

Frequency-Aware Spatial-Angular Gaussians for Efficient Global Illumination Precomputation and Real-Time Rendering

I. Shah^{1,2}  S. Lucas^{1,2}  G. Drettakis^{1,2} 

¹INRIA

²Université Côte d’Azur

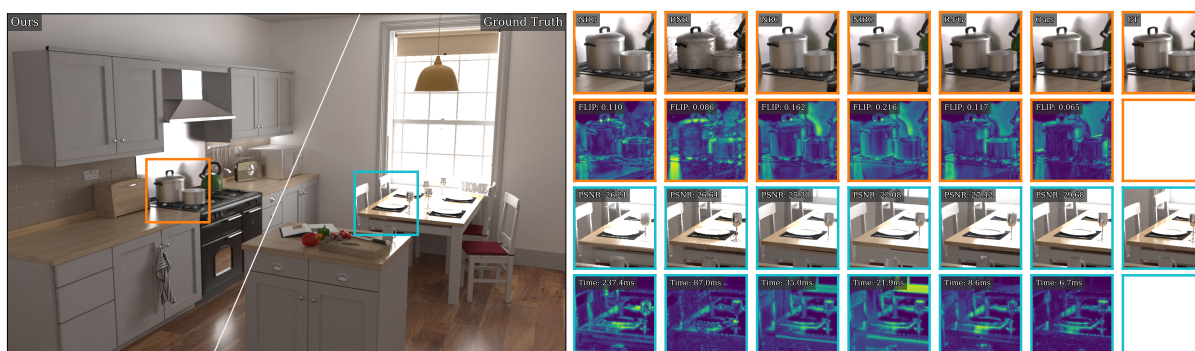


Figure 1: We present a new precomputation method for Global Illumination, allowing real-time rendering. We introduce a spatial-angular Gaussian Mixture Model and a frequency-based sampling scheme that allows efficient optimization and evaluation. Our method represents glossy-glossy transfer with high quality compared to real-time path/photon tracing solutions, and has faster rendering speed than precomputation-based methods, often with better quality.

Abstract

Precomputation and efficient approximate material models are well-known approaches for achieving real-time rendering with global illumination. Recent research in radiance fields and in particular 3D Gaussian Splatting has demonstrated the power of Gaussian primitives both to represent radiance and for fast optimization. Inspired by these advances, we propose a new representation to precompute global illumination, by using 3D Gaussians for the spatial and spherical Gaussians for the directional component of lighting. Critically, we precompute and store incoming radiance, thus requiring the optimization of a spatially smooth function, and we exploit the properties of anisotropic spherical Gaussians for very fast evaluation during rendering. We introduce a frequency-aware sampling method, that guides both the placement of precomputed ground truth illumination samples and the placement of the Gaussians, both spatial and angular. Our approach provides very fast optimization, for a total precomputation time of less than 6 minutes for our test scenes, and better quality rendering than competing approaches at the same frame rates.

1. Introduction

Real-time walkthroughs of 3D scenes with complex global illumination are still an outstanding challenge in computer graphics, despite impressive progress over the years. Such walkthroughs are important for many applications such as architecture, cultural heritage etc., or for creating multi-view image datasets for machine learning (ML) tasks [PMGD21, LGL*25]. Existing solutions for such walkthroughs suffer from visual artifacts for challenging light paths, expensive precomputation times or even slow rendering for neural network methods. We introduce a new lightweight representation

and reconstruction algorithm that provides real-time rendering of static scenes with fast precomputation, exploiting the power of recent primitive-based scene representations and efficient ML-driven optimization.

Traditional solutions for real-time walkthroughs often involve precomputation of radiance in a scene, stored in probes or some other structure [SL17]. Such solutions require sampling radiance at a set of points in space, often by regular placement of probes [MMNL17], or using heuristics [RLP*20], resulting in high memory usage. Additionally, such solutions often suffer in vi-

sual quality due to errors in visibility and parallax when sampling the probes. Recent machine learning approaches try to *learn* a representation of radiance in a scene, usually stored in a neural network (e.g., [DPD22, HCZ21, CDD*24, SDR*24]). These solutions often have long training times – sometimes hours – and slow rendering due to expensive neural network evaluation. High-dimensional (non-neural) primitives have been proposed, but suffer from similar shortcomings [DZK*24]. Real-time path-tracing has been the holy grail of rendering, and methods that re-use samples [BWP*20, MRNK21, OLK*21, LKB*22, KBG24, DKHD25] have provided significant advances in this domain. Nonetheless, there are always light paths that are sufficiently complex that they can only be resolved by precomputation. In addition, path-tracing is inherently a view-dependent, image-space method, often recomputing radiance for every pixel at every frame, which can be wasteful.

We overcome these limitations with a new representation that 1) can be optimized quickly for a given scene, 2) is compact (1-5 MB), and 3) allows high-quality rendering with stable appearance and minimal artifacts. To do this we need to represent illumination in some form, while allowing efficient evaluation of the rendering equation. The rendering equation takes incoming radiance L_i as input, integrates it with the appearance model and outputs outgoing radiance L_o , i.e., the final pixel color. We thus have to decide which of these two quantities to store, and how to optimize this quantity. Storing outgoing radiance would require our representation to model all the intricacies of texture and materials; for this reason, we instead choose to store the simpler *incoming radiance* L_i . However, the final goal is to render *outgoing radiance* L_o , and thus we do not need to have a very accurate representation of L_i , as long as L_o is reproduced efficiently and accurately during rendering. For efficiency, we use Anisotropic Spherical Gaussians (ASG), that allow real-time evaluation of L_o during rendering. For accuracy, we optimize against *outgoing radiance*, by evaluating L_o with our representation for a set of given 3D points and directions, and comparing with ground truth (GT), path-traced L_o at the same points and directions during optimization.

Concretely, we represent incoming radiance with a new Gaussian Mixture Model (GMM) of 3D Gaussian primitives, each augmented with an ASG. Spatial variation (e.g., due to shadows) is modeled with a distribution of spatial 3D Gaussians, while complex view-dependent effects will result in more spatial Gaussians in the same location, resulting in more ASGs representing more directions. We optimize this representation using adaptive initialization of primitives, and by using a frequency-based interpretation of our representation to allow adaptive sampling of GT points and adaptive control of the density of primitives. Once the optimization is complete, we can render the scene in real-time by directly evaluating the GMM in regions with diffuse or high roughness materials, and by performing a second bounce on highly specular surfaces.

In summary, we introduce three main contributions:

- A compact spatio-angular GMM representation for incoming radiance using Gaussian primitives, allowing fast optimization and real-time rendering of outgoing radiance.
- An efficient optimization strategy that exploits a frequency-

based interpretation of our GMM to adaptively control both the GT points used in optimization and the primitives themselves.

- A real-time renderer for global illumination of static scenes that efficiently evaluates outgoing radiance from our representation, which is used either directly when possible or after an additional bounce for low roughness materials.

Our evaluation shows that our solution can generate an optimized representation for typical test scenes in 3-6 minutes on average, and provide state-of-the-art visual quality while rendering at over 100 frames per second, outperforming previous methods. We will release our source code on publication.

2. Previous Work

The literature on Global Illumination (GI) and Gaussian representations is vast. We review the most relevant real-time global illumination methods, focusing on traditional and neural approaches, and we briefly discuss the most relevant literature on Gaussian representations. We refer the reader to surveys or books for more comprehensive reviews for each field (e.g., [TTM*22, WYZ*24, PJH23]).

2.1. Traditional Global Illumination

Computing GI with Physically Based Rendering (PBR) has been one of the major goals of Computer Graphics from its very early days, with the introduction of the radiosity method [GTGB84]. Radiosity can be seen as a precomputation solution, since the mesh with computed radiosity values can then be displayed interactively. Kajiya [Kaj86] introduced path tracing, which uses Monte Carlo integration to solve the rendering equation, requiring the expensive computation of GI for each pixel at each frame.

Cheap approximations for lighting have an even older history, with environment maps [BN76] used as a simple approximation to realistically light objects in a virtual scene. Light probes [Deb06] can be used to capture a real lighting environment, often placed at several positions in a scene. Rendering glossy materials requires blending of multiple probes, creating ghosting artifacts if we do not compensate for the parallax effect. This can be addressed by reprojection [RLP*20]. A similar idea was used in path-guiding [RHL20] where a mixture of spherical Gaussians is transformed to compensate for parallax. We also use spherical Gaussians, and provide a solution for parallax.

Another approach for dealing with the expense of GI is pre-computation. In its simplest form, lighting is simply “baked” into a texture map, an approach used widely in industry [SSS*20]. A very popular technique is Precomputed Radiance Transfer (PRT) [SKS02] that encodes radiance using spherical harmonics, leading to many followup methods [NRH03, TS06, LZT*08]. However, PRT based methods are usually limited to scenes being lit by distant environment maps. This is because computing and storing the lighting coefficients for spatially varying lighting is prohibitively expensive. The use of 3D Gaussians in our representation allows us to model spatially varying illumination efficiently.

Irradiance caching [WH92, SJJ12] is limited to diffuse surfaces; glossy surfaces require additional ray tracing until a diffuse hit

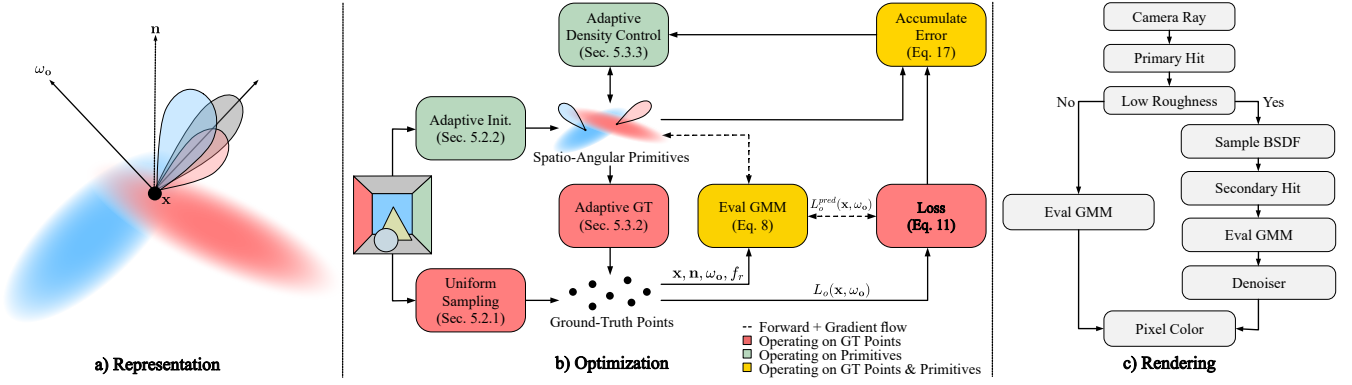


Figure 2: Overview of our method. a) Representation evaluated at a shading point: blue and red lobes are ASGs; gray lobe is the BRDF. b) Optimization pipeline: we uniformly sample the scene to initialize GT points and adaptively sample primitive positions. Adaptive GT adds GT points and Adaptive Density Control adjusts the primitive count throughout optimization. c) Rendering process: two branches are taken depending on surface roughness at the shading point.

is found. Radiance caching [KGPB05, SNRS12] addresses this by storing full incoming radiance at cache records and interpolating at runtime, using first- or second-order approximations [SJJ12]. In contrast, our method uses gradient-based optimization to directly minimize the difference between our representation and observed ground-truth radiance.

