Real-Time Global Illumination using Precomputed Light Field Probes

Morgan McGuire
Williams College and NVIDIA

Mike Mara
Stanford University

Derek Nowrouzezahrai
McGill University

David Luebke
NVIDIA

Final Image | Top View of Probe Locations | Path of one Ray through the Probes

Figure 1: *Left to right: a living room scene with global illumination computed in **4.7 ms** using our light field probes (i.e., there is no artificial ambient or environment map term). Note the glossy reflection on the floor, table, and television, as well as interreflections throughout the scene. We visualize the 4^3 grid of 1024×1024-texel light field probes, as well as the path of a single ray traced through four of these probes.*

Abstract

We introduce a new data structure and algorithms that employ it to compute real-time global illumination from static environments. *Light field probes* encode a scene's full light field and internal visibility. They extend current radiance and irradiance probe structures with per-texel visibility information similar to a G-buffer and variance shadow map. We apply ideas from screen-space and voxel cone tracing techniques to this data structure to efficiently sample radiance on world space rays, with correct visibility information, directly within pixel and compute shaders. From these primitives, we then design two GPU algorithms to efficiently gather real-time, viewer-dependent global illumination onto both static and dynamic objects. These algorithms make different tradeoffs between performance and accuracy. Supplemental GLSL source code is included.

Keywords: global illumination, irradiance, light field

Concepts: •**Computing methodologies** → **Rendering;**

1 Introduction

Radiance and irradiance probes are widely adopted techniques that, when combined with screen-space ray tracing and ambient occlusion, form the core for approximating realistic shading effects in most interactive graphics engines today.

Despite their popularity and flexibility, probes and screen-space techniques are not without limitations. Expensive, manual placement of light probes and proxy geometry is often necessary to reduce light and dark leaking artifacts and misaligned reflection effects (see Figure 13). High-performance screen-space techniques add important visual features, naturally handle dynamic objects, and map well to modern GPUs (e.g., exploiting high coherence texture accesses); however, they are limited to locally-visible surfaces and can suffer from view-dependent aliasing.

We present a new probe representation that combines the quality of precomputed shading with the ability to perform true world-space ray tracing, all with similar performance to modern screen-space ray tracers. To do so, we populate and encode an augmented representation of a scene's light field in a compact, GPU data structure.

Concretely, our technical contributions are:

- a *light field probe* data structure that encodes a scene's radiance and geometric information for visibility-aware light field queries,
- an efficient light field probe construction algorithm that does not require any surface parameterization,
- a world-space ray tracing algorithm using light field probes that can be invoked directly from a shader,
- extension of prefiltered (ir)radiance maps with visibility-aware sampling and interpolation to eliminate light and dark leaks, and
- demonstration and evaluation of light-transport simulation algorithms in a real-time deferred rendering context.

Fortunately, while the derivation and optimization process are somewhat complex, the final implementation is relatively simple and we include executable shader code for it. The code structure for the first illumination algorithm resembles a Monte Carlo ray tracer atop a hierarchical ray marcher. It is appropriate for interactive applications such as digital content creation. Because rays sampled from a Lambertian BSDF lobe tend to yield high variance, we include a second illumination algorithm that uses the prefiltered irradiance maps with that lobe. The prefiltered result are fast and noise-free, but biased. This algorithm is appropriate for real-time applications such as games and virtual reality experiences.

Figure 1 illustrates a living room scene with global illumination computed with our method, including diffuse and glossy indirect light, mirror reflections, and area sources. We visualize light field probe locations, as well as the path that a world-space ray marches in our octahedral parameterization of the scene geometry (encoded as part of a light field probe): here, apparent changes of the ray's direction occur at octahedral face boundary crossings, and the ray jumps from one probe to another when visibility cannot be resolved by the first probe (see Section 4). We use a two-level ray traversal

I3D '17, February 25 - 27, 2017, San Francisco, CA, USA
ISBN: 978-1-4503-4886-7/17/03…$15.00
DOI: http://dx.doi.org/10.1145/3023368.3023378

hierarchy: orange lines are traced at $1/16^2$ resolution, and yellow segments at full resolution (for fine-scale geometric details).

2 Previous Work

Light fields. The light field (or plenoptic function) describes the spatial-angular distribution of radiance in free space for an environment, dating back to models proposed in physics literature in the early 1930's. The first works to use light fields in graphics introduced the concept of a *light slab*, encoding a rectangular subset of the light field [Levoy and Hanrahan 1996; Gortler et al. 1996]. This light slab has been employed in many products and productions, and methods exist for reconstructing it in real-time with the slab region. However, ours is the first method to encode and apply to real-time rendering a light field parameterization that can be evaluated at *any* point in space, with correct occlusion.

Image-based lighting. The state of the art in interactive global illumination for games relies heavily on image-based lighting variants, many of which are documented exclusively in GDC and SIGGRAPH talks (e.g., [Martin and Einarsson 2010; Ritschel et al. 2009; Mickael Gilabert 2012; Sébastien and Zanuttini 2012; Hooker 2016]). In most cases, probes are placed densely inside the volume of a scene, each of which encoding a spherical radiance map. Additional pre-filtered radiance maps are often stored in order to approximate diffuse and glossy reflections. Sometimes, manually-placed (i.e., by an artist) box or sphere proxies are used to warp the lookups into these maps, in order to better approximate local reflection variation. Moreover, manually-placed convex proxy geometry sets can be used to bound blending weights used during reflection lookups, in order to prevent light leaks.

We refer readers to a comprehensive survey of these initial works, and other advanced image-based lighting techniques [Debevec 2006]. Image-based lighting techniques are ubiquitous in modern offline and real-time rendering.

While these production-level image-based lighting systems generate convincing illumination effects, practitioners agree that manual probe and proxy placement remains an important open problems in production [Hooker 2016]. Without these manual adjustments light and dark (i.e., shadow) leaks, and displaced reflections, are unavoidable. Some applications rely instead on screen-space ray tracing [Valient 2014] for pixel-accurate reflections; however, this fails when a reflected object is not visible from the camera's point of view.

Light field probes automatically resolve these issues by encoding additional information about the scene geometry (Section 3). No manual placement is necessary and all of our results use a naïve uniform grid placement of the probes: reflections appear (consistently) where they should due, in part, to our accurate world-space ray tracing algorithm (Section 4), and *visibility-aware* blending weights are computed automatically when sampling filtered radiance probes (Section 5) without the need for manually placed geometric proxies about each probe.

Interactive tracing & shading. Many recent interactive rendering approaches for resolving point-to-point queries (e.g., like mutual visibility) have shaped the 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 they use to compute accurate secondary diffuse and glossy reflections using virtual point lights (shot, e.g., using a ray-tracer). Our work is motivated by voxel cone tracing [Crassin et al. 2011]. At a high-level, one can interpret our ray tracing solution (Section 4) tracing rays against spherical voxel cells, as opposed to the octree representation constructed for voxel cone tracing. Two key differences, that lead to many practical advantages, are: first, we *explicitly* encode geometric scene information (i.e., normals and radial depths) instead of relying on the *implicit* octree structure to resolve local and global visibility details; and second, neither our spatial parameterization nor our filtering rely on the scene geometry, which allows us to completely sidestep the light (and dark) leaking artifacts present in voxel cone tracing solutions. Moreover, 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*.

