Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields

Zander Majercik
NVIDIA

Jean-Philippe Guertin
Université de Montréal

Derek Nowrouzezahrai
McGill University

Morgan McGuire
NVIDIA and McGill University

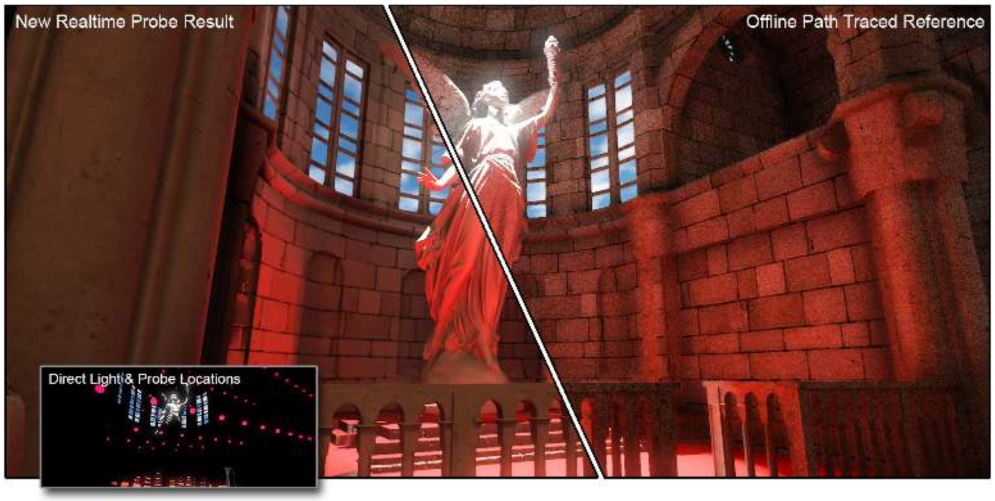

New Realtime Probe Result
Offline Path Traced Reference
Direct Light & Probe Locations

Figure 1. Combined with state of the art glossy ray tracing and deferred direct shading, our method (left) generates full global illumination in dynamic scenes that are visually comparable to offline path-traced results (right) but several orders of magnitude faster: 6 ms/frame, versus 1 min/frame in this scene (on GeForce RTX 2080 Ti at 1920×1080). Insets isolate the direct lighting contribution and visualize the probe locations.

Abstract

We show how to compute global illumination efficiently in scenes with dynamic objects and lighting. We extend classic irradiance probes to a compact encoding of the full irradiance field in a scene. First, we compute the *dynamic* irradiance field using an efficient GPU memory layout, geometric ray tracing, and appropriate sampling rates without down-sampling or filtering prohibitively large spherical textures. Second, we devise a robust filtered irradiance query, using a novel visibility-aware moment-based interpolant. We experimentally validate performance and accuracy tradeoffs and show that our method of *dynamic diffuse global illumination* (DDGI) robustly lights scenes of varying geometric and radiometric complexity (Figure 1). For completeness, we demonstrate results with a state-of-the-art glossy ray-tracing term for sampling the full dynamic light field and include reference GLSL code.

1. Introduction

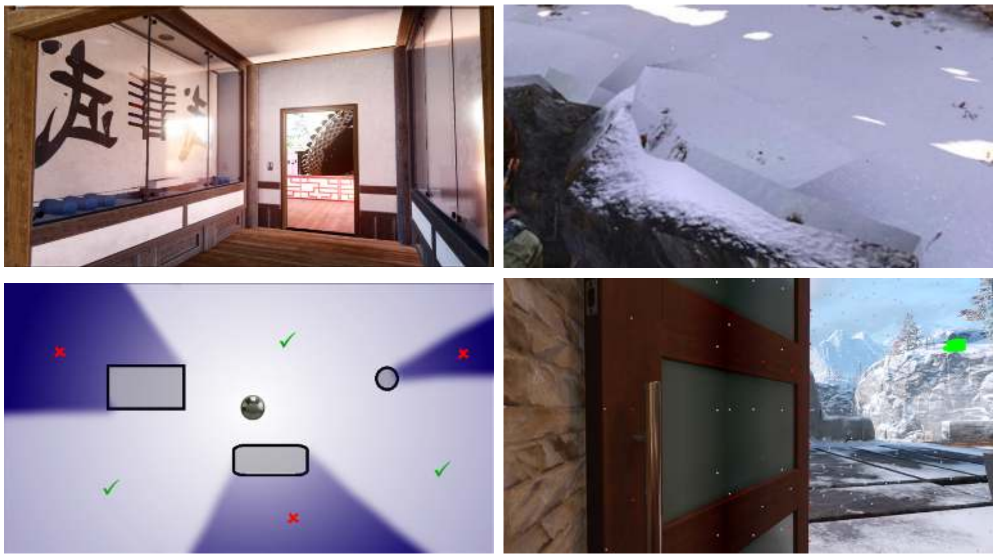
Probe-based global illumination. Synthesizing images with accurate global-illumination (GI) effects contributes significantly to the believability of computer-generated imagery. Accurately solving physics-based GI formulations is a longstanding area of research, and doing so with offline numerical solvers remains a time-consuming cost. In an interactive rendering context, a significant amount of work on generating *convincing* real-time GI effects has led to many different solutions, each with specific tradeoffs among accuracy, flexibility, and performance.

Recent work on *light field probes* strikes one such tradeoff. That representation encodes the static local light field of a scene using a specialized encoding of precomputed probes placed statically in a scene [McGuire et al. 2017b]. The probe representation has many benefits, including the ability to perform efficient and accurate world-space (filtered) ray tracing for glossy and near-specular indirect transport, as well as supporting irradiance probe-like queries that are robust to light-leaking artifacts for smoother indirect diffuse illumination.

Given the query and sampling operations exposed by the light field probe representation, many shading algorithms can be implemented using this representation as a basis, often resulting in high-fidelity images generated at high performance rates. The main limitations of standard light field probes lie in their precomputed nature and the manner in which they sample lighting in the scene. Precomputing the probe data can be costly, and therefore only fixed lighting and geometric conditions are handled. Moreover, the irradiance spatial interpolation and prefiltered glossy sampling schemes can lead to aliasing and light-leaking in the diffuse and specular indirect illumination.

Real-time GI. Unlike offline rendering, global-illumination solutions for real-time applications, such as video games, currently rely fundamentally on lighting data that can be rapidly read from spatial-angular data structures, and it is usually precomputed or limited to slow updates from static geometry for dynamic lighting. Examples include lightmap representations, irradiance and radiance probes, and voxelized representations of the scene or lighting information. Each of these representations strikes a particular tradeoff between compactness, runtime flexibility, accuracy, and cost. In geometrically and/or radiometrically complex scenes, these methods all have well-documented undesirable artifacts that manifest as a result of undersampling and reconstruction. The most significant of these artifacts is light- and shadow-leaking in areas of complex visibility. Many recent GDC and SIGGRAPH talks isolate and discuss these issues. We highlight two representative ones in Figure 2.

Typically, heuristic workarounds are used. These vary with the art and technical constraints of a particular production. In cases where only static geometry and/or lighting are treated, a largely manual post-processing intervention is often done. Of course, such an approach scales poorly with scene complexity and still requires sig-



Light leaks due to undersampling in classic irradiance probes [Hooker 2016]

Light and shadow leaks along lightmap seams (top) [Hooker 2016] and in voxels (bottom) [Iwanicki 2013]

Figure 2. Previous interactive GI methods suffer from artifacts that often necessitate heuristic solutions, typically based on art-direction or technical constraints. Visual artifacts in these methods can manifest themselves in various forms: (from left to right) light leaking, lightmap seams, visibility/occlusion undersampling, and inter-voxel visibility mismatches.

nificant offline precomputation. This problem is further exaggerated in the context of dynamic environments, where the scene geometry and lighting can change at runtime, precluding manual intervention. As such, there is a great practical need for automatic caching solutions that are robust to dynamic scenes and do not sacrifice the high-performance nature of pre-cached global-illumination solutions.

The core problem underlying prior techniques is not inherent in the representations, which are often efficient and well-designed for capturing either radiance (light energy along a ray used for the glossy portion of shading) or irradiance (cosine-weighted integral of radiance necessary for the diffuse portion of shading). Rather, the problem is that the techniques lack visibility information and thus cannot encode the full light field or *irradiance field* (irradiance taking occlusion into account).

This paper describes a method for extending classic irradiance probes to a representation of the full irradiance field, shows how to efficiently update that representation at runtime, and then evaluates the performance and quality of that method. The academic term for the quantity computed is the dynamic indirect irradiance field; we call the new probe technique *dynamic diffuse global illumination* (DDGI) in keeping with game-industry jargon.

The specific contributions of this work are:

- Extension of irradiance probes with accurate and dynamic occlusion information by an incremental scheme that leverages a compact, GPU-tailored data layout and compute schedule for converged "infinite"-bounce diffuse global illumination,
- An algorithm for ray tracing irradiance probes independent of the primary visibility resolution and frame rate, avoiding the cost of denoising or prefiltering prohibitively high-resolution spherical textures,
- A spatial interpolation, occlusion, and filtering scheme more robust to irradiance queries in scenes with temporally-varying geometry and lighting,
- Evaluation of a system for producing results nearly identical to (offline) path tracing in many cases, combining dynamically-updated occlusion-aware irradiance with GPU ray-traced glossy and specular reflections, reducing aliasing artifacts in these indirect contributions, and
- Open source reference shaders for implementing DDGI, taken directly from and compatible with the open source G3D Innovation Engine [McGuire et al. 2017a].

2. Related Work

Works on interactive global illumination span several decades. We review the areas most relevant to our work.

Image-based lighting. Image-based lighting methods form the basis of most interactive pre-caching lighting solutions in modern video games [Martin and Einarsson 2010; Ritschel et al. 2009; McAuley 2012; Hooker 2016]. Here, a common workflow involves placing light probes (of various types) densely inside the volume of a scene, each of which encodes some form of a spherical (ir)radiance map. Prefiltered versions of these maps can also be stored to accelerate diffuse and glossy runtime shading queries.

One interesting variant of traditional light probes allows digital artists to manually place box or sphere proxies in a scene, and these proxies are used to warp probe queries at runtime in a manner that better approximates spatially-localized reflection variations [Lagarde and Zanuttini 2012]. Similarly, manually-placed convex proxy geometry sets are also used to bound blending weights when querying and interpolating between many light probes at runtime, in order to reduce light leaking artifacts—one of the predominant artifacts of such probe-based methods.

These probe- and image-based lighting techniques are ubiquitous in modern offline and real-time rendering, and we refer interested readers to a comprehensive survey [Reinhard et al. 2006].

