Incrementally Baked Global Illumination

Christan Luksch
VRVis Research Center
luksch@vrvis.at

Michael Wimmer
TU Wien

Michael Schwärzler*
Delft University of Technology

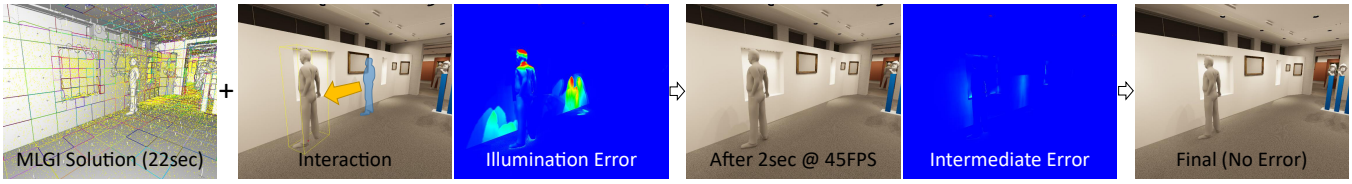

Figure 1: A *Many-Light Global Illumination* approach is used to bake a compact light transport representation to a lightmap, while real-time frame rates are maintained. At any time, scene interactions (e.g., moving objects, adjusting lights) are allowed, and the illumination is seamlessly transformed from the old state to the new. A prioritizing update scheme resolves illumination errors in a way that the most significant changes are treated within the first seconds without noise or flickering artifacts.

ABSTRACT

Global Illumination is affected by the slightest change in a 3D scene, requiring a complete reevaluation of the distributed light. In cases where real-time algorithms are not applicable due to high demands on the achievable accuracy, this recomputation from scratch results in artifacts like flickering or noise, disturbing the visual appearance and negatively affecting interactive lighting design workflows.

We propose a novel system tackling this problem by providing *incremental updates* of a baked global illumination solution after scene modifications, and a re-convergence after a few seconds. Using specifically targeted incremental data structures and prioritization strategies in a many-light global illumination algorithm, we compute a differential update from one illumination state to another. We further demonstrate the use of a novel error balancing strategy making it possible to prioritize the illumination updates.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Reflectance modeling**.

KEYWORDS

Global Illumination, Instant Radiosity, Lightmaps

ACM Reference Format:
Christan Luksch, Michael Wimmer, and Michael Schwärzler. 2019. Incrementally Baked Global Illumination. In *Symposium on Interactive 3D Graphics and Games (I3D '19), May 21–23, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3306131.3317015

1 INTRODUCTION

Developing new approaches to efficiently and accurately solve the rendering equation – describing the light transport in a scene (see Equation 1) – has been a major challenge in computer graphics for years. The recursive nature of this so-called *Global Illumination (GI)* problem implicates that if any element of a scene is changed, a complete re-evaluation is required to calculate the new solution. Real-time GI solutions exist, but are limited in terms of accuracy and complexity. Applications requiring accurate, high-quality results have to rely on computationally expensive methods (i.e., offline rendering algorithms), often requiring hours to compute. As a consequence, today's games engines often apply direct lighting using physically-based shading in real-time, while indirect diffuse or low-frequency GI is integrated using a baked solution provided by an offline rendering algorithm. This combination achieves almost photo-realistic impressions for scenes with (limited) dynamics and interactions. Nevertheless, for designers illuminating static parts of the scene, the high computational effort prevents a continuous, interactive workflow with illumination-based decisions.

We target this issue by describing an interactive system that provides a baked GI that remains consistent while the user can modify the scene freely. A key property of workflows in CAD modeling, lighting design or game level editing is that scenes are typically modified very locally. We propose an algorithm that keeps track of the complete state of the GI, and calculates the differences caused by the local modifications after a modeling step, allowing to update from one illumination state to another *incrementally*. In addition, the tracked light path information is used to estimate the illumination error, control the refinement of the illumination detail, and to prioritize and accelerate updates. While full illumination consistency after a modification is reached within a few seconds even in large scenes, interactive frame rates, editing capabilities and a noise-free visualization are retained during the adaptive simulation. We also demonstrate that (even in worst case scenarios) the transition to the new illumination state is performed seamlessly and without popping artifacts or flickering.

*Also with VRVis Research Center.

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

Specifically, our contributions are:

- a GI method tracking the light path information, allowing to incrementally compute accurate GI after scene modifications,
- strategies for controlling the GI error and – based thereon – a prioritization scheme for the incremental updates,
- and the incorporation of all these aspects in an interactive design tool for baking lightmaps for large scenes without delays, visual inconsistency, or noise.

2 RELATED WORK

The computationally complex task of GI is approached by different algorithms to solve the rendering equation [Kajiya 1986]

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f(x, \omega_o, \omega_i) L_i(x, \omega_i)(\omega_i \cdot n) d\omega_i, \quad (1)$$

where L_o is the radiance of a point x transfered in outgoing direction ω_o, as sum of the surface emission L_e and the integral of all incidence directions ω_i over its hemisphere Ω. f is the bidirectional reflectance distribution function (BRDF) describing the surface reflectance properties. For an extensive overview of interactive approaches, we refer to the survey by Ritschel et al. [2012].

Instant Radiosity, introduced by Keller [1997], reduces the solution of the rendering equation to the evaluation of a set of so-called *Virtual Point Lights* (VPLs). They are created at every intersections by tracing particle paths from the light source through the scene. Methods developed on this basis are called *Many-Light Global Illumination* solutions. The survey by Dachsbacher et al. [2014] gives a detailed overview. In contrast to other rendering algorithms, they do not produce noisy images, but are biased due to necessary culling and clamping when the virtual lights are evaluated. The extensions of *Lightcuts* [Walter et al. 2005] and *Bidirectional Lightcuts* [Walter et al. 2012] significantly speed up the rendering times and yields sub-linear scalability. *Matrix Row-Column Sampling* [Hašan et al. 2007] and *Matrix Slice Sampling* [Ou and Pellacini 2011] formulate the radiance transport of virtual lights as matrix and build a reduced solution allowing the use of *Shadow Mapping* [Williams 1978].