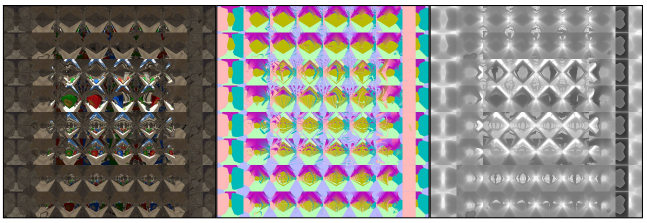

Figure 2: *Light field probes: radiance, normal, and distance maps are packed together for all the probes in a the Sponza scene.*

Other works. We use Cigolle et al.'s [2014] octahedral mapping from the sphere to the unit square for storing our spherical distributions, since it has slightly less distortion and admits simpler border handling than cube maps. As we enable *true world-space* ray-tracing within a pixel shader, our technique can be interpreted as a generalization of many previous, e.g., real-time environment map Monte Carlo integration techniques [Stachowiak 2015; Wyman 2005; Toth et al. 2015; Jendersie et al. 2016] .

Preliminary investigations for our work were undertaken by Evangelakos [2015], who showed that ray tracing with a single probe is correct for star-shaped geometric regions, and Donow [2016], who proposed a multiple-probe tracing algorithm.

3 Light Field Probe Structure

A scene's continuous)light field $\mathcal{L}(\mathbf{x}, \omega)$ encodes the spatial-angular distribution of radiance for all points and directions in the scene, where $\mathbf{x} \in \mathrm{R}^3$ is a point in free-space (i.e., within the bounding volume of scene) and $\omega \in \mathrm{S}^2$ is a unit outgoing direction at $\mathbf{x}$.

We build an augmented, discrete representation of this distribution, and so we require a discretization of positions in space and directions. We discretize spatial variation of the light field using a regular grid inside the scene's bounding box. While this regular structure allows us to simplify our tracing algorithm (Section 4), nothing prevents us from using spatial samples jittered within each volumetric grid cell during construction- and query-time. Such jittering is useful to allow artists to displace spatial samples that fall within the interior of an object to increase efficiency or cooperative sampling patterns such as Latin squares to avoid certain worst-case behavior for scene elements exactly aligned with the grid, but is never required because our ray tracing algorithm automatically disregards results from fully-occluded probes.

At each discrete sample position $\mathbf{x}'$, we store a spherical *light field probe* that map directions ω about $\mathbf{x}'$ to the following quantities:

- a (discretized) spherical "slice" of the light field at $\mathbf{x}'$ (i.e., $\mathcal{L}(\mathbf{x}', \omega)$) that corresponds to the local incident radiance distribution at $\mathbf{x}'$ from surrounding scene geometry and light sources
- the surface normal $\vec{\mathbf{n}}_{\mathbf{x}''}$ at points $\mathbf{x}''$ closest to $\mathbf{x}'$ in (outgoing) direction ω, and
- the radial distance $r_{\mathbf{x}' \leftrightarrow \mathbf{x}''}$ between $\mathbf{x}'$ and $\mathbf{x}''$.

We populate the geometric information (e.g., normals $\vec{\mathbf{n}}_{\mathbf{x}''}$ and radial distances $r_{\mathbf{x}'\leftrightarrow\mathbf{x}''}$ at points $\mathbf{x}''$) of the probes using rasterization, and the spherical light field slice radiance $\mathcal{L}(\mathbf{x}', \omega)$ distribution is generated using a numerical lighting simulation: depending on the application, this simulation can either be computed during rasterization using existing interactive shading techniques (i.e., deferred renderers with shadow maps), using an offline path tracer, or even generated through iterative application of our shading algorithm (Section 5). All of our results do the latter: specifically, for the first bounce of direct illumination, $\mathcal{L}(\mathbf{x}', \omega)$ simply contains the emission profiles of lights in a scene, and then we repeat the shading to iteratively populate $\mathcal{L}(\mathbf{x}', \omega)$ with additional light bounces. Because the algorithm can use rasterization to compute the probes themselves, we do not require a separate geometric ray tracer in the manner of most light map generation algorithms.

Section 4 details a ray tracing algorithm that allows us to populate, update and query physically-based light transport quantities using this data structure but, before doing so, we provide practical guidelines for realizing this discrete representation on modern GPUs.

Implementation Details. We use the engine's usual rasterizer path to render high-resolution cube map faces at each probe location, and then resample them into lower-resolution octahedral projections [Cigolle et al. 2014] (see our supplemental code). Octahedral parameterization maps an entire sphere to a square that has lower distortion, fewer boundaries, and simpler boundaries than a cube map at comparable resolution. This is not essential to the derivation of the algorithm, however it gives approximately a factor of four space and bandwidth reduction at the same quality level compared to storing cube maps. Octahedral mapping also preserves the piecewise-linear projection required for efficient ray marching, unlike most other sphere-to-square mappings.

We store each probe's square mappings as one layer of a 2D texture array. The high-dynamic range radiance values are encoded with the `R11G11B10F` (32 bits/pixel) format, the normals $\vec{\mathbf{n}}_{\mathbf{x}''}$ with `RG8` (16 bpp), and the radial distances $r_{\mathbf{x}'\leftrightarrow\mathbf{x}''}$ with `R16F` (16 bpp), for a total of **8 bytes per octahedral texel**. Figure 2 illustrates these light field probe textures for a 4^3 grid for the Sponza scene.

For static illumination and geometry, we can alternatively use the compressed `BC6H` (8 bpp) format for radiance and two `BC4` textures (2×4 bpp) for normals, reducing the storage cost to **4 bytes per texel**. Unfortunately, current GPUs lack compressed 16-bit scalar compression suitable for encoding radial distances. Since BC compression is slow and our shading algorithm's bandwidth is almost independent of the normal and radiance size, our results all use uncompressed textures to reduce light field generation time. We also use an octahedral map to encode unit normal vectors in 2D.

For shading applications (see Section 5) it is often beneficial to prefilter the light field radiance "slice" $\mathcal{L}(\mathbf{x}', \omega)$, additionally storing a cosine-weighted incident irradiance probe $\mathbf{E}(\mathbf{x}', \vec{\mathbf{n}})$ for all possible receiving surfaces at $\mathbf{x}'$ (with orientation $\vec{\mathbf{n}}$). Here, we found that octahedral resolutions above 128^2 resulted in no additional accuracy in the diffuse shading and, in many scenes, probe resolutions as low as 32^2 still yield good results (depending on the scene's visibility complexity). We also store irradiance probes in 8 bytes per octahedral texel, simply replacing the normals from the radiance representation with 16-bit floating point squared depths needed for the Chebychev test explained later, and retaining the other channels.

For radiance probes, the required resolution depends on the image resolution and the lowest microfacet roughness on scene objects: e.g., in order to avoid reflectance aliasing on perfect mirrors, we require an octahedral probe resolution with texel solid angles no larger than the solid angle subtended by a pixel at the probe center $\mathbf{x}'$. We recommend 1024^2 radiance maps and use them for all our results, except for experiments explicitly on resolution parameters.