While production-quality image-based lighting systems generate convincing illumination effects, practitioners agree that eliminating manual probe and proxy placement remains an important open problem in production [Hooker 2016]. Currently, without manual adjustment, it is impossible to automatically avoid probe placements that lead to light and dark (i.e., shadow) leaks or displaced reflection artifacts. To avoid these issues, some engines rely instead on screen-space ray tracing [Valient 2013] for pixel-accurate reflections. These methods, however, fail when a reflected object is not visible from the camera's point of view, leading to inconsistent lighting and view-dependent (and so temporally unstable) reflection effects.

Light field probes [McGuire et al. 2017b] automatically resolve these issues in scenes with static geometry and lighting by encoding additional information about the scene geometry into spherical probes. A solution for dynamic lighting is presented in [Silvennoinen and Lehtinen 2017], but this solution only supports coarse dynamic occluders and requires complex probe placement based on static geometry. We inherit the advantages of the representation in [McGuire et al. 2017b], which we extend fundamentally to treat *dynamic* geometry and lighting variations at runtime (Section 5). No manual placement is necessary and a naïve uniform-grid probe placement results in artifact-free renderings. Reflections appear (consistently) where they should due, in part, to an accurate world-space ray-tracing algorithm (Section 4.2). *Visibility-aware* blending weights allow for automatic filtered radiance sampling (Section 5) without the need for manually placed proxy geometry. As such, light field probes can be leveraged in both the context of traditional (prefiltered) radiance lookups, as well as in shader-enabled world-space ray tracing.

Interactive ray tracing and shading. Many recent interactive rendering approaches treat the problem of resolving point-to-point visibility queries, shaping modern solutions used in practice today. Ritschel et al.'s [2008] imperfect shadow maps encode a sparse, low-resolution representation of point-to-point visibility in a scene, which is then used to compute accurate secondary diffuse and glossy reflections using virtual point lights (generated, for example, with a ray tracer). Our work is motivated by another such solution: voxel cone tracing [Crassin et al.]. At a high level, one can interpret our ray-tracing technique (Section 4.2) as tracing rays against a spherical voxelized representation of the scene (i.e., as opposed to the octree representation constructed for traditional voxel cone tracing). Two important differences that contribute to many of the practical advantages of our representation are: first, that we *explicitly* encode geometric scene information (i.e., radial depth and depth-squared) instead of relying on the *implicit* octree structure to resolve local and global visibility details; and, second, that neither our spatial parameterization nor our filtering rely on scene geometry. This allows us to completely sidestep the light- (and dark-) leaking artifacts present in traditional voxel cone tracing. Finally, we are able to resolve *centimeter-scale* geometry at about the same cost (in space and time) as a voxel cone tracer that operates at *meter-scale*.

Representation. We use Cigolle et al.'s [2014] octahedral mapping from the sphere to the unit square to store and query our spherical distributions. This parameterization has slightly less distortion and provides simpler border management than, for example, cube maps. Since we target *true world-space* ray tracing in a pixel shader, and not just screen-space ray tracing, our technique can be seen as a generalization of many previous real-time environment map Monte Carlo integration methods [Stachowiak and Uludag 2015; Wyman 2005; Toth et al. 2015; Jendersie et al. 2016] .

We are also motivated by the preliminary investigations of [Evangelakos 2015] and [Donow 2016] that validate the accuracy of single-probe ray tracing and the feasibility of multi-probe traversal. Specifically, a single probe can perfectly sample the geometry of a region with *star-shape topology*, and thus ray tracing with a single probe in these regions will incur no visibility error (outside of errors due to probe-directional discretization).

3. Dynamic Diffuse Global Illumination Probes: Overview

As in [McGuire et al. 2017b], we encode geometric and radiometric scene data into spherical distributions at discrete probe locations. We combine efficient GPU ray tracing to enable accurate shader-based world-space ray tracing (using either a probe-based marching approach, or native GPU ray-tracing APIs), with filtered irradiance queries to compute diffuse, glossy, and specular indirect illumination at real-time rates.

Specifically, we encode the spherical diffuse irradiance (in `GL_R11G11B10F` format at 8×8 octahedral resolution), spherical distance and squared distances to the nearest geometry (both in `GL_RG16F` format at 16×16 octahedral resolution). We pack each of these square probe textures into a single 2D texture atlas with duplicated gutter regions to allow bilinear interpolation without any boundary artifacts (see Figure 3).

Instead of precomputing the probe data once at scene initialization, we dynamically update the probes to capture variations due to dynamic geometry and lighting. This allows us to enable truly dynamic high-fidelity global illumination. Our method retains the high performance of [McGuire et al. 2017b]; Figure 4 illustrates our ability to compute fully converged multi-bounce global illumination.

At each frame we are able to efficiently blend updated ray-traced illumination into our probe atlas in addition to interpolating probe depth information to adapt to changes in scene geometry. In a forward or deferred shading render pass, probes can

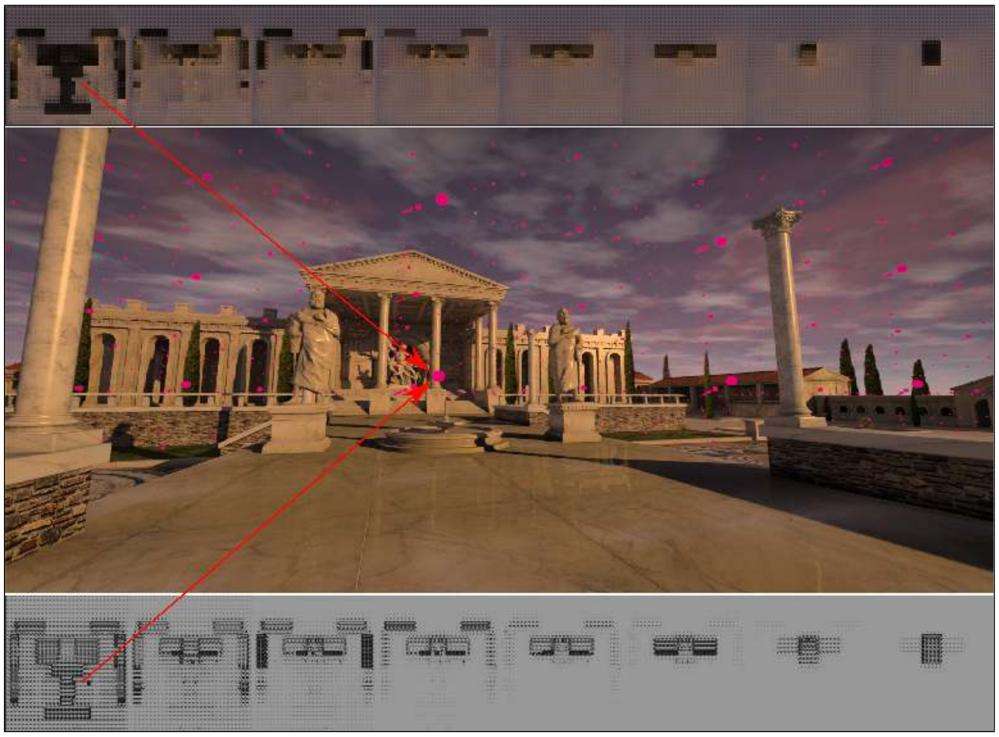


Figure 3. Spherical irradiance and depth textures. We encode spherical data in an octahedral parameterization, packing all the probes in an atlas. One-pixel texture gutter/border ensures correct bilinear interpolation, and additional padding aligns probes on 4×4 write boundaries.

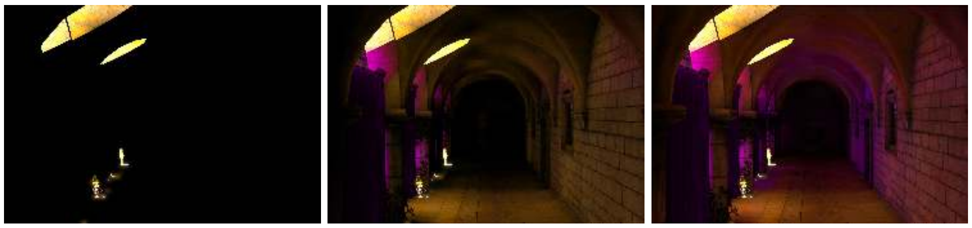


Figure 4. Left to right: direct illumination only, direct illumination with *one bounce* of indirect diffuse illumination (computed with spherical irradiance updated by our dynamic filtered ray-casting approach), and the fully converged *multi-bounce* global illumination that iteratively incorporates bounced lighting computed across probes.

effectively be treated as indirect lighting buffers analogous to standard precomputed environment maps.

We detail our method for updating dynamic diffuse global illumination probe distributions in Section 4 before discussing how to use probes at runtime to efficiently compute dynamic global illuminations in Section 5.

4. Updating Dynamic Diffuse Global Illumination Probes

We place our probes as in [McGuire et al. 2017b] before incrementally updating the probe content (Section 4.1). At every frame, we follow a multi-step process to update probe information in order to incorporate the effects of dynamic geometry and lighting:

1. *generate* and *trace* n primary rays from each of the m *active probes* in a scene, storing geometry for (up to) $n \times m$ surface hits in a G-buffer-like structure of *surfels* with explicit position and normals (Section 4.2);

2. *shade* the surfel buffer with direct and indirect illumination (Section 4.3), with the same routine used to shade final image pixels, i.e., those directly visible from the camera (Section 5); and

3. *update* the texels in the octahedral representations of the m active probes by blending in the updated shading, distance, and square-distance results for each of the n intersected surfels (Section 4.4).

We discuss several methods to select *active probes* to update in Section 4.2; however we employ a conservative selection approach and set *all* the probes in a scene to active. As such, our rendering-performance metrics are a conservative upper bound on the expected performance of our algorithm.

Shading the probe-intersected surfels (Section 4.3) relies on lighting and probe data from the previous frame, which serves two purposes: first, this allows us to amortize the cost of computing multiple indirect bounces over several frames; second, when combined with our blending approach (Section 4.4), this allows a smooth transition between sharp geometric and radiometric discontinuities (over time). A negative side effect, of course, is that indirect illumination can sometimes appear to "flow" in and out of areas with dramatic visibility changes, due to the latency in the indirect illumination update. Given the relative smoothness of indirect illumination, compared to direct illumination (which we update precisely at every frame), we follow the guidelines and observations of prior work that indicate that these specific artifacts remain an acceptable perceptual tradeoff for viewers [2015].