We build on the work of Xu et al. [XSD*13] who used anisotropic spherical Gaussians, and relied on the closed form integral of anisotropic spherical Gaussians to efficiently compute the rendering equation by using Gaussians to model incoming radiance and the BRDF. More recently, Currius et al. [CDAS20] optimize a CNN which outputs a list of spherical Gaussians per texel which is compared against a local lightfield map at each texel. This results in long “baking” and training times. In contrast, we develop a solution where precomputation of GI is very fast, in the order of tens of minutes.

Since the introduction of the RTX ray-tracing accelerator [Bur20], the graphics community has developed numerous solutions striving towards the goal of real-time global illumination, typically through path tracing. Despite the efficiency of the hardware, the sheer complexity of the problem makes a brute force solution infeasible. Initially designed for the computation of direct illumination, ReSTIR [BWP*20] proposed reusing light samples over time and on neighboring pixels. This method has been further extended to GI [OLK*21, LKB*22, KBG23]. While these methods improve the result, renderings are still noisy and thus rely heavily on the use of denoisers [BVM*17, Áfr25]. In addition, spatial and temporal reuse significantly improve sampling efficiency but lead to significant artifacts when cameras move quickly. Our solution also employs a denoiser, but our results show that we have significantly reduced visual artifacts.

2.2. Neural Global Illumination

Deep learning has been adopted extensively for global illumination. Neural Radiosity [HCZ21] encodes scene properties as GBuffers [ST91] which are then fed into a neural network to

retrieve the outgoing radiance of the scene. Building upon this work, [CDD*24] proposed a spatial and temporal encoding for dynamic scenes, albeit restricted to a small set of variable parameters. [DPD22] improved the network training efficiency by exploring the training space using Markov Chain Monte Carlo (MCMC). Recently, [ZHM*23] and [ZHH*24] introduced object-oriented neural rendering. [MZL*25] improved the rendering of glossy and specular objects through the use of an additional set of GBuffers storing second hit information. These methods precompute global illumination for dynamic scenes, but only work for a small, fixed range of dynamic parameters, have long training times, and have rendering (inference) speeds that are too slow for real-time rendering. We chose to focus on static scenes with real-time performance.

The conceptual link between PRT and neural networks was identified very early on, where [RWG*13] fit a neural network to pre-computed indirect illumination data for dynamic viewing and lighting conditions. More recent work has followed a similar trend for glossy scene relighting [RXL*23] and to study the various design tradeoffs between traditional and neural approaches [RBRD22]. Xu et al. [XZW*22] propose a lightweight neural representation able to handle operations such as rotation and double/triple product integral extensively used in PRT. Like traditional PRT based methods, these methods are also limited to scenes lit by environment maps.

Another use of machine learning is to use neural networks as radiance caches. [MRNK21] train a neural network on the fly to model outgoing radiance. More recently, [DKHD25] model incoming radiance, but since the neural network has trouble representing high frequencies, the radiance cache is usually evaluated after the ray intersects fairly rough surfaces.

While neural methods show great promise for the future of global illumination, we prefer to avoid them to achieve the best possible performance.

2.3. Gaussian based representation

While we render synthetic geometric models with material properties and lights, we have been inspired by recent work in radiance fields, and in particular the use of Gaussian primitives to represent radiance. Kerbl, Kopanas et al. [KKLD23] introduced 3D Gaussian Splatting (3DGS), utilizing 3D Gaussians for novel view synthesis. This method achieved real-time rendering after a very efficient optimization procedure using gradient descent to fit Gaussian primitives to the input images. Spherical harmonics were used to represent outgoing radiance. There has been a very significant body of followup work, with several attempts to change the primitive type and/or the representation for outgoing radiance [LSC*25, MGW*25]. A major problem in the 3DGS literature is the placement of primitives, that depends on both spatial and directional frequencies [DHS*05]. Some exploratory work has been proposed, e.g., [KRS*25] who reallocate Gaussians where needed assuming that training is a MCMC sampling process.

Related to our approach, [DZK*24] use N-D Gaussians to model correlation arising between scene properties by fitting higher dimensional Gaussians to the scene. Mixture models have also been applied for Path Guiding. [VKŠ*14, RHL20] fit von Mises-Fisher (vMF) distributions, while [DPÖM22, SHJD22] use 5-D and 9-D mixture models respectively for path guiding. Recently [JSL*25] extend radiosity by handling non-binary visibility and semi-opaque primitives (using 2D Gaussians). In these approaches and for synthetic scenes, scene properties such as intersections, albedo, roughness or normals are generated using GBuffers for fast evaluation. We also use GBuffers, but opt to represent the spatial component of incoming radiance with 3D Gaussians, and use Anisotropic Spherical Gaussians (ASG) for the directional component, instead of an N-D representation. This makes reasoning about spatial and directional frequencies easier, and allows us to benefit from the fast evaluation of ASGs at rendering time.

3. Overview

In what follows, we first discuss our representation of incoming radiance (see Fig. 2(a)), explaining our use of 3D Gaussians to represent spatial variation such as shadows, and ASGs for directional variation that allow efficient integration (Section 4). The optimization process is shown in Fig. 2(b) and described in Section 5. GMM primitives are initialized adaptively, and then optimized by computing the loss based on a set of GT points on which $L_o(\mathbf{x}, \omega_o)$ is computed using path tracing. These GT points are initialized using uniform sampling and then sampled adaptively during optimization. In Fig. 2(b), operations on GMM primitives are shown in green boxes, those on GT points in red, while operations involving both are shown in yellow. Finally, in Fig. 2(c) we show how real-time rendering is achieved by either directly evaluating the GMM for high roughness objects, or using a second bounce for shinier appearance.

4. Representation

Our goal is to compute outgoing radiance $L_o(\mathbf{x}, \omega_o)$ from a point $\mathbf{x}$ with normal $\mathbf{n}$, in direction ω_o given by the rendering or Light



Figure 3: Visualization of outgoing radiance (left) vs incoming radiance (right). Since incoming radiance is a spherical function, we visualized $\int_{\Omega} L_i(\mathbf{x}, \omega_i) d\omega_i$. The incoming radiance is much smoother than the outgoing radiance.

Transport equation (LTE):

$$L_o(\mathbf{x}, \omega_o) = \int_{\Omega} f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) (\omega_i \cdot \mathbf{n}) d\omega_i, \quad (1)$$

where $f_r(\mathbf{x}, \omega_i, \omega_o)$ is the Bi-directional Reflectance Distribution Function (BRDF) which describes the appearance of the surface, $L_i(\mathbf{x}, \omega_i)$ is the incoming radiance from direction ω_i at the shading point $\mathbf{x}$, and $(\omega_i \cdot \mathbf{n})$ is the cosine foreshortening term. Since closed-form solutions to this integral are generally not available, we will use precomputation and an efficient representation to allow real-time rendering.

Figure 3 illustrates the difference between modelling incoming radiance $L_i(\mathbf{x}, \omega_i)$ and outgoing radiance $L_o(\mathbf{x}, \omega_o)$, showing that $L_i(\mathbf{x}, \omega_i)$ is much smoother and does not vary with the object’s appearance; it is thus simpler to represent this function.

Our representation must capture both the spatial variation of $L_i(\mathbf{x}, \omega_i)$ (e.g., shadows; see Fig. 3, right) and its angular variation with sufficient accuracy to evaluate $L_o(\mathbf{x}, \omega_o)$ at each shading point. The required angular fidelity depends on surface roughness: since convolution with the BRDF acts as a low-pass filter, diffuse surfaces need only the average incoming radiance, while glossy surfaces require a more detailed representation. Figure 4 illustrates this—a single ASG suffices at a diffuse shading point (orange inset), whereas multiple ASGs are needed at a glossy one (cyan inset).

To capture spatial variations, we use 3D Gaussians, that have proven to be extremely efficient at optimizing radiance fields [KKLD23]. An un-normalized 3D Gaussian is defined as follows:

$$G_s(\mathbf{x}; \mu_s, \Sigma_s) = e^{-\frac{1}{2}(\mathbf{x} - \mu_s)^T \Sigma_s^{-1} (\mathbf{x} - \mu_s)}, \quad (2)$$

where μ_s is the position, and Σ_s is the covariance of the 3D Gaussian.

In addition to the 3D Gaussian that models spatial variation, each of our primitives has an ASG to represent directional variation. ASGs have been shown to be very effective for real-time rendering [XSD*13, CDAS20], because they allow very efficient evaluation of the rendering equation at a shading point; we show how we use this property below (Section 4.3). An ASG is defined as

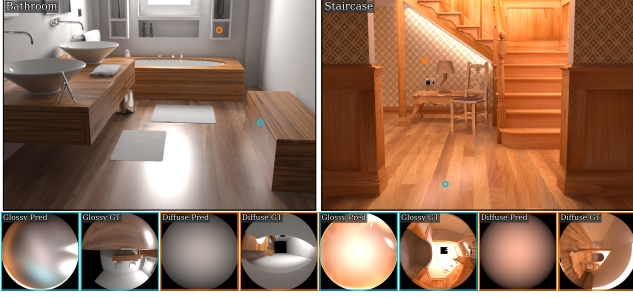


Figure 4: Visualization of learnt incoming radiance at a glossy (cyan inset) & a diffuse (orange inset) point. Since we optimize on outgoing radiance, and the BRDF acts as a low pass filter, our representation learns the incoming radiance only to the required fidelity

follows:

$$G_a(\omega; [\mathbf{x}, \mathbf{y}, \mathbf{z}], [s_x, s_y]) = (\omega \cdot \mathbf{z}) e^{-\left(\frac{\omega \cdot \mathbf{x}}{s_x}\right)^2 - \left(\frac{\omega \cdot \mathbf{y}}{s_y}\right)^2}, \quad (3)$$

where $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ are orthogonal vectors (tangent, bi-tangent and mean) forming the basis of the ASG and $[s_x, s_y]$ is the spread of the ASG. An ASG can also be written in a more general form using a covariance matrix Σ_a as follows:

$$G_a(\omega; \Sigma_a) = (\omega \cdot \mathbf{z}) e^{-\omega^T \Sigma_a^{-1} \omega}. \quad (4)$$

More details on ASGs are provided in the appendix Section A.

Our final representation uses the product of the 3D Gaussian and the ASG to represent incoming radiance $L_i(\mathbf{x}, \omega)$ as follows:

$$L_i(\mathbf{x}, \omega) \approx \sum_{j=1}^N \mathbf{w}_j \mathcal{G}(\mathbf{x}, \omega; \mu_s^j, \Sigma_s^j, \Sigma_a^j), \quad (5)$$

where $\mathbf{w}_j$ is the RGB weight of each primitive which will be optimized. Our primitive $\mathcal{G}(\mathbf{x}, \omega; \mu_s^j, \Sigma_s^j, \Sigma_a^j)$ is the product of the spatial 3D Gaussian $G_s(\mathbf{x}; \mu_s^j, \Sigma_s^j)$ and the ASG $G_a(\omega; \Sigma_a^j)$:

$$\mathcal{G}(\mathbf{x}, \omega; \mu_s^j, \Sigma_s^j, \Sigma_a^j) = G_s(\mathbf{x}; \mu_s^j, \Sigma_s^j) G_a(\omega; \Sigma_a^j). \quad (6)$$

4.1. Spatial-Angular Correlation

The 3D Gaussian models spatial variation of radiance and the ASG models angular variation; however, we also need to model the correlation between spatial and angular components. An example is parallax, shown in Fig. 5: the result of the GMM evaluation depends on the spatial position at which we evaluate the 3D Gaussian. We need a way to change the orientation of the ASG based upon this position.

