Implicit Visibility and Antiradiance for Interactive Global Illumination

Carsten Dachsbacher
REVES/INRIA Sophia-Antipolis

Marc Stamminger
University of Erlangen

George Drettakis
REVES/INRIA Sophia-Antipolis

Frédo Durand
MIT CSAIL

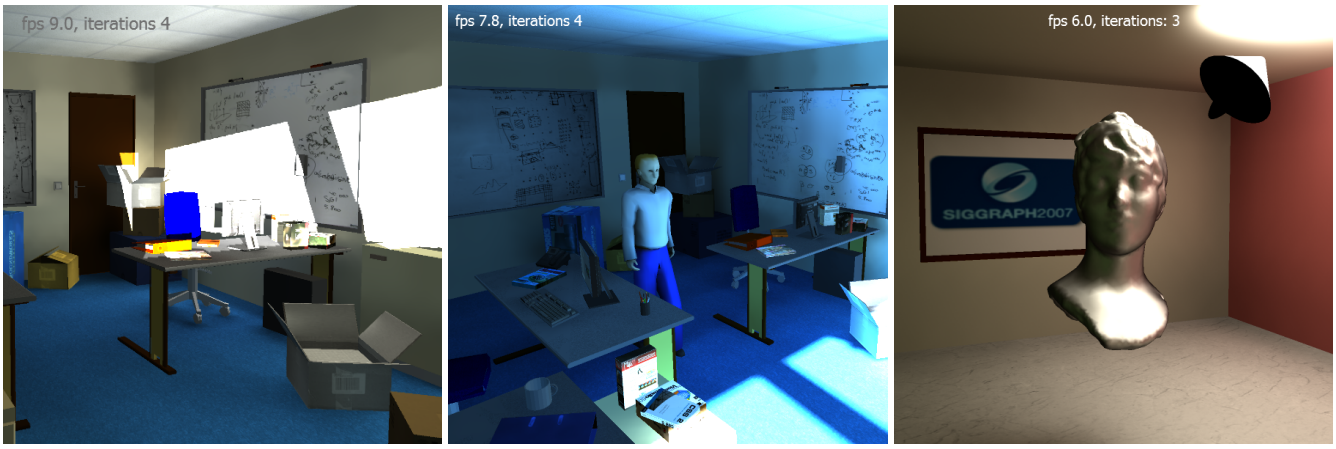

Figure 1: Left: An office which receives light through a window on the right wall while the rest of the scene is lit indirectly. Our method avoids explicit visibility computation and our GPU implementation enables the manipulation of the light or objects at 9 frames per second (fps). This includes four iterations of radiance and antiradiance per frame. Center: We can also include an animated character; indirect light is updated at 7.8 fps. Right: Our method naturally handles glossy surfaces; the glossy bust is lit indirectly, since the light points at the ceiling.

Abstract

We reformulate the rendering equation to alleviate the need for explicit visibility computation, thus enabling interactive global illumination on graphics hardware. This is achieved by treating visibility *implicitly* and propagating an additional quantity, called *antiradiance*, to compensate for light transmitted extraneously. Our new algorithm shifts visibility computation to simple local iterations by maintaining additional directional antiradiance information with samples in the scene. It is easy to parallelize on a GPU. By correctly treating discretization and filtering, we can compute indirect illumination in scenes with dynamic objects much faster than traditional methods. Our results show interactive update of indirect illumination with moving characters and lights.

Keywords: Global illumination, visibility, GPU

1 Introduction

Interactive global illumination is still an elusive goal, despite an abundance of heuristic solutions in recent years. The main expense in most methods is the cost of visibility; a primary goal of this paper is to enable global illumination without requiring explicit visibility computation. We start from the observation that if visibility is ignored, some light is transmitted extraneously through opaque objects and needs to be canceled out. This is why we introduce a new quantity called *antiradiance* that corresponds to light that needs to be removed. We present a reformulation of the rendering equation that requires the consideration of both radiance and antiradiance but enables the treatment of visibility in an *implicit* manner.

Our reformulation dramatically simplifies the treatment of pairwise interaction between objects: the propagation of radiance and antiradiance between two objects does not depend on occlusion due to other objects, which greatly facilitates parallelization on graphics hardware. The price to pay is that we now need to simulate two quantities, radiance and antiradiance, and the directional distribution of antiradiance must be considered. However, for indirect illumination (or any low frequency illumination), visibility quickly becomes "averaged out" after a few bounces of light and the directional discretization of antiradiance does not need to be extremely fine. For high-frequency direct lighting, we can use a separate traditional technique such as shadow maps. Furthermore, the treatment of directional quantities is desirable to handle glossy materials, although they incur extra cost for the local shading integral.

Our paper makes the following contributions:

A reformulation of the rendering equation, where visibility is implicit and antiradiance compensates for extraneous transport.

Two iterative solutions, one that provably converges but requires more iterations, and one that converges in practice and is cheaper.

Hierarchical discretization based on a spatial and directional data structure, with appropriate refinement and pre-filtering.

An efficient GPU implementation, which allows real-time global illumination for scenes which are hard to treat with all previous methods (moving objects and lights, varying material properties, mainly indirect illumination).

We show that our method can compute solutions which are close to reference images from a path-tracer [Pharr and Humphreys 2004], and illustrate our algorithm on several test examples.