4.1. Probe Grid Placement

We place probes in the volume of the scene at the vertices of an axis-uniform 3D grid. We use a power-of-two resolution per axis, simplifying probe indexing to simple bitwise operations. We can scale per-axis grid cell spacing independently for scenes that require different spatial discretization per axis.

We note that visibility-aware probe selection and sampling affords us a certain latitude when placing probes: probes that fall inside walls or other geometry will be ignored by visibility-query metrics [2017b]. Also, other than simplifying probe indexing, no aspect of the probe generation or shading *requires* a uniform grid placement; indeed, probes can be placed according to schemes used in other probe-based algorithms, including tetrahedral grids.

Every point in space is associated with a *cage* of vertices corresponding to the eight vertices of the grid cell that contains the point. We recommend using a grid resolution and scale that results in at least one full cage of vertices in each room-like space. This is needed to ensure a sufficient sampling of local-illumination variation inside each separated/distinct space in a scene. For human-scale scenes, we found a spacing of one to two meters sufficient, however we illustrate results with a variety of grid spacings.

4.2. Generating and Tracing Probe-Update Rays

We update texels in probes to account for dynamic geometry and lighting variation, blending in their results over time in order to smoothly account for the effects of these dynamic changes on the final rendered result.

At each of the m active probes, we uniformly sample n spherical directions according to a stochastically-rotated Fibonacci spiral pattern, similar to [McGuire et al. 2017b]. We then spawn n rays with these directions and a (shared) origin of the probe center. We lay out the rays across the m probes in a thread-coherent fashion, casting all of them in one batch.

Technical notes. While Vulkan Ray Tracing and DirectX Ray Tracing APIs permit ray dispatching from primary shaders, we benchmarked our ray batching and observed that it minimizes register pressure and facilitates debugging through inspection of intermediate shading results.

We experimented with several probe and ray sub-sampling schemes: e.g., updating only a subset of the probes in a scene, such as those within a certain radius of the camera; or varying the ray count based on distance to the camera. While these adaptive schemes led to some expected performance improvements, they introduced many additional scene-dependent user parameters and additional bookkeeping.

We instead opted for simplicity in our final results; our reference implementation updates every probe at every frame (i.e., sets every scene probe as *active* during probe updates), as well as dispatches the same number of rays per probe. More complex usage scenarios, such as large open-world environments, could benefit from a probe-streaming scheme as well as from probe LODs operating at several grid scales (see Section 7.1 for more discussion).

4.3. Secondary Surfel Shading

We employ a unified shading model for both probe updates and final rendering. Specifically, we compute global illumination in two contexts at runtime: first, when updating the shading on the $m \times n$ probe-sampled surfels, and, finally, when shading pixels from the camera for the final output image. Both of these contexts use the same shading routine, composed of a direct illumination pass that uses state-of-the-art interactive practices and an indirect lighting pass that leverages our probe data. We discuss the details of the shading routines in Section 5, focusing on the subtle differences in its application during probe surfel updates, below.

To abstract over the differences in how shading queries are made during probe update and final rendering, our shading routines expect a shading position, normal, and viewing direction as input (Section 5). For probe-traced surfel shading updates, we pass the intersected surfel locations and normals, as well as the direction from the surfel to the probe center, as input to the shading routine.

4.4. Probe Surfel Updates

After surfel shading, each of the $m \times n$ surfel points will have an updated shading value, and the sampled surfel distances (and squared distance) are also updated relative to their associated probe centers.

We update the probe texels (associated to each surfel) by alpha-blending in the new shading results at a rate of $1 - \alpha$, where α is a *hysteresis* parameter that controls the rate at which updated shading overrides shading results from previous frames (Equation (1)). We set α between 0.85 and 0.98 for all of our results.

$$\begin{aligned} &\text{newIrradiance}[\text{texelDir}] = \text{lerp}(\text{oldIrradiance}[\text{texelDir}],\\ &\sum_{\text{ProbeRays}} (\max(0, \text{texelDir} \cdot \text{rayDir}) * \text{rayRadiance}), \text{hysteresis}) \end{aligned} \tag{1}$$

We directly compute the filtered irradiance using a moment-based filtered shadow query, allowing us to avoid brute-force prefiltering of a (higher-resolution) incident radiance map. This smooth incident irradiance field will be used to compute diffuse indirect illumination (Section 5.2), and we can optionally maintain a higher-frequency shading map for glossy and specular indirect shading.

Technical Notes. We purposefully lay out our data and update computation in order to promote coherence in execution: probe texels operate in (near) lockstep to their neighbors, often operating on the same ray, blending in its result. This yields not only coherent memory fetches on the GPU, but also coherent compute. We update irradiance and depth texels against a cosine lobe distribution, a necessary step for correct irradiance representation [Akenine-Möller et al. 2018]. In the case of the depth and depth-squared buffers (as with [McGuire et al. 2017b]), we employ an additional *depth sharpening*, warping them according to a cosine-power lobe distribution. We do not update texels weighted below a threshold (we used 0.001) in the cosine-power lobe distribution.

Note that, while the updated spherical irradiance distributions will be used to shade *view-independent* diffuse reflectance effects, they are updated to correctly account for any glossy/mirror *view-dependent* shading due to dynamic geometry and lighting in the environment. For a in-depth discussion of irradiance and computing irradiance using light probes, we refer readers to [Akenine-Möller et al. 2018] (pg.268, 490).

5. Shading with Dynamic Diffuse Global Illumination Probes

We compute multi-bounce global illumination effects with diffuse, glossy, and specular transport. We will separately discuss the shading procedure for each of these transport components. We motivate and outline the novel contributions of our technique below. Instead of explicitly detailing every algorithmic detail and/or parameter setting of our implementation, we provide a full source reference implementation as a supplemental reference for the exact technical details.

5.1. Direct Illumination

We compute direct illumination from point and directional light sources using a deferred renderer with variance shadow mapping [Thaler 2011; Donnelly and Lauritzen 2006].

We can also handle direct illumination from extended area light sources using our indirect illumination pipeline: all one-bounce indirect lighting contributions (Section 5.2) compute one bounce of lighting seeded by the direct illumination in a scene. Multiple bounces of indirect illumination are instead seeded by the previous bounce of indirect lighting in the scene (Section 5.3). With this in mind, we can compute direct illumination from area lighting by seeding our indirect illumination shading routine with the area lighting emission profile in the scene (i.e., instead of the direct-illumination profile).

The last row of Figure 5 gives a sense of this area lighting setup: apart from the “direct light” on the window geometry, the remainder of the shading in the bathroom scene is computed as an “indirect” contribution from the window light.

With this approach, we can avoid approximating direct illumination from area lights, instead relying on the robustness of our probe-based shading technique to compute smooth area shadows and reflections. We detail this method, below.

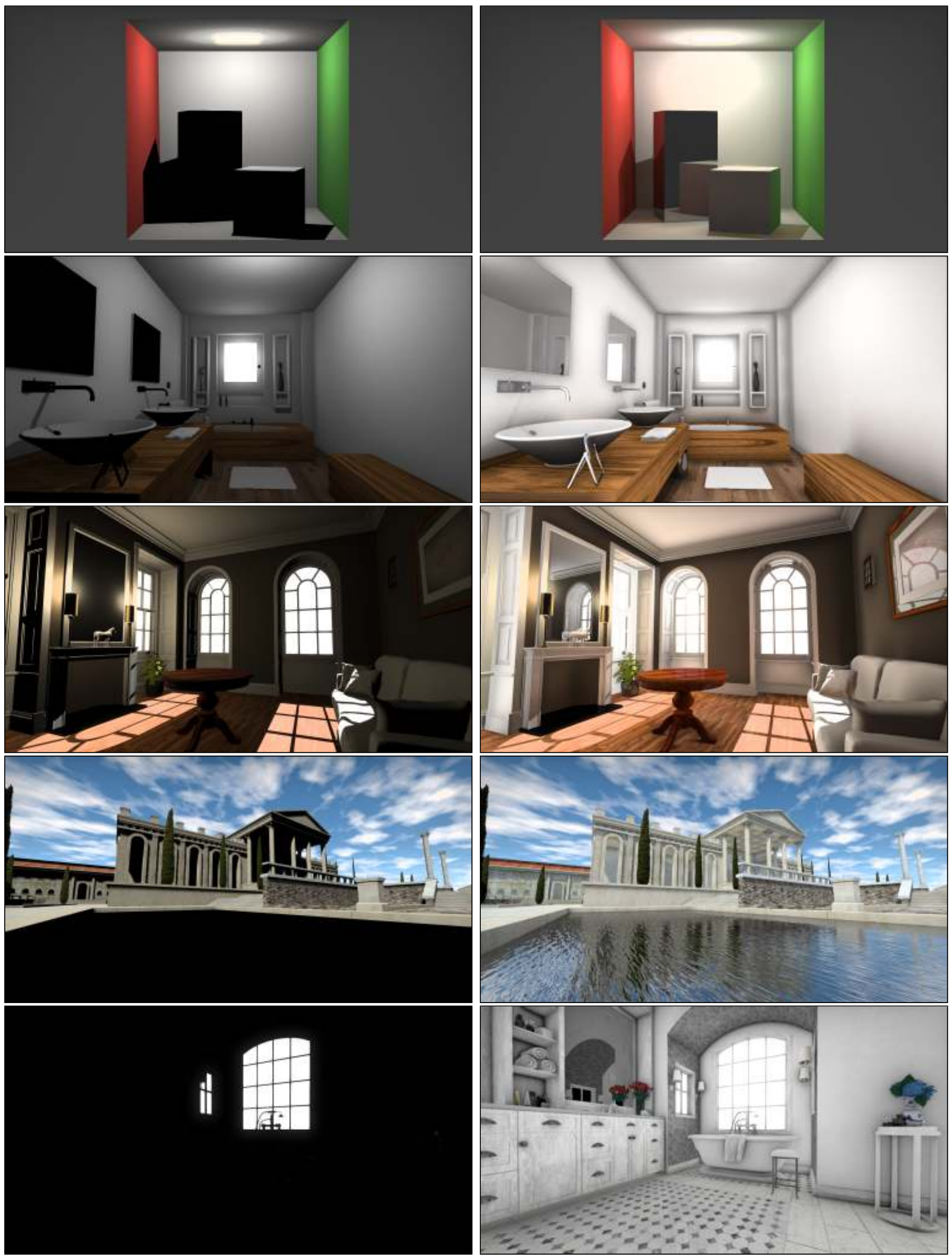


Figure 5. Direct illumination (left) versus full (diffuse, glossy & specular) global illumination (right) computed using dynamically-updated irradiance probes and ray-traced reflections.

