Fast Non-uniform Radiance Probe Placement and Tracing

Yue Wang
McGill University

Soufiane Khiat
Ubisoft

Paul G. Kry
McGill University

Derek Nowrouzezahrai
McGill University

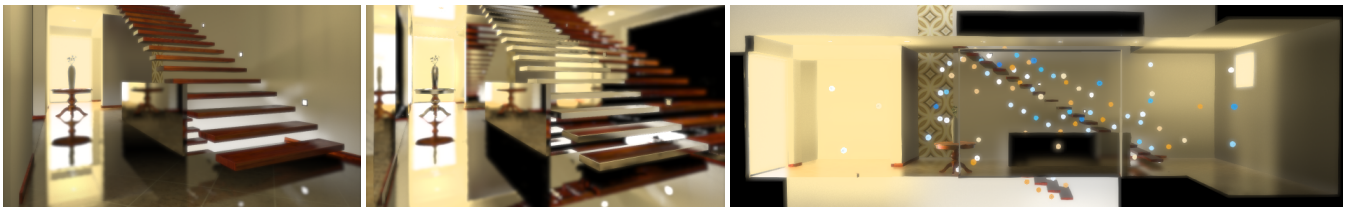

Figure 1: Modern Hall – Global illumination, glossy indirect shading and a side view of non-uniform visibility-aware placement (108 probes). Shading (17.2ms) is 20% faster and has 50% less error than [McGuire et al. 2017], at equal probe count.

ABSTRACT

Light field probes extend standard precomputed light probes to reduce light leaks and enable efficient filtered world-space ray tracing queries. When probes are placed uniformly in the scene volume, they permit an efficient querying algorithm. Manually increasing the grid resolution, however, is the only way to eliminate geometric feature undersampling, increasing the memory and computation cost of the approach. We present an automatic non-uniform probe placement method to correctly sample visibility information and eliminate superfluous probes. We organize non-uniform probes in an efficient structure for fast run-time ray tracing. Our probe placement relies on 3D scene skeletons and a gradient descent-based refinement to achieve full geometric coverage and reduce grazing angle sampling biases. Our adaptive probe ray tracer caches visibility information in a sparse voxel octree, augmenting probes with metadata used to apply a hierarchical-Z acceleration when marching rays in distant probes. We benchmark our approach on a variety of scenes and consistently demonstrate better performance, and fewer probes, in equal-quality comparisons to the state-of-the-art.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Ray tracing**.

KEYWORDS

Interactive rendering, global illumination, radiance probes

ACM Reference Format:
Yue Wang, Soufiane Khiat, Paul G. Kry, and Derek Nowrouzezahrai. 2019. Fast Non-uniform Radiance Probe Placement and Tracing. In *Symposium on Interactive 3D Graphics and Games (I3D '19), May 21–23, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3306131.3317024

I3D '19, May 21–23, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6310-5/19/05…$15.00
https://doi.org/10.1145/3306131.3317024

1 INTRODUCTION

Interactive global illumination remains a longstanding problem in computer graphics. Modern video game solutions trade between the types of supported shading effects, memory costs and run-time constraints. Most industrial rendering engines rely on some form of *light probe* data to accelerate run-time global illumination.

Light field probes resolve many of the limitations of existing probe solutions, reducing light leaking artifacts and enabling efficient filtered world-space ray-tracing [McGuire et al. 2017]. Here, probes are placed using a regular 3D grid to allow for a straightforward and fast run-time probe sampling algorithm. Despite this, regular grid placement suffers from two important limitations. First, they do not adapt to variations in scene geometry or lighting complexity, where higher sampling density may be needed to capture important light field signal variations. Secondly, in virtual environments that combine large open regions (e.g., outdoor) with finer-scale entities (e.g., buildings and rooms with clutter geometry), uniform probe placement fails: placing probes so as to guarantee proper geometric coverage will result in oversampling, undersampling, and wasted probes that either fall *inside* objects or that duplicate information stored in other probes.

We present a non-uniform probe placement method that samples geometric and lighting features at appropriate spatial resolutions. We additionally propose an efficient run-time ray querying algorithm that operates with non-uniform probes. Figure 1 shows a preview of our results. Concretely, our contributions are as follows:

- automatic non-uniform placement that minimizes the number of probes needed to capture visible scene geometry,
- fast GPU ray queries using a novel caching structure, and
- a distance data augmentation for hierarchical ray traversal.

2 RELATED WORK

Work on interactive global illumination spans nearly two decades. We discuss the prior art most relevant to our method. One key differentiating factor to consider when discussing our goals in the context of prior art is that we not only store (ir)radiance in our probes, but also geometric information like visible surface normals and radial distances. These signals, and the manner in which they need to be sampled for accurate ray-queries at runtime, differ from traditional (ir)radiance-only probe data. Here, we require

full geometric visibility coverage (i.e., the ensemble of our probes should sample every scene point *at least* once).

Light Probes. Greger et al.'s [1998] seminal *irradiance volumes* work cached (discretized) spherical irradiance distributions in a spatial grid, allowing diffuse dynamic objects to be lit by their surrounding environment at interactive rates. While self-occlusion and lighting from dynamic objects back onto the scene were not accounted for, this representation is powerful and variants extend the form, format, and use of these spatial-angular irradiance distributions [Tatarchuk 2005]. Among these, spherical harmonics (SH) prove to be an efficient representation for the angular distributions [Ramamoorthi and Hanrahan 2001], paving the way for precomputed radiance transfer (PRT) methods that allow for a certain degree of (low-frequency) self-occlusion [Sloan et al. 2002]. Jendersie et al. [2016] use a dynamically relit hierarchical surfel scene representation to support dynamic object relighting. However, scaling dynamic solutions to complex scenes remains an open problem. Silvennoinen and Lehtinen [2017] combine ideas from light probes and PRT, factorizing the light transport matrix into global and local reflection components, then placing sparse radiance probes to reconstruct smooth indirect lighting.

We too target sparse (and adaptive) spatial placement of angular probes, although we rely on *light field probe* distributions which account for scene geometry [McGuire et al. 2017]. McGuire et al.'s method uses inter-probe marching to enable efficient shader-level filtered ray tracing and irradiance queries. At runtime, shading can be computed using both Monte Carlo sampling-based methods and visibility-aware irradiance probe queries that are robust to light leaking artifacts. The efficiency of their tracing and querying algorithms rely fundamentally on a regular grid probe placement.