1.1 Related Work

Global illumination is a vast field and excellent introductory books exist, e.g. [Dutré et al. 2006; Pharr and Humphreys 2004]. We only discuss work that is most directly related to our approach.

"Negative light" was used to allow for incremental radiosity updates, where it compensates for the different visibility configuration between two frames [Buckalew and Fussell 1989; Puech et al. 1990; Chen 1990]. Shadow photons [Jensen and Christensen 1995] are also related to the idea of negative light. Note that visibility computation is still required, in contrast to our use of antiradiance. In particular, methods for the efficient update of global illumination [Shaw 1997; Drettakis and Sillion 1997] require searching for regions that need recomputation of visibility. Our reformulation avoids this search and does not require any explicit visibility computation.

Bunnel [2005] uses "negative light" to approximate ambient occlusion and simulates the effect of inter-reflection. In contrast to our work, his negative light is not directional, and this heuristic does not respect the rendering equation.

The most related work is by Pellegrini [1999] who also derives a new rendering equation where explicit visibility is avoided. We use similar ideas where negative light is transmitted through each surface to compensate for the lack of occlusion treatment. However, his work does not demonstrate an implementation and does not discuss convergence and directional discretization issues.

A number of solutions for dynamic global illumination perform partial computations and use caching or reprojection of results of previous frames. An example is the Shading Cache [Tole et al. 2002]; this approach caches samples from a path tracer and uses graphics hardware for interpolation. However, these approaches still require visibility to compute the sparse samples.

Our approach builds on ideas from clustering methods for hierarchical radiosity [Sillion 1995; Smits et al. 1994], since we create links between clusters of objects to accelerate computations for more complex scenes. These approaches however require sophisticated data structures and global random-access visibility computations for form-factors. Extensions to this method stored directional radiance using wavelets [Stamminger et al. 2000] or spherical harmonics [Sillion et al. 1995]. In contrast, we store a directional information of "negative light", thus *avoiding* visibility computation.

Precomputed Radiance Transfer is a method to quickly compute the reflection integral for a given environment, e.g. [Sloan et al. 2002; Kautz et al. 2002; Sloan et al. 2005]. Local inter-reflection between an object and the environment can be approximated, e.g. [Sloan et al. 2002], but all these approaches require significant precomputation and do not simulate the balance of light due to global illumination. Our technique is complementary and combining our global simulation with precomputed local interaction is an exciting avenue of future work. Kristensen et al. [2005] simulate global illumination but for a fixed geometry and require substantial precomputation. Kontkanen et al. [2006] use a 4D discretization and a hierarchy, but they need to compute explicit visibility and require lengthy precomputation to treat static scenes only. Hasan et al. [2006] precompute a direct-to-indirect light transfer matrix for interactive GPU-based relighting, but are restricted to a fixed camera and static scene.

Ren et al. [2006] proposed a different reformulation of visibility using the exponentiation of spherical harmonics in the context of soft shadows. They however still need to determine the occluders between an object and the light and do not treat global illumination.

A number of authors have mapped global illumination algorithms on graphics hardware, e.g. [Coombe et al. 2004; Purcell et al. 2003]. However, they need to cope with visibility computations that are not GPU-friendly, raising important implementation challenges and performance limitations. Keller [1997] places a large number of virtual light sources, requiring visibility treatment, and accumulates their contribution with hardware shadow mapping and blending.

1.2 Overview

To achieve rapid and GPU-friendly global illumination, we get rid of explicit visibility computation such as ray casting and reorganize the traditional rendering equation by reformulating occlusion in an *implicit* manner. This allows us to consider pairwise object interaction as if all other objects were transparent and did not cause occlusion, thereby greatly simplifying computations.

To compensate for missing occlusion, every surface point emits incident light backwards as "negative light". We refer to this "negative light" as *antiradiance*. Antiradiance is propagated in the same way as normal radiance and cancels out the light that incorrectly traversed surfaces due to omitted occlusion. We thus propagate two quantities: radiance and antiradiance and it is critical to consider the directional distribution of antiradiance. At this price, we can replace the very costly occluded propagation of traditional global illumination by a much cheaper unoccluded propagation step.

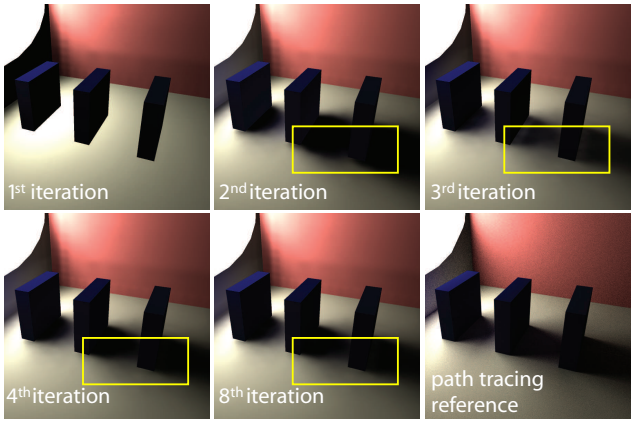

Figure 2: Top, left to right: after one iteration there are no shadows, since antiradiance has not been propagated. At two iterations shadows are too dark; after three they are too bright (yellow box). Bottom: with 4 iterations the solution is almost converged, compared to 8 iterations and the path-traced reference.

We illustrate the basic concept in Fig. 2. We formalize this process by reformulating the rendering equation and demonstrating the correctness of this reformulation. We also provide an iterative solution method, and discuss speed and convergence choices.