The reduced light transport complexity of many-light solutions makes them very interesting for real-time applications. *Reflective Shadow Mapping* [Dachsbacher and Stamminger 2005] is one of the first techniques that provides a single bounce indirect GI in real-time. *Imperfect Shadow Maps* [Ritschel et al. 2008] provide fast visibility tests for a large number of lights that significantly improves the possible complexity and performance of real-time approaches. Dong et al. [2009] demonstrated that using clustered visibility for a group of virtual lights in combination with a suitable real-time soft shadow mapping technique also yields comparable results. Ritschel et al. [2011] describe a view-adaptive approach that provides a good distribution of VPLs according their contribution to the image. Georgiev et al. [2010] use a *Russian Roulette* strategy to select VPLs generated from multiple bounces with balanced contribution to the image. Olsson et al. [2015] use hardware-supported virtual cube-map shadows to efficiently implement high-quality shadows from hundreds of light sources.

An additional aspect that especially real-time approaches need to address is temporal aliasing that occurs when VPLs are created independently every frame. A strategy employed by Laine et al. [2007] is to reuse as many VPLs as possible between frames. Prutkin et al. [2012] use a k-means clustering where the seeds are initialized with the cluster centers from the previous frame. Barak et al. [2013] extend the method from Ritschel et al. by using Metropolis Hastings to select the VPLs from an importance buffer. Hedman et al. [2016] solve temporal stability similar to Laine et al. in combination with a sampling technique that selects VPLs of multiple light bounces efficiently according to their contribution to the image. VPLs managed in a *Lighting Grid Hierarchy* [Yuksel and Yuksel 2017] provides an offline rendering technique that efficiently avoids temporal aliasing. It achieves a huge speed-up for complex animated volumetric effects and general many-light problems.

Luksch et al.[2013] avoid temporal aliasing in their many-light solution by constructing the VPLs view-independently, allowing to bake lightmaps with complex diffuse GI in the order of seconds using shadow mapping and rasterization. Their method employs a point cloud segmentation algorithm on the VPLs in a first step and then performs a top down kd-tree clustering per segment afterwards that creates a set of well-fit virtual polygonal [Baum et al. 1989] and spherical lights [Hašan et al. 2009]. To incorporate interactions and bridge the calculation delay, a combination of real-time visualization of intermediate illumination results and blending is used.

A current limitation of real-time many-light approaches is that they are only applicable in scenes with diffuse surfaces. Voxel based GI techniques [Crassin et al. 2011; Kaplanyan and Dachsbacher 2010; Thiedemann et al. 2011] use voxelized scene representations in order simplify the computations. This allows to perform a GPU friendly ray marching that is able to provide glossy surface reflections. In practice the GI is only computed for one indirect light bounce as recursive evaluation or injection of a multi-bounce illumination into the voxel data structure are too expensive for real-time applications.

A series of recent techniques that combine ideas from different approaches with precomputations allow real-time GI with dynamic lighting in static scenes. Jendersie et al. [2016] use surfels that are shaded in real-time and a clustered hierarchy with precomputed light links to propagate the indirect illumination to (ir)radiance caches. The technique of Silvennoinen et al. [2017] use radiance probes with precomputed transport terms to iteratively propagate the indirect illumination over multiple bounces to a lightmap. McGuire et al. [2017] use light field probes and describe algorithms for diffuse indirect illumination and glossy reflections. All those techniques can be extended to provide coarse indirect illumination for moving objects, but those have no or limited effect on the GI.

3 SYSTEM BASELINE

In our proposed approach, we strive for a solution to solve the rendering equation with high accuracy, but provide interactivity during its calculation. That means, we aim for computation times of several seconds on a GPU, and display intermediate results while retaining interaction capabilities. Luksch et al [2013] have successfully demonstrated the practicability of such ideas, but have the major drawback that each single scene modification triggers a complete re-computation of the light distribution. While this may be compensated by clever blending techniques in small scenes, these attempts don't scale in cases with several hundred light sources or large extents, as the computation time is too high, introducing waiting times until enough light is distributed for a decision on

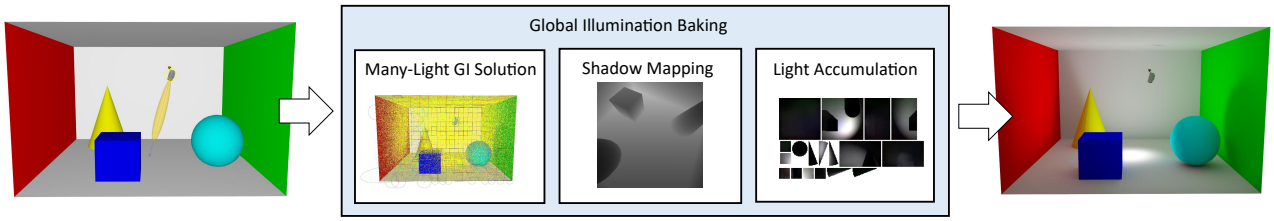

Figure 2: Overview of the system baseline. A many-light solution is baked into a lightmap using shadow mapping and rasterization, providing complex diffuse GI independent from the view-point. Baked solutions typically require complete recalculations, interrupting a continuous work flow. Our proposed approach extends such a system by an efficient update mechanism performing an incremental transition to the new illumination state.

the next (modeling) steps. We base our novel method on a Many-Light Global Illumination (MLGI) algorithm, that incrementally updates a baked high-quality diffuse GI state in lightmaps after scene modifications. This has two major advantages:

- It is not necessary to re-evaluate the complete light propagation, but only account for those parts where the accuracy is affected. This is especially beneficial in large scenes.
- By using a many-lights approach and lightmaps, we display the current illumination state in a noise-free way while the simulation is still running after modifications. We also show how necessary updates can be prioritized, so that the visual impact caused by an incomplete update is minimal.

3.1 Simulation Setup

In our context, a scene consists of individually transformable polygonal surfaces with material information, and a set of light sources (see Section 3.2). In order to calculate the indirect illumination, particles are emitted from these light sources (Section 3.3), traced in the scene, and grouped into clusters (Section 3.4). These particle clusters act as indirect light sources.

The actual light transport is then performed using the direct and indirect light sources, and a rasterization-based approach. Our proposed incremental update mechanism assumes that there is a suitably fast mechanism available that is able to accumulate the contribution of the virtual lights (both direct and indirect) to the baked state (e.g., using shadow mapping, achieving an evaluation of up to 1000 lights per second, and taking visibility into account approximately). We use lightmaps, for which the uv-coordinates are generated automatically, as our method for storage and representation of the baked world-space MLGI.