Regular grid probe placement poses several problems. At low-resolution, sampling complex scene geometries can be difficult, and increasing grid resolution does not necessarily solve the problem. Furthermore, using a conservatively high resolution can be doubly wasteful: since typical interactive graphics and gaming environments combine large expanses with densely-populated building/room geometry, regular grid probe placement can simultaneously oversample open areas, undersample regions with denser fine-scale geometry, and inevitably places probes *inside* objects (partially or fully). This wastes memory and also leads to reconstruction artifacts at runtime.

We propose a non-uniform light field probe placement algorithm that guarantees geometric coverage, and we design new tracing and visibility-aware irradiance querying algorithms to support this non-uniformity. We show that, for equal quality, our method consistently requires both fewer probes *and* achieves faster run-time shading.

Non-Uniform Probe Placement. Non-uniform probe placement methods vary on the data they store at probe locations, and how they use it for shading at runtime. Most methods either rely on distance-, visibility- and/or user-controlled heuristics, or on lifting and merging schemes that operate from a pre-initialized probe set.

Greger et al. [1998] use a bi-level regular grid, placing irradiance probes on a coarse grid before subdividing cells that contain geometry. This approach suffers from the same sampling issues as regular grids, including light leaking and darkening. Robustness to the darkening artifacts, caused by probes placed *inside* objects, can be improved by casting rays during probe placement [Gilabert and Stefanov 2012], and a hierarchical regular grid (with > 2 levels) can also be employed [Stefanov 2016]. Still, the regular structure of probe positions provides no assurance that geometric features are sufficiently sampled. Our method resolves mutual visibility, placing probes adaptively along the *geometric skeleton* of a scene's volume.

Cupisz [2012] uses a Delaunay tetrahedralization of scene volumes to place and interpolate non-uniform irradiance probes. In our initial experiments, the tetrahedron barycentric coordinate nearest probe search is too costly to perform (e.g., per ray) at runtime, but a simpler Voronoi-based representation with augmented neighborhood information proved useful.

Tatarchuk [2005] adaptively sample irradiance volume probes starting from a dense uniform sampling, pruning and injecting probes based on a numerical estimate of the spatial gradient of irradiance. Bowald [2016] extends this idea with a binary bit mask that accounts for mutual probe visibility during the gradient computation.

Chajdas et al. [2011] place environment map probes according to local variations in incident lighting intensity, color and directionality. This method begins with a dense uniform grid sampling, pruning using a similarity metric based on a discretized irradiance gradient field. While this typically results in better coverage than naïve uniform placement, there is no guarantee regarding scene geometry coverage, and the gradient field tends to degenerate by driving probes towards light sources. To improve geometric coverage, Silvennoinen and Timonen [2015] also begin with a uniform grid of probes, then retain a subset of these probes based on visible surface determination and distance-to-surface metrics. We also rely on probe-surface visibility, using a visibility atlas to more robustly treat complex indoor scene environments. Silvennoinen and Lehtinen [2017] apply a greedy placement strategy: after voxelizing the scene, probes are placed only in voxels that occupy open space. A heuristic based on visibility and probe influence radii is used to merge this initial dense probe set into a sparser one. While this serves to avoid placing probes *inside* objects (something our placement algorithm also achieves), the heuristic tends to favor placing probes close to geometry. Such placement is undesirable for our ray tracing needs, since angular distortion of the geometric information stored in the probes would lead to instabilities in the tracing algorithm.

3 ADAPTIVE PROBE PLACEMENT

We present an automatic method to place the *fewest number of* light field probes needed to adequately sample the geometric details of a scene. Our approach comprises two stages: first, we compute the *geometric skeleton* of a scene, using an intermediate medial axis or signed distance representation (Section 3.1); secondly, after placing candidate probes using the skeleton, we optimize their placement using visibility- and sampling-based metrics designed to improve the robustness of runtime (ir)radiance queries and ray-tracing using the geometric probe data (Sections 3.2 and 3.3).

3.1 Visibility-aware Probe Placement

The *straight skeleton* is a representation of a polygonal scene that resembles a piecewise linear approximation of the medial axis. We

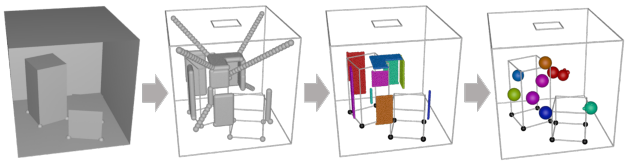

Figure 2: Left to right: we construct an SDF of the Cornell Box, apply voxel culling/clustering to obtain an approximate skeleton, then cluster skeleton elements before placing probes on the clusters' centroids.

arrive at a scene's straight skeleton by sweeping polygonal edges inwards towards each other, forming skeleton elements as the set of non-adjacent swept polygonal edge intersections.

A 3D straight skeleton comprises points, line segments and planes, and its applications vary from vertex placement in graph drawing algorithms [Bagheri and Razzazi 2012] to automated engineering and architectural design [Kelly et al. 2017].

We can exploit an important property of a straight skeleton: each of its elements defines a region of *constant scene geometry visibility information*. As such, placing probes along every straight skeleton element is an automatic way to guarantee full scene visibility. Moreover, doing so in a bijective fashion guarantees full visibility with a minimal number of probes.

Despite this seemingly straightforward solution, straight skeletons are difficult to compute in practice: for scenes with general 3D assets, 3D straight skeleton computation can be numerically unstable, with many edge cases and robustness issues. Fortunately, by relaxing the requirement of a *perfect* straight skeleton, we can apply fast approximate skeleton construction algorithms that meet our practical needs and are suitable for use with real-world digital assets. We present two such approaches: signed distance function-based and medial axis-based representations.

Signed Distance Function Skeletonization. The signed distance function (SDF) of a scene maps 3D points to distances: every point in space is mapped to the shortest distance (from the point) to a point on the surface of the scene. The distance is signed, with positive values for points outside the scene volume and negative values for points inside. Algorithms for constructing SDFs from general meshes are well understood, and we adopt the openly-available OpenVDB library to compute scene SDFs. After building a discrete SDF, we exclude voxels outside the scene and those with small distance values (i.e., regions close to scene surfaces). The remaining voxels are clustered to form an approximate skeleton topology for the scene (Figure 2). The average position of each cluster is used as an initial candidate for individual probe placement.