Since antiradiance is a directional quantity, we use spatial and directional finite elements to solve the system of equations. We use appropriate pre-filtering in space and directions for form-factor computation, and develop a hierarchical solution to deal with complexity. Our solution has been designed to be stream-processor friendly. The actual GPU implementation includes several important design choices, allowing us to achieve interactive global illumination updates, with moving objects and moving lights.

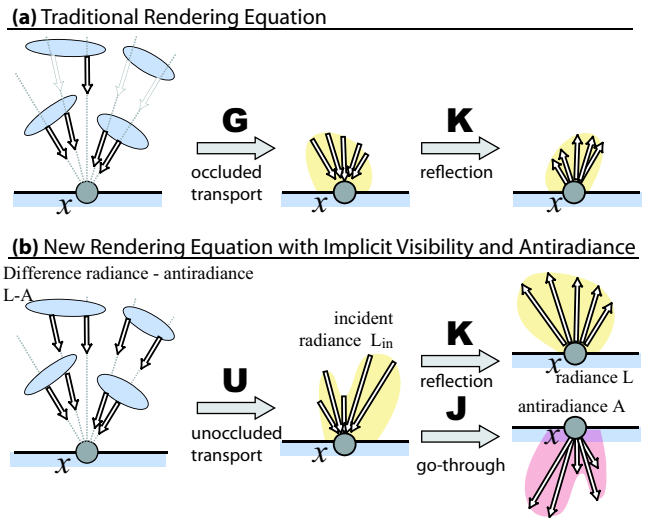

Figure 3: (a) Operator formalization of the rendering equation. (b) Our reformulation uses unoccluded transport but creates antiradiance to compensate for extraneous transport.

2 Reformulating the Rendering Equation

Traditional rendering equation The Rendering Equation [Kajiya 1986] can be expressed in terms of linear operators that act on radiance $L(x,\omega)$, where x is a point in space and ω a direction [Arvo et al. 1994] (Fig. 3). Please note that we follow the convention of [Arvo et al. 1994] for the directions and signs in the operator formulation. The *reflection* operator $\mathbf{K}$ takes incident radiance L_{in} and performs the shading integral to output outgoing radiance L at each point x,

$$(\mathbf{K}L_{\text{in}})(x,\omega) = \int_{\Omega_{n_x}} f(x;\omega_{\text{in}} \to \omega) L_{\text{in}}(x,\omega_{\text{in}}) \langle n_x | -\omega_{\text{in}} \rangle d\omega_{\text{in}} \quad (1)$$

where n_x is the normal at x, f the BRDF and $\langle|\rangle$ the dot product. Incident light L_{in} is obtained through the (occluded) *geometry* operator $\mathbf{G}$ which uses an explicit visibility function $\text{ray}(x,\omega)$ that determines the first front-facing point starting at x in direction $-\omega$ (rays that reach no hit point return a virtual black point p_{inf}):

$$L_{\text{in}}(x,\omega) = (\mathbf{G}L)(x,\omega) = L(\text{ray}(x,\omega),\omega) \quad (2)$$

$E(x,\omega)$ is the self-emission, and the rendering equation becomes:

$$L(x,\omega) = E(x,\omega) + (\mathbf{K}\mathbf{G}L)(x,\omega). \quad (3)$$

2.1 Reformulation

We seek to replace the transport operator $\mathbf{G}$ by an operator $\mathbf{U}$ that does not require explicit visibility. That is, we define $\mathbf{U}$ using a modified ray function RAY that returns the *set of all* intersection points (Fig. 3b) in direction $-\omega$:

$$L_{\text{in}}^{\text{unocc}}(x,\omega) = (\mathbf{U}L)(x,\omega) = \sum_{y \in \text{RAY}(x,\omega)} L(y,\omega) \quad (4)$$

Informally, using $\mathbf{U}$ directly would propagate excessive light and we need to compensate for it. This is why we simulate a second quantity, *antiradiance*, that cancels out this extraneous radiance. Note that each time light reaches a surface, it gets extraneously propagated through it by $\mathbf{U}$. This is why we create antiradiance that corresponds exactly to the amount of light incident to an object. In this manner, the light wrongly transmitted during a propagation gets canceled by antiradiance during the following step.

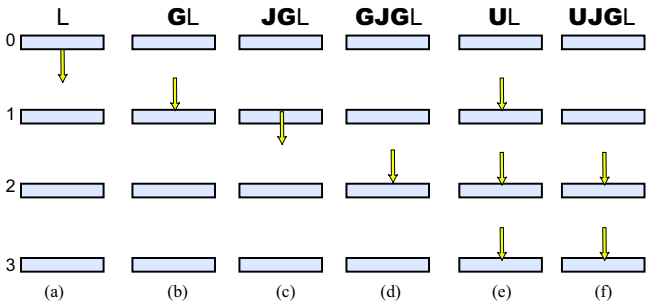

Figure 4: Radiance after unoccluded transport $\mathbf{U}$ is the sum of radiance after a series of occluded transport $\mathbf{G}$ and "go-through" $\mathbf{J}$.