To model this, instead of optimizing the Σ_a directly, we optimize a transform $\mathcal{T} : \mathbb{R}^3 \rightarrow SO(3)$ which maps the position at which the primitive is efficiently evaluated to the orientation of the ASG. Concretely, we get $q = \text{normalize}(\mathbf{W}\hat{\mathbf{x}} + \mathbf{b})$, where $\mathbf{W}$ and $\mathbf{b}$ are a 4×3 matrix and a 4D vector respectively which represent the transform. $\hat{\mathbf{x}}$ is the position of the point in local coordinates and q is the quaternion representing the rotation of the ASG. The quaternion q

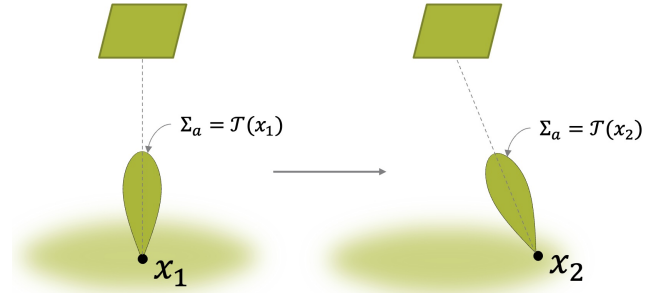


Figure 5: Our representation has a 3D Gaussian to represent the spatial component (blurred ellipse) and an Anisotropic Spherical Gaussian for the directional component. We need to take into account the parallax effect when evaluating radiance at a different point of the 3D Gaussian.

is then converted to a rotation matrix to get the final covariance matrix Σ_a of the ASG (Eq. (23) in appendix Section A). This transform $(\mathbf{W}, \mathbf{b})$ is optimized for each primitive separately. This process is illustrated in Fig. 5. It is important to note that the purpose of the correlation is to model smooth changes. Abrupt changes (caused by visibility for example) are handled by placing more primitives (see Section 5.3.3) or by performing a secondary bounce (see Section 6.2).

4.2. Per Object Primitives

Given that we have a full scene description, it is important that the representation corresponds directly to the objects in the scene, notably to avoid light leaking across surfaces. For this, we optimize a separate set of primitives per object. This is done by storing the ID of the shape (object) on which the primitive was initialized along with the properties of the primitive. During evaluation of the GMM, we check if the shape of the shading point matches with that stored in the primitive and only then compute its contribution. This helps reduce light bleeding artifacts. It also improves performance as we can further limit the bounds of the primitive to be that of the shape it belongs to Section 6.1.1.

4.3. Integral and Outgoing Radiance Evaluation

To exploit properties of ASGs in real time rendering, we represent the BRDF at the shading point using Isotropic Spherical Gaussians (SG), allowing real-time, closed form evaluation [XSD*13]. An SG is defined as:

$$G_{iso}(\omega; \mu, \lambda) = e^{\lambda(\mu \cdot \omega - 1)}, \quad (7)$$

where λ is the sharpness (concentration) of the lobe and μ is the mean direction. We use the convolution property of ASGs (see Eq. 25 in the appendix Section A) to compute the integral in the LTE efficiently [XSD*13]. To this end, we first approximate the BRDF

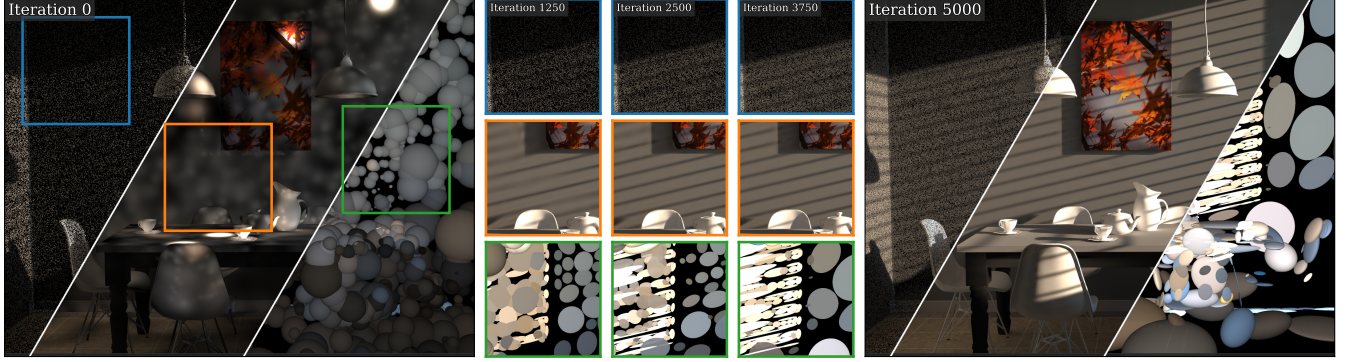


Figure 6: We begin by uniformly distributing path-traced samples (blue inset) and initializing primitives using an adaptive strategy (orange inset, Section 5.2.2). The distribution is then optimized over the path-traced samples (middle columns), with additional samples introduced in high-frequency regions via our adaptive GT algorithm (Section 5.3.2). Primitives are iteratively densified and culled based on their frequencies (Section 5.3.3) to efficiently model the scene. The green inset visualizes the primitives as ellipsoids. Over the course of optimization, primitives are progressively redistributed to regions where they are most needed.

times the geometry term with a mixture of SGs as follows:

$$f_r(\mathbf{x}, \omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) \approx \sum_{i=1}^N \mathbf{w}_i(\mathbf{x}, \omega_o) G_{iso}(\omega_i; \mu^i(\mathbf{x}, \omega_o), \lambda^i(\mathbf{x}, \omega_o)). \quad (8)$$

We use SGs rather than ASGs for the BRDF because the SG–ASG convolution admits a closed-form solution that is cheaper to evaluate than the ASG–ASG convolution [XSD*13], and we found no discernible quality difference between the two choices in practice.

The parameters of the Isotropic SGs and their weights can be computed analytically on the fly for the widely used Cook-Torrance model [CT81] or be precomputed for more complex BRDFs. We refer the reader to [WRG*09, XSD*13] for further details about obtaining these parameters.

By substituting Eqs. (5), (8), and (25) in Eq. (1), outgoing radiance in direction ω_o is evaluated as follows with our representation:

$$L_o(\mathbf{x}, \omega_o) = \int_{\Omega} f_r(\mathbf{x}, \omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) L_i(\mathbf{x}, \omega_i) d\omega_i \\ \approx \sum_{i=1}^N \sum_{j=1}^M \mathbf{w}_i \mathbf{w}_j G_s(\mathbf{x}; \mu_s^j, \Sigma_s^j) \hat{G}(\mu^i, \lambda^i, \Sigma_a^j), \quad (9)$$

where $\hat{G}(\mu, \lambda, \Sigma_a)$ is the result of the convolution of ASG with SG as defined in Section A. We drop the dependence of the parameters of Isotropic SG on shading point and view direction for brevity. Note that $\mathbf{w}_i$ here are the SG weights for the BRDF decomposition (functions of $\mathbf{x}$ and ω_o), distinct from the GMM primitive weights $\mathbf{w}_j$ in Eq. (5). With this, we can efficiently compute the integral at the shading point $\mathbf{x}$. Most BRDFs, such as isotropic GGX, can be represented with a single SG, so in practice we use $N = 1$ except for plastic where $N = 2$ for diffuse+specular. We provide the exact formulas used for computing the final integral with our supported BRDF models in the supplementary.

5. Optimization

Next, we describe how we optimize our GMM representation of incoming radiance $L_i(\mathbf{x}, \omega_i)$ to allow high-quality rendering. As discussed above (Section 4), we only need to represent incoming radiance $L_i(\mathbf{x}, \omega_i)$ at sufficient accuracy to render *outgoing radiance* $L_o(\mathbf{x}, \omega_o)$. We describe our loss, discuss initialization of both GT points and the Gaussian primitives, and finally discuss our adaptive optimization process which is guided by a frequency-based interpretation of our Gaussian representation.

5.1. Loss

We optimize the GMM parameters via gradient descent, minimizing the loss between the path-traced $L_o(\mathbf{x}, \omega_o)$ at a set of 3D points and the corresponding values $L_o^{pred}(\mathbf{x}, \omega_o)$ predicted by our GMM. Since the outgoing radiance has high dynamic range, we use an absolute log loss on outgoing radiance following previous work [DAD*18, WTC*25]. The loss on outgoing radiance is:

$$\mathcal{L}_{rad}(i) = \left| \log(L_o^{gt}(\mathbf{x}_i, \omega_i) + 1) - \log(L_o^{pred}(\mathbf{x}_i, \omega_i) + 1) \right|. \quad (10)$$

We also add a scale regularizer during optimization to encourage primitives to become smaller. It is defined as:

$$\mathcal{L}_{scale}(i) = \sum_{k=1}^3 \sqrt{\text{eigval}_k(\Sigma_s^i)}. \quad (11)$$

This regularizer ensures spatial primitives do not expand in empty space, avoiding unnecessary primitive evaluation, and it helps with pruning since reducing the number of samples covered by a primitive will eventually lead them to be pruned; see Section 5.3.3.

Along with the outgoing radiance loss (Eq. (10)), the final loss becomes:

$$\mathcal{L} = \frac{1}{M} \sum_{i=1}^M \mathcal{L}_{rad}(i) + \lambda \frac{1}{N} \sum_{i=1}^N \mathcal{L}_{scale}(i). \quad (12)$$

We found $\lambda = 0.01$ to work well in all our test scenes.

5.2. Initialization

The optimization process creates a set of our spatio-angular Gaussian primitives, and optimizes their parameters to represent incoming radiance based on path-traced radiance values at 3D GT points. We next describe how we initialize the set of GT points and the set of primitives.

5.2.1. GT Point Initialization

We start with a uniform distribution of 3D points on the geometry in the scene, and compute path-traced $L_o(\mathbf{x}, \omega_o)$ on each point, using Mitsuba 3 [JSR*22]. Each GT point has a position $\mathbf{x}$, normal $\mathbf{n}$, an outgoing view direction ω_o and material properties f_r . This is illustrated in Fig. 6, blue inset. We place $10\times$ more samples on glossy surfaces, to capture both the spatial and angular variability in illumination.

5.2.2. Adaptive Primitive Initialization

Since our GMM represents incoming radiance, there are two factors that affect the density of primitives required to accurately represent this signal. First, *shadows* due to direct illumination, which create strong discontinuities that the primitives need to reproduce, and second, regions where the geometry has *high curvature* and corners require more primitives to correctly reconstruct incoming light.

During initialization, we first sample 1 million primitive positions on the scene with uniform probability. At each sampled position, we need to efficiently estimate the probability of being close to a visibility discontinuity, and a probability of being a high curvature region. For this, we compute two values, the visibility factor $\mathcal{V}(\mathbf{x})$ and harmonic distance $D(\mathbf{x})$ [DHS*05] respectively.