5.2. Diffuse Indirect Illumination

We compute spherical incident irradiance distributions at each probe, and we extend the original visibility-aware probe-weighting scheme of [McGuire et al. 2017b] to

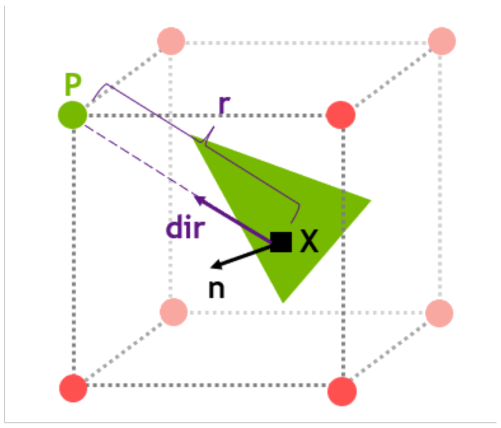


Figure 6. Shading of a surfel X. We sample each probe in the eight-probe cage using the surface normal n in world space. We backface-weight each probe P using dir, the direction from X to P. The mean distance stored for P is represented by r. To avoid sampling visibility near the visibility-function boundary (i.e., the surface), we offset from the world-space position at X based on the surface normal and the camera-view vector.

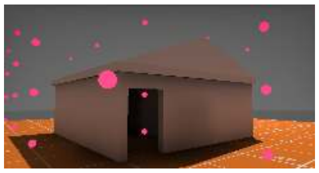
query diffuse irradiance from probes at shading points in the scene. We modulate incident irradiance by spatially-varying diffuse surface albedo in order to compute one bounce of indirect outgoing diffuse radiance [Akenine-Möller et al. 2018].

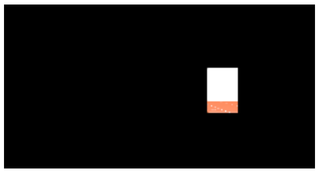
To compensate for the fact that the incident diffuse irradiance at a probe location does not account for local occlusion around a shading point, we (optionally) modulate outgoing diffuse reflection by a screen-space ambient occlusion variant [Shanmugam and Arikan 2007]. Note, however, that we do not include this local occlusion when computing surfel shading updates (Section 4.3): the impact of omitting this term on *secondary* lighting (i.e., computed as the diffuse, glossy, or specular reflection of the surfel shading) is significantly less than on the lighting of directly-visible surfaces.

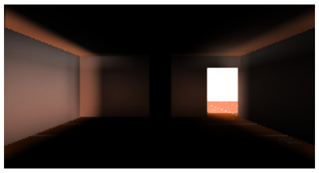
Our indirect diffuse interpolation and sampling technique differs from that of [McGuire et al. 2017b], incorporating ideas from the ray-tracing and shadow-mapping literature that are designed to increase robustness to dynamic geometry and lighting. Specifically, after computing the indices of the eight-probe cage that contains the shading point, we compute interpolation weights for each irradiance probe from its position and direction (relative to the shading point), as follows (see Figure 6):

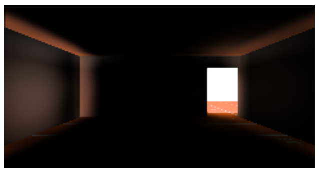
Figure 7 illustrates the visual impact that each of these weighting stages has on the final rendering, highlighting how each factor contributes to eliminating artifacts, starting from a traditional irradiance-probe rendering and progressing through each of our factors, above.

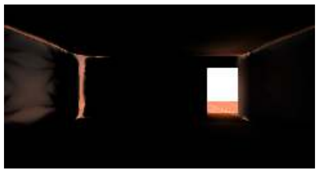
- we backface-cull probes that lie below the shading point's tangent plane, using a soft threshold that falls off smoothly as the dot product of the shading normal with the direction towards a probe approaches zero,

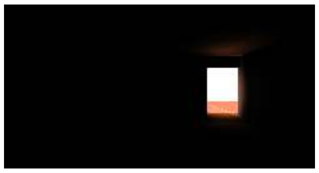

(a) House exterior

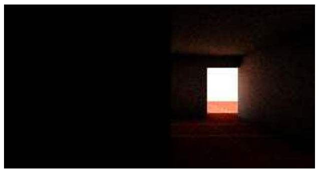

(b) Interior cutaway; direct light only


(c) + classic irradiance probes


(d) + new backface test


(e) + new visibility test


(f) + normal bias


(g) Path traced reference

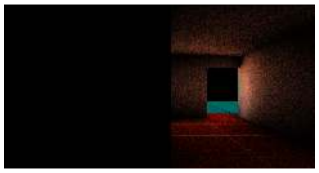

(h) Absolute error $\times$ 2

Figure 7. Irradiance interpolation and sampling in a closed room scene (a), where light enters only from a single door opening (b). Images (b) through (i) place the camera at the back corner of the room, illustrating how light-leaking artifacts from traditional irradiance probes (c) are progressively compensated for using the terms in our novel interpolant (d) – (f). We visualize a $2\times$ error image (h) between our final result (f) and the path-traced reference (g).

- we apply a perceptually-based weighting to account for the human visual system's sensitivity to (relatively) low-intensity lighting in otherwise dark regions (i.e., light leaks): we reduce the contribution of very low-irradiance values (i.e., less than 5% of the representable intensity range) according to a monotonically-decreasing curve profile,

- we apply mean- and variance-biased Chebyshev interpolants, as detailed in the variance shadow-mapping method, to our visibility queries (see Figure 7) in order to appropriately filter radiance queries,

- we offset the shading point according to a bias proportional to the shading normal and the direction to the probes: this improves the robustness of the visibility-based interpolation weights by moving away from potential shadowed-unshadowed discontinuities, and

- we then perform a standard trilinear interpolation based on the distance between the shading point and the probe centers, using the aforementioned weighting and biasing factors.

Each of these weighting terms are appropriately bound using conservative epsilon tests in order to avoid numerical issues when normalizing the weights, e.g., when per-probe weights approach zero.

Note that shading with standard irradiance probes results in significant light-leaking artifacts, as expected and similar to those highlighted in related work (see Figure 2), whereas our final renderings agree much more closely with path-traced ground truth. Any color banding artifacts in our results are due to conversion from HDR to LDR into the PDF-embedded PNG format; these artifacts are not present on display.

Technical Note. Previous work uses $128 \times 128 \times 6$ high-precision cube maps to store depth information, however our additional weighting criterion allow us to scale down to 16×16 medium-precision depth values without incurring any numerical issues.

5.3. Multiple Bounces of Global Illumination

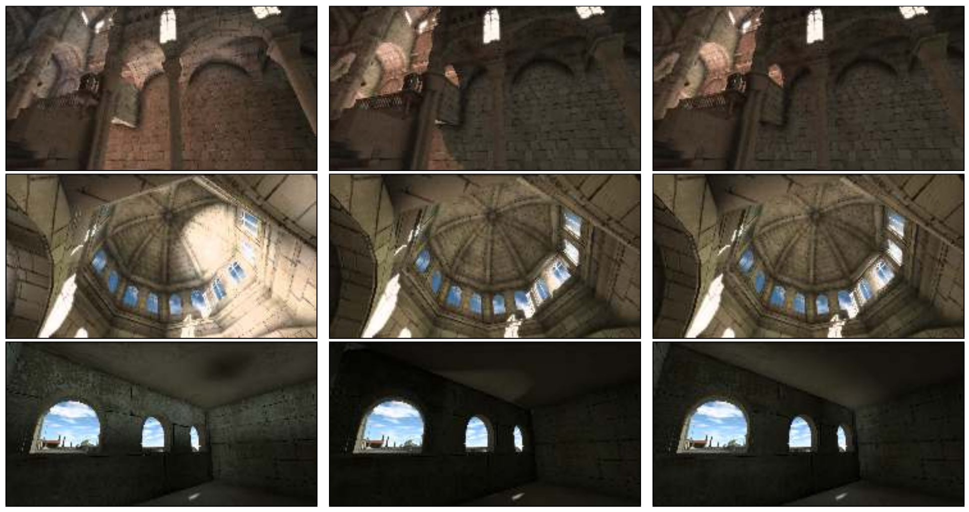
We compute multiple bounces of indirect illumination recursively, across frames, by seeding the radiance buffers with the previous bounce of light, similar to McGuire et al. [2017b]. This leads to a time-lag artifact for indirect bounces that is most evident in static scenes viewed by a static camera, which is not the use case in which we are primarily interested: when the view, lighting, and/or scene geometry is dynamic, the lag in indirect bounces is not noticeable.

Our approach could easily be adapted to collect the per-bounce results (up to a maximum bounce bias) before display, if fixed-view and geometry-usage scenarios are a priority. Given the performance of our approach (see Section 6), we would still reach interactive shading rates despite not being able to amortize the cost of multiple bounces across frames.

6. Results

We benchmark our approach on scenes with a mix of geometric and radiometric complexity. We explore the impact that probe count, resolution, and pixel format play in final rendering quality. We also compare the quality of our final rendering to path-traced references (computed offline).

We show results with direct illumination, glossy ray-traced reflection, shadows, tone-mapping, bloom, and other standard rendering terms to show the diffuse GI in the context of a full renderer. The code for this pipeline is available as the open source G3D Innovation Engine (`https://casual-effects.com/g3d`), where we injected the diffuse and glossy GI terms using the G3D::DefaultRenderer's path for reading from two screen-resolution textures. The glossy ray tracing is simply brute force mirror-ray tracing per pixel followed by a bilateral blur pass based on glossiness and distance, which is a standard practice [2013]. For a state-of-the-art ray-traced glossy approach, see [Schmid et al. 2019].