We now formalize this and relate the unoccluded $\mathbf{U}$ and occluded $\mathbf{G}$ operators. To make the fact that $\mathbf{U}$ ignores occlusion explicit, we define a "go-through" operator $\mathbf{J}$ that lets incident light through opaque objects (Fig. 3b lower right). It is essentially the identity:

$$(\mathbf{J}L_{\text{in}})(x,\omega) = L_{\text{in}}(x,\omega). \quad (5)$$

With the help of $\mathbf{J}$, we can describe the relation between $\mathbf{G}$ and $\mathbf{U}$. The idea is shown in Fig. 4 with L a Dirac impulse at the top patch 0. $\mathbf{G}L$ is non-zero at patch 1 only. By letting light pass ($\mathbf{J}$) and propagate it again using $\mathbf{G}$, it reaches patch 2. Every further application of $\mathbf{GJ}$ propagates light one layer further. $\mathbf{U}$ is finally the sum over all layer depths:

$$\mathbf{U}L = \mathbf{G}L + \mathbf{GJG}L + \mathbf{GJGJG}L + \ldots \quad (6)$$

where the terms become zero when $\mathbf{G}(\mathbf{JG})^i$ reaches scene depth. This gives us the fundamental equation that relates $\mathbf{G}$ and $\mathbf{U}$:

$$\begin{aligned} \mathbf{U}L &= \mathbf{G}L + (\mathbf{G} + \mathbf{GJG} + \ldots)\mathbf{JG}L \\ &= \mathbf{G}L + \mathbf{UJG}L \\ \to \mathbf{G}L &= \mathbf{U}L - \mathbf{UJG}L \end{aligned} \quad (7)$$

where the left (positive) term $\mathbf{U}L$ is unoccluded propagation of radiance and the right term $-\mathbf{UJG}L$ propagates "negative light" to compensate for the lack of occlusion. We call $\mathbf{JG}L$ *antiradiance*, and denote it by A:

$$A = \mathbf{JG}L \quad (8)$$

Our definition of antiradiance still includes visibility, but we can replace $\mathbf{G}L$ by Eq. 7 to get a recursive equation for A without $\mathbf{G}$:

$$\begin{aligned} A &= \mathbf{J}(\mathbf{U}L - \mathbf{UJG}L) \\ &= \mathbf{JU}(L - A) \end{aligned} \quad (9)$$

This equation allows us to determine the antiradiance field A for any radiance distribution L. We start with $A = 0$ and iteratively update A. After n iterations, where n is the maximum scene depth complexity, the computation converges and we obtain the antiradiance distribution A corresponding to L by an iteration using $\mathbf{U}$ (instead of a single application of $\mathbf{G}$). With the knowledge of A, we can compute one iteration step of classical light transport:

$$\mathbf{G}L = \mathbf{U}L - \mathbf{UJG}L = \mathbf{U}(L - A) \quad (10)$$

We finally reformulate the rendering equation (3) by replacing **G** with its expression in terms of **U** and antiradiance A (Eq. 7, 8)

$$L = E + \mathbf{KG}L = E + \mathbf{KU}(L - A),$$

i.e., we replace **G** by **U** and propagate $L - A$ instead of L. Our new rendering equation is

$$L = E + \mathbf{K}\,\mathbf{U}(L - A) \quad (11)$$
$$A = \mathbf{J}\,\mathbf{U}(L - A). \quad (12)$$

Note the similarities between Eqs. 11 and 12. Both are recursive and propagate $L - A$ using the unoccluded transport **U**. L is reflected via a BRDF in **K**, antiradiance A with the simple operator **J**, which is essentially the BRDF of a transparent surface. The recursion is interdependent, since L and A are defined based on each other.

2.2 Iterative Solution

Before presenting our finite-element solution to the new rendering equation, we discuss iterative strategies and convergence. We propose two schemes that differ in the relative number of steps of Eqs. 11 and 12. The first scheme has been described in the previous derivation and is asymmetric. It applies one radiance propagation and then antiradiance steps until convergence to emulate one step of $\mathbf{G}L$. It corresponds to traditional global illumination and hence provably converges. The second method is symmetric and alternates Eqs. 11 and 12. While we cannot formally prove convergence, our experiments have shown that it is stable in practice; we present our intuition on why this is the case.

Asymmetric Iteration In our first technique, after each step of radiance propagation (Eq. 11), we iterate over Eq. 12 to propagate antiradiance until convergence. So after one step of

$$L^{(i+1)} = E + \mathbf{K}\,\mathbf{U}(L^{(i)} - A^{(j)}), \quad (13)$$

we apply multiple steps of

$$A^{(j+1)} = \mathbf{J}\,\mathbf{U}(L^{(i)} - A^{(j)}), \quad (14)$$

where $L^{(i)}$ is frozen while j iterates. The antiradiance iteration effectively computes $A = \mathbf{JG}L$; the required number of iterations is bound by the scene's depth complexity.

Symmetric Iteration In our second technique, we solve the joint equations 11 and 12 by iteratively propagating radiance (Eq. 13) and antiradiance (Eq. 14). Such a scheme converges only if the iteration matrix does not have eigenvalues with magnitude higher than one. In traditional radiosity, this can be easily proven because the matrix is diagonal-dominant. We cannot make such a claim here, because the unoccluded transport operator **U** can create energy.

In fact, in simplified 2D examples with multiple occlusion and perfect mirror-BRDFs, eigenvalues larger than one occur. However, in practice the symmetric propagation seems to converge faster and we did not observe divergence. We hypothesize that there is much natural dampening due to the shading integral. Highly-specular BRDFs might create more problems and deserve further investigation in future work.