Ball Shrinking Skeletonization. We use *ball shrinking* to compute the scene's medial axis [Ma et al. 2012]. It operates on a point-based scene input, first placing spheres of (sufficiently large, user-prescribed) radii r tangent to each point p (centered at $p - r\overrightarrow{n_p}$) and progressively shrinking them until a maximal inscription criterion is met. The centers of the resulting sphere-set are then taken as the medial axis vertices (and probe locations). We initialize the point-based input with the vertices of a sufficiently fine tessellation

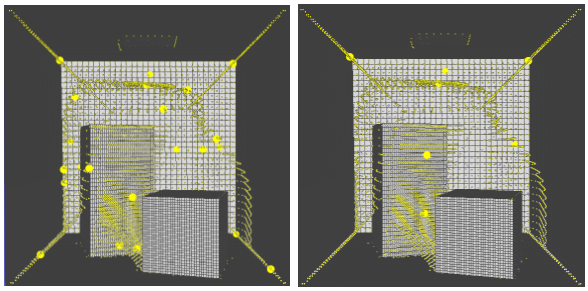

Figure 3: Left to right: decreasing probe density by preclustering vertices during medial axis construction.

of the scene, and control probe density by pre-clustering vertices according to position-orientation similarity (Figure 3).

3.2 Position Optimization Metrics

After skeleton-based probe placement, we can apply an optional optimization pass to refine probe locations. We define several optimization metrics below, and optimize the probe placement according to these objectives using gradient descent (Section 3.3).

Visibility. We define an objective to measure how much of the scene surfaces are visible from a set of probes. To do so, we map binary surface visibility (from each probe's octahedral directional map) to the scene's texture atlas, parameterized by texels (i, j) (see Figure 4). This binary measure is maximized when the *union of per-probe visibility atlases* $\cup_k V^k_{i,j}$ fully cover the scene.

Probe View Distortion. In addition to optimizing visibility, it is also important to consider the projected solid angle distortion at each probe imposed by octahedral directional maps. Specifically, if a probe's view of a surface points occurs at grazing angles, ray-tracing to these points using the probe may suffer from numerical imprecision. To account for this, we preferentially weight surface points (visible from a probe) according to a term proportional to their subtended solid angle $\overline{\Omega}_k = -\omega \cdot \overrightarrow{n}_{p(\omega)}$, where ω is the view direction for a texel in the octahedral map of the k^{th} probe, and $\overrightarrow{n}_{p(\omega)}$ is the normal of surface point visible to the probe at direction ω.

Distance to Surface. Optimizing for visibility and relative view orientation can lead to local maxima, where probes settle into regions very close to object surfaces. To address this, we can promote global scene coverage by maximizing the sum of distances to all

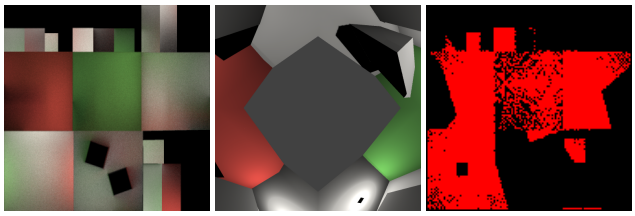

Figure 4: Left to right: texture atlas for the Cornell box; radiance texture for a single probe; mapping of the points visible from this probe, onto the atlas.

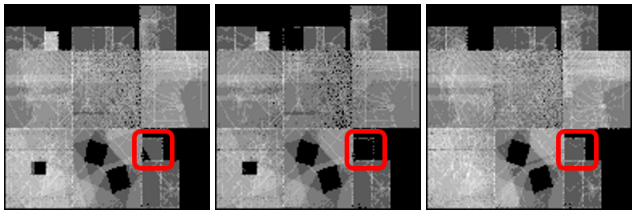

Figure 5: Global visibility atlas. Left: all probes accumulated, one corner on the bottom right is not visible to any probes; Middle: all except current testing probe, the surface in the red square is not covered; Right: all probes accumulated after 100 iterations of Gibbs sampling, full coverage is achieved

visible surfaces of the k^{th} probe, $\overline{d}_k = \sum_\omega |\overline{p} - p(\omega)|$, where $\overline{p}$ is the probe location we will optimize (in Section 3.3).

3.3 Gradient Descent Optimization

We define the fitness of a *probe set proposal* $P = (p_1, p_2, ..., p_n)$ as

$$R(P) = \sum_{i,j} \pi_{i,j}, \tag{1}$$

where $\pi_{i,j}$ is the fitness associated to the $(i,j)^{\text{th}}$ texel in the our global fitness atlas, which comprises a weighted combination of the individual fitness metrics (described in Section 3.2) as

$$\pi_{i,j} = \sum_{k=1}^{n} \left(w_1 \, \overline{\Omega}_k + w_2 \, \overline{d}_k \right) \, V_{i,j}^k \,, \tag{2}$$

where weights w_1 and w_2 scale the relative importance of the orientation and distance metrics, and we have n probes.

When evaluating Equation 1 during optimization (and the individual terms in Equation 2), we subsample view directions ω at each probe (instead of every octahedral direction). We update our probe set proposal by incrementally updating individual probe locations, in sequence. When updating probe k's location p_k to p'_k, yielding a new proposal $P' = (p_1, \ldots, p'_k, \ldots, p_n)$, we compute a gradient-based update as

$$p'_k = p_k + \gamma \, \nabla_k R, \tag{3}$$

where γ is a step size (see below), and $\nabla_k R \approx (\partial / \partial p_k) R$ is an approximated gradient of the fitness with respect to the candidate probe's current position p_k. Observing that traditional finite differences performs poorly, we based this approximated gradient on sampling eight directions on the corners of a cube with the current probe at the center. We compute the sum of the visibility atlas as a new reward for each support probe, and then move the probe position in the direction of the support probe that gives us largest reward improvement.

Setting the *initial* step size to the support probe grid diagonal is reasonable but, after many iterative optimization updates, oscillatory behavior can emerge, with some probes moving back and forth between two positions. Furthermore, performing probe updates sequentially and locally (i.e., without global knowledge of other probe updates) will not guarantee convergence to a global maximum. We apply two strategies during optimization to improve exploration: Gibbs sampling and stochastic direction sampling.