During the accumulation process, intermediate results can be visualized in an interleaved way with the simulation using the information from the lightmaps, retaining interactivity at all times. Note that it is possible to subtract stored light information again by adding "negative" light energy – a requirement for the adaption to incremental computations.

3.2 Light Source Description

Direct light sources are placed and modified by an artist, and are described by a transformation, aperture geometry, emission profile and intensity. Figure 3 illustrates a linear area light with annotated components. The aperture geometry defines the surface where

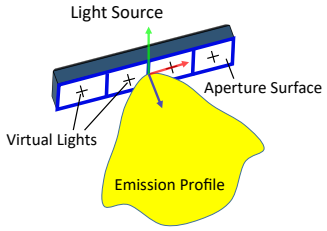

Figure 3: Light source described by an aperture surface (white) from where light is emitted according to an emission profile (yellow). The direct illumination is rendered using virtual lights (blue polygons) that subdivide the aperture.

the light is emitted from (see Section 3.3), and can be a polygon, but also a singular point, or a sphere. The emission profile can be Lambertian, a parametric spot, or in the form of an arbitrary spherical profile (IES/LDT photometries as used in lighting design). Intensity and transformation are parameters that could be implicitly given by the geometry and the emission profile, but are deliberately separate as they play an important role during incremental updates.

In order to support arbitrarily complex lighting of a large number of light sources and area lights, the direct illumination is accumulated in the lightmaps as well. The direct light emission is split up into a suitable amount of virtual lights that are managed equally to indirect virtual lights (see *Direct Virtual Lights* in Section 4.1).

3.3 Particle Tracing

A particle simulation similar to Photon Tracing is the first step of the virtual light creation. A random walk based on quasi-Monte Carlo integration where each path transports the same amount of light energy p is used. At each intersection of a path with the scene, an oriented VPL with intensity of p times the surface reflectance is created. The total number of paths and the maximum number of light bounces can be configured. In order to prioritize illumination updates in our system, the created VPLs are linked to the object at the path intersection and will be used to gain instant insight on what paths are invalid after a modification of the scene.

3.4 Virtual Light Clustering

The huge number of VPLs created from the particle tracing are clustered to form more expressive virtual lights that give an accurate

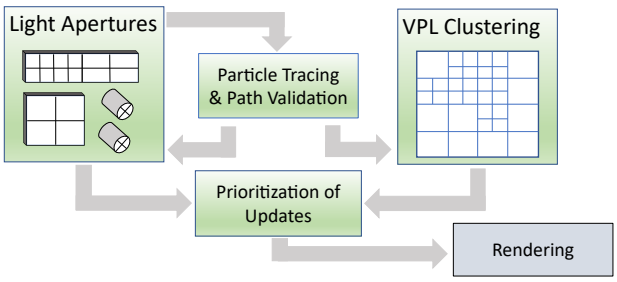

Figure 4: Outline of the incremental GI update components. The illumination is represented by direct virtual light on light apertures and the octree VPL clustering (Section 4.1). Particle tracing and path validation (Section 4.2) identifies changes that accumulate invalid energy affected VPL clusters. The illumination updates are then performed in a prioritized process (Section 4.3 and 4.4).

but very compact representation of the GI. This process employs a top-down clustering scheme based on a balancing metric (see Section 5) that balances contribution and error of the clusters for a given budget N. The clustered VPLs are defined very similarly to direct light sources, but are used for computing indirect illumination, and have a lambertian emission characteristic. Previous work of Luksch et al.[2013] demonstrated the superior accuracy of polygonal light sources on planar regions due to the well-defined borders to neighboring clusters. Therefore, the clustering should be able to identify planar regions in the VPLs whenever possible, and place a polygonal virtual lights there. In all other cases, disk lights are used.

4 INCREMENTAL GI UPDATES

Based on the described light baking method, we propose a novel strategy that is capable of dealing with updates in the light distribution after scene changes incrementally, i.e., instead of a complete recalculation, the difference between two GI states is calculated. In theory, due to the recursive nature of the GI problem, every cluster (and therefore every indirect light source) has to be updated and re-evaluated after a change anyway. The benefit of an incremental computation lies in the *visual coherence* between two illumination states, which is a lot higher compared to a complete re-evaluation (see Figure 1). Moreover, we propose different strategies for prioritizing updates with a high visual impact (Section 4.4).

The core task of the update process is to provide an illumination response after scene changes as fast as possible. This means, that illumination information that has become invalid after a modification has to be removed from the lightmap, and new lighting has to be added in a consistent way. In a light simulation system as described above, this implies solving the following challenges (see also Figure 4):

- To allow updating the illumination information incrementally, the underlying data structures must support corresponding, efficient update mechanisms (Section 4.1).
- Changes in the illumination of the scene have to be reliably detected. Sometimes this can be directly derived from the type of scene interaction (e.g., moving a light source), but most information on illumination changes can be gained through the particle path validation (Section 4.2).
- Due to the invalidation of VPLs and the creation of new ones, the clusters with invalid energy have to be updated efficiently (Section 4.3). This will also require to re-balance the selected clusters according to a balancing metric (see Section 5).
- Necessary updates to the baked illumination state have to be performed in the form of atomic updates that always provide a consistent state without missing or double counted light energy. In order to increase the user experience, we prioritize the cluster updates according to the amount of invalid light energy (Section 4.4).

4.1 Incremental VPL Clustering Data Structure

The first challenge is to find a clustering scheme that is suited for incremental operations, i.e. it is required to have the capability to add VPLs to and remove VPLs from a cluster arbitrarily. These operations should also only have a local effect on the affected cluster, and should not cause balancing issues in the used data structure.

In existing many-light approaches, clustering schemes are mainly driven by a similarity metric that combines reflectance, spatial and directional variation. This cluster property is important to minimize the illumination error, and we seek to fulfill this property as well.