Discussion We measured convergence for the office scene from Fig. 1. The average diffuse reflectance in the scene is 0.66. The results are shown in Fig. 5. In the asymmetric iteration we alternately apply one radiance propagation and four antiradiance propagation steps, which become visible as steps in the graph.

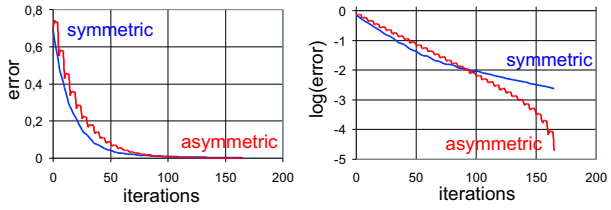

Figure 5: Convergence comparison of asymmetric (red) and symmetric iteration (blue) for our office scene (Fig. 1). (a) linear plot. (b) log plot.

For both schemes, we did more than 300 iterations. No divergence could be observed, the difference between both final results was less than 10^{-4}. The graph shows that the symmetric iteration is better in the beginning, however the asymmetric iteration overtakes after 90 steps. Visually, convergence is achieved after less than 10 iterations, and in this range the symmetric iteration has better behavior. However, the graph does not differentiate between antiradiance and radiance iteration steps. In particular for complex BRDFs antiradiance propagation is generally faster in computation time, because the application of **J** is very cheap compared to **K**. For the rest of this paper, we focus on the symmetric scheme.

3 Discrete Finite Element Solution

To solve our new rendering equation, we use spatial and directional finite elements. Our approach uses a hierarchy to handle complexity. The refinement strategy uses both spatial and directional information, and is well adapted to moving objects and lights. We also show how different direct lighting approaches can be used such as directional lights or environment maps.

3.1 Discretization

We discretize the scene into a hierarchy of patches P_i with centers x_i, and the space of directions into m *bins* Ω_j with main directions ω_j. Each bin covers a solid angle of $\omega_{bin} = 4\pi/n$, in practice bins span between 6 and 20 degrees.

For each patch P_i, we need to store a discretization of exitant radiance $L(x,\omega)$ and antiradiance $A(x,\omega)$. To make hierarchy maintenance simpler, it is convenient to use intensities $I_L(x_i,\omega_j)$ and $I_A(x_i,\omega_j)$, i.e. flux over solid angle. We only treat opaque solids, so that operators **K** and **J** update separate hemispheres (Fig. 6(a)). Thus we can store the quantity *exitant intensity* $I_{ex}(x_i,\omega_j) = I_L(x_i,\omega_j) - I_A(x_i,\omega_j)$ in a single data structure. We also store incident light as radiance $L_{in}(x_i,\omega_j)$. Finally, we store total intensity I_{total}, the final intensity used for display.

The iterative computation consists of two steps: in the *global pass* exitant intensity $I_{ex} = I_L - I_A$ is propagated from a sending to a receiving patch (depending on geometric properties of the two samples, but not on visibility) and contributes these to $L_{in}(x_i,\omega)$. This corresponds to the operator **U**. In the subsequent *local pass*, this incident radiance is transformed into exitant intensity and antiradiance for the next iteration, thus computing operators **K** and **J**.

Global pass The global pass is very similar to the link pass of standard radiosity, and in its simplest form it is an $O(n^2)$ process, because each of the n patches exchanges light with each other.

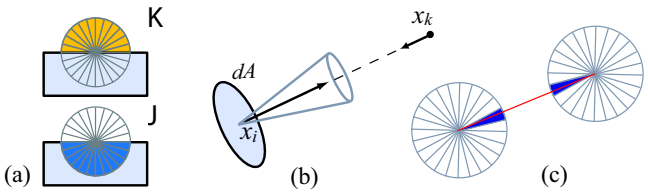

Figure 6: (a) We can store I_{ex}, which is the difference $I_L - I_A$ because the operators J and K affect separate hemispheres. (b) The interaction between x_i and x_k. (c) Finding the interacting bins.

Consider a particular pair of patches (P_i, P_k) with P_i being the receiver and P_k the sender. We first assume that, thanks to our hierarchical decomposition, patch P_k occupies only one directional bin as seen from the center of P_i; we lift this restriction later.

The emitted intensity at P_k towards P_i is $I^{i\leftarrow k} = I_{ex}(x_k, x_i \leftarrow x_k)$, where $x_i \leftarrow x_k$ is the direction linking the centers of the two patches. To compute the influence at x_i, we position a virtual infinitesimal surface element dA at x_i that directly faces x_k (see Fig. 6 (b)). The flux from x_k emitted towards dA is then:

$$d\Phi = \frac{I^{i\leftarrow k} dA}{||x_i - x_k||^2}. \tag{15}$$

With our discretization, we first have to find the directional bins, via which sender and receiver interact (see Fig. 6 (c)). Let the directional bins at x_i and x_k, in which this light arrives, be ω_j and ω_l. We have to distribute the incident flux uniformly over this receiving bin, so the incident radiance at x_i is:

$$L_{\text{in}}^{ij\leftarrow kl}(x_i, \omega_j) = \frac{d\Phi}{dA\omega_{\text{bin}}} = \frac{I_{ex}(x_k, \omega_l)}{||x_i - x_k||^2 \omega_{\text{bin}}}. \tag{16}$$

The "form factor" coupling the exitant light at P_k and the incident light at P_i is thus:

$$F^{ij\leftarrow kl} = \frac{1}{||x_i - x_k||^2 \omega_{\text{bin}}}. \tag{17}$$

Note the difference to standard radiosity form factors: the cosine at the sender and the sender's area are implicitly stored in the intensity distribution $I(x_k, \omega_l)$. The cosine at the receiver is computed during the local pass described below.