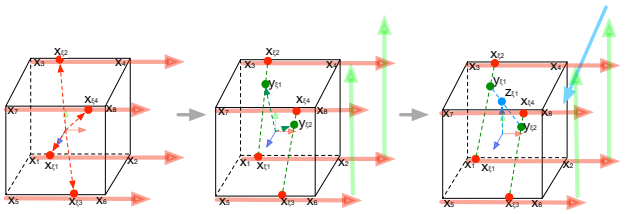

Figure 6: Stochastic direction sampling. Left to right: we construct a stochastic probe update direction by sampling points $x_{\xi,i}$ ($i \in [1,4]$) proportional to a linear model of expected fitness along the x-adjacent support probe grid, then repeating for y-points $y_{\xi,j}$ ($j \in [1,2]$) and z-point $z_{\xi,1}$.

Gibbs Sampling. Instead of accepting each new proposal probe set P', we perform an unbiased stochastic update. We accept a new proposal if $\xi < R(P')/R(P)$, where ξ on $[0,1]$ is a uniform canonical random number. If each P' only updates a single probe and each ξ is drawn independent and identically distributed (i.i.d.), this stochastic process is guaranteed to generate samples proportional to the steady-state fixed-point R distribution.

Stochastic Direction Sampling. Instead of computing the position update vector $\nabla_k R$ using the fixed support probe grid locations p_k^s, we perform a stochastic direction sampling to reduce coherent structure in the exploration. Our stochastic direction sampling proceeds incrementally along each canonical axis and assumes that the distribution of R along any axis follows a (locally) linear model.

Starting with the x-axis, we pair the *four* x-adjacent support probes on the corners of the support *cube*. We fit four linear models $f(x) = ax + b$ to the four fitness pairs $[R(P'_s), R(P'_{s+1})]$ (for $s \in \{1,3,5,7\}$) obtained by substituting p_k in P with p_k^s in P'_s. After solving for a and b, we can draw a sample for the x-component of $\nabla_k R$ proportional to each of the $f(x)$, using the inversion method: the CDF of f is $F(x) = (a/2)x^2 + bx$ and its inverse is $F^{-1} = (-b \pm \sqrt{b^2 - 2ax})/a$. Thus, evaluating $F^{-1}(\xi_i)$ at four i.i.d. canonical random numbers ξ_i yields a sample point along each of the four x-adjacent support probe edges (left of Figure 6).

We repeat this process along the y-axis next, now considering the *two* y-adjacent support probes on the corners of the (x-presampled) support *square*, to sample *two* y points. Finally, we repeat along the z-axis, considering the *only remaining* z-adjacent support probes on the endpoints of the (xy-presampled) support *line*.

This process yields a randomized gradient-step direction $\nabla_k R$ that can be substituted into the standard gradient descent process but that additionally considers the local variation of R around p_k. We also adjust our Gibbs sampling procedure to simply consider this stochastic direction-sampled proposal instead of the one we would have picked using the fixed support grid. Stochastic direction sampling improves global scene visibility coverage. Our supplemental video illustrates its subtle yet important effect on probe placement.

4 SHADING WITH NON-UNIFORM PROBES

We first explored a Voronoi-based approach to select non-uniform probes for shading, inspired by Cupisz [2012]. Our implementation was no faster than the original algorithm [McGuire et al. 2017]. We instead propose a sparse voxel octree-based (SVO) approach.

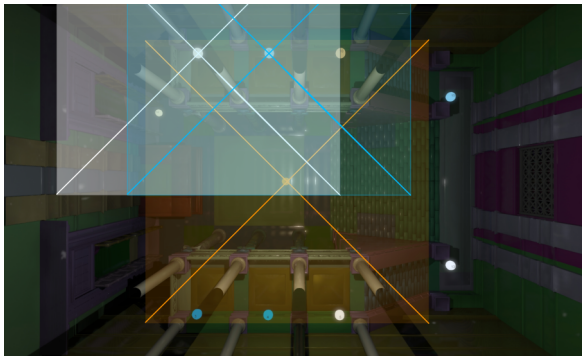

Figure 7: Top view of influence regions. The top left pillar of the temple is covered by an orange probe and a white probe.

4.1 Baking Probes to Sparse Voxel Octrees

SVO representations are widely used to accelerate (potentially filtered) geometric queries. We encode probe indices into an SVO in order to efficiently determine the subset of probes best suited for shading, given any world-space position in the scene. Unlike traditional SVOs [Crassin and Green 2012], we only populate leaf nodes (with probe indices) and leave interior nodes empty, for two reasons: first, any failed ray intersection due to a backface hit will need to recursively query the tree at the backface hit position to re-select a new probe (and, so, only baking probes at geometric boundaries is insufficient); second, dynamically-added geometry can appear anywhere in the scene, not just close to existing surfaces.

We associate each probe with a *region of influence* according to its visible geometry [Sébastien and Zanuttini 2012]: the influence distance for each face of a probe's bounding cube is set as the median of the probe's sampled radial distances (Figure 7).

As our placement strategy ensures full visibility coverage, every voxel is visible to at least one probe (hence, every voxel will be assigned at least one probe). Influence volumes for each probe often overlap; we conservatively store up to four probes in each voxel.

When selecting a probe for shading from a voxel, we can reuse the placement metric terms in Equation 2. For probe placement we needed to avoid placing probes inside objects, and so we favored larger probe-to-surface distances; for shading, however, we know that probes are not inside objects and so we simply weigh the four stored probes according to their projected solid angle-weighted visibility: $(\overline{\Omega_k} \,/\, \overline{d}_k) V^k_{i,j}$ for $k \in [1, 4]$ and where (i, j) are evaluated according to the runtime ray query. Since we consider mutual visibility during probe selection, the graph of probes a ray can visit before terminating at a valid intersection does not have any cycles: each ray-trace is guaranteed to complete (or return no intersection) without looping over the same probe more than once.

4.2 Hi-Z Ray Tracing in Probe Textures

McGuire et al. [2017] use a bi-scale ray marching algorithm: a query ray is first marched using a lower-resolution MIP of the radial distance probe data, only continuing on to a full-resolution march if the low-res march returned a hit.

We extend this approach to fully hierarchical traversal, inspired by recent hierarchical-Z ray tracing work [Uludag 2014]: since octahedral maps are piecewise linear, we can perform hi-Z probe ray marching along individual piecewise linear ray segments (projected onto the octahedral map).

We need to take care when constructing the MIP-hierarchy: directly mipmapping the octahedral textures yields incorrect distortion. We instead build probe cubemap mipmaps and separately resample each mip-level into an octahedral map.