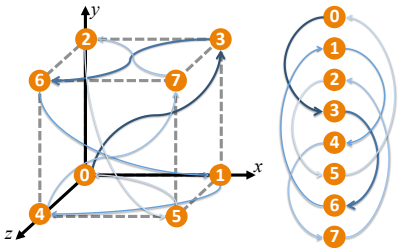

Figure 3: *An indexed cube of probes. Arrows illustrate the order in which probes are considered when testing ray-scene intersections.*

Figure 1 is rendered at 1920×1080, with shading computed *at full resolution*. This is in stark contrast to modern screen-space ray tracers that operate at, e.g., $1/4\times$ screen resolution.

4 Light Field Probe Ray Tracing

Generally speaking, every physically-based rendering algorithm can be implemented atop a ray tracer, where (potentially incoherent) rays are used to resolve visibility and to sample radiometric quantities in the scene[1]. Concretely, a ray is a half-line with points $\mathbf{p}(t) = \mathbf{r}_o + t\,\mathbf{r}_d$ parameterized by the distance t from the ray's origin $\mathbf{r}_o \in \mathrm{R}^3$ along its direction $\mathbf{r}_d \in \mathrm{S}^2$. Resolving visibility and querying the scene amounts to intersecting rays against the scene geometry, solving for the (nearest) intersection point $\mathbf{p}(t') \equiv \mathbf{p}'$.

Radial distances and visible normals in the light field probes encode the geometric information used to resolve *incoherent, world-space ray intersections*, and the radiance (and irradiance) probes encode radiometric quantities of interest, i.e., the outgoing radiance $\mathcal{L}(\mathbf{p}', -\mathbf{r}_d)$ (and incident irradiance $\mathbf{E}(\mathbf{p}', \vec{\mathbf{n}} \equiv -\mathbf{r}_d)$).

Our light field ray tracer (Listing 1) first selects a probe, then invokes a *single-probe ray intersection routine* to march across the geometry defined by that probe. If the single-probe routine is unable to accurately determine a `hit` or `miss` event, then the algorithm advances to the next candidate probe and reapplies the single-probe routine.

As such, the single-probe routine returns one of three possible results: a definitive `hit`, a definitive `miss`, or an `unresolvable` exit code; the latter occurs when the ray passes into a 3D region that is occluded from the probe's center of projection. In other words, when a probe "knows" that it does not have the required visibility information at a region of space, an intersection for a ray that passes through that region cannot be (reliably) computed using only information from the one probe. Note, however, that by increasing the density of light probes stored in the scene, we are (very quickly) able to guarantee accurate world-space ray-object intersection resolution *across probes*. We provide the details of the single-probe and full ray tracing algorithms, below and in Listing 1.

Listing 1: *Pseudocode for the ray tracing algorithm.*

```
def lightFieldTrace(ray):
    result = UNKNOWN
    while result == UNKNOWN:
        choose the next probe
        (result, endpoint) = singleProbeTrace(ray, probe)
        ray.origin = endpoint # Advance the ray to the last point checked
    return result
```

[1] Rasterization can resolve coherent visibility queries, and so many approximate solutions to the rendering equation need not rely on a ray tracer.

```
def singleProbeTrace(ray, probe):
    compute the four 2D polyline segments
    for each polyline segment:
      for each 2D pixel and corresponding 3D point on the segment:
        compare the voxel in the radial distance texture to the ray:
          if hit: return (HIT, point)
          if hidden behind surface: return (UNKNOWN, point)
          # (otherwise, keep iterating)
  return (MISS, last polyline endpoint) # Reached the end of the line
```

4.1 Light Field Ray Tracing Algorithm

The light field ray tracer **selects a probe**, traces within that **single probe**, and then **repeats** that process until a definitive hit or miss.

Selecting a probe. With the exception of simple scenes, any ray origin $\mathbf{r}_o$ inside a scene's bounding box will be contained within a cubic cage of eight light field probes (Figure 3). Due to spatial locality, these probes will have similar scene visibility information, and so they are reasonable candidates to use to test for the ray's intersection with the scene. Of course, this heuristic can fail in scenes with high depth complexity and low probe density, however even sparsely placed probes were sufficient for the scenes we used.

The probe whose center lies closest to the line containing the ray nearly always yields the shortest octahedral projection of the ray[2], so we begin with that probe. Choosing the first probe this way has some additional advantages: first, it will likely minimize the number of texture fetches needed during visibility marching (see below); and, second, this probe is also likely to "observe" the entire unoccluded portion of the ray until its true (i.e., world-space) `hit`.

Note that probe selection only affects performance, not accuracy, except for depth aliasing within a texel and finite precision. That is, any point observed by multiple probes is represented at the same world-space location in all of those probes, and probes that cannot observe the point contain the position of the nearest occluder. The issues are exactly the same for those in the shadow mapping algorithm, which also compares different discrete projections of the same point. So, since *any* probe selection process will lead to same ray-intersection result within available precision, we prefer a heuristically efficient selection.

Single-probe hierarchical tracing routine. A line segment in R^3 maps to a polyline (with *at most* four segments) in R^2, after spherical octahedral projection. Each polyline segment lies on a separate octahedral face, and the 2D coordinates of the segment endpoints are the 2D projections of the intersections of the 3D line with the three principal axis-aligned planes that pass through the probe center. These define eight quadrants that correspond to the octahedral faces, after projection.

To trace a ray *within* a single probe, we compute the 2D polyline endpoints and optimally march along the polyline in the octahedral map. Intersection tests are computed at each texel by depth testing the polyline point against the radial depthmap value stored in the probe at the texel: when the stored radial depth of the scene is less than the radial distance from the probe's center to the 3D polyline at the currently marched polyline texel, the ray has either hit a surface or passed behind the surface.

Within an octahedral face, this tracing routine is a GPU-optimized variant of hierarchical algorithms for voxel and heightfield tracing [Amanatides and Woo 1987; Musgrave et al. 1989; McGuire and Mara 2014]. We use a min-MIP map to hierarchically refine the test. Each octahedral texel describes a planar patch (i.e., surfel)

[2]Because an octahedron is not a sphere, it is possible to construct cases where a probe that is slightly farther away instead meets the criterion.

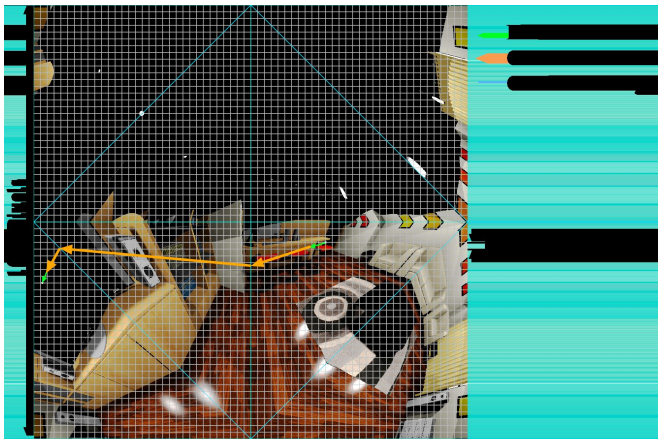

Figure 4: *The single-probe ray trace routine in the living room scene: we visualize the coarsest resolution grid in our two-level hierarchy, octahedral faces after spherical projection, and coarse- (orange) and fine-level (green) ray march tests. Most of visibility tests occur at $1/16^2\times$ resolution (orange). Since the center of the room has no occluders, this ray never changes probes.*

at distance $r_{\mathbf{p}'\leftrightarrow\mathbf{x}''}$ from the probe origin, with normal $\vec{\mathbf{n}}_{\mathbf{x}''}$. We bound that patch by a spherical voxel based on the normal and representation precision. The currently-marched point along the ray $\mathbf{p}'$ intersects the voxel if its bounds at the corresponding texel overlap the voxel's. Using the observed normal $\vec{\mathbf{n}}_{\mathbf{x}''}$ from the octahedral normal maps, we can exclude back face intersections relative to the ray (not the probe center). This back-face discrimination is necessary for scenes with (unrealistically thin) two-sided geometry (e.g., curtains in Sponza).