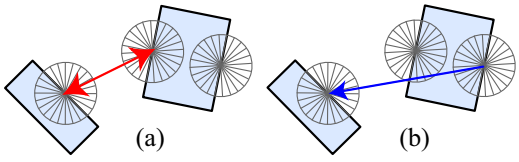

Figure 7: We only treat solids: (a) Bidirectional links are only established between bins both in the positive hemisphere. (b) Bins in the negative hemisphere only have outgoing links.

Interacting pairs of sender and receiver are stored as *links*. For storage convenience, we only consider solid objects. This implies that we establish bidirectional links between bins which are both in the positive hemisphere, while bins in the negative hemi-sphere only have outgoing links; see Fig. 7.

Up to now we computed the incident light at the center of P_i and assumed that the energy of the sender P_k is contracted to its center. The first assumption corresponds to point collocation in standard radiosity [Sillion and Puech 1994]. The latter assumption, however, can lead to artifacts if the sender patch covers a large solid angle and spreads over several bins. Our oracle refines a link if the solid angle of the sender is too large. However, when maximum refinement or patch area thresholds are reached, this is not always the case.

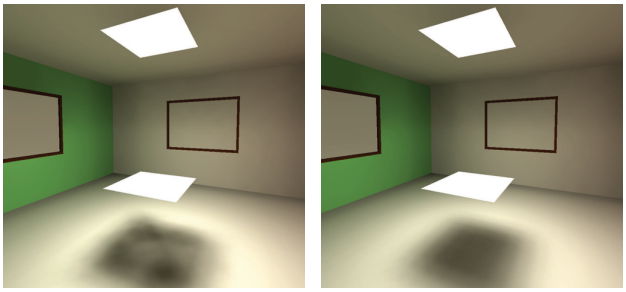

Figure 8: Image (a) without filtering (b) with filtering. We use 4x4 samples on the blocker, accentuating the effect for illustration.

For an accurate CPU implementation, we advocate the precise computation of exchanges between all pairs of bins in the cone subtended by the sender as seen from the receiver. However, for graphics hardware, we need to make computations as local as possible and use the following heuristic: for each link, we compute the solid angle of the sender. If this is larger than that of a bin, we "spread" and filter its energy to the bin's neighbors. That is, we select a circular neighborhood of size σ, the solid angle subtended by the receiver and in a sense filter the form factor with a Gaussian blur similar to prefiltering for antialiasing. Assuming an optimal uniform bin distribution the angle between a bin and its closest neighbor is $\sqrt{\frac{4\pi}{m}}$; this is the standard deviation of our Gaussian. This means that the appropriate bins receive energy, but that this energy is taken from the wrong bin: that defined by the direction between the patch centers. However, link refinement ensures that an exchange usually covers only one or very few bins; Fig. 8 demonstrates the effectiveness of this heuristic.

Local pass In the local pass, we transform incident radiance L_{in} to reflected radiance and antiradiance, represented as intensities I_L and I_A. The reflection is defined by Eq. 1, however we have to transform exitant radiance to intensity. At this step, the surface area a_i represented by a sample i is needed as well as the sample's normal n_i. We can then derive:

$$\begin{aligned} I_L(x_i, \omega_j) &= a_i \langle n_i | \omega_j \rangle L(x_i, \omega_j) \\ &= a_i \langle n_i | \omega_j \rangle \omega_{\text{bin}} \textstyle\sum_{l=0}^{m-1} f(x_i, \omega_l \rightarrow \omega_j) \langle n_i | - \omega_l \rangle L_{\text{in}}(x_i, \omega_l) \end{aligned} \tag{18}$$

For antiradiance, we have to transfer the light incident from bin ω_j to the opposite direction and transform it to intensity:

$$I_A(x_i, \omega_j) = a_i \langle n_i | - \omega_j \rangle L_{\text{in}}(x_i, \omega_j) \tag{19}$$

3.2 Hierarchy and Refinement

To make computations tractable, we solve the above finite element problem hierarchically in a spirit similar to Hierarchical Radiosity with Clustering [Smits et al. 1994; Sillion 1995]. We use a hierarchy over the scene and simulate exchanges through links at the appropriate level. The refinement is the same for both quantities.

Our algorithm starts with the creation of links, which is the only step which must happen on the CPU since it requires random global

access to the scene. We start by considering exchanges from the root cluster to itself. These self-links are always subdivided, i.e. replaced by links between all pairs of children. For a non-self-link we use a simple oracle for refinement: if the solid angle of the sender with respect to the receiver is larger than the solid angle of a bin of the directional discretization, its light will arrive at the sender within several bins of the receiver's incident light discretization. In this case, we subdivide the link, i.e. we subdivide sender and/or receiver and consider the links between the children recursively. Otherwise, we establish the link by computing its form factor and storing the link. By this, the epsilon threshold of Hierarchical Radiosity that controls the accuracy of the solution is linked to the directional discretization accuracy. Evidently, link refinement is stopped if sender and receiver become smaller than a discretization threshold. This step works very well with clustering: we do not need a normal and thus do not have to use average normals, nor consider self-occlusion.