Our hierarchical-Z ray marching approach uses fewer steps on average compared to bi-scale marching [McGuire et al. 2017], resulting in at least a 30% performance improvement (in equal-quality comparisons; see Section 5). We conservatively avoid the coarsest levels of the hierarchy to avoid numerical edge cases.

5 IMPLEMENTATION DETAILS AND RESULTS

We evaluate our method on scenes of varying geometric and radiometric complexity. We compare quality, performance and memory requirements of our placement and tracing against the method of McGuire et al. [2017]. We implemented our algorithm in the same G3D software engine [McGuire and Mara 2017] as McGuire et al. [2017], benchmarking on a 3.6 GHz Intel Xeon E5-1650 v4 CPU PC with 64GB of RAM and an NVIDIA GeForce GTX 1060 GPU. In all cases, direct illumination is computed using shadow mapping in a deferred renderer, and our light field probes are used to compute glossy indirect illumination. We focus on glossy indirect, since diffuse indirect varies more smoothly. The direct lighting radiance stored in probes is computed using the same direct illumination renderer. Multiple bounces of indirect lighting can be computed by iteratively re-injecting indirect illumination radiance into the probes. We always employ a probe resolution of 512×512.

We build the SVO from static scene geometry prior to baking probe indices (Section 4), and so visibility information is bounded by the static scene bounds. This serves the additional benefit of bounding probe influence to the scene boundaries. We populate SVO leaves with probe indices using a compute shader pass (Section 4.1), with one thread per node: each thread first culls probes that are not visible from current voxel, before sorting the remaining probes based on their projected solid-angle weighted visibility.

5.1 Quality Comparisons

We compare non-uniform vs. uniform probe placement under *equal-quality* and *equal probe count* scenarios. We compute error in the glossy indirect component (the effect by far most affected by probe placement). We compute ground truth with a Monte Carlo path tracer, measuring image mean squared error (MSE). We generally demonstrate the highest quality one could hope to attain with uniform grids, using impractically high grid resolutions.

Equal Quality – Varying Probe Placement. Figure 8 illustrates a variant of the Cornell Box: both boxes and the room walls are glossy. The first row visualizes probe placement: non-uniform placement uses seven probes (one probe is hidden by the tall box) generated using our SDF placement (Section 3.1). Even in this relatively simple scene, regular grid placement suffers from reflection artifacts (e.g., on the tall box, illustrated in the second row). Non-uniform placement generates artifact-free reflections at twice the performance.

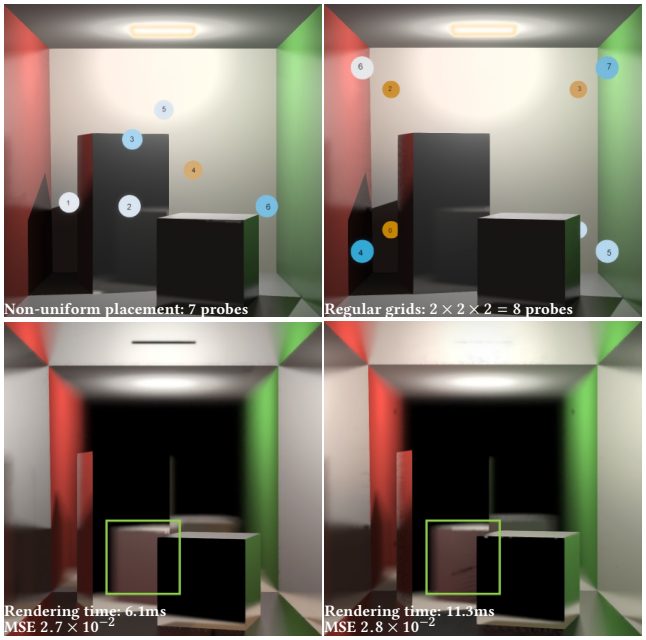

Figure 8: Comparison of non-uniform and regular grid in final rendering results (first row) and filtered glossy components (second row) of the Cornell box.

Figure 10 illustrates results in the Iconic Temple, a scene with spatially-varying geometric complexity. This scene further illustrates the need (and utility) of non-uniform probe placement. For example, a *hand-optimized* $4 \times 2 \times 4$ regular grid still results in eight probes occluded under the stairs and in objects. Each cylindrical pillar requires at least three probes to obtain full visibility coverage, and our placement automatically arrived at such a solution using SDF placement (see Figure 7). Comparing the indirect glossy rendering component, our non-uniform placement correctly captures glossy reflections of, e.g., pillars off walls and walls off the floor (despite using *half* the number of probes compared to a $4 \times 2 \times 4$ regular grid). In all cases, our probes are rendered faster than their (roughly) equal-error uniform grid counterparts.

We observe similar results in a modification of the Crytek Sponza scene used by McGuire et al. [2017]. We place 54 to 84 non-uniform probes using the medial axis approach (Section 3.1). Figure 11 compare our results with 54 and 64 probes to three regular grid resolution settings, illustrating probe placement (top row), rendered results (middle row) and rendering error map (bottom row). Once again, the regular structure of the grid precludes adequate sampling of fine- and coarse-scale visibility relationship around the statue and behind the curtains. To achieve full coverage and good rendering result with no apparent artifacts, an $8 \times 4 \times 4$ regular grid is needed (i.e., compare columns 1 and 2 of Figure 11).

Once again, our approach generates *lower error* results with *fewer probes*, all with *faster rendering performance*.

Equal Probe Count. Figures 10 and 11 columns 3 and 4 also illustrate equal probe count comparisons in the Iconic Temple and modified Crytec Sponza scenes.

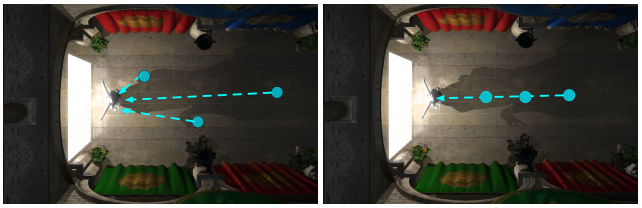

Figure 9: Non-uniform probe placement improves view variation, e.g., when sampling the statue geometry. Uniform probes duplicate the same view towards the statue.