We visualize debug ray trace output in Figure 4. Here, we see the three-segment polyline of a single ray on the octahedral projection of a probe in the living room scene. We overlay the polyline atop the radiance texture for illustration purposes, however the actual tracing uses the radial distance and normal probe data.

Selecting fall-back probes. If the single-probe hierarchical ray cast is unable to find a definitive `hit` or `miss` event on the previously selected probe, then we must pick a new probe and repeat the process. To do so, we advance the ray origin to the point $\mathbf{r}'_o = \mathbf{r}_o + t'\,\mathbf{r}_d$ where the previous probe trace terminated. If $\mathbf{r}'_o$ lies within a different cube of bounding probes, then we reuse the same selection procedure to pick the next probe from the new eight candidate probes. However, if $\mathbf{r}'_o$ remains within the bounding box of the original eight probes, we heuristically choose the untested probe farthest from the previous one: this heuristic relies on the assumption that moving as far as possible from the previous probe will avoid any locally occluding geometry. Figure 5 visualizes the probe indices used to resolve ray intersections in a typical scene.

We can define a simple and efficient *procedural* iteration sequence that respects this selection heuristic (Figure 3). Without loss of generality, assume that we translate and scale the scene (including the ray) to place the eight probes within the 1 $\mathbf{m}^3$ cube with the origin at the lower corner. The index of the probe with the smallest x-, y-, and z- coordinates is $\lfloor \mathbf{r}_o \rfloor$, and the relative indices of adjacent probes $0 \le i < 8$ are $\lfloor \mathbf{r}_o \rfloor + (i \bmod 2, \lfloor i/2 \rfloor \bmod 2, \lfloor i/4 \rfloor \bmod 2)$.

An exhaustive search of the permutations yields an optimal eight-element probe index selection sequence: namely, one that consistently moves to the farthest probe (without revisiting a probe). The sequence, starting at the relative index i closest to the ray, is given by $i' = (i + 3) \bmod 8$. This selection sequence can be efficiently evaluated with bitwise operations.

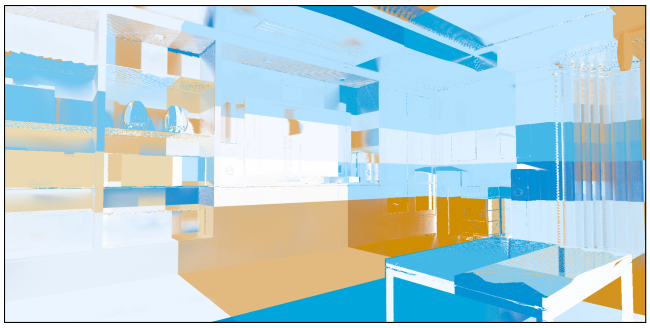

Figure 5: *Living Room color-coded by the probe (matching Figure 1) in which a hit was detected for a glossy ray from that pixel.*

Again, we note that the selection procedure does not affect ray tracing accuracy, only performance. If, after testing against all eight probes we still fail to find a true world-space intersection (i.e., when none of the probes are able to observe the path of the ray behind some densely-scattered, complex occluders), we simply assume that the ray `hit` the occluder at the last observed location.

4.2 Discussion

Our probe selection criterion results in the (nearly) shortest possible 2D polyline, after octahedral projection. This leads to a convenient memory-performance trade-off: reducing the length of the 2D polyline reduces the number marched polyline texels (and, so, the number of radial depth tests). Increasing probe density not only increases accuracy, but also increases performance by decreasing the length of projected rays. In the limit of infinite probe density, an *entire* ray will project onto a *single* texel of a probe, as there will always be a probe with center atop the ray origin and only a single texel will need to be depth tested against in this probe.

Implementation Details. Optimizing any algorithm for high throughput on modern GPUs requires careful management design and optimization: GPU compilers are often not sophisticated enough to perform important parallel processing-sensitive data and computation transformations. Our implementation seeks to minimize peak register count (since increased register usage reduces thread count and parallelism). We also re-arrange branches to increase convergence since, when a warp of threads diverges at a branch the instruction cost is doubled at both sides. Our supplemental code includes a complete optimized GLSL implementation of the tracing algorithm, suitable for use in both pixel and compute shaders.

In practice, we observe optimal performance by fixing a two-level deep min-MIPmap hierarchy for radial depth tests, where the second level is at $1/16^2\times$ resolution. This behavior can be explained by the fact that shader programs cannot use a stack without spilling registers to main (video) memory, and so the asymptotic advantages of using a full hierarchy are overweighed by the large constant-factor cost incurred by accesses to main memory.

5 Shading with Light Field Probes

We now describe three global illumination algorithms, the later two of which are practical: a brute force importance-sampling Monte-Carlo renderer, an more efficient extension that reduces the number of ray samples required by **post**-filtering separate Lambertian and glossy terms, and an extremely fast but biased solution that **pre**-filters Lambertian reflection and only traces rays for glossy reflection (which is then also post-filtered).

A naïve renderer. Given the ray tracing algorithm in Section 4, we can readily implement a naïve, hybrid rasterization/Monte Carlo ray tracing solution to the rendering equation that exploits the GPU.

We rasterize directly visible surfaces and compute direct light using shadow maps (and a deferred renderer) in a pixel shader. Then, in the same pixel shader, we ray trace to estimate indirect lighting using Monte Carlo integration: we trace n random rays according to an importance-sampling distribution, each returns a sampled incident radiance value from the light field, we scale samples by the cosine-weighted bidirectional reflection distribution function (BRDF) and normalize by the sampling distribution and n. The integral will converge for any (valid[3]) sampling distribution.

This naïve algorithm captures indirect effects on both static and dynamic objects, and we could also use it to progressively populate the light field probes' radiance distributions for an arbitrary number of scattering events, by re-rendering into the probes. This approach does, however, have two limitations: viewpoint biasing in higher-order indirect lighting effects, and performance.

Specifically, second- and higher-order indirect lighting (i.e., from iteratively re-rendered probes that rely on the shading computed from previous shading iterations), such as mirror reflections of objects reflected *by other mirrors*, will be viewpoint biased according to probe locations. As this is error is substantially less objectionable than the distorted *first-bounce* reflections that, e.g., state-of-the-art screen-space ray tracing and warped environment maps suffer from, we leave this problem as a potential area of future work.

We instead focus proposing a solution to the more pressing performance issue, especially since we target real-time rendering applications. Any (imperfect) Monte Carlo sampling process will result in variance, which manifests itself as image noise in the case of uncorrelated pixel sampling. Offline renderers address this by increasing the sampling rate n and applying a post-process denoiser, both of which incur minutes of additional frame rendering time. We instead extend our naïve rendering algorithm with approximations and optimizations appropriate to modern real-time rendering pipelines. These necessarily introduce bias, but significantly reduce noise and increase performance to levels that are likely acceptable for many current real-time applications.