Note that we do not consider the emitted light of the sender as this is usually done in radiosity (BF-refinement). This would make it necessary to refine links after each iteration, which does not fit to our GPU implementation.

Maintenance of the hierarchy The hierarchical representation must be maintained using a push-pull method. We push all light gathered in interior nodes to the leaves after the global pass. Then the local pass is performed at leaves. After the light has been reflected and antiradiance generated, the light is pulled up to update the inner nodes' light. Since we pull intensity, an inner node's intensity is simply the sum of its children's intensities.

3.3 Direct light and Dynamic Scenes

Direct Light Initialization Our approach is geared towards indirect illumination. For direct illumination, we can initialize the incident radiance of the samples in the scene, using one of the many existing methods. We can for example initialize all the L_{in} using an environment map. For direct shadows, we can use shadow maps or one of the more recent soft shadow methods [Guennebaud et al. 2006]. We evaluate direct lighting on each of the samples in the scene, for example with the shadow map, and initialize the exitant radiance. No antiradiance is generated in this pass, because shadowing of the direct illumination is done by the shadow map. During rendering, both parts of the lighting computation, fast per-pixel direct lighting with shadows and indirect lighting from the coarser finite-element illumination is summed in screen-space.

Moving Objects and Lights Moving or deforming objects are handled naturally and easily with our approach. Since we do not consider visibility, we only have to update links to and from moving objects, and we do not have to search for all links that are modified by changing visibility. The situation is more involved for hierarchical computations, because for a moving object the optimal link hierarchy changes with the position. However, if the position of moving objects can be reasonably bound, we can also bound the oracle and generate a link mesh based on the worst case. The link hierarchy thus does not need to be updated, but only the unoccluded form factors need to be recomputed, which in turn is a cheap operation. At the price of a less stringent oracle, the sender's emission usually also can be limited by reasonable bounds. One powerful feature of our approach is that moving lights can be handled in the same way, albeit with the same limitations.

4 Implementation on Graphics Hardware

One motivation for our approach is to obtain a GPU friendly global illumination algorithm. The global and local passes described above can be easily and efficiently performed on graphics hardware. The CPU only needs to compute the hierarchy of links which is cheap, because no visibility has to be computed. We now describe data layout and the mapping of the most important operations.

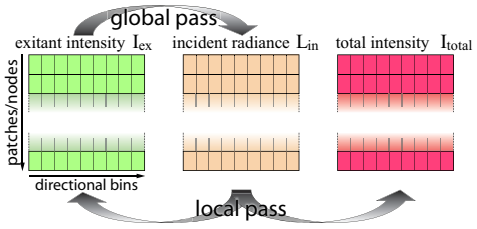

Figure 9: The global pass transfers exitant energy between elements. The local pass first reflects light and generates antiradiance and second tracks the element's intensity for display.

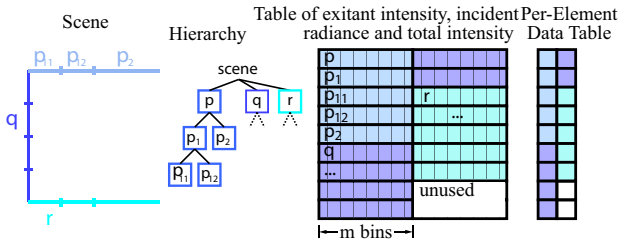

Figure 10: Left: a simple 2D scene and the corresponding hierarchy. Center: the layout of intensity/radiance tables. Right: Per-element attributes (position, normals, area) are stored in textures with the same layout.

4.1 Data Structures

The element hierarchy is laid out in depth-first order, which allows for efficient push/pull operations as we will see later. We use three tables with n rows (n being the number of elements) each with m (number of bins) columns, storing exitant intensity I_{ex}, incident radiance L_{in} and total intensity I_{total} for every element (Fig. 9). Note that when stored in textures the tables may be split into multiple texture columns as texture sizes are restricted by hardware (Fig. 10). All per-element attributes required for computations (normal, position, area, color, specular exponent) are stored in textures with the same layout but smaller width, so that we can use the same texture coordinates as we use for accessing the element bins.

Links are stored as render primitives in vertex buffers (Fig. 11, left) that includes the indices of the sender, receiver, and bin as well as the form factor value and the cone angle for filtering.

4.2 Operations

For each iteration, we perform the following four operations working on the aforementioned tables stored as textures and used as render targets. As GPUs do not allow the use of a render target as texture at the same time we use intermediate render targets or ping-pong rendering where necessary.

Global Pass In the global pass (Fig. 11) energy is transferred from one outgoing bin of a sender element, i.e. one entry in the I_{ex} table, to the bins of a receiver. Sender and receiver positions and sizes are used to find the bins used in our form-factor approximation (Sec. 3.1). The bins affected and appropriate filtering are

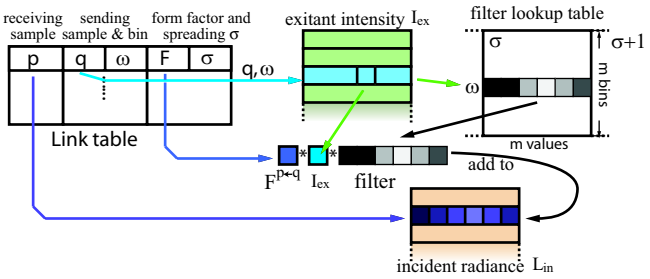

Figure 11: Energy transfer via a link by rendering a line primitive: the receiver index of p gives the position, sender index q and outgoing bin ω determine the bin and precomputed spreading/filtering.

precomputed for a finite number of cone sizes σ. For each σ we store an m^2 convolution table (Fig. 11 upper right): for each outgoing direction (rows), we store the contribution using the filtered form-factor to all bins of a receiver (columns). Some receiver bins may not receive energy; in this case the corresponding lookup table entry is zero.