(a) Classic irradiance probes **(b)** Low-res raycast visibility **(c)** New variance visibility

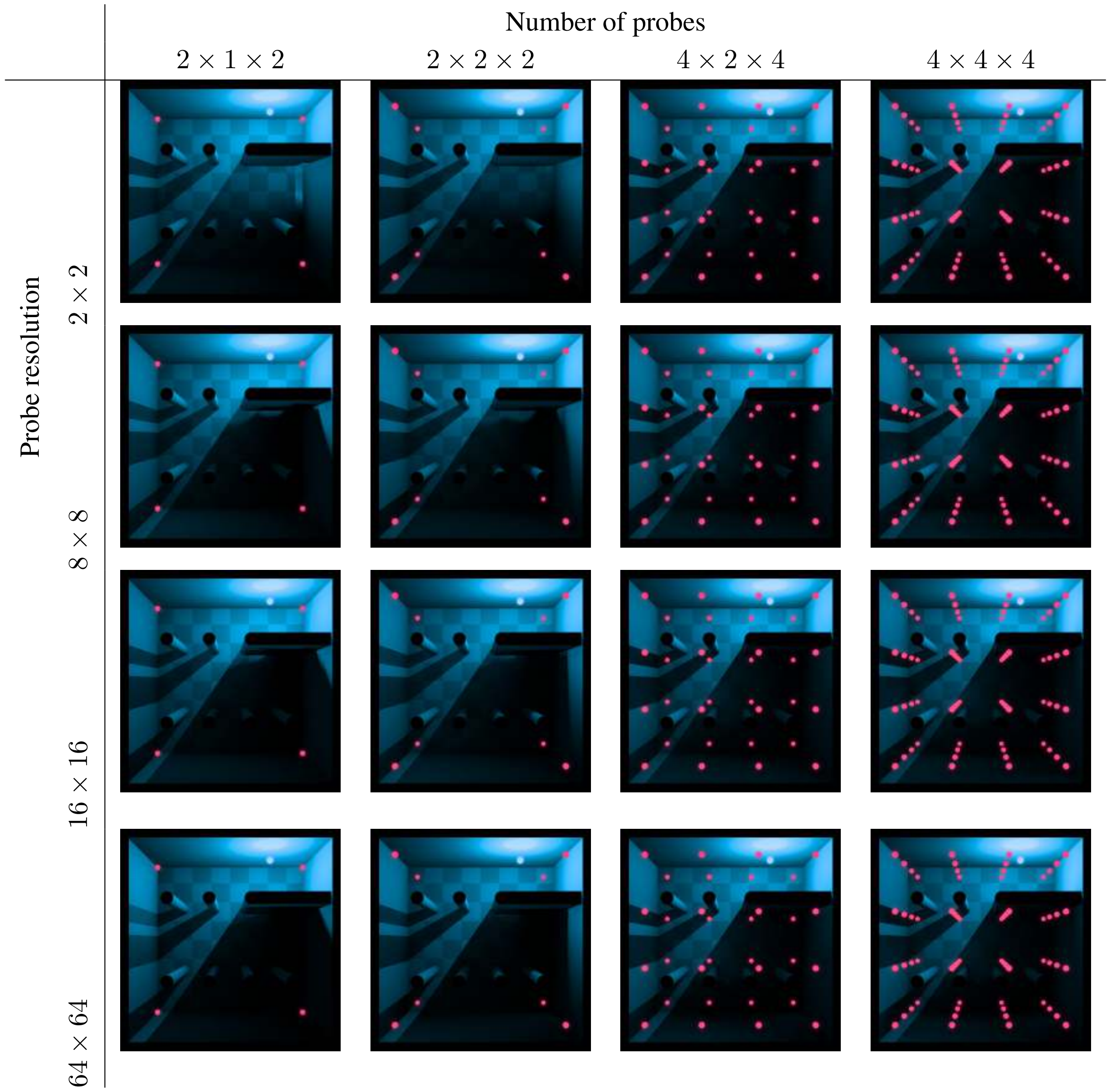
Figure 8. Comparison of probes with no visibility test, visibility test by a low-res ray cast, and our new variance visibility.

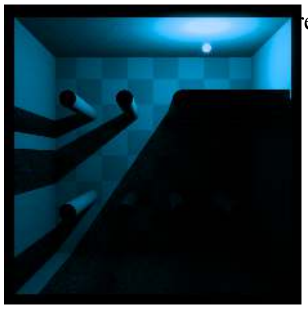
Since the rays will be convolved with a clamped cosine, glossy reflections on second-hit surfaces will be indistinguishable in most cases. We recommend adding the energy from the glossy portion of the BRDF to the matte portion in those second-hit surfaces, which we did in all of our figures. This handles today's practical cases well and gives a slight speedup to shading. However, in the theoretical case of extremely dense and high-resolution probes, this step would affect correctness, so we left a disabled code path in the reference implementation that performs the full BRDF shade including glossy for the surfaces seen by probes.

For consistency when reporting performance statistics, we conservatively update every probe every frame, with the understanding that many probes that may not impact the final rendering quality will still be updated (and, so, we will incur a performance cost). This is especially true in large scenes (e.g., Figures 6, 8 and 16); here, many probes either fall outside the camera frustum and/or do not contribute to any directly (or indirectly) visible scene geometry. We leave optimized probe selection and adaptive probe updates to future work.

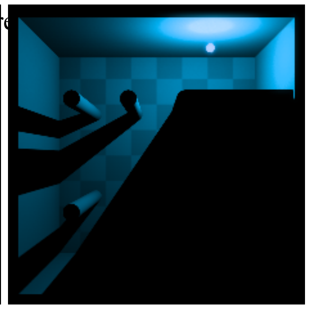
6.1. Probe Count and Resolution

Figures 9 and 10 illustrate the impact that probe count and resolution have on the final rendering. Figure 11 illustrates the impact of probe count in the specific case of indirect shadows. Probes are all initialized in a simple uniform 3D grid, scaled to the bounding volume of the scene. There is no need for manual probe placement due to the visibility-aware sampling of probe data, one of the main advantages of the probe representation.



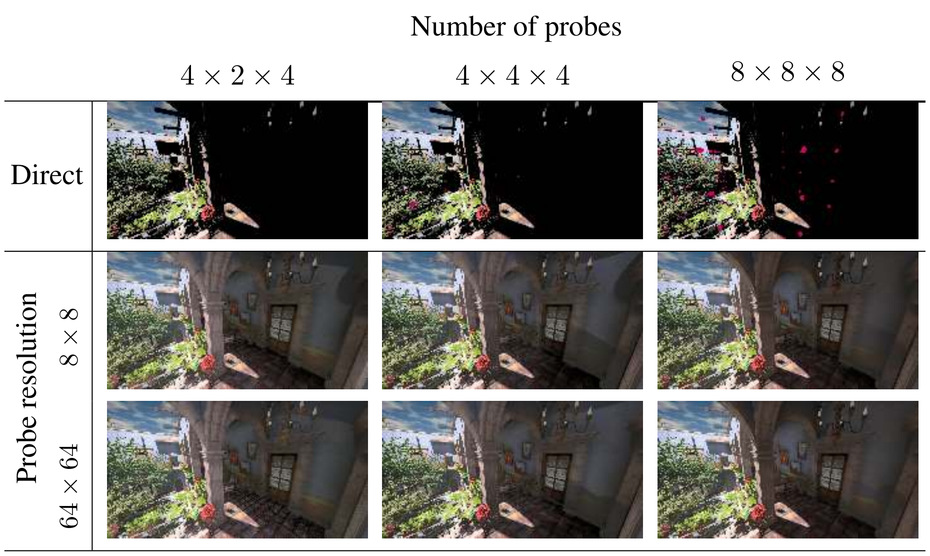


(b) Path-traced reference

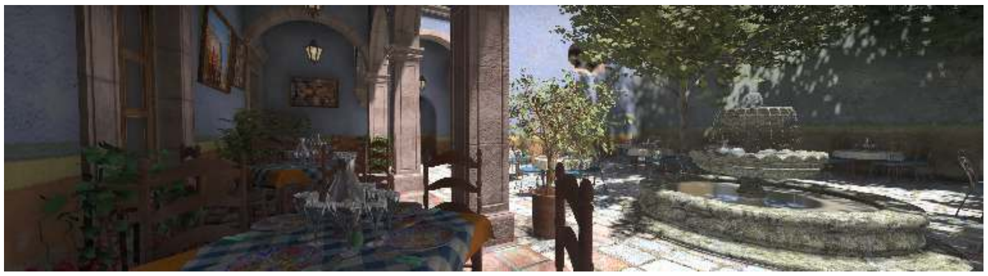


(c) Direct only

Figure 9. Quality comparison across a selection of probe resolutions and densities. Probe resolution is given as $X \times Y \times Z$, with Y increasing towards the camera. Note that even at low resolution, an image rendered with a sufficient number of probes looks identical to the path-traced reference.

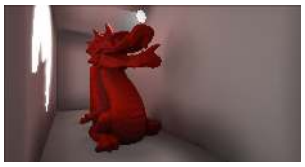


(a) Probe count and resolution comparison

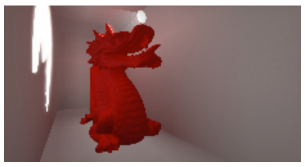


(b) Second view, indirect and direct: $8 \times 8 \times 8$ probe grid at 64×64 resolution

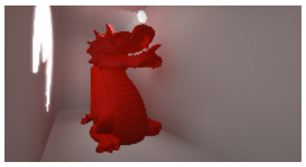
Figure 10. Quality comparison across a selection of probe resolutions and densities for a more complex scene.



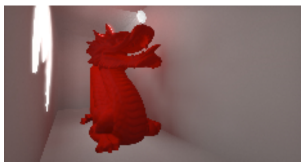
(a) 4×4 probes, AO



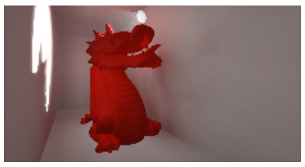
(b) 4×4 probes, no AO



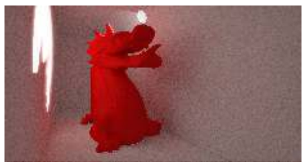
(c) 8×8 probes



(d) 16×16 probes



(e) 64×64 probes



(f) Path-traced reference

Figure 11. Indirect shadows with increasing probe grid resolution. As the resolution of the probe grid increases, the indirect shadow approaches the pathtraced reference without the overdarkened look of SSAO.

We can conclude that probe count plays a larger role than probe resolution: rendering with low (angular) resolution probes still leads to results that converge favorably compared to path traced reference, however low probe count/density can lead to subtle light leaking artifacts. These, however, are resolved with modestly chosen probe density settings.

6.2. Ray Tracing Performance

Table 1 shows throughput for ray casts with varying probe densities and rays/probe. Except in extreme cases, throughput is above 1.5 GRays/second. Table 2 shows timings for our algorithmic contributions within our rendering pipeline, and gives some context to the times in Table 1. The total time of our contributions is 7ms. However, note that our implementation updates all probes every frame, and thus incurs a high probe update cost. Adaptive probe selection (Section 7.1) would reduce this time considerably.

		Number of probes		
		$16 \times 8 \times 16$	$32 \times 8 \times 32$	$32 \times 16 \times 32$
Rays/probe	32	1.63 GRays/s	1.66 GRays/s	1.6 GRays/s
	64	1.63 GRays/s	1.62 GRays/s	1.59 GRays/s
	128	1.65 GRays/s	1.6 GRays/s	1.5 GRays/s
	256	1.65 GRays/s	1.5 GRays/s	1.48 GRays/s

Table 1. Ray throughput in Gigarays per second. Timings were taken on our Greek Temple Scene (876127 primitives) using an NVIDIA RTX 2080 Ti. For reference, we used $32 \times 8 \times 32$ probes with 64 rays/probe for our largest scenes.