The visibility factor is defined as:

$$\mathcal{V}(\mathbf{x}) = \int_{\Omega} \mathcal{V}(\mathbf{x}, \omega) d\omega, \quad (13)$$

where $\mathcal{V}(\mathbf{x}, \omega)$ is 1 if the ray starting from $\mathbf{x}$ in direction ω hits a light source and 0 otherwise. This value captures the regions with shadows due to direct illumination (similar to the shadow term in [HHM18]). We want to place more primitives at the boundaries of such shadows. To find these regions, we compute the variance of the visibility factor $\text{Var}[\mathcal{V}(x)]$ in the local neighborhood. The details of how the variance is computed are given in the supplementary.

The harmonic distance is defined as:

$$D(\mathbf{x}) = \int_{\Omega} \frac{1}{d(\mathbf{x}, \omega)} d\omega, \quad (14)$$

where $d(\mathbf{x}, \omega)$ is the distance to the closest object from $\mathbf{x}$ in direction ω . This value captures the regions where there is a sharp change of curvature.

We use Monte-Carlo integration to estimate both values. We compute these with 4K rays per sample; since we only perform a single ray trace cast per sample, the computation is relatively cheap ($\sim 4 - 6$ s). We normalize $|\text{Var}[\mathcal{V}(x)]|$ and $D(\mathbf{x})$ using quantile normalization (10%-90%) separately and build a PDF:

$$p(\mathbf{x}) = \frac{1}{2}(\hat{\mathcal{V}}(\mathbf{x}) + \hat{D}(\mathbf{x})), \quad (15)$$

where, $\hat{\mathcal{V}}(\mathbf{x})$ and $\hat{D}(\mathbf{x})$ are the quantile-normalized values ($\hat{\cdot}$ represents quantile normalization throughout). Finally, we subsample the positions used in initialization based upon the PDF so that we have N points which are converted into primitives. The value of N is scene dependent and chosen by the user. The visibility factor, harmonic distance and the final probability are illustrated in Fig. 7. Further details about initialization of parameters of each primitive are given in the supplementary.

5.3. Optimization Process

The cost of optimization depends on the number of expensive GT path-traced computations we perform, and the number of primitives that need to be evaluated at each iteration. To reduce both of these, we introduce adaptive sampling of GT points, and adaptive density control of primitives. Our Gaussian primitives can be naturally interpreted to provide frequency information that we exploit for these adaptive solutions.

5.3.1. Frequency-Based Interpretation

Durand et al. [DHS*05] performed analysis on computing frequency content of light transport. However, computing a discrete Fourier Transform (DFT) of our GMM on points in space is expensive. Instead, we interpret the frequency content of our spatio-angular primitives and use this to guide our computation. The Fourier Transform of a 3D Gaussian is also a 3D Gaussian with inverse scale, i.e., a narrower Gaussian has higher frequency. In contrast, a closed form solution for the Fourier Transform of an ASG does not exist. However, in a similar spirit to the interpretation of frequency for 3D Gaussians, we see that ASGs with smaller spread have higher angular frequency.

Given the above, we define the spatial frequency of a 3D Gaussian to be:

$$F_s(i) = \frac{1}{\sqrt{\prod \text{eigval}(\Sigma_s^i)}}, \quad (16)$$

since the eigenvalue eigval of the covariance Σ_s^i provides a measure of size. Similarly, we define the angular frequency of an ASG to be:

$$F_a(i) = \frac{1}{s_x^i s_y^i} \quad (17)$$

Similarly to the spatial case, the scale factor s_x^i, s_y^i provides the angular spread. Since our GMM approximates incoming radiance, we will use its frequency content to adaptively sample more GT points in the regions where the frequency content is higher and to guide primitive placement.

5.3.2. Adaptive GT Point Sampling

We start with a uniform distribution of GT points on the geometry in the scene on which path-traced $L_o(\mathbf{x}, \omega_o)$ is computed and start optimizing our GMM on these points. Every 500 iterations, we compute $F_s(i)$ and $F_a(i)$ on each primitive i . We then sample a new set of points on the scene but we do not compute $L_o(\mathbf{x}, \omega_o)$, thus avoiding the expensive path-tracing step.

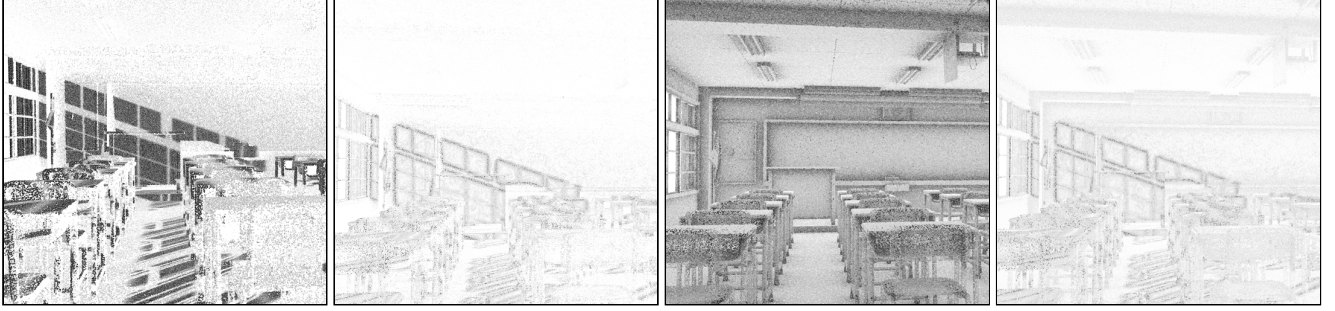


Figure 7: Left to right: The visibility factor, its variance, the harmonic distance and the final probability used to place primitives. The values are inverted for better readability, darker colors correspond to higher values.

Instead, at each such point we compute:

$$\mathcal{F}_s = \max_{i=1}^N \mathbf{1}[J(i, \mathbf{x}, \omega_o) > \tau] \mathcal{F}_s(i)$$

$$\mathcal{F}_a = \max_{i=1}^N \mathbf{1}[J(i, \mathbf{x}, \omega_o) > \tau] \mathcal{F}_a(i)$$

where, $J(i, \mathbf{x}, \omega_o) = \int_{\Omega} w_i \mathcal{G}(\mathbf{x}, \omega; \mu_s^i, \Sigma_s^i, \Sigma_a^i) f_r(\mathbf{x}, \omega_i, \omega_o) (\omega_i \cdot \mathbf{n}) d\omega$ is the contribution of primitive i at the given point and viewing direction, and $\mathbf{1}[\cdot]$ is the indicator function, serving as a filter to consider only primitives relevant for the sample; we found $\tau = 0.001$ to work well in practice. We then normalize both $\mathcal{F}_s$ and $\mathcal{F}_a$ over all points separately using quantile normalization (10%-90%). We build a PDF $p = \frac{1}{2}(\mathcal{F}_s + \mathcal{F}_a)$ and subsample this initial set of uniform points based upon this PDF. We then compute $L_o(\mathbf{x}, \omega_o)$ only on the subsampled points. These points are added to the existing set of path-traced samples which is used for future optimization. We add 10% of the existing number of samples every 500 iterations. In Fig. 6, the middle columns (top row) show the adaptive point sampling in action.

5.3.3. Adaptive Density Control

Even though our initialization strategy tries to allocate primitives where they are required the most, it can still miss some regions. This happens because the heuristics can fail in certain cases, e.g., for complex paths such as multiple glossy bounces (see blue inset in Fig. 11), where the direct visibility factor is insufficient. Hence, we need a densification and pruning strategy which adds and removes primitives during the optimization as required.

5.3.3.1. Primitive Addition. Similar to the path-traced samples, we want to place more primitives in regions with higher frequency. We thus use our frequency estimate to guide new primitive addition.

For each primitive, we first compute $F_s(i)$ and $F_a(i)$, and also compute its error contribution $E(i)$. The error contribution is computed as:

$$E(i) = \frac{1}{M} \sum_{j=0}^M \mathcal{L}_{rad}(j) \mathcal{G}(\mathbf{x}, \omega; \mu_s^j, \Sigma_s^j, \Sigma_a^j), \quad (18)$$

where M is the total number of path-traced samples affected by the primitive (see Eq. (19)).

Similar to Section 5.2.2, we independently normalize $F_s(i)$,

$F_a(i)$, and $E(i)$ using quantile normalization (10%-90%) and then assign each primitive a probability $p(i) = \frac{1}{2}(\hat{F}_s(i) + \hat{F}_a(i)) \times \hat{E}(i)$. We then sample primitives based upon this probability and clone them. This strategy places more primitives in regions of high spatial or angular frequencies. The sum of spatial and angular frequency acts as an OR operator, so primitives are likely to be cloned when either their spatial or angular frequency is high. The multiplication with the error term prevents cloning of primitives if the error is low, by reducing the probability of cloning, preventing cloning in regions where reconstruction quality is already good. We clone all the properties of the primitive, and halve the weights of both the original and cloned primitive to avoid change in the optimization state. The densification and culling is performed at an interval of 100 iterations starting from 500 iterations.

5.3.3.2. Pruning. We prune primitives based upon the number of path-traced samples it affects ($N(i)$), which is computed as:

$$N(i) = \sum_{j=0}^M \mathbf{1}[J(i, \mathbf{x}, \omega_o) > 0.01 \times L_o(\mathbf{x}, \omega_o)]. \quad (19)$$

We prune out the primitive if $N(i)$ is smaller than some threshold N_{min} which is decided based upon the number of points sampled per shape N_{shape} . We found that $N_{min} = \min(500, 0.01 \times N_{shape})$ works well in all scenes. The bottom row of the middle columns in Fig. 6 show how the primitives are densified and pruned to fit where they are needed.

6. Rendering

We can render using our optimized GMM representation either by directly evaluating the GMM at a shading point, or after a second bounce off the shading point. The former method can be used for surfaces that have high roughness, i.e., are closer to diffuse. The latter is required since for specular or near specular surfaces we would require a very accurate representation of the incoming radiance to allow direct evaluation; instead, we choose the additional bounce which can be computed very efficiently.

6.1. Direct GMM Evaluation

When evaluating the GMM directly, we simply evaluate Eq. (9) to obtain $L_o(\mathbf{x}, \omega_o)$ at the shading point $\mathbf{x}$ in the view direction ω_o

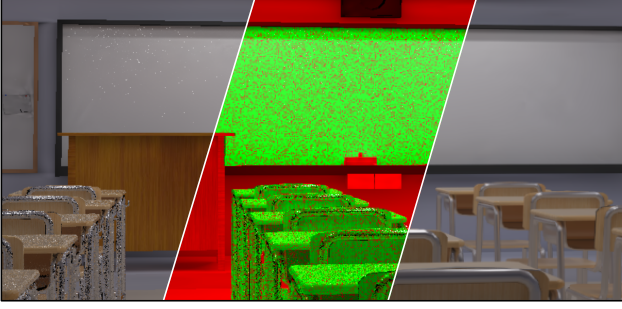


Figure 8: Visualization of the second bounce selection based upon BTH Eq. (20). Red denotes ray termination at the primary hit-point, and green denotes continuation. The leftmost image shows the noisy image and the rightmost shows the denoised image.

defined by the ray from the eye through the given pixel. To perform this evaluation, we need to select the M primitives that will be considered in the sum; evaluating all primitives is prohibitively expensive as well as being wasteful since most of them will have almost no contribution to a given shading point.