In summary this leads to the following requirements:

- Well-defined cluster boundaries (Section 3.4)
- Suitability for incremental operations
- Locality of updates
- Clustering by similarity
- Ability to keep balance (Section 4.3)

Spatial Clustering: Bounding volume hierarchies or loose space partitioning trees would not have well-defined boundaries and are therefore not suited. Kd-trees produce well-adapted and balanced clusters, but variants that are able to keep balance when updated are complex, and most implementations require multiple nodes to be re-clustered, which does not meet or requirement that updates should be local. Hence, in order to meet our requirements, we opted for a two-stage VPL clustering approach to generate the virtual light sources. We first perform *spatial clustering using an octree* (which is easy to maintain and not prone to balancing issues), followed by a similarity clustering within the octree leaf nodes. We intend to refine the spatial clustering to a level where the light emission of each leaf can be represented by a small number K of clustered virtual lights. In case the distribution of VPLs changes, it is possible to adjust this refinement locally without effecting other regions. Figure 5 illustrates two octree cells (C_A, C_B) an the clustered virtual lights created in each (L_i).

Similarity Clustering: In each octree leaf node, virtual lights with similar orientation are created using a two-stage *hierarchical clustering* approach. As mentioned in Section 3.4, polygonal clusters are the preferred way to transport light with high accuracy. In order to avoid illumination artifacts the polygons need to fit the scene surfaces precisely. Therefore, we define that polygons must have support of at least $p_{min} = 10$ VPLs and allow just a small configurable orientation tolerance $\sigma_n = 1°$ and an offset tolerance

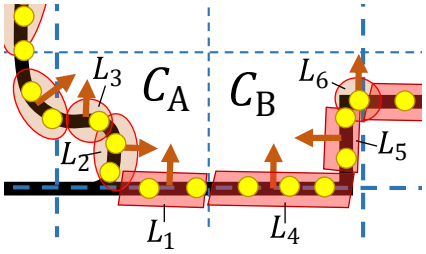

Figure 5: Illustration of spatial clustering using an octree where per leaf cell (C_A, C_B) VPLs are clustered by similarity (L_i). In particular polygonal lights (rectangles) are used to represent planar regions and disk lights (ellipses) are used for curved surfaces.

of $\sigma_d = 0.01$ times the node's cell size. Polygons are then detected in the following way: For a VPL at position p, its plane is given as $x = (n_x, n_y, n_z, d)$, where n is the normal vector of the surface the VPL lies on, and d is the plane normal distance given by $d = n \cdot p$. In order to be able to quantify the similarity of two VPLs planes using the euclidean distance of 4D vectors, we reformulate the plane normal distance as $\tilde{d} = n \cdot (p - c)\sin(\sigma_n)/\sigma_d$, where c is the node center. As the chosen tolerances are relatively small, a greedy cluster growing algorithm with epsilon $e = \sin(\sigma_n)$ is employed.

The number of virtual lights per octree node should not exceed a certain number K, as the update costs when adjusting the refinement of the octree need to be considered. Typically we use $K = 4$ in order have enough polygonal lights that fit well to corners and other cases where multiple surface planes meet or fall within the same octree cell. If the first clustering yields a result $> K$ clusters, only the $K - 1$ largest plane clusters are converted to polygonal virtual lights similar to Luksch et al. [2013] and clipped by the octree cell boundaries. The remaining VPLs are clustered according to their orientation using a k-means algorithm with an angular distance metric, where k is set to K minus the number of polygonal light clusters. They are interpreted using a disk light formulation similar to the one used by Prutkin et al. [2012] to avoid self-illumination artifacts, and are placed at the VPL closest to the cluster center with an orientation derived from the cluster's best fitting plane.

Border Handling: As in an octree, spatial clustering determines to predefined cells, suboptimal cluster separations will be introduced (see annotated lights in Figure 5):

- VPLs on coplanar surfaces to cell boundaries are potentially assigned randomly to one of the cells due to numerical issues (L_1 and L_4)
- Planar surfaces expanding short distances along cell boundaries will introduce small disk lights where extending the neighboring polygonal light would be desired (L_6)

Such cases are resolved in a second step after the virtual lights of a cell have been built.

Direct Virtual Lights: As stated in Section 3.2, the aperture of the direct light sources are represented using virtual lights as well. Each aperture provides a root node and employs a hierarchical refinement scheme. As we consider the geometry to be rigid, we use a kd-tree here that subsequently splits along the major extend of its cell. The emitted particle paths are linked to each leaf, which allows to apply a unified balancing metric for direct and indirect VPL clusters (see Section 5).

4.2 Path Validation

The particle paths represent all the distributed light energy in our system. After each scene modification, all particle paths are validated, and the changed paths are incorporated into the virtual light clustering state. In order to identify essential changes efficiently, we perform the validation per light source where the paths have been scattered starting from its aperture and employ a prioritization. This prioritization is based on invalid radiant flux (i.e., energy of light paths known to be invalid) that is identified immediately after scene modifications. If the aperture geometry is transformed, the corresponding emitter gets a priority with all its radiant flux. In case of general changes, we utilize the stored links from objects to particle paths by tracking them back to their originating light emitters, and accumulating the invalid radiant flux.

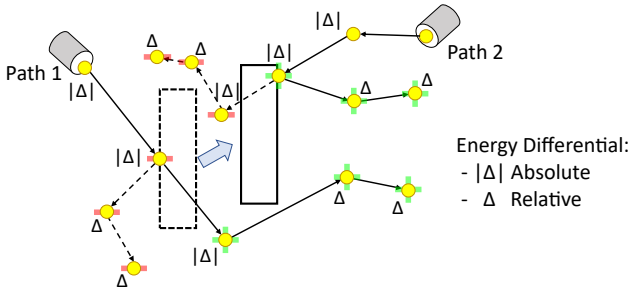

Figure 6: Illustration of path validation and invalid energy tracking after a scene modification (movement of black box). Prioritization using object-VPL links updates Path 1 before Path 2. Eventually, all invalid VPLs are removed and, new path segments and their VPLs are added. The changes also accumulate invalid energy in their clusters, with distinction between *absolute* |Δ| and *differential* Δ accumulation.

Figure 6 illustrates a scene modification and showcases two paths of different light sources. The prioritized validation will start with Path 1 as its connected to the moved object, but eventually all paths are validated and will identify all VPLs that are invalid. Those VPLs, and the VPLs that emerged from them due to further scattering, are removed from their current VPL cluster. New VPLs, including their scattered follow-up VPLs, are added. Changes in a path intersection also implicates that the originating cluster has changed light-receiving elements. In order to allow subsequent updates in an efficient order, each cluster accumulates these changes in the form of radiant flux. In particular, we distinguish *absolute* |Δ| and *differential* Δ changes:

- Differences caused by scattered particles that are removed and added to clusters, and thereby change the cluster's radiant flux, are tracked differentially, meaning that additions and removals might cancel each other out.
- Changes in the path intersections and their originating clusters (in case of a moved object) are accumulated absolutely.