		Time (ms)
Render Passes	Ray generation	0.1
	Ray cast	0.8
	Ray shade	0.4
	Probe update	0.7 (0.3 color + 0.4 depth)
	Sample irradiance probes for primary ray shading	0.5
	Deferred direct shading	0.1
	Total	2.6
	Add brute-force ray-traced glossy	+2.4

Table 2. Timings for the indirect light components of a single frame render using $32 \times 8 \times 32$ probes with 64 rays/probe. Probes were at 8×8 resolution using RGB10A2 format for color and RG16F format for depth. Timings were taken using glTimerQuery. We allocate time for the combined ray cast according to the proportion of rays for diffuse and glossy. We did not profile the unoptimized modular passes for parts of the system outside our contributions (shadow maps, AO, G-buffer generation) though we include unoptimized glossy ray cast and unoptimized glossy indirect shade in the final row for context.

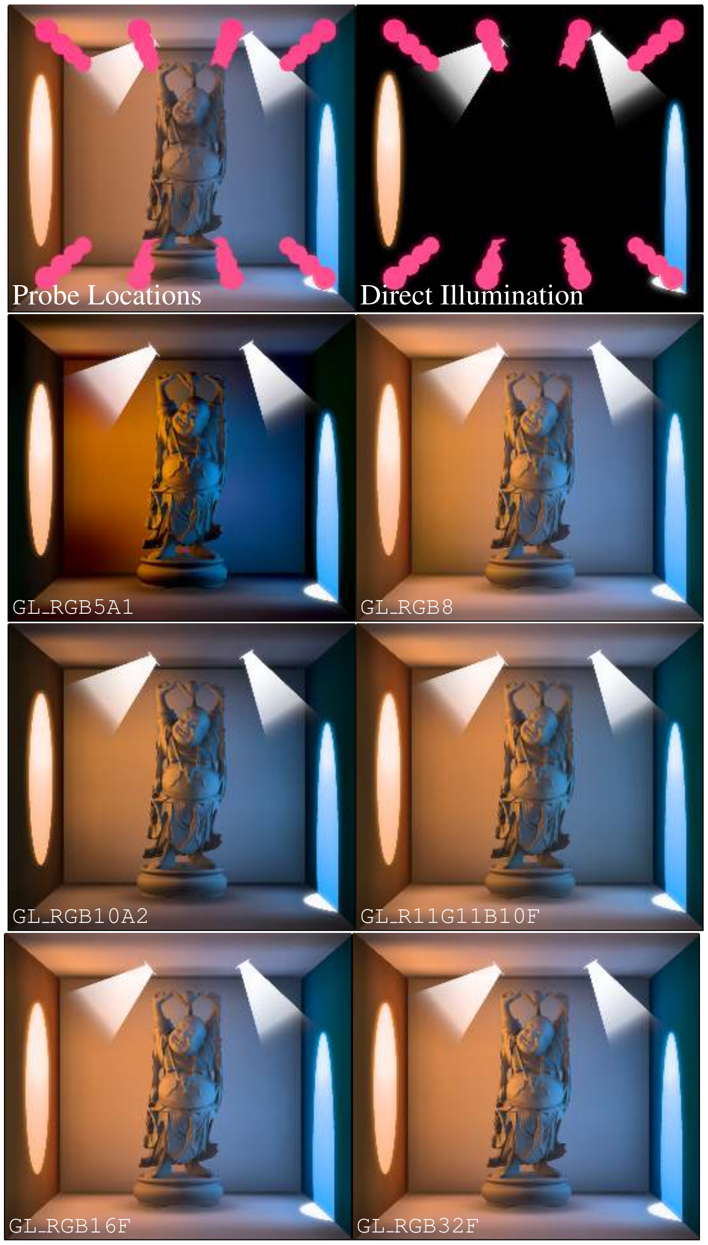


Figure 12. Color precision at 128 rays/probe/frame. `GL_RGB5A1` requires a low hysteresis $\alpha = 0.8$ in order to not fall below the blending threshold with many rays and suffers from flicker and oversaturation. The other formats, using $\alpha = 0.95$, are nearly indistinguishable from one another although `GL_R11G11B10F` has a greenish tint because it cannot represent exact grays. `GL_RGB10A2` balances quality and size. Note that `GL_RGB8` gives less precision but requires the same 32 bits/texel storage on modern GPUs due to word alignment. `GL_RGB10A2` and `GL_RGB8` are too dark because they lack the dynamic range of floating point.

6.3. Probe Texel Format

Figure 12 illustrates the impact of probe texel formats on the quality of our final rendering. Using eight-bit integer pixel formats can lead to artifacts that vanish at 16-bit floating-point representations. Experimentally, we find that 11-bit floating-point representations strike a good balance between precision and storage: at this bit depth, we maintain the visual fidelity of the 16-bit floating-point representation while reducing storage by a factor of 45%.

6.4. Quality Comparisons

Figure 13 compares our method to a path-traced reference on a dynamic scene using a grid of $4 \times 2 \times 4$ probes at 8×8 color resolution and 16×16 depth resolution. Our results are almost indistinguishable from the path-traced reference, rendered at several orders of magnitude faster. Note the variations of subtle diffuse indirect illumination

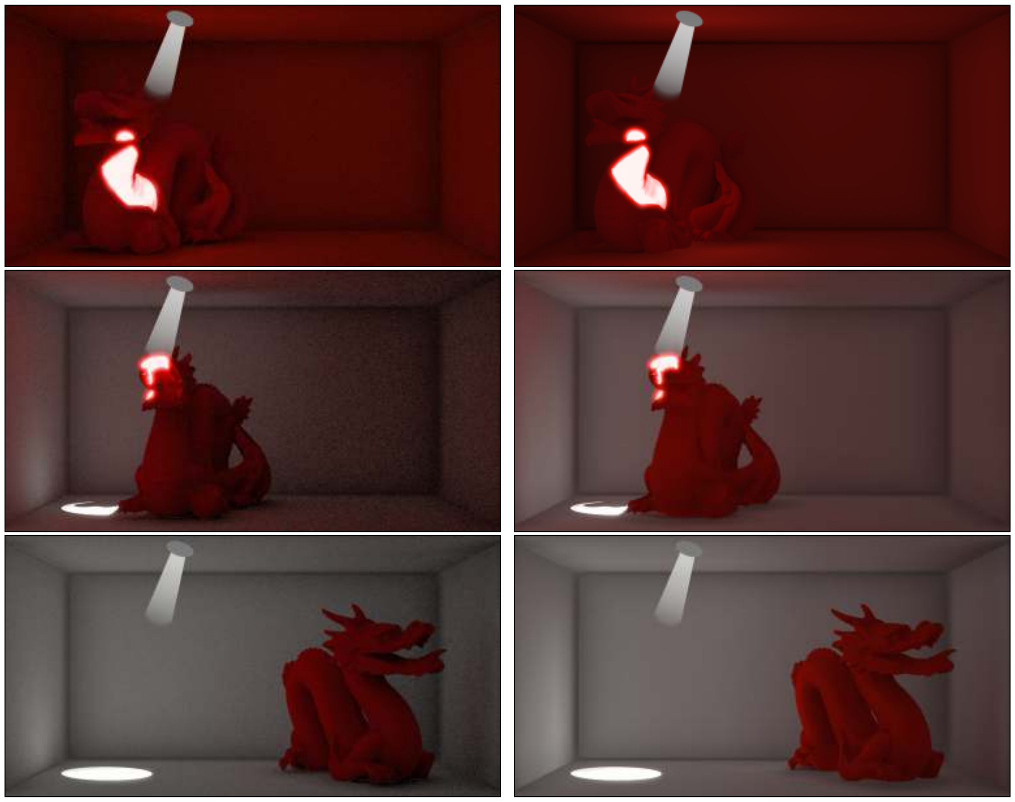

(a) Path-traced reference **(b)** New irradiance field

Figure 13. Comparing path tracing (left) to our dynamic diffuse global-illumination probes (right), with full diffuse global illumination, in a box with a dynamically translating dragon. See our video supplement for a real-time animation.