6.1.1. Determining Bounds of a Primitive

To limit the number of primitives we consider, we define the spatial bounds of our Spatio-Angular primitives such that their contribution $J(i, \mathbf{x}, \omega_0)$ remains above a threshold of w_{min} . Specifically, for each primitive we want the largest bounds such that $J(i, \mathbf{x}, \omega_0) \geq w_{min} \forall \omega_0 \in \Omega$. To derive this, we must find an upper bound for $J(i, \mathbf{x}, \omega_0)$. Let $w_i = \max_c(\mathbf{w}_i)$ denote the maximum channel of the RGB weight $\mathbf{w}_i$. Since we know that $\int_{\Omega} f_r(\mathbf{x}, \omega_i, \omega_0)(\omega_i \cdot \mathbf{n})d\omega \leq 1 \forall \omega_0 \in \Omega$, the following holds:

$$\begin{aligned} J(i, \mathbf{x}, \omega_0) &\leq \max w_i G(\mathbf{x}, \omega; \mu_s^i, \Sigma_s^i, \Sigma_a^i) \\ &\leq w_i G_s(\mathbf{x}; \mu_s^i, \Sigma_s^i). \end{aligned}$$

Hence, the bound of the primitive can be determined by finding the spatial region where $w_i G_s(\mathbf{x}; \mu_s^i, \Sigma_s^i) \geq w_{min}$. This region is an ellipsoid with semi-axis lengths of:

$$a_j = \sqrt{2 \text{ eigval}_j(\Sigma_i) \ln \left(\frac{w_i}{w_{min}} \right)},$$

We found that $w_{min} = 0.001$ works well in practice.

6.1.2. Finding relevant primitives

After setting the bounds for each primitive, we need to find the list of primitives which enclose each shading point. We model this problem as a Ray-Intersection problem and use the Ray-Tracing cores available in modern GPUs to find such primitives efficiently. Specifically, we build a BVH on our 3D Gaussians which are represented as axis-aligned bounding boxes (AABBs). Because we know that a primitive only contributes to the shape it belongs to, we can further limit the AABB by computing its intersection with the AABB of its shape. We then use it as the input to the BVH building algorithm. This improves traversal performance by reducing empty space covered by the AABBs where it would never be evaluated.

Then we launch a very small finite-ray [KBG23] from each shading point and use an *intersection* shader to accept primitives which enclose the given point. Finally, we use an *anyhit* shader to accumulate the results into the output buffer. During optimization, we update the BVH after every step and rebuild it after every 100 steps to improve performance in subsequent steps.

For shapes with few primitives, BVH traversal overhead can exceed the cost of simply testing all primitives directly. We therefore fall back to a linear scan over the shape’s primitives when the primitive count is below a threshold $N_{lin} = 1000$, bypassing BVH traversal entirely.

6.2. Second bounce

We observe that for specular or near-specular BRDFs the variance introduced by sampling the BRDF lobe is quite low. Hence, we choose to sample the BRDF lobe, trace a ray and evaluate the GMM at the second bounce. Similar approaches have been applied before such as the *Spread Angle Heuristic* (SPH) [MRNK21] and the *Balance Termination Heuristic* (BTH) [DKHD25]. We found that BTH works well for our method. Concretely, at the first bounce, we decide to continue tracing the ray with probability:

$$P_s = \frac{p(\omega_i)}{p(\omega_i) + \frac{1}{\pi}}, \quad (20)$$

where $p(\omega_i)$ is the probability of the BRDF sample. The probability is high for smooth specular surfaces and low for rough surfaces. Figure 8 visualizes the pixels on which the second bounce is performed. We use a denoiser (NRD [NVI]) on the path traced surfaces to handle the little variance introduced by BRDF sampling. We show all results with second bounce and denoising, unless mentioned otherwise. For plastic BRDFs (diffuse+specular), we always evaluate the diffuse component on the first bounce and apply the above heuristic only to the specular component.

7. Implementation, Results and Comparisons

7.1. Implementation

We implemented our method in Python with PyTorch and Slang.D for the optimization step including an interactive viewer [SMLD]. We apply an exponential activation function to the scaling and weight parameters to ensure positivity. We also normalize the quaternions to ensure they represent valid rotations. No constraints are applied on the other parameters of the primitives.

Ground truth path-tracing samples are generated using Mitsuba3 [JSR*22]. For the second bounce, we use the Falcor framework [KCK*22] as it includes the NRD denoiser [NVI] and its architecture is more suited for real-time rendering. We modified the BRDF models in Falcor so that they are compatible with those of Mitsuba, and verified that the converged ground truth images are the same up to a reasonable precision. Details of the modifications made are provided in the supplementary. We also ensured that the geometry of the scenes in the PBRT format read by Falcor and the Mitsuba versions is the same. We will provide all our source code and the consistent versions of the scenes on publication.

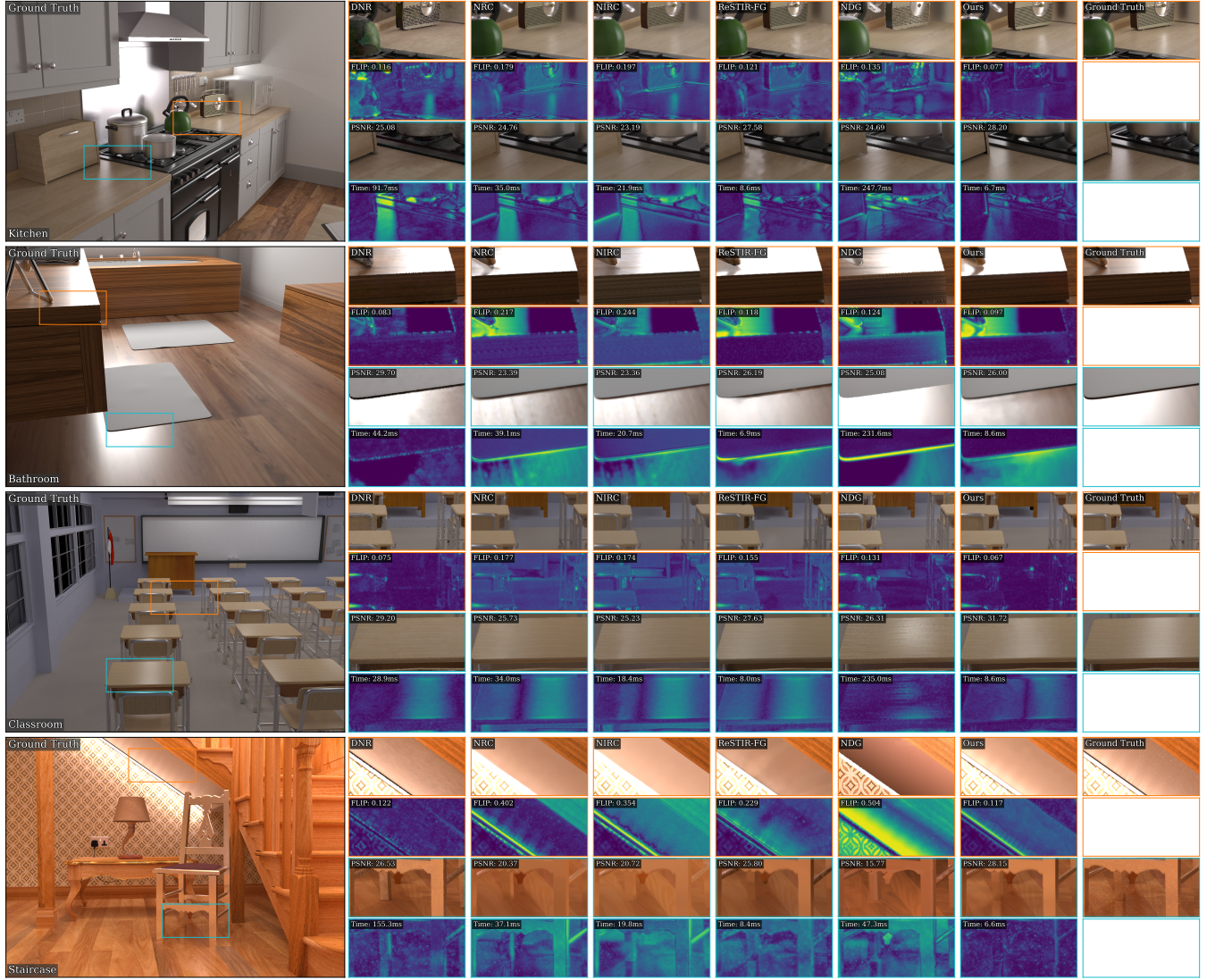


Figure 9: Equal-time qualitative comparison on four scenes (KITCHEN, BATHROOM, CLASSROOM, STAIRCASE). All path-tracing baselines (NRC, NIRC, R-FG) are matched to the frame time of OURS and are denoised; precomputation-based methods (NDG, DNR) produce noise-free output independently of SPP. For each method we show two magnified insets (orange & cyan) alongside their per-pixel Δ LIP error maps, and report Δ LIP, PSNR, and render time for the frame.

7.2. Comparisons

We evaluated our method on a set of standard scenes used in global illumination, from the Bitterli dataset [Bit16]. Specifically, we test on KITCHEN, BATHROOM, CLASSROOM, and STAIRCASE. We modified STAIRCASE to add more shiny materials, making the light transport harder.

We evaluate our method by comparing with several previous methods broadly falling in the two categories, precomputation based and real-time path tracing. For precomputation based methods, we compare with N-D Gaussians (NDG) [DZK*24] and Dynamic Neural Radiosity (DNR) [SDR*24]. For real-time path tracing, we compare with Neural Radiance Caching

(NRC [MRNK21]), Neural Incoming Radiance Caching (NIRC [DKHD25]) and the best available sample reuse method which works at realtime frame rates ReSTIR FG (R-FG [KBG24]). For NDG and DNR, we modify the method to remove the scene description vector from the input as we only deal with static scenes. For NRC, we use the implementation provided in the code release for NIRC as the original paper didn't publish their code. We choose not to compare against reconstruction methods such as 3D Gaussian Splatting (3DGS) [KKLD23], since it is known that they struggle with glossy surfaces which are present extensively in our scenes. Furthermore, since 3DGS learns geometry, appearance and lighting, it's much heavier. Since in synthetic scenes, geometry

	KITCHEN						BATHROOM					
	FLIP ↓	PSNR ↑	MRSE ↓	RT (ms) ↓	PT (s) ↓	Size (MB) ↓	FLIP ↓	PSNR ↑	MRSE ↓	RT (ms) ↓	PT (s) ↓	Size (MB) ↓
NDG	0.136	24.17	0.053	253.33	3704	1.87	0.110	29.17	0.044	187.60	3164	2.15
DNR	0.102	25.61	0.020	93.03	7844	111.02	0.080	32.63	0.027	43.71	4364	111.02
NRC	0.171	24.75	0.404	34.99	—	—	0.224	23.92	0.108	39.15	—	—
NIRC	0.218	21.60	0.410	21.89	—	—	0.207	24.89	0.066	20.73	—	—
R-FG	0.126	26.60	0.014	8.64	—	—	0.123	27.41	0.059	6.94	—	—
OURS	0.075	28.46	0.012	6.67	355	2.08	0.093	29.08	0.078	8.63	142	0.88

	STAIRCASE						CLASSROOM					
	FLIP ↓	PSNR ↑	MRSE ↓	RT (ms) ↓	PT (s) ↓	Size (MB) ↓	FLIP ↓	PSNR ↑	MRSE ↓	RT (ms) ↓	PT (s) ↓	Size (MB) ↓
NDG	0.544	15.56	0.265	56.09	4797	1.21	0.121	27.51	0.044	243.28	3657	1.70
DNR	0.127	26.41	0.663	163.49	7210	111.02	0.079	28.96	0.041	46.17	4337	111.02
NRC	0.392	21.10	0.366	37.12	—	—	0.173	26.26	0.072	34.01	—	—
NIRC	0.398	19.91	0.378	19.80	—	—	0.164	26.34	0.075	18.43	—	—
R-FG	0.207	26.35	0.053	8.44	—	—	0.165	26.92	0.057	7.97	—	—
OURS	0.117	28.43	0.046	6.57	444	2.92	0.070	30.99	0.029	8.63	300	3.13

Table 1: Average $\mathcal{FLIP}$, $PSNR$, $MRSE$, $Rendering\ Time\ (RT)$, $Precomputation\ Time\ (PT)$, and $model\ Size$ over paths in the scene for all methods. Highlight legend: rank 1 (best), rank 2, rank 3.

and materials are available, it’s much more efficient to learn only the lighting.