The sum of both terms gives the invalid energy Φ_{inv} that will be used for prioritization for updating the illumination of a cluster. Changes to the VPL distribution also implicate a changed balancing metric (Section 5), which therefore needs to be re-evaluated (Section 4.3). The update selection process (Section 4.4) will then decide when to perform *Validation* updates.

4.3 Incremental Cluster Balancing

In order to ensure an efficient and balanced selected level of detail of the incremental clustering data structure described in Section 4.1, a balancing metric B (see Section 5) is employed, providing an abstract quantification of the illumination error for a spatially clustered octree node. In contrast to Lightcuts [Walter et al. 2005], this task is performed for a global set of virtual lights and not a per-pixel-basis, and combined with incremental updates adapting to scene changes.

The cluster balancing state is defined by the set of selected nodes, and their corresponding balancing metric B. At first, all root nodes (including the VPL octree and one node per light aperture) are selected. The state can be modified using the following operations:

- *Refine*: Removes the node from selected set and adds its sub-nodes.
- *Merge*: Remove siblings and select their parent. Only valid if all sibling nodes are contained in the selected set.

The runtime procedure aims to maintain a *balanced* state of selected nodes, where all nodes should have the most similar balancing metric B for a given budget of N nodes. After additions and removals of VPLs to the octree nodes due to scene modifications (see Section 4.2), the metric will change and thereby invalidating the *balanced* state. In order to incrementally transition to the desired state, the *Merge* operation that introduces the smallest B is considered. If it introduces a B smaller than the currently largest B, the operation will be performed, otherwise we consider the state as *balanced*. In case of a *Merge*, the reduced number of selected nodes allows to perform a new *Refine* operation. The complete process is depicted in Algorithm 1. Note that its execution is interleaved with illumination updates (see Section 4.4).

```
balanced ← false
while not balanced do
    if nodeCount < N then
        take node with largest B
        perform Refine
    else
        find node to merge introducing smallest B
        if introduced B < largest B of selection then
            perform Merge
        else
            balanced ← true
        end
    end
end
```

Algorithm 1: Cluster balancing algorithm in closed-form. After updating the balancing metric of all clusters after a scene change, *Merge* and *Refine* updates are performed alternately until a fixed point is reached. Note that those updates are actually performed interleaved to *Validation* updates (see Section 4.4).

4.4 Update Selection & Prioritization

In Sections 4.2 and 4.3, we outlined how invalid radiant flux is tracked, and how the virtual light clustering is balanced. The final stage of our incremental system defines the execution order of the actual illumination updates. Assuming both processes (Path *Validation* and *Balancing*) provide one next update each, we use a heuristic to decide which to perform first, as invalid radiant flux Φ_{inv} is in no direct relation to the balancing metric B. Typically, significantly more *Validation* updates are required to transition from one state the next, but *Balancing* is essential to provide refined clustered lights in areas where geometry is added or moved to. Therefore, we perform them interleaved, but by an adjusting rate. The ratio of invalid radiant flux to total radiant flux $\frac{\Phi_{inv}}{\Phi}$ of the next *Validation* update is used to adjust the rate from exclusive *Validation* updates to 50:50 interleaved updates using the quadratic function:

$$ratio = \max\left(0.5, \left(\frac{\Phi_{inv}}{\Phi}\right)^{\frac{\log(0.5)}{\log(t)}}\right). \tag{2}$$

$t = 0.1$ can be used to specify at what ratio of invalid energy the update sequence will be performed completely interleaved.

The application of this scheduling heuristic results in immediate updates of the direct illumination (since all radiant flux of the direct virtual lights is invalidated after interactions), as well as a high prioritization of illumination updates in regions where a large number of VPLs was changed.

Considering that updates are performed on octree cells, a single update involves accumulation of multiple virtual lights, where some will add, and some will remove illumination. Each light subtraction needs to be performed using the exact same scene state as its original addition. This includes the set of scene objects and their transformation (material parameters only need to be considered if required for in the shadow mapping stage). Note that additions or removals of scene geometries work implicitly using this scheme, assuming that eventually all illumination is subtracted using the old state and added to the new one.

- *Validation* updates subtract the old illumination, and add new illumination from the same or new virtual lights (depending on whether VPLs directly within the cluster changed, or whether only different scattered hits occured).
- *Balancing* updates typically involve a higher number of individual operations, depending on the structure of the VPL clusters and the number of non-empty octree nodes.

The selected updates are then performed in the main rendering procedure in an atomic way, while interactive or real-time frame rates are maintained. In our evaluation we found that an average of 4-8 accumulations per update needs to be performed (see Section 6).

5 CLUSTER BALANCING METRIC

Balancing the contribution of virtual lights is the key aspect of an efficient many-light rendering process. The number of virtual lights should be as small as possible, while the illumination error is minimized at the same time. In our system this challenge needs to be solved for a global set of virtual lights while incorporating information from the traced particle paths. Therefore, we introduce a balancing metric for the top-down clustering process that balances

subdivision the VPL-octree (see Section 4.1). Figure 7 illustrates an octree cell (left) and the simplified representation of a VPL cluster (right) used to calculate the balancing metric. Note that we perform the refinement of clusters on the octree data structure, so a cluster in this section refers to the VPLs of an octree cell. The combined balancing metric B for a cell (VPL cluster) is defined as the product of the following derived terms:

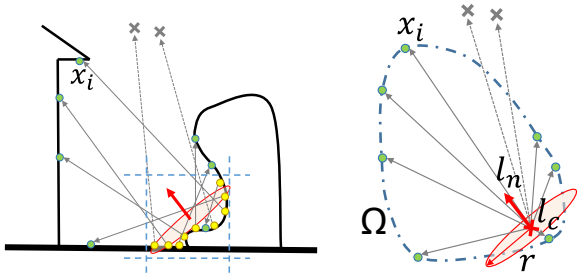

Figure 7: Illustration (left) of an octree cell (blue) with its VPLs (yellow) and the scattered particle paths that potentially intersected with the scene (green). The simplification (right) with normal l_n, position l_c and radius r is used to estimate the contribution and potential illumination error considering the light receiving environment Ω.