In the Iconic Temple, it is challenging to limit the number of grid-based probes to match our non-uniform placement while still achieving good coverage, especially with the irregularity introduced by, e.g., the stairs. Indeed, even our best hand-tuned uniform grid places four probes under the stairs (making them useless), and every (reasonable) uniform placement we tried fails to reach full surface visibility coverage along the center corridor, leading to shading artifacts due to undersampling.

In the modified Crytech Sponza scene, uniform grid placement cannot generate equal quality renderings at equal probe count (see, e.g., 6×3×3 vs. our 54-probe placement). At least four vertical layers are needed to fully cover the scene geometry with uniform grids. In addition to undersampling artifacts on the statue, we observe artifacts due to thickness and grazing angles undersampling in the $6 \times 3 \times 3$ result. Our non-uniform placement alleviates these problems, reducing the uniform structure in the final probe set (Section 3). Despite using only one vertical layer of probes in the corridor, our placement still samples a sufficiently diverse set of views towards the statue and area light (see Figure 9).

5.2 Performance

Table 1 summarizes our performance benchmarks for rendering indirect glossy illumination. Every scene is rendered at a resolution of 1900×1080, and to normalize across the number of glossy pixels, we render *all* surfaces as glossy for the purpose of benchmarking.

Table 1: Performance comparison (measured in milliseconds) of different placement methods.

Scene	Probes	Tracing Algo.	Performance
Cornell Box	$2 \times 2 \times 2$ grid	Original	11.3
	$2 \times 2 \times 2$ grid	Hi-Z	8.2
	7 non-uniform	Original	7.8
	7 non-uniform	Hi-Z	**6.1**
Iconic Temple	$2 \times 2 \times 4$ grid	Original	32.5
	$4 \times 2 \times 4$ grid	Original	27.7
	16 non-uniform	Hi-Z	**17.3**
Modern Hall	$4 \times 4 \times 8$ grid	Original	24.1
	50 non-uniform	Original	17.8
	50 non-uniform	Hi-Z	**14.50**
Crytek Sponza	$8 \times 4 \times 4$ grid	Original	30.0
	64 non-uniform	Original	23.1
	64 non-uniform	Hi-Z	**24.8**
Crytek Sponza	$6 \times 3 \times 3$ grid	Original	24.1
	54 non-uniform	Original	20.6
	54 non-uniform	Hi-Z	**19.0**

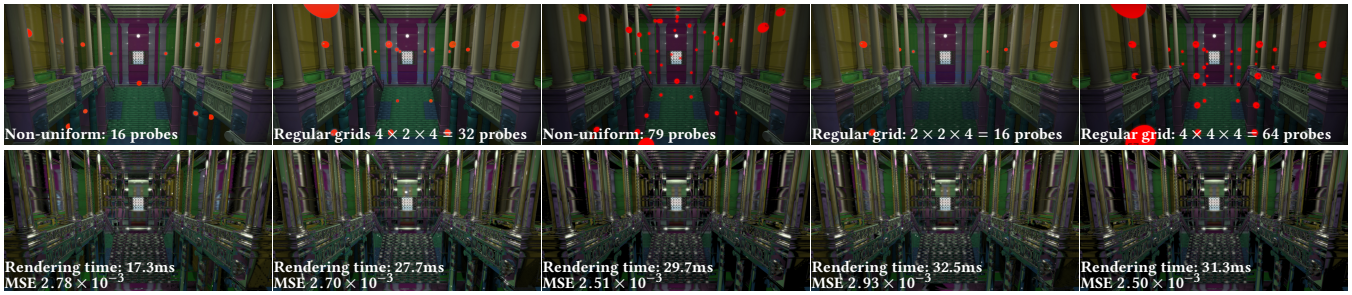

Figure 10: Iconic Temple – equal quality (cols. 1 & 2), equal probe count (cols. 3 & 4), and equal performance (cols. 3 & 5). Top to bottom: probe placement, glossy indirect rendering. At similar MSE, non-uniform placement uses fewer probes.

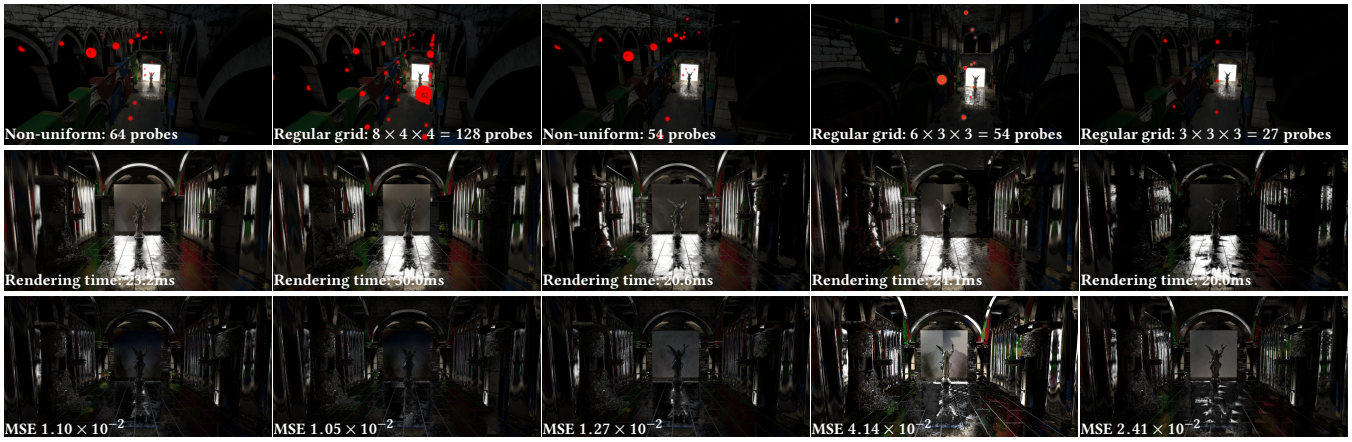

Figure 11: Crytek Sponza – equal quality (cols. 1 & 2), equal probe count (cols. 1 & 4) and equal performance (cols. 3 & 5) comparisons. Top to bottom: probe placement, glossy indirect component and glossy indirect error visualization. At similar MSE, non-uniform placement uses fewer probes. For equal probe count *and* at equal performance, non-uniform placement achieves lower MSE and fewer visual artifacts.