5.1 Spatial-Temporal Radiance Denoising

We introduce an optimization suitable for deferred and forward+ renderers: we compute a denoised estimate of indirect incident illumination in a separate pass using a G-buffer, applying the cosine-weighted BRDF and adding the result to direct illumination computed (and stored in a framebuffer) in an earlier pass.

Most BRDFs can be factored into a Lambertian term and one or more narrow lobes or impulses combined by a directionally-varying Fresnel coefficient. With BRDF importance sampling, directions drawn proportional to the Lambertian component tend to produce extremely noisy results when traced because they evenly cover the entire hemisphere of illumination about the surface normal. Perfect mirror reflectors correspond to impulse terms that produce no noise – they result in a single explicit sampling direction. Sampling proportional to narrow glossy lobes, such as from a microfacet BRDF models, produce noisy glossy reflection phenomena but with much less variance than the Lambertian estimate.

We choose to factor the BRDF into two terms and draw samples proportional to these two factors: a uniform **Lambertian** hemispherical distribution, and a **"glossy"** term that can combines perfect specular impulses, retroreflection, and glossy lobes. Our indirect illumination pass traces one ray sampled from each of these

[3] Normalized, with non-zero probability wherever the BRDF is non-zero.

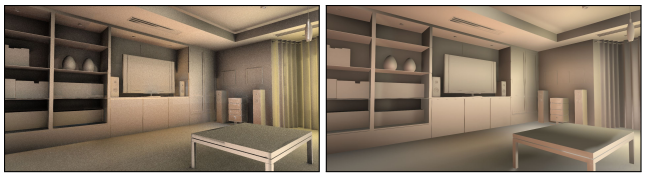

Figure 6: *Left: Incident radiance from one ray per pixel for the Living Room scene. Right: Irradiance result with filtered visibility, directly approximating a denoised result.*

(with cosine weighting) and writes the results to (noisy) Lambertian and glossy *incident* radiance buffers.

For Lambertian reflection, we compute and store integrated *incident* irradiance, instead of albedo-modulated *outgoing* radiance. This allows us to denoise our numerical estimate without blurring or biasing the surface albedo. We denoise this incident irradiance in a second pass with standard cross-bilateral and reprojected temporal filters, using G-buffer normals and depths to measure the difference between surfaces[4]. Here, the benefits of separating Lambertian and glossy terms become apparent: the Lambertian term is often noisier (at equal sampling rate), but also spatially slower-varying. As such, it benefits much more from filtering and admits the use of wider filters. Our results use a 12 pixel radius and 2 pixel stride, and an exponentially-weighted moving average temporal filter (with 98% hysteresis). Since illumination sampled according to the glossy component is less noisy, but also faster-varying in space, we only apply a 3 pixel radius spatial filter and a 75% hysteresis temporal filter (Figure 7).

This approach is similar to standard offline rendering practice [Bitterli et al. 2016] but much lighter weight. With only two indirect rays per pixel, our denoising process allows us to achieve equal-quality results compared images computed using a sampling rate on the order of hundreds of samples per pixel. Denoising produces some reflection blurring (both in space and time), and filter parameters allow the user to trade between noise and blur depending on the needs of their application and the structure in the scene materials.

5.2 Irradiance with (Pre-)Filtered Visibility

Rays that sample the Lambertian reflection capture less information per pixel than glossy samples (i.e., compare the left sides of Figures 7 and 6), since they estimate a function that varies more smoothly in space (except at edges, which the bilateral filter detects). Meanwhile (and ironically), Lambertian rays are *more* expensive to trace due to their incoherence: e.g., Figure 5 shows the coherence of the probes used to resolve glossy `hits`; an analogous visualization for single-sample Lambertian integration would yield

[4]Directly applying the open-source G3D Innovation Engine `TemporalFilter` and `BilateralFilter` implementations.

Figure 7: *Visualizing the glossy indirect buffer. Left: One glossy sample per pixel. Right: After denoising.*

an essential randomly colored image. Note, however, that complex visibility events may introduce "probe-indexing noise" even with the coherent glossy rays; see the bookshelf in Figure 5 – this is a representative example of how our light field probes and ray tracing combine to provide a much more robust solution to, e.g., manually-placed probes and occlusion proxy geometry.

Assuming a reasonable probe density, incident irradiance varies slowly within the eight probe cube volume, except when visibility changes rapidly within the cube. We precompute 128^2 octahedral irradiance maps with a (quasi-)Monte Carlo integral estimate using the radiance maps. At runtime, we estimate incident irradiance at shading points by trilinearly interpolating irradiance maps at the eight probes within the cube of the intersection. This reduces tens-to-hundreds of texture operations that result from exhaustive sampling down to only eight prefiltered fetches at runtime.

One major limitation of previous methods that interpolate incident irradiance this way is that lighting and darkness can leak "through" objects inside the volume of the irradiance probe cube grid (see Figure 13): specifically, when the mutual visibility assumption between a shading point inside the cube volume and one or more probes is invalidated. We introduce a geometry-aware approach for using precomputed incident irradiance that avoids these artifacts by also filtering the geometric data used for ray marching.

While precomputing cosine-filtered irradiance maps, we also compute filtered maps of the radial distance and squared radial distance. These correspond to first and second moments of radial distance. We filter these with a cosine-power lobe to account for aliasing in the discrete distance values while avoding overblurring them. Figure 8 illustrates these maps at three probe locations in the living room scene.

At runtime, in order to compute *geometry-aware* weights for interpolating from surrounding irradiance maps, we combine the trilinear weighing with two additional terms: a clamped dot product of the shading point's surface normal and the direction towards each probe center (i.e., projected area, corresponding to a spatially-smooth backface test), and by a Chebyshev visibility test of the local radiance distance distribution, using the two filtered moments as spherical analogues of a variance shadow map test [Donnelly and Lauritzen 2006] with variance $\sigma^2 = |E[r]^2 - E[r^2]|$. We normalize the result by the sum of all eight weights. See our code supplement for the preferred implementation and filter weights.

Why not spend the texture space from the irradiance probe storage simply computing high-quality irradiance light maps for surfaces? There are two reasons. First, that requires a corresponding high-quality surface parameterization, which is a continual problem in practice. Any such parameterization will still place texels where they cross lighting conditions, such as on a floor half inside and half outside of a wall, unless the scene is itself modeled without any geometric interpenetration, and create interpolation discontinuities wherever there are seams in the texture atlas. Second, precomputed irradiance maps are only useful on static surfaces. Our irradiance probes with prefiltered visibility allow gathering irradiance anywhere in the space of the scene, including on dynamic objects.

6 Results

We render all results at 1920×1080 on a GeForce 1080 with 1024^2 texel light field probe textures and 128^2 texel filtered-visibility irradiance probes, except where noted. Our implementation is built on the G3D [McGuire and Mara 2016] engine using its deferred shading mode and the shaders from our supplement.