Visual Contribution: The visual contribution Φ_Ω of a virtual light can be calculated as the radiant flux Φ multiplied by the average reflectance of the surrounding environment Ω. Considering the traced particle paths that where scattered using a PDF p correlating to the light emission $L_o(\omega_o)$, this gives the following equation:

$$\Phi_\Omega = \frac{1}{2\pi N_h} \sum_{i=1}^{N_h} f(x_i) \underbrace{\frac{L_o(\omega_o)}{p(x_i)}}_{I} \tag{3}$$

$$= \frac{\Phi}{N_s} \sum_{i=1}^{N_h} f(x_i), \tag{4}$$

where f is the surface albedo in direction of x_i, N_s is the number of scattered particles and N_h the number of actual hits with the scene. The application of importance sampling during the particle tracing reduces the term I to the normalization factor of the PDF. Its integral gives the radiant flux and can therefore be substituted by Φ, which is typically calculated during the VPL clustering as the sum of all particle energies p weighted by the surface reflectance in the cluster. As not all scattered particles might hit the environment (i.e., they are scattered out of the scene), the ratio of scattered and hit particles $\frac{N_h}{N_s}$ is applied. In general, this contribution-based balancing term has its most significant impact in outdoor scenes, where most light energy is reflected outside the scene.

Illumination Error: The worst case illumination error of a virtual light can be assumed to be proportional to its maximum radiant intensity I_{max}. This means that the different types of virtual lights need to be considered (see Figure 8). Building on the contribution term Φ_Ω that is proportional to the radiant flux Φ, a directionality term D is introduced in order to drive the cluster balancing based on I_{max} insteadof radiant flux.

$$D = \frac{I_{max}}{\Phi} \tag{5}$$

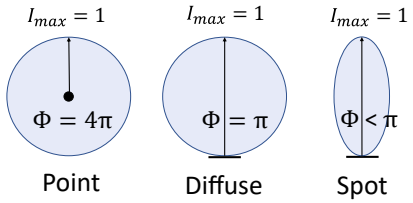

Figure 8: Distribution of radiant intensity $I(\omega_o)$ for a point, diffuse area and spot light with equal maximum of 1 and their corresponding radiant flux Φ. I_{max} is put into relation to express the worst case illumination error.

Visibility Error: The visibility of clustered VPLs is evaluated from a single point only, and thereby introducing visibility errors in penumbra regions of the virtual light cluster. In order to estimate occlusion errors, neighboring particle hits could be put in relation, and their depth gradient could give an estimate. In the context of irradiance caching, an Occlusion Hessian [Schwarzhaupt et al. 2012] placement error metric has been demonstrated to be very efficient. A spherical mesh needs to tessellated from the path directions, then the depth gradients can be calculated per vertex and summed up to an error estimate. On an octree level where there are > 500 paths this metric provides suitable results in situation where occlusion artifacts could be observed. Unfortunately, we did not manage to integrate this in a scalable form. It turned out that this calculation is too expensive for our application where the error needs to be updated for a lot of clusters every time paths change. We use a simplification based on an estimate of the penumbra size, that can be calculated in general using the distance from the cluster to the light receiver and the shadow caster. Assuming the illumination of all clusters is affected by a shadow caster at the same arbitrary distance, implicates that each cluster introduces a visibility error E_v proportional to its area. This visibility error estimate can be generalized by incorporating the average distance d_{avg} of light receiving points x_i, and assuming the error to be proportional to the projected area:

$$E_v = \frac{\pi r^2}{(d_{avg} + r)^2} \tag{6}$$

The cluster radius r is used to define the area, and additionally acts as a bias in the denominator in order to guarantee a valid balancing metric. Note that any constant factors (e.g. π) can be excluded when applying the balancing metric.

Clustering Error: Polygonal lights are built with small tolerance, and disk lights effectively suppress self-illumination artifacts while the light transport difference to distant points is marginal compared to other error sources (i.e., visibility), see Figure **??**. We therefore assume a clustering term of 1.

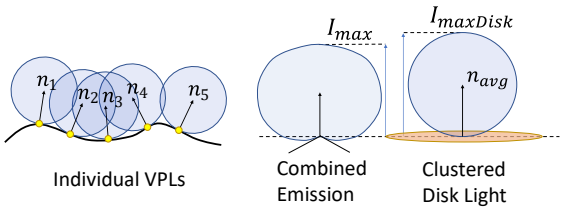

Figure 9: Illustration of illumination error when using a disk light for a VPL cluster. The light transport difference to distant points is marginal compared to other error sources. Self-illumination artifacts are suppressed by this formulation.

6 EVALUATION

All evaluations were performed on a desktop workstation with an Intel i7 4790K CPU, 32GB RAM and a NVIDIA Geforce GTX 980 graphics card at 1920x1080 resolution with 4x MSAA.

Table 1 (top) lists details of the scenes used in the evaluation, their frame time when rendered using the lightmap, and the average accumulation time needed to bake a virtual light. Visibility calculations were performed using cube shadow mapping with resolutions of 1024 for direct and 512 for indirect illumination. The lightmap has 32-bit floating point precision per channel, and the number of actually used texels is listed.

Table 1: Scene characteristics and update performance

	Triangles	Lights	Lightmap	Frame	VL
Museum	1,012,755	164	8 MP	5.2ms	2.5ms
Town Square	174,720	35	10 MP	2.9ms	2.1ms
Warehouse	349,079	24	4.5 MP	3.5ms	0.8ms

	Paths/Bc	Clusters	VLs	Upd-Sz	FPS	Upd/s
Museum	400K / 3	2737	5293	4.2	42	45
Town Square	500K / 3	3052	5734	3.9	36	50
Warehouse	1M / 4	5894	15728	7.8	30	120

Table 1 (bottom) gives insight on the update performance during interactive editing for the listed scenes and the used simulation settings with number of paths and bounces (Bc), balanced cluster count and virtual light count. Our implementation is able to validate about 200k paths per second and update the octree accordingly, which is fast enough for the prioritization to work efficiently. Additionally, it is to consider that newly scattered VPLs will require the the balancing metric of a lot of clusters to be re-evaluated constantly, but its simplicity allows incremental computations when single VPLs are added or removed. The update size (Upd-Sz) is the average of the number of accumulated virtual lights per *Balancing* and *Validation* update. It is affected by the geometric complexity within the octree cells, where each is allowed to create a maximum of $K = 4$ virtual lights. Finally, the targeted frame rate sets the limit on the updates that are possible interleaved with the rendering.