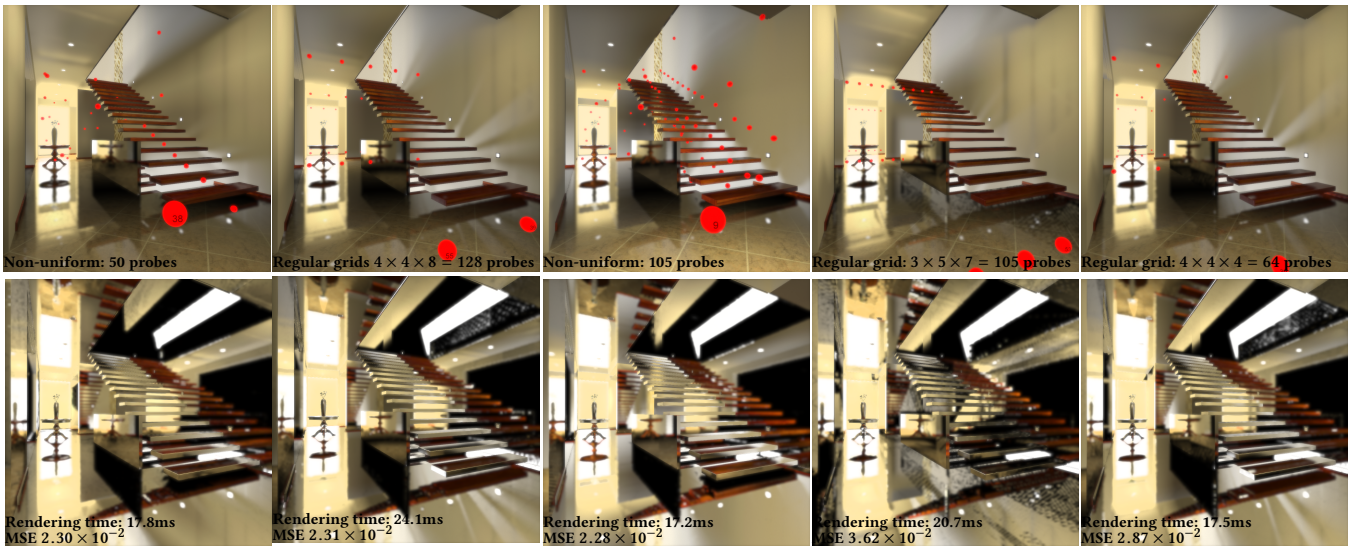

Figure 12: Modern Hall – equal quality (cols. 1 & 2), equal probe count (cols. 3 & 4) and equal performance (cols. 3 & 5) comparisons. Probes (top) & glossy indirect shading (bottom). At similar MSE, non-uniform placement uses fewer probes.

We consistently outperform the approach of McGuire et al. [2017]. Our hierarchical-Z ray tracer accelerates the tracing process, resulting in render times roughly 30% to 53% faster than those of their approach, at equal visual quality. Generally speaking, our method maintains good performance even at extreme, dense non-uniform probe counts. Since runtime probe selection accounts for visibility, we avoid querying probes that are unlikely to contribute an intersection. Regular grids, on the other hand, do not generalize well with increased visibility complexity (e.g., the decorations between pillars in the iconic temple).

Apart from ray tracing, the most time-consuming component of our runtime is the SVO traversal. A more efficient structure, such as a BVH, may improve this cost.

Generating the skeleton (either from an SDF or a medial axis) for one-time probe placement take $\sim$ 10 seconds per scene, and depends on the resolution and complexity of the geometry. Baking probe indices into the SVO takes $\sim$ 5 seconds at initialization. Total probe placement and initialization time is roughly the same as [McGuire et al. 2017], requiring about a minute depending on the number of probes (and their resolution, which we keep fixed).

Memory Footprint. Table 2 summarizes our memory usage statistics. We separate the storage costs of the probe distance data, SVO, and other probe data (radiance, irradiance), as the former are the key differentiators between our method and McGuire et al. [2017]. We use the same texture formats as McGuire et al. [2017]. We store probe positions in a 1D texture, adding a negligible cost (a few KB) incorporated into the "*other*" category.

Generally speaking, our method uses more memory *per probe* than the original algorithm: probe positions cannot be computed implicitly from the grid, and the SVO incurs the majority of our additional memory requirements. Depending on the resolution and depth of the SVO, memory usage varies. We store SVO probe indices in RGBA8 format and each voxel has at most four probe candidates.

We encode radial distances in a R16F format 512×512 texture. For hierarchical-Z tracing, we mipmap distance probes in RG16F format, storing minimum and maximum depth in the R and G channels.

6 CONCLUSION AND FUTURE WORK

We provide a new approach for non-uniform placement of radiance probes, which identifies positions that maximize visibility and suitability for use in light field ray tracing. Our approach can be extended to open spaces with a bounding volume intersection. For the run-time shading algorithm, we provide an efficient structure to organize probes which is visibility-aware. We devise a new hierarchical ray marching algorithm adapted to non-uniform probes that outperforms the state-of-the-art.

Table 2: Memory usage (in MB) with same quality.

Scene	# Probes	Probe Data	Other	SVO	Total
Cornell	non-uni 7	9.33	12.25	15.41	36.99
	grid 8	4.57	14.88	–	19.45
Temple	non-uni 16	21.33	28	61.21	110.54
	grid 24	13.71	59.52	–	73.23
Hall	non-uni 50	66.70	87.60	195.25	349.55
	grid 128	146.29	238.08	–	384.37
Sponza	non-uni 54	72	94	243.23	409.23
	grid 128	146.29	238.08	–	384.37

In all test scenes, our solution achieves higher rendering quality with fewer probes (in comparison with the path traced references) and maintains better performance than regular grids. Our optimized non-uniform probe positions capture much more precise indirect glossy reflections in comparison to the same number of probes organized in a regular grid.

Future Work. There are several avenues for future work. To better identify effective positions for placement, we can analytically compute probe-to-surface visibility by adopting irradiance isocountour gradients [Arvo 1995]. In order to evaluate the visibility of each probe, we unwrap the geometry of a 3D scene into an atlas. However, for large scale scenes with complex geometry, mapping all surfaces into a single texture atlas leads to loss of detail. We could search for a segmentation of the geometry that allows us to map each part into a local atlas such that details are preserved and all surfaces are fully represented at a common scale.