Irradiance visualizations. Figure 8 visualizes the prefiltered irradiance and radial distance moments (in lat-long parameterization) in the living room scene. Probe #11 is near the yellow curtain, and

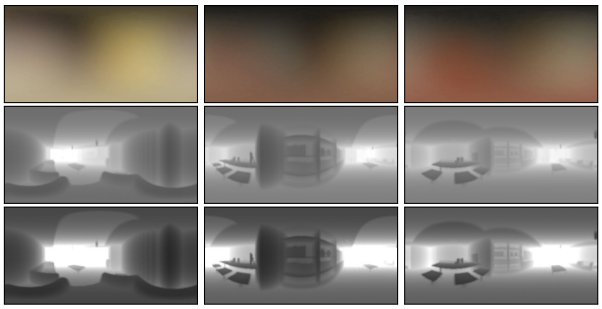

Figure 8: *Columns: Three irradiance + filtered distance probes from the Living Room scene. Rows: Irradiance, filtered radial distance, filtered squared radial distance.*

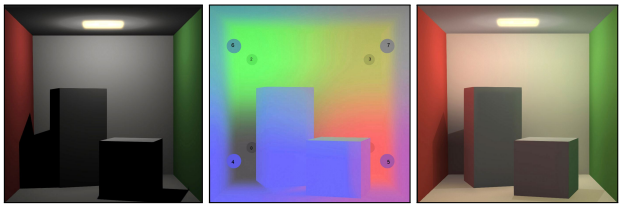

Figure 9: *Cornell Box with direct light, visualization of eight light field probes and their relative contributions (after visibility filtering), and the scene with direct and one-bounce indirect.*

probes #40 and #41 are adjacent to each other, near the red stool and bookcase. The tops of the probes are dark since this example only has a single scattering event, so indirect light has yet to scatter off onto the ceiling. Adjacent probes (#40 & #41) have similar irradiance, confirming its slow spatial variation and the fact that it is a reasonable candidate for interpolation. However, their radial distance maps (affecting visibility) differ significantly, which justifies our extended visibility filtering and testing process.

Figure 9 visualizes the relative interpolation weights for eight irradiance probes in the Cornell Box. The color gradient in the center image is dominated by the trilinear spatial weighing, but changes abruptly with normal for backfaces, and weight "shadows" appear on the floor where boxes occlude the red and lavender probes.

Light leaking. Light and darkness leaks arise in two situations with existing irradiance probes, voxels, and lightmaps: first, when a probe (or texel or voxel) lies *inside* an object, it is completely black and leaks darkness around itself; second, when the probe is in free space but illumination conditions vary rapidly around it (i.e., due to occlusions) then light can leak, e.g., through the wall.

Figure 10 (left) illustrates the case of a probe inside an object with

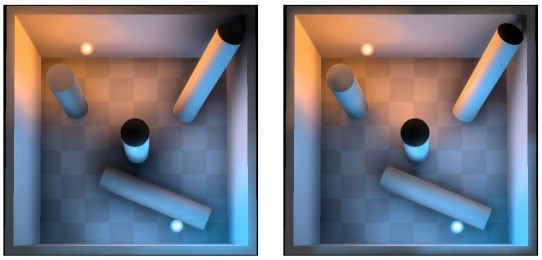

Figure 10: *Left: traditional irradiance probes leak darkness when geometry covers probes, as with the center and upper right cylinders; right: our filtered visibility eliminates these leaks.*

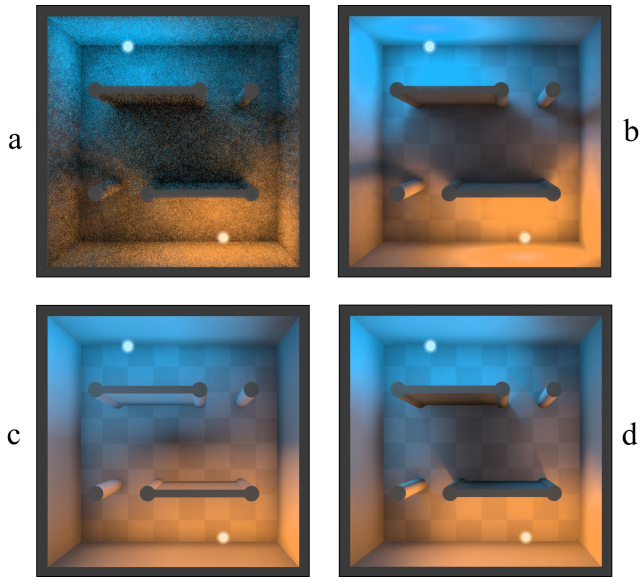

Figure 11: *(a) Our ray-cast indirect illumination [13ms], (b) after spatio-temporal denoising, (c) traditional irradiance maps, and (d) our irradiance sampled with filtered visibility [**0.2 ms**].*

a state of the art irradiance mapping trilinear interpolation scheme. The scene comprises a box with four cylinders, modeled with a $3 \times 2 \times 3$ probe grid. We only visualize the indirect lighting contribution. The center and right cylinders collectively overlap three of these probes, all of which record black (for all directions). Note the pools of darkness at the center and upper right of the left subimage; on the right side of the figure, our prefiltered visibility tests correctly downweights the contribution from these "covered" probes, preventing darkness from leaking into the scene.

Figure 11 shows four versions of indirect lighting in a scene with occluding shapes, modeled with a $3 \times 2 \times 3$ probe grid. The top row visualizes a single-sample global illumination sample per pixel traced into the probe grid, and the denoised result. This results renders soft indirect light shadows without any light leaks around internal walls in 13ms (with filtering). In the bottom row, traditional irradiance probes leak light into the center region, whereas our prefiltered visibility prevents light leaks and casts indirect shadows into the center. The softer indirect shadows from the edges and pillars in (b) are lost with the prefiltered visibility, however this approximation renders without noise about $65\times$ faster, in **0.2ms**.

Figure 12 illustrates light leak prevention in Sponza. Only about 20 pixels are directly lit (the small bright triangle) and traditional irradiance probes (left) leak light from the center of the atrium through the curtain and pillars; our prefiltered visibility (right) correctly blocks the indirect light, except where it should correctly propagate through the gaps beneath and beside the curtains.

Figure 13 further illustrates how, even in geometrically simple scenes, radiometrically complex situations arise where light leaks

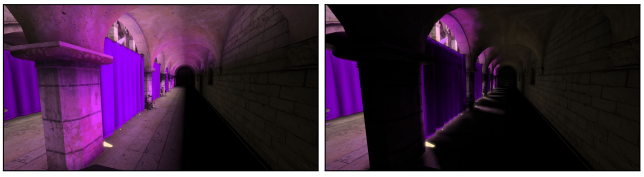

Figure 12: *Left: light leaks from traditional irradiance probes; right: correct indirect shadows using our prefiltered visibility.*