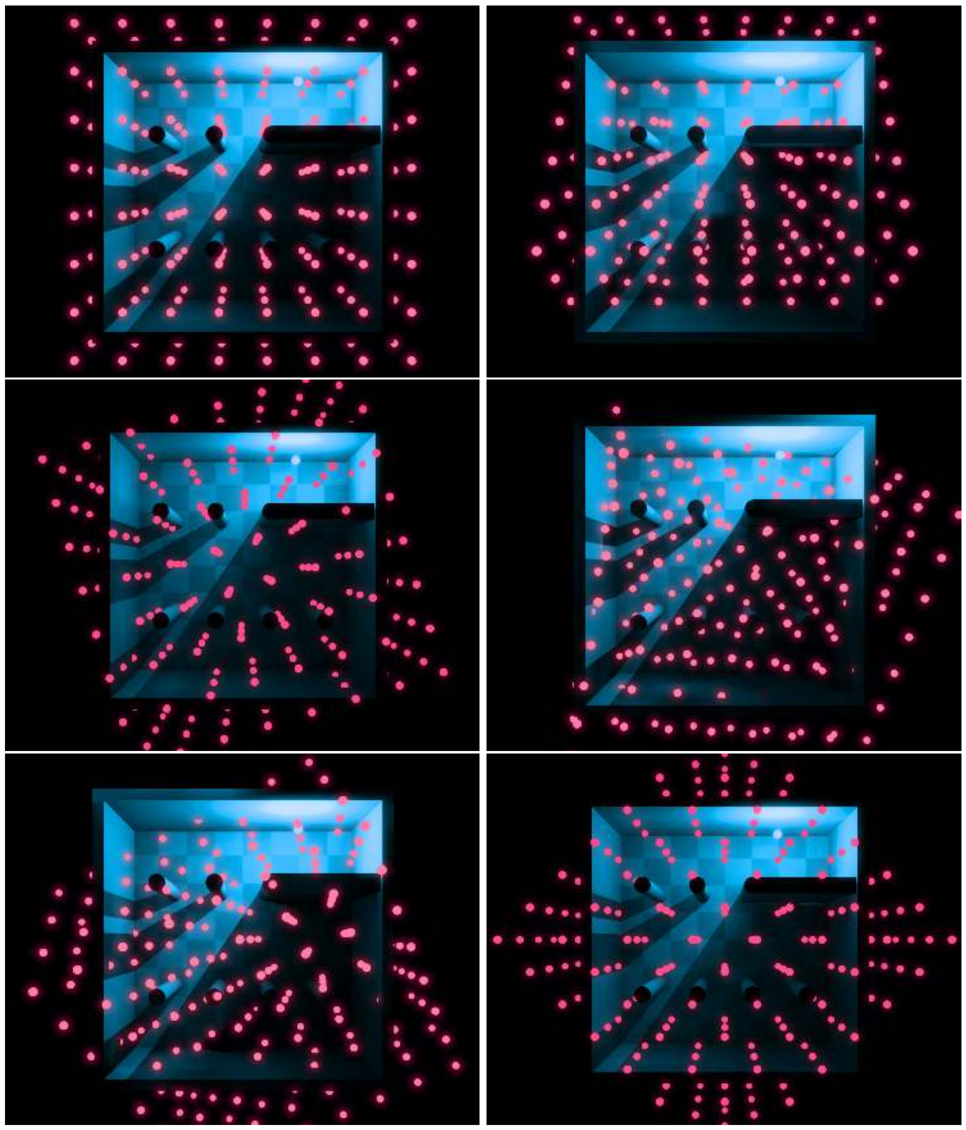


Figure 14. Our pillars test scene initialized with an enclosing $8 \times 4 \times 8$ probe grid. The grid is perfectly aligned with the box in the top-left image. All other images have a rotation and translation of the grid. Some images are chosen to intentionally break the algorithm by leaving part of the scene uncovered by probes (bottom left). Others are chosen at random. As long as there are probes covering the area being shaded, our algorithm is robust to rotation and translation of the probe grid. See our supplemental video for a demonstration of multiple random rotations, including some failure cases for positions outside the rotated and translated probe grid.

caused by the reflection of the red dragon onto the white walls of the box as the dragon passes under the light.

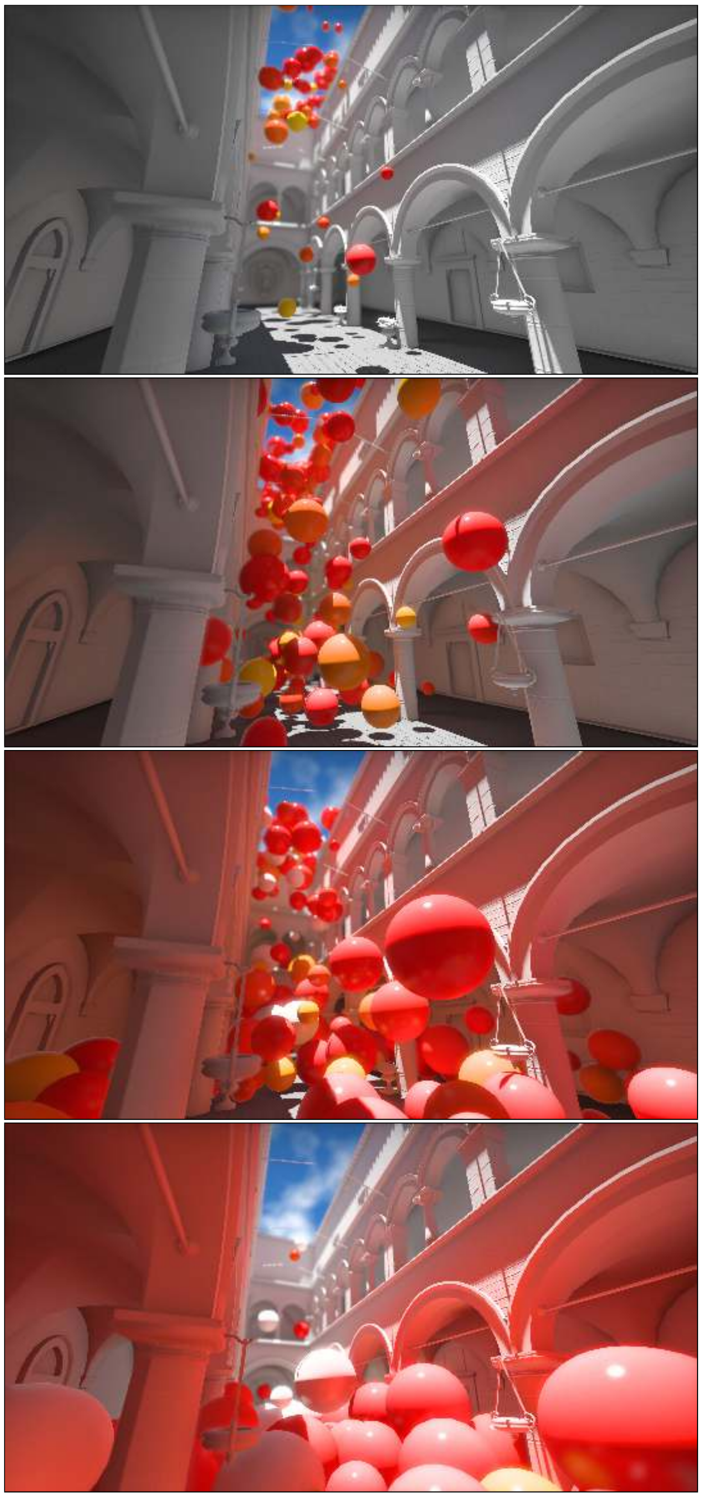
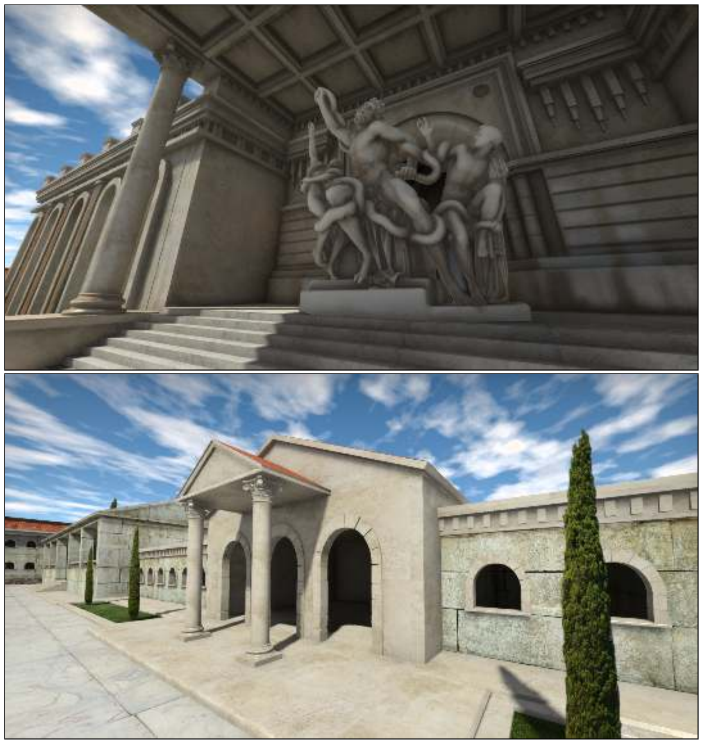
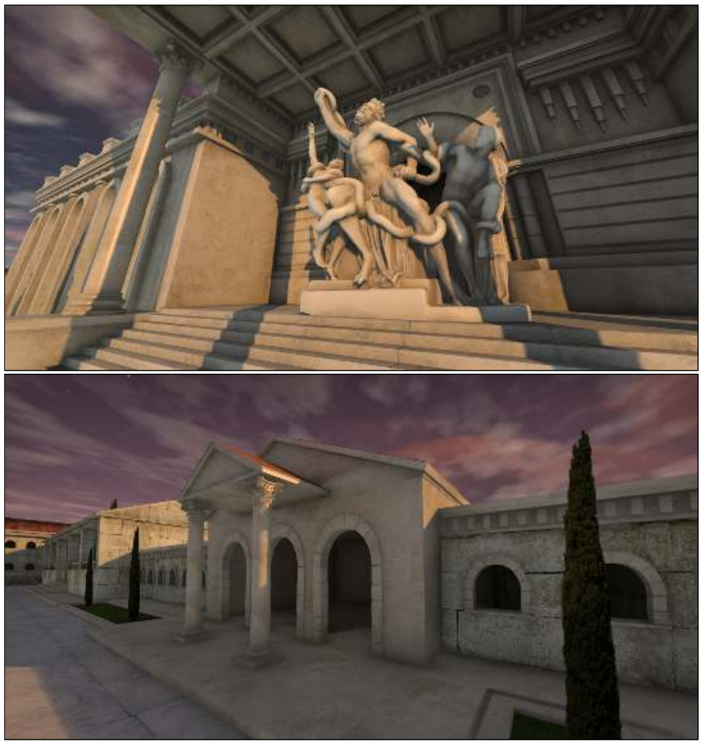


Figure 15. Time-lapse images showing fully dynamic GI with moving geometry. In this example, not only are a large number of spheres animating and casting complex GI, but they are also moving *through* the probes, which would lead to objectionable shadow leaks without correct occlusion. See our supplemental video for a real-time animation.



(a) Noon



(b) Evening

Figure 16. Time-lapse showing different times of day simulated with dynamic lighting. See our supplemental video for a real-time animation.

Figure 7 also illustrates the impact that the individual components of our robust diffuse indirect weighitng scheme has on rendering.

7. Conclusion and Discussion

We present an approach for updating and interpolating the irradiance field, as represented in dynamic diffuse global-illumination probes, in the presence of dynamic scene geometry and lighting, robustly treating temporal occlusion and lighting variation. Our method does not suffer from light- or shadow-leaking artifacts, suppressing aliasing due to undersampling. We compute accurate diffuse, glossy, and specular global-illumination effects in arbitrarily dynamic scenes at high performance.

This is due, in part, to an efficient data-packed probe layout that enables ray and shading computation to be dispatched in a coherent manner across probes. Our occlusion-aware spatial irradiance interpolation scheme is more robust to variations in smooth diffuse illumination, compared to the original scheme presented by [McGuire et al. 2017b].

We demonstrate how traditional forward- and deferred-rendering architectures that leverage precomputed lighting can be combined with modern GPU-enabled ray tracing in order to leverage the advantages of both of these enabling technologies.

Indeed, we build on the idea that an efficient ray tracer should be used not as a *substitute* for rasterization, but rather as a means to *complement* rasterization when incoherent visibility queries are needed. When combined with design strategies commonly used in the interactive rendering community, such as temporal amortization and probe-based precomputed lighting, our hybrid rendering solution generates results that amount to more than the sum of its technological parts.

Our indirect diffuse shading relies on a fundamental assumption about the spatial and angular relationship of radiance in a scene: here, we assume that the incident light at a shade point is similar to the incident light at the probes that surround it, *if the probes and the point are mutually visible*. The error induced by this assumption increases as probe density decreases (Figure 9).