7.2.1. Denoised Results

We evaluate our method against all baselines on four scenes, measuring rendering quality and reporting an equal-time comparison. We report $\mathcal{FLIP}$ [ANA*20], $PSNR$, and $MRSE$ against a converged 4096 spp path-traced reference at Full-HD resolution (1920×1080) in Table 1. Metrics are evaluated over camera paths of 150–250 frames per scene; a subset of frames and full-path videos are provided in the supplemental. We apply the denoiser only for OURS, NRC, NIRC, and R-FG, as the precomputation-based methods already produce noise-free output.

For a fair equal-time comparison, we match all path-tracing baselines to the average frame time of OURS. Since OURS and R-FG are the fastest methods, this reduces all other baselines to 1 SPP. A qualitative comparison is shown in Fig. 9.

As shown in Table 1, our method outperforms all baselines on most scenes. It successfully captures complex light transport, including table reflections in KITCHEN and staircase-bottom reflections in STAIRCASE (see Fig. 9). On BATHROOM, our method scores below NDG and DNR due to denoiser lag on specular highlights (see insets in Fig. 9); nevertheless, it outperforms all other real-time path tracing methods while remaining significantly cheaper to render and optimize.

7.2.2. Undenoised 1-spp

We also compare undenoised 1-SPP (OURS-ND) outputs against the real-time path tracing baselines (NRC, NIRC, and R-FG), reporting $\mathcal{FLIP}$ [ANA*20] and $MRSE$ over the same camera paths as Section 7.2.1 in Table 2. Our method outperforms all baselines by a large margin: rough and diffuse surfaces are evaluated directly from our representation and are therefore variance-free, while very specular surfaces are handled by path tracing. This hybrid design

	KITCHEN		BATHROOM	
	FLIP ↓	MRSE ↓	FLIP ↓	MRSE ↓
NRC	0.342	0.243	0.269	0.848
NIRC	0.325	0.223	0.247	0.437
R-FG	0.207	0.201	0.178	0.855
OURS	0.092	0.090	0.107	0.756

	STAIRCASE		CLASSROOM	
	FLIP ↓	MRSE ↓	FLIP ↓	MRSE ↓
NRC	0.527	1.802	0.255	4.663
NIRC	0.480	1.456	0.215	1.601
R-FG	0.366	1.519	0.226	2.749
OURS	0.127	0.934	0.074	0.538

Table 2: Average $\mathcal{FLIP}$ and $MRSE$ over paths for 1-SPP undenoised methods. Highlight legend: rank 1 (best), rank 2, rank 3.

combines the strengths of both approaches. Figure 10 shows qualitative results on STAIRCASE and CLASSROOM.

7.3. Ablations

7.3.1. Second bounce

In Fig. 12, we show BATHROOM & KITCHEN scenes rendered with (OURS) and without the second bounce (OURS-FB). We see that in many cases the direct evaluation provides reasonable results, but the second bounce is required to reproduce high-frequency reflections.

7.3.2. Importance of each component

We perform an ablation on various components of our method. We set the number of primitives initialized to be the same across all

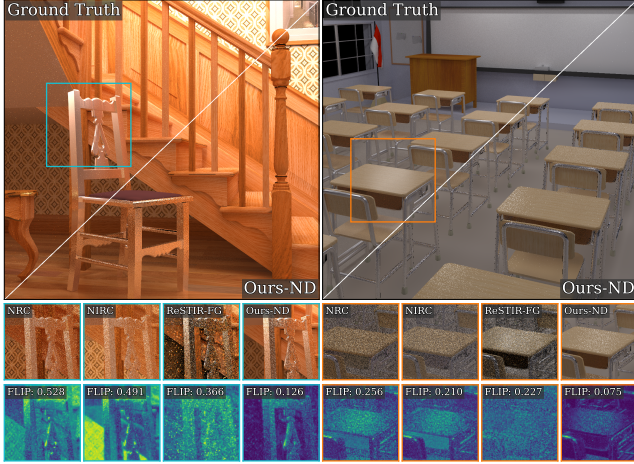


Figure 10: Qualitative comparison of undenoised 1-SPP renderings on STAIRCASE & CLASSROOM. All methods are rendered at 1 SPP without denoising. For each method we show a magnified inset alongside its per-pixel FLIP error maps. OURS-ND produces variance-free output on rough and diffuse surfaces, while a secondary ray is cast with BRDF sampling on glossy surfaces.

methods. We start with a baseline with uniform initialization of primitives with densification and pruning disabled and adaptive GT sampling disabled. This struggles to reconstruct the light transport well because of undersampling in regions with complex light transport (orange inset in Fig. 11). We then enable the adaptive initialization strategy for our primitives which improves the results by placing primitives in regions estimated to have complex light transport. However, some details are missing where the initialization heuristic fails (blue inset in Fig. 11). Furthermore, regions which are over-sampled have streak like artifacts (on the walls). We then enable densification and culling which successfully captures the highlight on the floor and removes primitives from the oversampled regions. Finally, we enable adaptive ground truth computation. With this we get results with comparable quality to densification and culling but also achieve a $\sim 2\times$ speedup in precomputation time. This is because we can start with a smaller number of path-traced samples and add more only where they are required, hence speeding up the iteration times. A missing highlight is also recovered, as adaptive GT sampling concentrates samples in high-frequency regions of the radiance field where they are most needed (red arrow). Figure 11 shows the effect of each component in the KITCHEN scene along with the quantitative metrics.

8. Additional Scenes

In Fig. 13 we show our method on three additional scenes HOUSE, DININGROOM (from Bitterli dataset) and OFFICE which was purchased from Evermotion and converted to Mitsuba. The HOUSE scene shows that our method also works on outdoor scenes and the OFFICE scene shows that our method works on an artist-authored scene with more complex geometry.

PRECOMPUTATION		RENDERING	
Primitive Init.	3.89 s	Primary Hit	0.15 ms
GT Radiance	30.8 s	EvalGMM (FB)	2.02 ms
Forward	32.5 s	EvalGMM (SB)	2.62 ms
Backward	275.4 s	Denoising	1.16 ms
BVH Updates	11.5 s	Misc	0.72 ms

Table 3: Left: precomputation time breakdown of KITCHEN. Right: per-frame rendering time breakdown on KITCHEN (average over path).

SCENE	#Tris	#Prims	PT (s)	RT (ms)	#GT
KITCHEN	1.47M	11K	355	6.67	2.9M
BATHROOM	1.01M	5K	142	8.63	1.5M
STAIRCASE	2.62M	17K	444	6.57	3.7M
CLASSROOM	100K	16K	300	8.63	2.8M
HOUSE	1.81M	9K	101	2.23	2.0M
DININGROOM	270K	6K	128	2.54	2.9M
OFFICE	4.2M	9K	245	12.0	3.7M
HOUSE5X	9M	40K	319	4.32	9.0M

Table 4: Scene statistics. We report the number of triangles (#Tris), number of primitives (#Prims), precomputation time (PT), rendering time (RT) and number of GT points (#GT) for all of the scenes we tested on.

8.1. Performance

We report some performance statistics and analyze the scalability of our method.

8.1.1. Precomputation & Rendering Time

We report the timing breakdown of different components of the precomputation and rendering on the KITCHEN scene in Table 3. The backward pass dominates precomputation, accounting for roughly 78% of the total time, while GT radiance sampling and the forward pass each contribute around 9–10%. Upon cursory profiling of the shaders, we found that the backward pass has low occupancy due to long range register dependencies. For rendering, GMM evaluation (first and second bounce combined) is the main cost at ~ 4.6 ms out of the 6.67 ms total per-frame budget.

8.1.2. Scalability

We investigate the scalability of our method. In Table 4 we report the statistics of all the scenes that we tested our method on. The table shows that the number of GT points is the main factor that affects the scalability of our solution. We perform an experiment to test our method on large scenes by duplicating the HOUSE scene five times to form a new scene HOUSE5X. The results are presented in Fig. 14.

9. Conclusion & Future Work

Our method is not without limitations. The use of a denoiser introduces temporal lag in regions where sampling variance is high (Fig. 15 top). Improving the sampling quality by product sampling



Figure 11: Ablation of our method on the KITCHEN scene. Uniform initialization fails in regions of high frequency light transport (orange inset); adaptive initialization improves placement but misses highlights (blue inset) and introduces streaking in oversampled areas; enabling densification and culling captures highlights and removes artifacts; adaptive GT sampling preserves quality while reducing precomputation time. All images are computed with the first bounce variant (OURS-FB) to better understand the effect of different components.

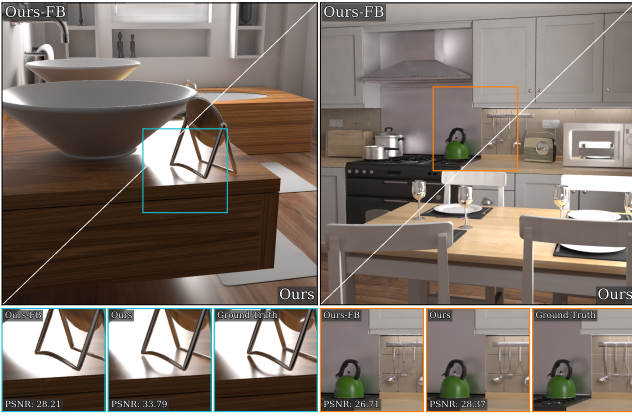


Figure 12: Comparison of GMM evaluation at the first versus second bounce. The representation struggles to capture fine details on highly specular surfaces; evaluating at the second bounce alleviates this issue.

the BRDF and representation could reduce variance and improve denoised results. We also noticed that our method can fail to reconstruct small-scale shadows (Fig. 15 bottom) as the culling strategy is dependent on the number of points a primitive covers. If there aren't enough points in the region of the shadow, the primitive representing it gets culled out. Improving the culling strategy to alleviate this could be explored in future work. Finally, instead of storing the complete global illumination (direct + indirect), one could store only the indirect lighting and rely on sample reuse methods for direct lighting [BWP*20]. As indirect lighting is much smoother, it could improve optimization speed, memory usage and rendering speed. Since our representation is already fast and lightweight, we don't add the dependency of sample reuse methods. This could be explored in future work.