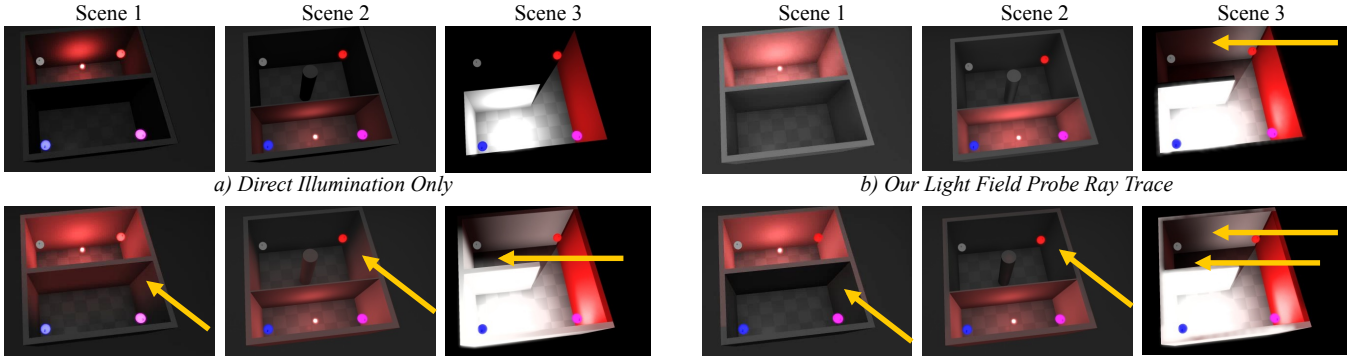

Figure 13: *Avoiding light leaks while capturing interreflections.*

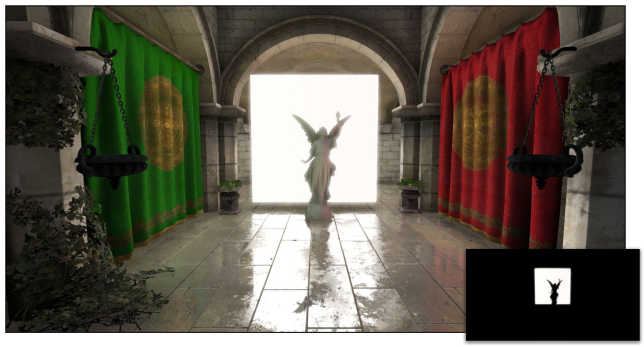

Figure 14: *An area light with correct shadowing, using light field probes. Here, we compute direct illumination with our ray tracer, instead of relying on a deferred renderer with shadow mapping. Incident radiance is exclusively from the emissive square (inset).*

occur using traditional irradiance map interpolation. Here, again, filtered visibility interpolation significantly reduces artifacts and produces results similar to a high-fidelity light field ray-trace.

Complex illumination. Figure 21 (on the last page of this paper) compares direct and full global illumination in scenes with complex materials and geometry. The top row also visualizes probe locations and, in each case, probes were re-rendered using their own previous lighting iterations for indirect illumination until convergence.

An advantage of light field global illumination is that it automatically handles direct light from area light emitters, including soft shadowing, which is not the case with recent analytic methods for glossy reflection of area lights [Lecocq et al. 2016; Heitz et al. 2016]. Figure 14 demonstrates this feature by reproducing the scene from Heitz et al.'s work exclusively using our light field probe ray tracer. We are able to place a statue in front of the area light and correctly shadow its reflection off the glossy floor, as well as reflecting the banners on the sides. There is no explicit direct illumination term in this scene and the only source of light is the emissive square. It is not surprising that this gives a correct result, since it is mathematically equivalent to path tracing with area sources. However, the light field probe data structure and ray tracing algorithm make it possible to employ this practically for the first time in a real-time renderer on consumer hardware, and have running time independent of the number of area light sources in the scene.

Figure 15 shows a challenging case handled well by our method. This kitchen scene is lit by dim direct illumination from overhead lights and a bright sun, however the sun only shines (through a window) on the door in the background. Indirect lighting is the dominant contributor at most pixels. The glossy reflections on the

Figure 15: *The faucet casts a shadow in the glossy reflection of indirect light in this kitchen counter.*

Figure 16: *Three dynamic objects reflect indirect light from the static scene.*

counter top are from indirect light and are correctly blocked by the faucet and teapot, demonstrating high-precision indirect light shadowing.

Since our implementation only generates probes once at scene initialization, we only cast indirect illumination *from* static objects; however, we can cast indirect illumination *onto* both static and dynamic objects at run time, as shown in Figure 16. This is a critical feature for games, as enabled by traditional light probes and voxel cone tracing however, unlike these solutions, light maps and precomputed radiance transfer methods [Sloan et al. 2002] are restricted to lighting only static objects.

Parameters. A single probe can capture the entire light field and visibility for a star-shaped volume, such as the Holodeck in Figure 17. For scenes with more complex visibility, there is a quality tradeoff at fixed storage cost between the number of probes and their resolution. Figures 19 and 18 vary these parameters independently for the Living Room scene. Although there are some subtle interactions with *visibility* aliasing in the bookcase, resolution mainly affects *radiance* aliasing as observed glossy reflections,

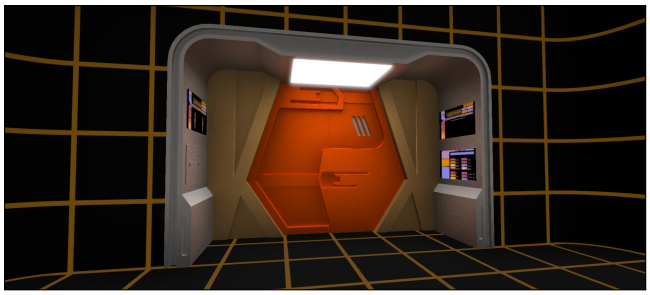

Figure 17: *The Holodeck rendered with direct and indirect light from an area light, with only a single light field probe.*

such as on the television and coffee table. The left-most subfigure uses only 64^2-texel probes; that's comparable to using a 48x48x6-texel environment map, so the reflections obviously show the texel boundaries. We show this to demonstrate how the technique degrades with inappropriate parameters and do not recommend such extreme undersampling. Glossy reflections are insensitive to probe *count*, provided that most surfaces are observed by a probe. Diffuse interreflection (such as on the ceiling and underneath the table), however, can have artifacts when there are too few probes since the prefiltered visibility cannot properly capture local variation.

Performance. Table 1 shows indirect illumination render time for scenes from our paper. Time for evaluating the Lambertian term varies slightly between scenes based on cache behavior arising from how many probes affect visible surfaces. The number of operations per pixel is constant and the glossy term evaluation varies significantly based on the number of surfaces with glossy reflection (e.g., few in Sponza and Cornell Box, nearly all in the other scenes), the visibility complexity of the scene, and the coherence of the glossy rays.

Complex visibility forces the tracing algorithm to frequently change probes along a ray, which incurs a setup cost when recomputing the projected polyline. Incoherent rays cause incoherent memory accesses with poor cache behavior. Both of these occur in San Miguel, but it is also significantly slower than the other scenes because every surface has a dim, broad glossy lobe in the BRDF that requires a wide post-filter. To improve the performance of such a scene for a real-time game, an artist could tune the BRDF to balance performance and appearance or flag low-magnitude glossy lobes as not receiving ray traced global illumination. Furthermore, they might replace the foliage with proxy geometry during light field probe computation to simplify the visibility, as is done today with Geomerics' light map and probe system.

Precomputation of the probes takes 1 to 2 minutes for every scene and is dominated by the cost of rendering the radiance probes at full resolution, which is the product the number of probes and the time to render a single frame (times six cube map faces). The projection to octahedral space and filtering cost only a few seconds more.