The virtual light visualization in Figure 10 shows how the octree and subsequent similarity clustering fits the lights to the scene geometry. The light transport in the Museum (left) and the Town Square (middle) of their large surfaces is represented well. In the Warehouse (right) the clustering operates on its limits, a higher cluster and light count might be preferable. Our system scales

Figure 10: Visualization of clustered virtual lights in Museum (left), Town Square (middle) and Warehouse (right) created with the configurations listed in Table 1.

well up to 100k clusters, allowing to trade illumination detail with feedback time. Compared to statically baked MLGI as proposed by Luksch et al. [2013], their segmentation and clustering approach provides virtual lights with better alignment to the scene structure, and in some extend to illumination gradients. Nevertheless, our incremental illumination update approach provides a huge benefit for interactive work flows, and is both scalable and achieves high-quality results at the same time.

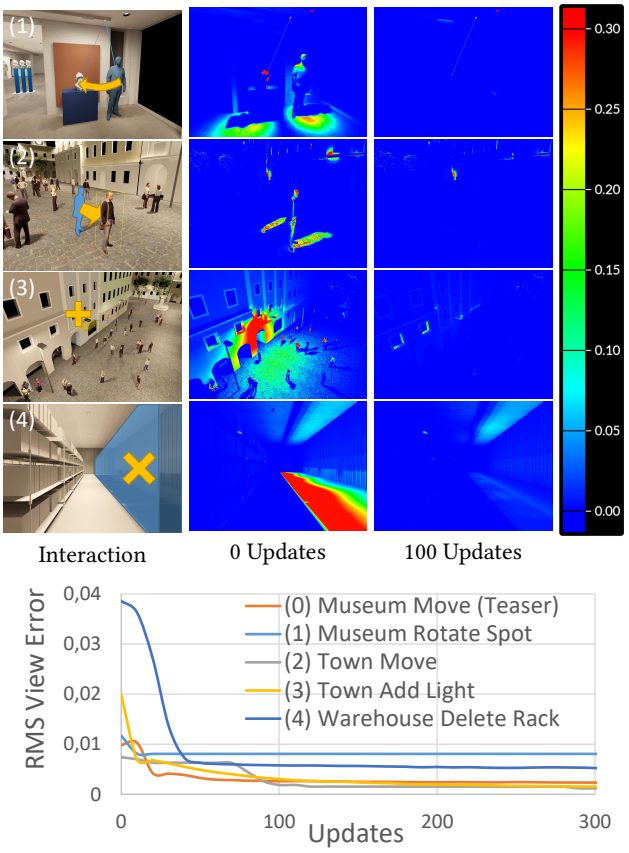

Figure 11: Top: 4 interaction scenarios with their initial illumination error, and the remaining error after 100 updates using the depicted transfer function based on the squared pixel difference. Bottom: The convergence of the viewport RMS error to the completely validated illumination.

Figure 11 (top) illustrates 4 interaction scenarios and (bottom) a graph with their corresponding convergence of illumination updates to the completely updated solution. Typically, the direct light and clusters that accumulated a significant amount of invalid radiant flux are validated within the first couple of updates. Even in severe editing operations (Scenario 4) most of the illumination error is resolved within the first 1-2 seconds. Indirect light converges then at a reduced pace and can require > 1000 updates in the Warehouse or Town Square scene, where lots of clusters have visibility of the interactions' location. In general, the effectiveness of our prioritization is limited by the number of scattered paths per cluster, but increasing the number of paths and keeping the same cluster count does not significantly impact the effectiveness in the evaluated scenarios. In addition, the accompanying videos extensively demonstrate the update performance in scenarios where multiple successive interactions are performed. Adding new objects and illumination of the footprint covered by moved objects turn out to be worst case scenarios, where the illumination error is resolved slowly while the complete validation is progressing.

7 CONCLUSIONS AND FUTURE WORK

Our *Incrementally Baked Global Illumination* approach is a successful method to maintain visual coherence in complex lighting design scenarios with a high demand on accuracy. We have demonstrated that it is possible to track changes in the GI caused by scene modifications, and compute and visualize a seamless transition from one illumination state to another without a significant loss in precision. This successfully avoids a global recomputation, and therefore flickering artifacts or noise based on a novel cluster balancing strategy.

Our approach could be used with other data structures that hold a baked GI, as long as there is a similarly fast accumulation mechanism for virtual lights. We therefore hope that the capabilities of our proposed system will find their way into today's real-time rendering engines, enabling further applications.

Advances in performance of GPU ray-tracing and integration into graphics APIs makes it interesting to replace shadow mapping and thereby gaining flexibility. Extending our system to maintain and incrementally update multiple locally selected cluster sets would be a challenging future work. Additionally, incorporating view dependency into our update prioritization strategy could enhance the visual coherence even more.

ACKNOWLEDGMENTS

We wish to express our thanks to Markus Billeter and Baran Usta (CGV group at TU Delft) for their insightful and valuable comments on this work. We also thank Bert Junghans, Christian Bauer and Holger Leibmann from Zumtobel for the ongoing fruitful collaboration and for providing the scenes for evaluating this work.

VRVis is funded by BMVIT, BMDW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174) which is managed by FFG.

REFERENCES

Tomáš Barák, Jiří Bittner, and Vlastimil Havran. 2013. Temporally Coherent Adaptive Sampling for Imperfect Shadow Maps. *Computer Graphics Forum (Proceedings of EGSR 2013)* 32, 4 (2013), 87–96.