In conclusion, we have presented a new method for precomput-

ing global illumination, allowing consistent, high-quality real-time rendering with only modest precomputation costs. Our representation for incoming radiance combines powerful 3D Gaussians to capture spatial variation and ASGs for angular variation. The former allow efficient optimization, while the latter allow efficient evaluation of radiance for rendering by exploiting the closed form solution for BRDF convolution. We demonstrate that with only 3-5 minutes of precomputation we can achieve rendering quality and speed that is state of the art, compared to previous methods, notably neural precomputation solutions as well as real-time sample reuse solutions.

10. Acknowledgments

This work was funded by the European Union, European Research Council (ERC) Advanced Grant NERPHYS, 101141721 <https://project.inria.fr/nerphys>. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see <https://www.grid5000.fr>). The authors would also like to thank Adobe and NVIDIA for software and hardware donations.

References

- [Áfr25] ÁFRA A. T.: Intel® Open Image Denoise, 2025. <https://www.openimagedenoise.org.3>
- [ANA*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3, 2 (2020), 15:1–15:23. doi:10.1145/3406183.11
- [Bit16] BITTERLI B.: Rendering resources, 2016. URL: <https://benedikt-bitterli.me/resources/>. 10



Figure 13: Our method on additional scenes (HOUSE, DININGROOM and OFFICE). We report PSNR and Δ LIP. Other scene statistics are reported in Table 4.



Figure 14: Experiment analyzing scalability of our method. We report PSNR and Δ LIP for the HOUSE5X variant.

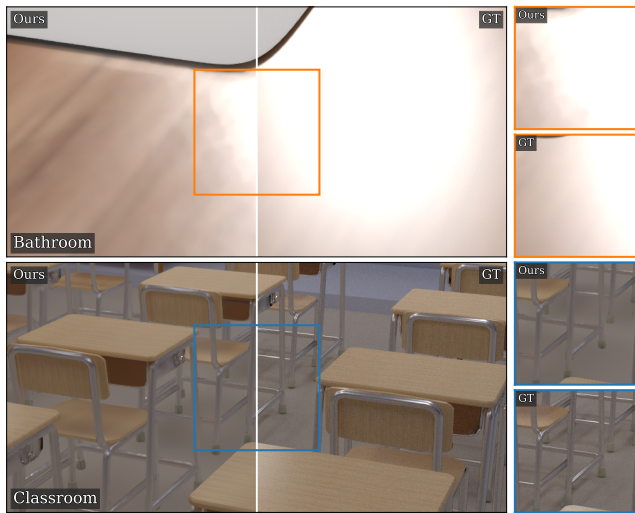


Figure 15: Some limitations of our method. Top: In regions of high variance, the denoiser introduces some temporal lag. Bottom: Our method struggles to capture small-scale shadows due to the culling strategy.

- [BN76] BLINN J. F., NEWELL M. E.: Texture and reflection in computer generated images. *Communications of the ACM* 19, 10 (Oct. 1976), 542–547. [doi:10.1109/dct2v6.2](https://doi.org/10.1109/dct2v6.2)
- [Bur20] BURGESS J.: RTX On: The NVIDIA Turing GPU. *IEEE Micro* 40, 2 (2020), 36–44. [doi:10.1109/MM.2020.2971677](https://doi.org/10.1109/MM.2020.2971677)
- [BVM*17] BAKO S., VOGELS T., MCWILLIAMS B., MEYER M., NOVÁK J., HARVILL A., SEN P., DEROSE T., ROUSSELLE F.: Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36, 4 (July 2017), 97:1–97:14. [doi:10.1145/gbxfbt.3](https://doi.org/10.1145/gbxfbt.3)
- [BWP*20] BITTERLI B., WYMAN C., PHARR M., SHIRLEY P., LEFONN A., JAROSZ W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). [doi:10.1145/gg8xc7.2.3.13](https://doi.org/10.1145/gg8xc7.2.3.13)
- [CDAS20] CURRIUS R. R., DOLONIUS D., ASSARSSON U., SINTORN E.: Spherical Gaussian Light-field Textures for Fast Precomputed Global Illumination. *Computer Graphics Forum (Proceedings of Eurographics)* 39, 2 (2020), 133–146. [doi:10.1111/cgf.15014](https://doi.org/10.1111/cgf.15014)
- [CDD*24] COOMANS A., DOMINICI E. A., DÖRING C., MUELLER J. H., HLADKY J., STEINBERGER M.: Real-time neural rendering of dynamic light fields. *Computer Graphics Forum* (2024). [doi:10.1111/cgf.15014](https://doi.org/10.1111/cgf.15014)
- [CT81] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *Computer Graphics (Proceedings of SIGGRAPH)* 15, 3 (Aug. 1981), 307–316. [doi:10.1145/br5ps6.6](https://doi.org/10.1145/br5ps6.6)
- [DAD*18] DESCHAINTRE V., AITALLA M., DURAND F., DRETTAKIS G., BOUSSEAU A.: Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)* 37, 4 (2018), 1–15.
- [Deb06] DEBEVEC P.: A median cut algorithm for light probe sampling. In *ACM SIGGRAPH Courses* (Boston, Massachusetts, 2006), ACM Press. [doi:10.1145/bcx6bk.2](https://doi.org/10.1145/bcx6bk.2)
- [DHS*05] DURAND F., HOLZSCHUCH N., SOLER C., CHAN E., SIL-LION F. X.: A frequency analysis of light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 24, 3 (July 2005), 1115–1126. [doi:10.1145/cdqv3w.4.7](https://doi.org/10.1145/cdqv3w.4.7)
- [DKHD25] DEREVIANNYKH M., KLEPIKOV D., HANIKA J., DACHS-BACHER C.: Neural two-level monte carlo real-time rendering. *Computer Graphics Forum* 44, 2 (2025), e70050. [doi:https://doi.org/10.1111/cgf.70050](https://doi.org/10.1111/cgf.70050)
- [DPD22] DIOLATZIS S., PHILIP J., DRETTAKIS G.: Active exploration for neural global illumination of variable scenes. *ACM Transactions on Graphics* (2022). [doi:10.1145/cgf.70050](https://doi.org/10.1145/cgf.70050)
- [DPÖM22] DODIK A., PAPAS M., ÖZTIRELI C., MÜLLER T.: Path guiding using spatio-directional mixture models. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 172–189.

- [DZK*24] DIOLATZIS S., ZIRR T., KUZNETSOV A., KOPANAS G., KAPLAYAN A.: N-dimensional Gaussians for fitting of high dimensional functions. In *ACM SIGGRAPH Conference Papers* (New York, NY, USA, July 2024), SIGGRAPH '24, Association for Computing Machinery, pp. 1–11. doi:10/gt48x3. 2, 4, 10
- [GTGB84] GORAL C. M., TORRANCE K. E., GREENBERG D. P., BATAILLE B.: Modeling the interaction of light between diffuse surfaces. *Computer Graphics (Proceedings of SIGGRAPH)* 18, 3 (July 1984), 213–222. doi:10/fsjssr. 2
- [HCZ21] HADADAN S., CHEN S., ZWICKER M.: Neural radiosity. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 40, 6 (Dec. 2021). doi:10/gnszcs. 2, 3
- [HHM18] HEITZ E., HILL S., MCGUIRE M.: Combining analytic direct illumination and stochastic shadows. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2018), ACM Press, pp. 2:1–2:11. doi:10/gfznb7. 7
- [JSL*25] JIANG K., SUN J.-M., LI Z., WANG D., LI T.-M., RAMAMOORTHY R.: Differentiable light transport with gaussian surfels via adapted radiosity for efficient relighting and geometry reconstruction. *ACM Transactions on Graphics (TOG)* 44, 6 (December 2025). 4
- [JSR*22] JAKOB W., SPEIERER S., ROUSSEL N., NIMIER-DAVID M., VICINI D., ZELTNER T., NICOLET B., CRESPO M., LEROY V., ZHANG Z.: Mitsuba 3 renderer, 2022. URL: <https://mitsuba-renderer.org>. 7, 9
- [Kaj86] KAJIYA J. T.: The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH)* 20, 4 (Aug. 1986), 143–150. doi:10/cvf53j. 2
- [KBG23] KERN R., BRÜLL F., GROSCH T.: Accelerated photon mapping for hardware-based ray tracing. *JCGT* 12, 1 (January 2023), 1–28. URL: <http://jcgt.org/published/0012/01/01/>. 3, 9
- [KBG24] KERN R., BRÜLL F., GROSCH T.: ReSTIR FG: Real-time reservoir resampled photon final gathering. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (2024), The Eurographics Association. doi:10.2312/sr.20241155. 2, 10
- [KCK*22] KALLWEIT S., CLARBERG P., KOLB C., DAVIDOVIČ T., YAO K.-H., FOLEY T., HE Y., WU L., CHEN L., AKENINE-MÖLLER T., WYMAN C., CRASSIN C., BENTY N.: The Falcor rendering framework, 8 2022. <https://github.com/NVIDIAGameWorks/Falcor>. URL: <https://github.com/NVIDIAGameWorks/Falcor>. 9
- [KGPB05] KRIVÁNEK J., GAUTRON P., PATTANAIK S., BOUATOUCH K.: Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (2005), 550–561. doi:10/csf2sw. 3
- [KKLD23] KERBL B., KOPANAS G., LEIMKUEHLER T., DRETTAKIS G.: 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 42, 4 (July 2023), 139:1–139:14. doi:10/gskvgh. 4, 10
- [KRS*25] KHERADMAND S., REBAIN D., SHARMA G., SUN W., TSENG J., ISACK H., KAR A., TAGLIASACCHI A., YI K. M.: 3d gaussian splatting as markov chain monte carlo, 2025. URL: <https://arxiv.org/abs/2404.09591>, arXiv:2404.09591. 4
- [LGL*25] LIANG R., GOJCIC Z., LING H., MUNKBERG J., HASSELGREN J., LIN C.-H., GAO J., KELLER A., VIJAYKUMAR N., FIDLER S., ET AL.: Diffusion renderer: Neural inverse and forward rendering with video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (2025), pp. 26069–26080. 1
- [LKB*22] LIN D., KETTUNEN M., BITTERLI B., PANTALEONI J., YUKSEL C., WYMAN C.: Generalized resampled importance sampling: Foundations of ReSTIR. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022), 75:1–75:23. doi:10/gqjn7b. 2, 3
- [LSC*25] LIU R., SUN D., CHEN M., WANG Y., FENG A.: Deformable beta splatting. In *ACM SIGGRAPH Conference Papers* (New York, NY, USA, 2025), SIGGRAPH Conference Papers '25, Association for Computing Machinery. URL: <https://doi.org/10.1145/3721238.3730716>. doi:10.1145/3721238.3730716. 4
- [LZT*08] LEHTINEN J., ZWICKER M., TURQUIN E., KONTKANEN J., DURAND F., SILLION F. X., AILA T.: A meshless hierarchical representation for light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 27, 3 (Aug. 2008), 37:1–37:9. doi:10/cbpbvx. 2
- [MGW*25] MEYER L., GRUEN J., WEIHERER M., EGGER B., STAMMINGER M.: Multi-spectral gaussian splatting with neural color representation. *ArXiv* (May 2025). URL: https://meyer1s.github.io/ms_splatting. 4
- [MMNL17] MCGUIRE M., MARA M., NOWROUZEZAHRAI D., LUEBKE D.: Real-time global illumination using precomputed light field probes. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (San Francisco, California, Feb. 2017), ACM Press, pp. 1–11. doi:10/ggd8nm. 1
- [MRNK21] MÜLLER T., ROUSSELLE F., NOVÁK J., KELLER A.: Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 40, 4 (July 2021), 36:1–36:16. doi:10/gmd76w. 2, 3, 9, 10
- [MZL*25] MO S., ZHENG C., LIN Z., XI D., YE Q., WANG R., BAO H., HUO Y.: Dual-band feature fusion for neural global illumination with multi-frequency reflections. *SIGGRAPH Conference Papers '25*. doi:10.1145/3721238.3730733. 3
- [NRH03] NG R., RAMAMOORTHY R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 22, 3 (July 2003), 376–381. doi:10/fmz579. 2
- [NVI] NVIDIA: Nvidia real-time denoising (nrd) library. URL: <https://github.com/NVIDIA-RTX/NRD>. 9
- [OLK*21] OUYANG Y., LIU S., KETTUNEN M., PHARR M., PANTALEONI J.: ReSTIR GI: Path resampling for real-time path tracing. *Computer Graphics Forum* 40, 8 (2021), 17–29. doi:10/gqwm dx. 2, 3
- [PJH23] PHARR M., JAKOB W., HUMPHREYS G.: *Physically based rendering: From theory to implementation*. MIT Press, 2023. 2
- [PMGD21] PHILIP J., MORGENTHAUER S., GHARBI M., DRETTAKIS G.: Free-viewpoint indoor neural relighting from multi-view stereo. *ACM Transactions on Graphics (TOG)* 40, 5 (2021), 1–18. 1
- [RBRD22] RAINER G., BOUSSEAU A., RITSCHER T., DRETTAKIS G.: Neural Precomputed Radiance Transfer. *Computer Graphics Forum (Proceedings of Eurographics)* 41, 2 (2022), 365–378. doi:10/gp8g7d. 3
- [RHL20] RUPPERT L., HERHOLZ S., LENSCH H. P. A.: Robust fitting of parallax-aware mixtures for path guiding. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). doi:10/gg8xc6. 2, 4
- [RLP*20] RODRIGUEZ S., LEIMKUEHLER T., PRAKASH S., WYMAN C., SHIRLEY P., DRETTAKIS G.: Glossy probe reprojection for interactive global illumination. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 39, 6 (Nov. 2020), 237:1–237:16. doi:10/gsmm87. 1, 2
- [RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 32, 4 (July 2013), 130:1–130:12. doi:10/gbdf7z. 3
- [RXL*23] RAGHAVAN N., XIAO Y., LIN K.-E., SUN T., BI S., XU Z., LI T.-M., RAMAMOORTHY R.: Neural free-viewpoint relighting for glossy indirect illumination. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 42, 4 (2023). doi:10/gsk2q3. 3
- [SDR*24] SU R., DONG H., REN J., JIN H., CHEN Y., WANG G., LI S.: Dynamic neural radiosity with multi-grid decomposition. In *ACM SIGGRAPH Asia Conference Papers* (New York, NY, USA, 2024), SIGGRAPH Asia Conference Papers '24, Association for Computing Machinery. URL: <https://doi.org/10.1145/3721238.3730716>. doi:10.1145/3721238.3730716. 4