Runtime selection of the most promising probe for any given ray is also an important direction for future work. During a ray tracing process, there could be multiple probe selection queries, depending on the complexity of the scene. Therefore we need a strategy to be efficient and GPU friendly. In our current algorithm, we use a sparse voxel tree which encodes light probe indices in each voxel. However, the sparse voxel tree is stored in a RGBA 3D texture, and due to the sparsity of storage, a lot of space is left empty. We can further optimize the memory usage of octree since we only store on leaves. For runtime, traversing down a octree involves reading the children index buffer and computing correct nodes. This operation can be expensive for large scenes because the depth of tree is greater if we keep the same resolution as our testing scenes. We believe different representations of probe structures can save memory, for example, bounding volume hierarchy (BVH) could be used to approximate the range of each probe for selection.

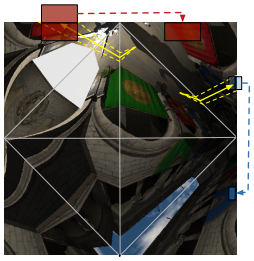

Figure 13: Radiance probe mipmap issues.

Finally, although we applied hi-Z ray tracing, we do not apply the cone tracing algorithm. In probe textures, the octahedron mapped to a square texture is sampled from six faces of the cubemap [Engelhardt and Dachsbacher 2008]. Because we bake radial probe distances, which cannot be used to pre-convolve visibility, we constructed visibility from real depth for each cube face. However, constructing mipmaps of radiance octahedral textures for runtime sampling is not trivial, since we need to stitch the seams of the octahedron together to have the correct cone projected. As shown in Figure 13, rays (in yellow) are traced in the probe texture and the hit texels that lie on the edges of the octahedron. To correctly mipmap the texels on boundaries, we must consider the texels that are reflective symmetric with respect to the midline of the boundary, as shown in red and blue. Ultimately, the cone constructed at the hit pixel requires a large radius in this case, where we need to mipmap the value correctly based on which edges are connected.

ACKNOWLEDGMENTS

The authors acknowledge funding from the NSERC/Ubisoft Industrial Research Chair in Believable Virtual Character Experiences.

REFERENCES

James Arvo. 1995. *Analytic Methods for Simulated Light Transport.* Ph.D. Dissertation. PhD thesis, Yale University.

Alireza Bagheri and Mohammadreza Razzazi. 2012. Drawing free trees inside simple polygons using polygon skeleton. *Computing and Informatics* 23, 3 (2012), 239–254.

Johan Bowald. 2016. *Global Illumination for Static and Dynamic Objects using Light Probes.* Master's thesis. Chalmers University of Technology.

Matthäus G Chajdas, A Weis, and Rüdiger Westermann. 2011. Assisted Environment Map Probe Placement. In *Proceedings of SIGRAD 2011. Evaluations of Graphics and Visualization - Efficiency; Usefulness; Accessibility; Usability; November 17-18; 2011; KTH; Stockholm; Sweden.* Linköping University Electronic Press; Linköpings universitet, 17–25.

Cyril Crassin and Simon Green. 2012. Octree-Based Sparse Voxelization Using The GPU Hardware Rasterizer. In *OpenGL Insights.* CRC Press, Patrick Cozzi and Christophe Riccio, 303–318. http://www.seas.upenn.edu/~pcozzi/OpenGLInsights/OpenGLInsights-SparseVoxelization.pdf,ChapterPDF

Robert Cupisz. 2012. Light Probe Interpolation using Tetrahedral Tessellations. In *Game Developers Conference (GDC).*

Thomas Engelhardt and Carsten Dachsbacher. 2008. Octahedron Environment Maps.. In *VMV.* 383–388.

Mickael Gilabert and Nikolay Stefanov. 2012. Deferred Radiance Transfer Volumes, Global Illumination in Far Cry 3. In *Game Developers Conference.*

Gene Greger, Peter Shirley, Philip M Hubbard, and Donald P Greenberg. 1998. The Irradiance Volume. *IEEE Computer Graphics and Applications* 18, 2 (1998), 32–43.

Johannes Jendersie, David Kuri, and Thorsten Grosch. 2016. Precomputed Illuminance Composition for Real-Time Global Illumination. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '16).* ACM, New York, NY, USA, 129–137. https://doi.org/10.1145/2856400.2856407

Tom Kelly, John Femiani, Peter Wonka, and Niloy J. Mitra. 2017. BigSUR: Large-scale Structured Urban Reconstruction. *ACM Trans. Graph.* 36, 6, Article 204 (Nov. 2017), 16 pages. https://doi.org/10.1145/3130800.3130823

Jaehwan Ma, Sang Won Bae, and Sunghee Choi. 2012. 3D Medial Axis Point Approximation using Nearest Neighbors and the Normal Field. *The Visual Computer* 28, 1 (2012), 7–19.

Morgan McGuire and Michael Mara. 2017. The G3D Innovation Engine. https://casual-effects.com/g3d/ https://casual-effects.com/g3d/.

Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, and David Luebke. 2017. Real-Time Global Illumination Using Precomputed Light Field Probes. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '17).* ACM, New York, NY, USA, Article 2, 11 pages. https://doi.org/10.1145/3023368.3023378

Ravi Ramamoorthi and Pat Hanrahan. 2001. An Efficient Representation for Irradiance Environment Maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01).* ACM, New York, NY, USA, 497–500. https://doi.org/10.1145/383259.383317

Lagarde Sébastien and Antoine Zanuttini. 2012. Local Image-based Lighting with Parallax-Corrected Cubemaps. In *ACM SIGGRAPH 2012 Talks.* ACM, 36.

Ari Silvennoinen and Jaakko Lehtinen. 2017. Real-Time Global Illumination by Precomputed Local Reconstruction from Sparse Radiance Probes. *ACM Trans. Graph.* 36, 6, Article 230 (Nov. 2017), 13 pages. https://doi.org/10.1145/3130800.3130852

Ari Silvennoinen and Ville Timonen. 2015. Multi-Scale Global Illumination in Quantum Break. In *SIGGRAPH '15 ACM SIGGRAPH 2015 Courses.* ACM.

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. Graph.* 21, 3 (July 2002), 527–536. https://doi.org/10.1145/566654.566612

Nikolay Stefanov. 2016. Global Illumination in Tom Clancy's The Division. In *Game Developers Conference.*

Natalya Tatarchuk. 2005. Irradiance Volumes for Games. In *Game Developer's Conference*, Vol. 3.

Yasin Uludag. 2014. Hi-Z Screen-Space Cone-traced Reflections. *GPU Pro* 5 (2014), 149–192.