Scene	Probes	Resolution	Lambertian	Glossy
Cornell Box	2x2x2	2048^2	0.358	0.000
Cornell Box	2x2x2	256^2	0.302	0.000
Sponza (Statue)	8x2x4	512^2	0.332	0.151
Living Room	4x4x4	1024^2	0.293	4.401
Luxury Kitchen	4x4x4	1024^2	0.358	9.334
San Miguel	8x2x8	1024^2	0.520	47.170

Table 1: *Rendering times for indirect light in milliseconds*

Figure 18: *Light field probe texture resolution primarily affects glossy reflection quality. From left:* 64^2, 256^2, 1024^2 *texels.*

Figure 19: *The number of probes primarily affects diffuse interreflection quality. From left:* $1\times4\times1$, $4\times4\times4$, $8\times4\times8$.

7 Future Work

Light field compression. With compressed textures, we can store 128 probes in San Miguel with 500 MB. Compared to the original scene's 2 GB of compressed geometry and texture maps, this corresponds to only a moderate overhead; however, there's obviously redundant data between probes. Light field-specific compression could exploit this for additional storage reductions [Chang et al. 2006], but implementing it with efficient GPU read operations remains an open problem.

Glossy sample prefiltering. It is tempting to pre-filter the light field probes by constructing MIP-map chains to enable cone tracing, in the same manner as voxel cone tracing. This would eliminate the need for post-filtering glossy term. However, while it is trivial to prefilter radiance maps, the optimal strategy for prefiltering normal and distance maps is nonobvious. Voxel cone tracing produces light and dark leaks due to filtering, and is based on a frequently-invalid assumption that occlusion of multiple surfaces is uncorrelated (exactly analogous to the way that image alpha compositing assumes that coverage locations are uncorrelated between layers). A better filtering strategies is required for voxels, and then for the spherical voxels in light field probes.

Spherical harmonics. Projection into a spherical harmonic basis is one way to represent and then filter low-order functions on the sphere, such as the prefiltered irradiance probes. They have the advantage of a much lower memory footprint than our octahedral probes. For these reasons we are interested in spherical harmonic representation as future work. We did not incorporate them into this work because, spherical harmonics also introduce ringing during reconstruction and require more bandwidth to read. The ringing can create light and dark leaks when it occurs in the depth model. The bandwidth is 2-3x higher. For example, bilinear filtering of an octahedral irradiance probe requires four 64-bit (RGB, Z, Z^2) samples = 256 bits. A 3rd-order spherical harmonic representation

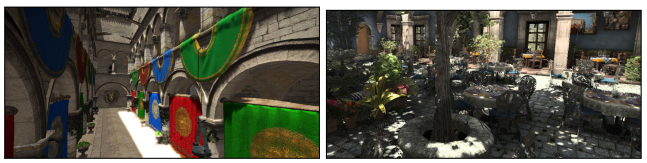

Figure 20: *Primary visibility traced directly from the light field probes. These images were produced with no explicit geometry in memory–just the probes and a full-screen compute shader.*

requires 576 bits, and in practice, 786 bits to extend the depth portion to 4th-order to approximately match our visibility prefiltering cosine-power filter width.

Real-time probe updates. Systems such as Geomerics re-render radiance probes and light maps in real-time using complex scheduling and threading. Our input data are similar, so a similar software engineering approach may be applicable to the light field probe representation to enable real-time updates of the global illumination solution.

Primary rays. The light field probe ray tracer produces unacceptable aliasing when used to render *primary eye rays* (Figure 20), however early results are promising and performance is higher than for glossy rays due to high coherence. Solving primary-ray aliasing would unlock interesting applications for virtual reality, such as directly tracing into the lens-warped projection, foveated rendering by adjusting screen-space sample density and filtering probes, and beam tracing to render each pixel just ahead of scanout.

References

AMANATIDES, J., AND WOO, A. 1987. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, 3–10.

BITTERLI, B., ROUSSELLE, F., MOON, B., IGLESIAS-GUITIÁN, J. A., ADLER, D., MITCHELL, K., JAROSZ, W., AND NOVÁK, J. 2016. Nonlinearly weighted first-order regression for denoising monte carlo renderings. *EGSR 35*, 4 (June).

CHANG, C.-L., ZHU, X., RAMANATHAN, P., AND GIROD, B. 2006. Light field compression using disparity-compensated lifting and shape adaptation. *Trans. Img. Proc. 15*, 4, 793–806.

CIGOLLE, Z. H., DONOW, S., EVANGELAKOS, D., MARA, M., MCGUIRE, M., AND MEYER, Q. 2014. A survey of efficient representations for independent unit vectors. *JCGT 3*, 2, 1–30.

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., AND EISEMANN, E. 2011. Interactive indirect illumination using voxel cone tracing: A preview. In *I3D*, ACM, 207–207.

DEBEVEC, P. 2006. Image-based lighting. In *SIGGRAPH Courses*, ACM.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *I3D*, ACM, 161–165.

DONOW, S., 2016. Light probe selection algorithms for real-time rendering of light fields.

EVANGELAKOS, D., 2015. A light field representation for real time global illumination.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *SIGGRAPH*, ACM, 43–54.

HEITZ, E., DUPUY, J., HILL, S., AND NEUBELT, D. 2016. Real-time polygonal-light shading with linearly transformed cosines. *ToG 35*, 4 (July), 41:1–41:8.

HOOKER, J., 2016. Volumetric global illumination at treyarch, August. SIGGRAPH Advances in Real-Time Rendering Course.

JENDERSIE, J., KURI, D., AND GROSCH, T. 2016. Precomputed illuminance composition for real-time global illumination. In *I3D*, ACM, 129–137.

LECOCQ, P., SOURIMANT, G., AND MARVIE, J.-E. 2016. Accurate analytic approximations for real-time specular area lighting. In *I3D*, ACM.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *SIGGRAPH*, ACM, 31–42.

MARTIN, S., AND EINARSSON, P., 2010. A real time radiosity architecture for video games, August. SIGGRAPH Advances in Real-Time Rendering Course.

MCGUIRE, M., AND MARA, M. 2014. Efficient GPU screen-space ray tracing. *JCGT 3*, 4 (Dec.), 73–85.

MCGUIRE, M., AND MARA, M., 2016. The G3D innovation engine. http://g3d.cs.williams.edu/.

MICKAEL GILABERT, N. S. 2012. Deferred radiance transfer volumes. GDC.

MUSGRAVE, F. K., KOLB, C. E., AND MACE, R. S. 1989. The synthesis and rendering of eroded fractal terrains. In *SIGGRAPH*, ACM, 41–50.

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. *ToG 27*, 5 (Dec.), 129:1–129:8.

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *I3D*, ACM, 75–82.

SÉBASTIEN, L., AND ZANUTTINI, A. 2012. Local image-based lighting with parallax-corrected cubemaps. In *SIGGRAPH Talks*, ACM, 36:1–36:1.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM ToG*.

STACHOWIAK, T., 2015. Stochastic screen-space reflections. SIGGRAPH Advances in Real-Time Rendering in Games course.

TOTH, R., HASSELGREN, J., AND AKENINE-MÖLLER, T. 2015. Perception of highlight disparity at a distance in consumer head-mounted displays. In *HPG*, ACM, 61–66.

VALIENT, M., 2014. Taking killzone shadow fall image quality into the next generation. GDC.

WYMAN, C. 2005. An approximate image-space approach for interactive refraction. In *SIGGRAPH*, ACM, 1050–1053.

Figure 21: *Left: Direct illumination only. Right: Global illumination with iteratively-rendered probes.*