- org/10.1145/3680528.3687685, doi:10.1145/3680528.3687685. 2, 10
- [SHJD22] SCHÜSSLER V., HANIKA J., JUNG A., DACHSBACHER C.: Path guiding with vertex triplet distributions. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 41, 4 (2022). doi:10.1111/cgfr.14582. 4
- [SJJ12] SCHWARZHAUPT J., JENSEN H. W., JAROSZ W.: Practical Hessian-based error control for irradiance caching. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (Nov. 2012), 1. doi:10/gbb6n4. 2, 3
- [SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 21, 3 (2002), 527–536. doi:10/fqg3kn. 2
- [SL17] SILVENNAINEN A., LEHTINEN J.: Real-time global illumination by precomputed local reconstruction from sparse radiance probes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 36, 6 (Nov. 2017), 230:1–230:13. doi:10/gcqbvn. 1
- [SMLD] SHAH I., MEULEMAN A., LANVIN A., DRETTAKIS G.: Graphdeco viewer. URL: <https://github.com/graphdeco-inria/graphdecoviewer>. 9
- [SNRS12] SCHERZER D., NGUYEN C. H., RITSCHER T., SEIDEL H.-P.: Pre-convolved radiance caching. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 31, 4 (June 2012), 1391–1397. doi:10/gfzwbr. 3
- [SSS*20] SEYB D., SLOAN P.-P., SILVENNAINEN A., IWANICKI M., JAROSZ W.: The design and evolution of the UberBake light baking system. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). doi:10/gg8xc9. 2
- [ST91] SAITO T., TAKAHASHI T.: Nc machining with g-buffer method. *ACM Siggraph Computer Graphics* 25, 4 (1991), 207–216. 3
- [TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 25, 3 (2006), 967–976. doi:10/dwsd3h. 2
- [TTM*22] TEWARI A., THIES J., MILDENHALL B., SRINIVASAN P., TRETSCHK E., YIFAN W., LASSNER C., SITZMANN V., MARTIN-BRUALLA R., LOMBARDI S., SIMON T., THEOBALT C., NIESSNER M., BARRON J. T., WETZSTEIN G., ZOLLHÖFER M., GOLYANIK V.: Advances in Neural Rendering. *Computer Graphics Forum (Proceedings of Eurographics)* 41, 2 (2022), 703–735. doi:10/gsmm7h. 2
- [VKŠ*14] VORBA J., KARLÍK O., ŠIK M., RITSCHER T., KŘIVÁNEK J.: On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33, 4 (Aug. 2014), 101:1–101:11. doi:10/f6c2cp. 4
- [WH92] WARD G. J., HECKBERT P. S.: Irradiance gradients. In *CE_EGWR93* (Bristol, UK, May 1992), Chalmers A., Paddon D., Sil-lion F. X., (Eds.), Consolidation Express Bristol, pp. 85–98. 2
- [WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 28, 5 (Dec. 2009), 133:1–133:10. doi:10/dfr74x. 6
- [WTC*25] WANG L., TRAN D. M., CUI R., TG T., DAHL A. B., BIGDELI S. A., FRISVAD J. R., CHANDRAKER M.: Materialist: Physically based editing using single-image inverse rendering. *arXiv preprint arXiv:2501.03717* (2025). 6
- [WYZ*24] WU T., YUAN Y.-J., ZHANG L.-X., YANG J., CAO Y.-P., YAN L.-Q., GAO L.: Recent advances in 3d gaussian splatting. *Computational Visual Media* 10, 4 (2024), 613–642. 2
- [XSD*13] XU K., SUN W.-L., DONG Z., ZHAO D.-Y., WU R.-D., HU S.-M.: Anisotropic spherical Gaussians. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 32, 6 (Nov. 2013), 209:1–209:11. doi:10/gbd5rd. 3, 4, 5, 6, 16
- [XZW*22] XU Z., ZENG Z., WU L., WANG L., YAN L.-Q.: Lightweight neural basis functions for all-frequency shading. In *ACM SIGGRAPH Asia Conference Papers* (New York, NY, USA, Nov. 2022), Association for Computing Machinery, pp. 1–9. doi:10/grqvdr. 3
- [YGS*24] YANG Z., GAO X., SUN Y.-T., HUANG Y.-H., LYU X., ZHOU W., JIAO S., QI X., JIN X.: Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. In *Advances in Neural Information Processing Systems* (2024), Globerson A., Mackey L., Belgrave D., Fan A., Paquet U., Tomczak J., Zhang C., (Eds.), vol. 37, Curran Associates, Inc., pp. 61192–61216. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/708e0d691a22212e1e373dc8779cbe53-Paper-Conference.pdf, doi:10.52202/079017-1956. 16
- [ZHH*24] ZHENG C., HUO Y., HUANG H., SHENG H., HUANG J., TANG R., ZHU H., WANG R., BAO H.: Neural global illumination via superposed deformable feature fields. SA '24. doi:10.1145/3680528.3687680. 3
- [ZHM*23] ZHENG C., HUO Y., MO S., ZHONG Z., WU Z., HUA W., WANG R., BAO H.: Nelt: Object-oriented neural light transfer. *ACM Trans. Graph.* 42, 5 (Aug. 2023). URL: <https://doi.org/10.1145/3596491>, doi:10.1145/3596491. 3

Appendix A: Anisotropic Spherical Gaussians

Anisotropic Spherical Gaussians (ASG) [XSD*13] are spherical radial basis functions that can model a spherical function. ASG have demonstrated their efficiency in representing view-dependent illumination [XSD*13, CDAS20, YGS*24], notably thanks to their properties allowing fast integration. An ASG $G_a(\omega; [\mathbf{x}, \mathbf{y}, \mathbf{z}], [s_x, s_y])$ is defined as follows:

$$G_a(\omega; [\mathbf{x}, \mathbf{y}, \mathbf{z}], [s_x, s_y]) = (\omega \cdot \mathbf{z}) e^{-\left(\frac{\omega \cdot \mathbf{x}}{s_x}\right)^2 - \left(\frac{\omega \cdot \mathbf{y}}{s_y}\right)^2}, \quad (21)$$

where $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ are orthogonal vectors (tangent, bi-tangent and mean) forming the basis of the ASG and $[s_x, s_y]$ is the spread of the ASG.

An ASG can also be written in a more general form using a covariance matrix Σ_a as follows:

$$G_a(\omega; \Sigma_a) = (\omega \cdot \mathbf{z}) e^{-\omega^T \Sigma_a^{-1} \omega}, \quad (22)$$

where Σ_a is defined as:

$$\Sigma_a = [\mathbf{x} \ \mathbf{y} \ \mathbf{z}] \begin{bmatrix} s_x^2 & 0 & 0 \\ 0 & s_y^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} [\mathbf{x} \ \mathbf{y} \ \mathbf{z}]^T. \quad (23)$$

Convolution with Spherical Gaussian The convolution of an ASG with a Spherical Gaussian (SG) can be approximated using a closed-form solution. An SG $G_{iso}(\omega; \mu, \lambda)$ is defined as:

$$G_{iso}(\omega; \mu, \lambda) = e^{\lambda(\mu \cdot \omega - 1)}, \quad (24)$$

where μ is the mean direction and λ is the concentration parameter of the SG. The convolution of an ASG $G_a(\omega; \Sigma_a)$ with an SG $G_{iso}(\omega; \mu, \lambda)$ is approximately given by [XSD*13]:

$$\hat{G}(\mu, \lambda, \Sigma_a) = \int_{\Omega} G_a(\omega; \Sigma_a) G_{iso}(\omega; \mu, \lambda) d\omega \approx \frac{\pi s_x s_y}{\sqrt{(1+\lambda s_x^2)(1+\lambda s_y^2)}} G_a\left(\mu; [\mathbf{x}, \mathbf{y}, \mathbf{z}], \left[\sqrt{\frac{1}{\lambda} + s_x^2}, \sqrt{\frac{1}{\lambda} + s_y^2}\right]\right). \quad (25)$$

This is useful to compute the LTE efficiently (see Section 4.3).