE, PETER, STRAUB, JULIAN, et al. "Deepsdf: Learning continuous signed distance functions for shape representation". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, 165–174 2.

[PHC*97] PULLI, KARI, HOPPE, HUGUES, COHEN, MICHAEL, et al. "View-based rendering: Visualizing real objects from scanned range and color data". *Rendering Techniques' 97: Proceedings of the Eurographics Workshop in St. Etienne, France, June 16–18, 1997 8*. Springer. 1997, 23–34 3.

[PMDS06] POPESCU, VOICU, MEI, CHUNHUI, DAUBLE, JORDAN, and SACKS, ELISHA. "Reflected-scene impostors for realistic reflections at interactive rates". *Computer Graphics Forum*. Vol. 25. 3. Wiley Online Library. 2006, 313–322 3.

[PZVG00] PFISTER, HANSPETER, ZWICKER, MATTHIAS, VAN BAAR, JEROEN, and GROSS, MARKUS. "Surfels: Surface elements as rendering primitives". *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, 335–342 3.

[QSMG17] QI, CHARLES R, SU, HAO, MO, KAICHUN, and GUIBAS, LEONIDAS J. "Pointnet: Deep learning on point sets for 3d classification and segmentation". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 652–660 2.

[QYSG17] QI, CHARLES RUIZHONGTAI, YI, LI, SU, HAO, and GUIBAS, LEONIDAS J. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". *Advances in neural information processing systems* 30 (2017) 2.

[Ris06] RISSER, ERIC. "True imposters." *SIGGRAPH Research Posters*. 2006, 58 3.

[SALP05] SZIRMAY-KALOS, LÁSZLÓ, ASZÓDI, BARNABÁS, LAZÁNYI, ISTVÁN, and PREMECZ, MÁTYÁS. "Approximate ray-tracing on the gpu with distance impostors". *Computer graphics forum*. Vol. 24. 3. Citeseer. 2005, 695–704 3.

[SCK08] SHUM, HEUNG-YEUNG, CHAN, SHING-CHOW, and KANG, SING BING. *Image-based rendering*. Springer Science & Business Media, 2008 2, 3, 5.

[VK06] VICHITVEJPAISAL, PHONGVARIN and KANONGCHAIYOS, PIZZANU. "Enhanced billboards for model simplification". (2006) 3.

[ZBX*21] ZHU, JUNQIU, BAI, YAOYI, XU, ZILIN, et al. "Neural complex luminaires: representation and rendering." *ACM Trans. Graph.* 40.4 (2021), 57–1 3.