7.1. Future Work

There are two immediate areas of future work that merit further investigation: the inclusion of more fall-back/alternative rendering paths, and adaptive selection for probe updates.

Alternative glossy and specular render paths. All of our glossy and specular transport is computed by sampling rays using our dynamically-updated probes. This brute-force sampling solution errs on the side of accuracy, at the cost of performance. Compared to public demonstrations of the recent interactive ray-tracing advances, such as the PICA PICA and Battlefield V engines [Andersson and Barré-Brisebois 2018], our glossy and specular shading solution is rather simple. These works give a sense of the potential performance gains that are possible by heuristically shortening rays, falling back to environment-mapped reflections, combining true ray-traced reflections with screen-space reflection approximations, using lower-resolution geometry LODs for distant intersections, and simplifying reflection shaders after the first bounce direct illumination.

These approximations are powerful—incorporating them in a manner that is both robust to different scene geometries and materials, and that allows direct control over error bounds, are interesting directions of future work.

Adaptive probe selection. In large and complex scenes, even conservatively culling probes can result in significant performance improvements. The scene-depth information we currently sample and store at each probe is immediately useful (and, likely, sufficient) to inform a more efficient probe-scheduling routine. For example, one can readily cull probes from the active probe list using a *furthest surface* heuristic (i.e., from every ray traced at every probe).

Acknowledgements

Thanks to Dylan Lacewell, Keith Morley, and James Bigler for invaluable advice on working with OptiX. Thanks to Michael Mara for the initial light-field probe code. Our implementation is built on the G3D Innovation Engine and benefits from the work of Michael Mara, Corey Taylor, and the many other open-source contributors to that platform.

References

AKENINE-MÖLLER, T., HAINES, E., HOFFMAN, N., PESCE, A., IWANICKI, M., AND HILLAIRE, S. 2018. *Real-Time Rendering 4th Edition*. A K Peters/CRC Press, Boca Raton, FL, USA, ch. 10.6, 425–433. 10, 11, 13

ANDERSSON, J., AND BARRÉ-BRISEBOIS, C. 2018. Shiny pixels and beyond: Real-time raytracing at seed. GDC 2018, EA SEED. URL: https://www.ea.com/seed/news/seed-gdc-2018-presentation-slides-shiny-pixels. 25

CIGOLLE, Z. H., DONOW, S., EVANGELAKOS, D., MARA, M., MCGUIRE, M., AND MEYER, Q. 2014. A survey of efficient representations for independent unit vectors. *Journal of Computer Graphics Techniques (JCGT) 3*, 2 (April), 1–30. URL: http://jcgt.org/published/0003/02/01/. 6

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., AND EISEMANN, E. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum 30*, 7, 1921–1930. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2011.02063.x. 5

CRASSIN, C., LUEBKE, D., MARA, M., MCGUIRE, M., OSTER, B., SHIRLEY, P., SLOAN, P.-P., AND WYMAN, C. 2015. CloudLight: A system for amortizing indirect lighting in real-time rendering. *Journal of Computer Graphics Techniques (JCGT) 4*, 4 (October), 1–27. URL: http://jcgt.org/published/0004/04/01/. 8

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06. ACM, New York, NY, 161–165. URL: http://doi.acm.org/10.1145/1111411.1111440. 11

DONOW, S. 2016. *Light probe selection algorithms for real-time rendering of light fields*. Master's thesis, Williams College, Williamstown, MA. 6

EVANGELAKOS, D., 2015. A light field representation for real time global illumination. Bachelor's Thesis, Williams College, Williamstown, MA. 6

HOOKER, J. 2016. Volumetric global illumination at treyarch. In *Advances in Real-Time Rendering 2016*, SIGGRAPH 2016. ACM, New York, NY. URL: https://www.activision.com/cdn/research/Volumetric_Global_Illumination_at_Treyarch.pdf. 3, 4, 5

IWANICKI, M. 2013. Lighting technology of the last of us. In *ACM SIGGRAPH 2013 Talks*, SIGGRAPH '13. ACM, New York, NY, 20:1–20:1. URL: http://doi.acm.org/10.1145/2504459.2504484. 3

JENDERSIE, J., KURI, D., AND GROSCH, T. 2016. Real-Time Global Illumination Using Precomputed Illuminance Composition with Chrominance Compression. *Journal of Computer Graphics Techniques (JCGT) 5*, 4 (December), 8–35. URL: http://jcgt.org/published/0005/04/02/. 6

LAGARDE, S., AND ZANUTTINI, A. 2012. Local image-based lighting with parallax-corrected cubemap. In *ACM SIGGRAPH 2012 Talks*, SIGGRAPH '12. ACM, New York, NY, 36:1–36:1. URL: https://seblagarde.wordpress.com/2012/11/28/siggraph-2012-talk/. 4

MARTIN, S., AND EINARSSON, P. 2010. A real time radiosity architecture for video games. In *Advances in Real-Time Rendering 2010*, SIGGRAPH 2010. ACM, New York, NY. URL: http://advances.realtimerendering.com/s2010/Martin-Einarsson-RadiosityArchitecture(SIGGRAPH%202010%20Advanced%20RealTime%20Rendering%20Course).pdf. 4

MCAULEY, S. 2012. Calibrating lighting and materials in Far Cry 3. In *Practical Physically Based Shading in Film and Game Production*, SIGGRAPH 2012. ACM, New York, NY. URL: https://blog.selfshadow.com/publications/s2012-shading-course/. 4

MCGUIRE, M., MARA, M., AND MAJERCIK, Z., 2017. The G3D innovation engine. URL: https://casual-effects.com/g3d. 4

MCGUIRE, M., MARA, M., NOWROUZEZAHRAI, D., AND LUEBKE, D. 2017. Real-time global illumination using precomputed light field probes. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '17, 2:1–2:11. URL: http://casual-effects.com/research/McGuire2017LightField/index.html. 2, 5, 6, 8, 9, 10, 12, 13, 15, 25

REINHARD, E., DEBEVEC, P., WARD, G., MYSZKOWSKI, K., SEETZEN, H., HESS, D., MCTAGGART, G., AND ZARGARPOUR, H. 2006. High dynamic range imaging: Theory and practice. SIGGRAPH 2006. ACM, New York, NY. URL: http://old.siggraph.org/publications/2006cn/course05.pdf. 5

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. In *ACM SIGGRAPH Asia 2008 Papers*, ACM, New York, NY, USA, SIGGRAPH Asia '08, 129:1–129:8. URL: http://doi.acm.org/10.1145/1457515.1409082, doi:10.1145/1457515.1409082. 5

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09. ACM, New York, NY, 75–82. URL: http://doi.acm.org/10.1145/1507149.1507161, doi:10.1145/1507149.1507161. 4

SCHMID, J., ULUDAG, Y., AND DELIGIANNIS, J. 2019. It just works: Ray-traced reflections in "Battlefield V". Presented at GPU Technology Conference, 2019. URL: https://schedule.gdconf.com/session/it-just-works-ray-traced-reflections-in-battlefield-v/863651. 15

SHANMUGAM, P., AND ARIKAN, O. 2007. Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07. ACM, New York, NY, 73–80. URL: http://doi.acm.org/10.1145/1230100.1230113. 13

SILVENNOINEN, A., AND LEHTINEN, J. 2017. Real-time global illumination by precomputed local reconstruction from sparse radiance probes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 36*, 6 (11), 230:1–230:13. URL: https://doi.org/10.1145/3130800.3130852. 5

STACHOWIAK, T., AND ULUDAG, Y. 2015. Stochastic screen-space reflections. In *Advances in Real-Time Rendering 2015*, SIGGRAPH 2015. ACM, New York, NY. URL: https://www.ea.com/frostbite/news/stochastic-screen-space-reflections. 6

THALER, J., 2011. Deferred rendering. Technical Report, TU Wien. URL: https://www.researchgate.net/profile/Jonathan_Thaler2/publication/323357208_Deferred_Rendering/links/5a8fce31aca272140560aaad/Deferred-Rendering.pdf. 11

TOTH, R., HASSELGREN, J., AND AKENINE-MÖLLER, T. 2015. Perception of highlight disparity at a distance in consumer head-mounted displays. In *Proceedings of the 7th Conference on High-Performance Graphics*, ACM, New York, NY, HPG '15, 61–66. URL: http://doi.acm.org/10.1145/2790060.2790062. 6

VALIENT, M. 2013. Killzone shadow fall demo postmortem. Sony Devstation 2013, Guerilla Games. URL: https://www.guerrilla-games.com/read/killzone-shadow-fall-demo-postmortem. 5, 15

WYMAN, C. 2005. An approximate image-space approach for interactive refraction. *ACM Trans. Graph. 24*, 3 (July), 1050–1053. URL: http://doi.acm.org/10.1145/1073204.1073310. 6

Index of Supplemental Materials

The supplemental materials (video and code) can be found at http://www.jcgt.org/published/0008/02/01/supplement.zip.

The video (DynamicLightFieldProbes.mp4) shows experiments to validate our method. They include: dynamic geometry (Figure 13), dynamic occlusion, robustness to probe grid perturbation (Figure 14), a dynamic occlusion stress test (Figure 15), and dynamic lighting (Figure 16).

The file sampleIrradianceField.pix contains code to compute irradiance from the probe grid; IrradianceFieldgenerateRandomRays.pix contains code for generating probe-update rays. The other glsl files contain helper functions. The file IrradianceField.h/.cpp contains details of our ray-cast and generation setup.

Author Contact Information

Zander Majercik
NVIDIA
2788 San Tomas Expressway,
Santa Clara, CA 95051, USA
amajercik@nvidia.com

Jean-Philippe Guertin-Renaud
Universitë de Montréa;
CP 6128, Succ. Centre-Ville
Montreal, Quebec H3C 3J7 Canada
contact@jpg.sh

Derek Nowrouzezaharai
McGill University.
3480 University Street
Montreal, Quebec Canada H3A 0E9
rderek@cim.mcgill.ca
www.cim.mcgill.ca/~derek/

Morgan McGuire
NVIDIA
431 King St. W
Toronto ON M5V 3M4 Canada
mcguire@nvidia.com
https://casual-effects.com

Majercik et al., Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields, *Journal of Computer Graphics Techniques (JCGT)*, vol. 8, no. 2, 1–30, 2019
http://jcgt.org/published/0008/02/01

Received:	2018-12-22		
Recommended:	2019-03-11	Corresponding Editor:	Andrew Willmott
Published:	2019-06-05	Editor-in-Chief:	Marc Olano