Daniel R. Baum, Holly. E. Rushmeier, and James M. Winget. 1989. Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. ACM, New York, USA, 325–334. https://doi.org/10.1145/74333.74367

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. 2011. Interactive Indirect Illumination Using Voxel Cone Tracing: A Preview. In *Symposium on Interactive 3D Graphics and Games (I3D '11)*. ACM, New York, NY, USA, 207–207. https://doi.org/10.1145/1944745.1944787

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable Realistic Rendering with Many-Light Methods. *Comput. Graph. Forum* 33, 1 (Feb. 2014), 88–104. https://doi.org/10.1111/cgf.12256

Carsten Dachsbacher and Marc Stamminger. 2005. Reflective Shadow Maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games (I3D '05)*. ACM, New York, NY, USA, 203–231. https://doi.org/10.1145/1053427.1053460

Zhao Dong, Thorsten Grosch, Tobias Ritschel, Jan Kautz, and Hans-Peter Seidel. 2009. Real-time Indirect Illumination with Clustered Visibility. In *Proceedings of the Vision, Modeling, and Visualization Workshop 2009, November 16-18, 2009, Braunschweig, Germany*. DNB, Leipzig, Germany, 187–196.

Iliyan Georgiev and Philipp Slusallek. 2010. Simple and Robust Iterative Importance Sampling of Virtual Point Lights. In *Eurographics 2010 - Short Papers*, H. P. A. Lensch and S. Seipel (Eds.). The Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 4. https://doi.org/10.2312/egsh.20101047

Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. 2009. Virtual Spherical Lights for Many-light Rendering of Glossy Scenes. In *ACM SIGGRAPH Asia 2009 Papers (SIGGRAPH Asia '09)*. ACM, New York, NY, USA, Article 143, 6 pages. https://doi.org/10.1145/1661412.1618489

Miloš Hašan, Fabio Pellacini, and Kavita Bala. 2007. Matrix Row-column Sampling for the Many-light Problem. In *ACM SIGGRAPH 2007 Papers (SIGGRAPH '07)*. ACM, New York, NY, USA, Article 26, 10 pages. https://doi.org/10.1145/1275808.1276410

Peter Hedman, Tero Karras, and Jaakko Lehtinen. 2016. Sequential Monte Carlo Instant Radiosity. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '16)*. ACM, New York, NY, USA, 121–128. https://doi.org/10.1145/2856400.2856406

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

James T. Kajiya. 1986. The Rendering Equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. ACM, New York, NY, USA, 143–150. https://doi.org/10.1145/15922.15902

Anton Kaplanyan and Carsten Dachsbacher. 2010. Cascaded Light Propagation Volumes for Real-time Indirect Illumination. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '10)*. ACM, New York, NY, USA, 99–107. https://doi.org/10.1145/1730804.1730821

Alexander Keller. 1997. Instant Radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 49–56. https://doi.org/10.1145/258734.258769

Samuli Laine, Hannu Saransaari, Janne Kontkanen, Jaakko Lehtinen, and Timo Aila. 2007. Incremental Instant Radiosity for Real-time Indirect Illumination. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques (EGSR'07)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 277–286. https://doi.org/10.2312/EGWR/EGSR07/277-286

Christian Luksch, Robert F. Tobler, Ralf Habel, Michael Schwärzler, and Michael Wimmer. 2013. Fast Light-Map Computation with Virtual Polygon Lights. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2013*. ACM, New York, NY, USA, 87–94. https://doi.org/10.1145/2448196.2448210

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

Ola Olsson, Markus Billeter, Erik Sintorn, Viktor Kämpe, and Ulf Assarsson. 2015. More Efficient Virtual Shadow Maps for Many Lights. *IEEE Transactions on Visualization and Computer Graphics* 21, 6 (June 2015), 701–713. https://doi.org/10.1109/TVCG.2015.2418772

Jiawei Ou and Fabio Pellacini. 2011. LightSlice: Matrix Slice Sampling for the Many-lights Problem. In *Proceedings of the 2011 SIGGRAPH Asia Conference (SA '11)*. ACM, New York, NY, USA, Article 179, 8 pages. https://doi.org/10.1145/2024156.2024213

Roman Prutkin, Anton Kaplanyan, and Carsten Dachsbacher. 2012. Reflective Shadow Map Clustering for Real-Time Global Illumination. In *Eurographics (Short Papers)*, Carlos Andújar and Enrico Puppo (Eds.). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 9–12. http://dblp.uni-trier.de/db/conf/eurographics/eg-short2012.html#PrutkinKD12

Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. 2012. The State of the Art in Interactive Global Illumination. *Comput. Graph. Forum* 31, 1 (Feb. 2012), 160–188. https://doi.org/10.1111/j.1467-8659.2012.02093.x

Tobias Ritschel, Elmar Eisemann, Inwoo Ha, James Dokyoon Kim, and Hans-Peter Seidel. 2011. Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes. *Comput. Graph. Forum* 30 (2011), 2258–2269.

Tobias Ritschel, Thorsten. Grosch, Min. H. Kim, Hans-Peter Seidel, Carsten. Dachsbacher, and Jan Kautz. 2008. Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. In *ACM SIGGRAPH Asia 2008 Papers (SIGGRAPH Asia '08)*. ACM, New York, NY, USA, Article 129, 8 pages. https://doi.org/10.1145/1457515.1409082

Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. 2012. Practical Hessian-Based Error Control for Irradiance Caching. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (nov 2012), 193:1–193:10. https://doi.org/10.1145/2366145.2366212

Ari Silvennoinen and Jaakko Lehtinen. 2017. Real-time Global Illumination by Precomputed Local Reconstruction from Sparse Radiance Probes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 36, 6 (Nov. 2017), 230:1–230:13. https://doi.org/10.1145/3130800.3130852

Sinje Thiedemann, Niklas Henrich, Thorsten Grosch, and Stefan Müller. 2011. Voxel-based Global Illumination. In *Symposium on Interactive 3D Graphics and Games (I3D '11)*. ACM, New York, NY, USA, 103–110. https://doi.org/10.1145/1944745.1944763

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: A Scalable Approach to Illumination. In *ACM SIGGRAPH 2005 Papers*. ACM, New York, USA, 1098–1107. https://doi.org/10.1145/1186822.1073318

Bruce Walter, Pramook Khungurn, and Kavita Bala. 2012. Bidirectional Lightcuts. *ACM Trans. Graph.* 31, 4, Article 59 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185555

Lance Williams. 1978. Casting Curved Shadows on Curved Surfaces. *SIGGRAPH Comput. Graph.* 12, 3 (Aug. 1978), 270–274. https://doi.org/10.1145/965139.807402

Can Yuksel and Cem Yuksel. 2017. Lighting Grid Hierarchy for Self-illuminating Explosions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4, Article 110 (2017), 10 pages. https://doi.org/10.1145/3072959.3073604