Using this method, global energy transfer becomes rendering of the line covering all bins of the receiver element in L_{in}. As a result exitant energy weighted by the filter is transferred to the receiver bins as incident radiance (Fig. 11). For dynamic links, the affected bins and filter parameters are determined on the fly, otherwise they are computed once in the linking step.

Push Operation After the global transfer, we push energy down the hierarchy. Due to depth-first order all child elements are stored consecutively and we can add a node's radiance to several (or all) children at once by rendering quad primitives.

Local Pass For the local pass, we iterate over all bins of every element: we convolve the incident radiance with the BRDF to obtain the reflected energy and transform the incident radiance of the opposite bin to antiradiance. The reflected intensity is also accumulated in the I_{total} table, which is used for display. For diffuse surfaces, the BRDF computation can be accelerated: The reflected intensity in the normal direction is computed, stored in intermediate textures and used to compute reflected energy for all bins.

Pull Operation After the local pass we pull energy from the leaves to their parents. We read the intensity bins of a leaf element (from I_{ex}) and add it to the bins of all its predecessors in the hierarchy (in I_{ex}). This is achieved by rendering textured lines. Vertex positions are chosen such that the lines cover all receiver bins, while texture coordinates define the source bins. We store render primitive data in vertex and index buffers. Note that intensity I_{ex} is stored after weighting by element area and thus no further scaling is required.

4.3 Interpolation by Splatting

For final rendering, we need to interpolate the sample values to obtain a smooth solution. For shared vertex meshes, we compute the total intensity at the vertex locations and use linear hardware interpolation. For flat surfaces, we use a three pass rendering approach. Similarly to splatting for point-based rendering [Lavignotte and Paulin 2003; Ren et al. 2002], we start with a pass which writes surface IDs to a render target: we interpolate across surfaces patches sharing the same ID, which is determined after linking as a preprocess. In a second pass we render triangle fans centered at elements' centers and accumulate radiances weighted by a smooth falloff. The distance to neighbor elements can be used to deform the fans and better preserve local features. A third pass re-normalizes the radiance values by the total weights.

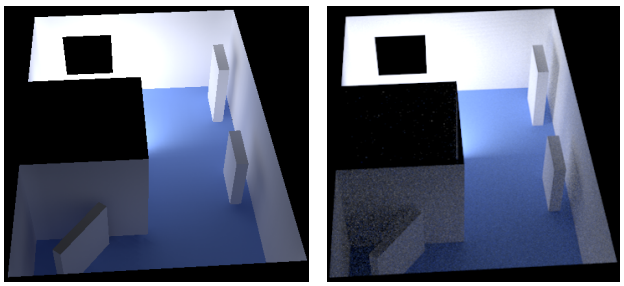

Figure 12: Left, our solution for an entirely indirect lighting situation: a directional light enters the window at the top of the scene; all shadows here are indirect. Right a path-tracing reference solution.

5 Results

All the tests were run on a Xeon processor running at 3Ghz, with an NVIDIA GeForce 8800GTX graphics card. Reference solutions are computed with PBRT [Pharr and Humphreys 2004], using the path-tracer module (faster algorithms exist but they tend to be biased and we focus here on accuracy).

Validation on Simple Scenes We present a few simple scenes to investigate some sampling issues and overall validation of our method. We first show a geometrically simple maze scene (Fig. 12), in which all lighting is indirect. A directional light enters from above, lighting the wall facing away from the viewer; all light and shadows in this image are indirect. This is a particularly tough case as can be seen with the noisy path-tracing solution, despite the 12K rays/pixel. Our solution shown here, with 1024 bins, spatial subdivision for a total of 6088 elements, and 8 iterations took 3min 45s (including linking) computed on the CPU, while the path tracing solution for the image (350×330 pixels) took 13h 39min.

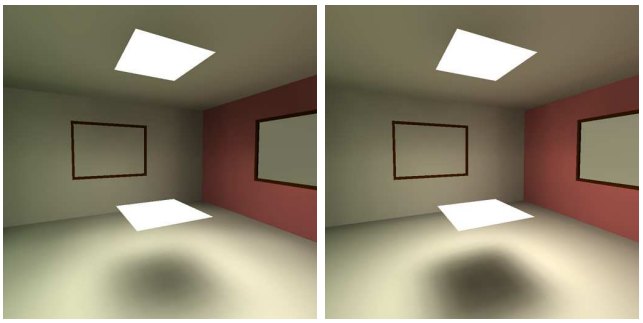

Figure 13: Effect of directional sampling. (a): 128 bins, (b) 512 bins; lower directional sampling rates result in blurring of shadows.

As with any finite element basis, sampling rate affects the result. The effect of directional sampling rate is intuitive, essentially resulting in blurring of shadows. Consider Fig. 13 in which the left image is computed with 128 bins and the right-hand image with 512 bins. We can clearly see the sharpening of shadow detail with higher directional sampling.

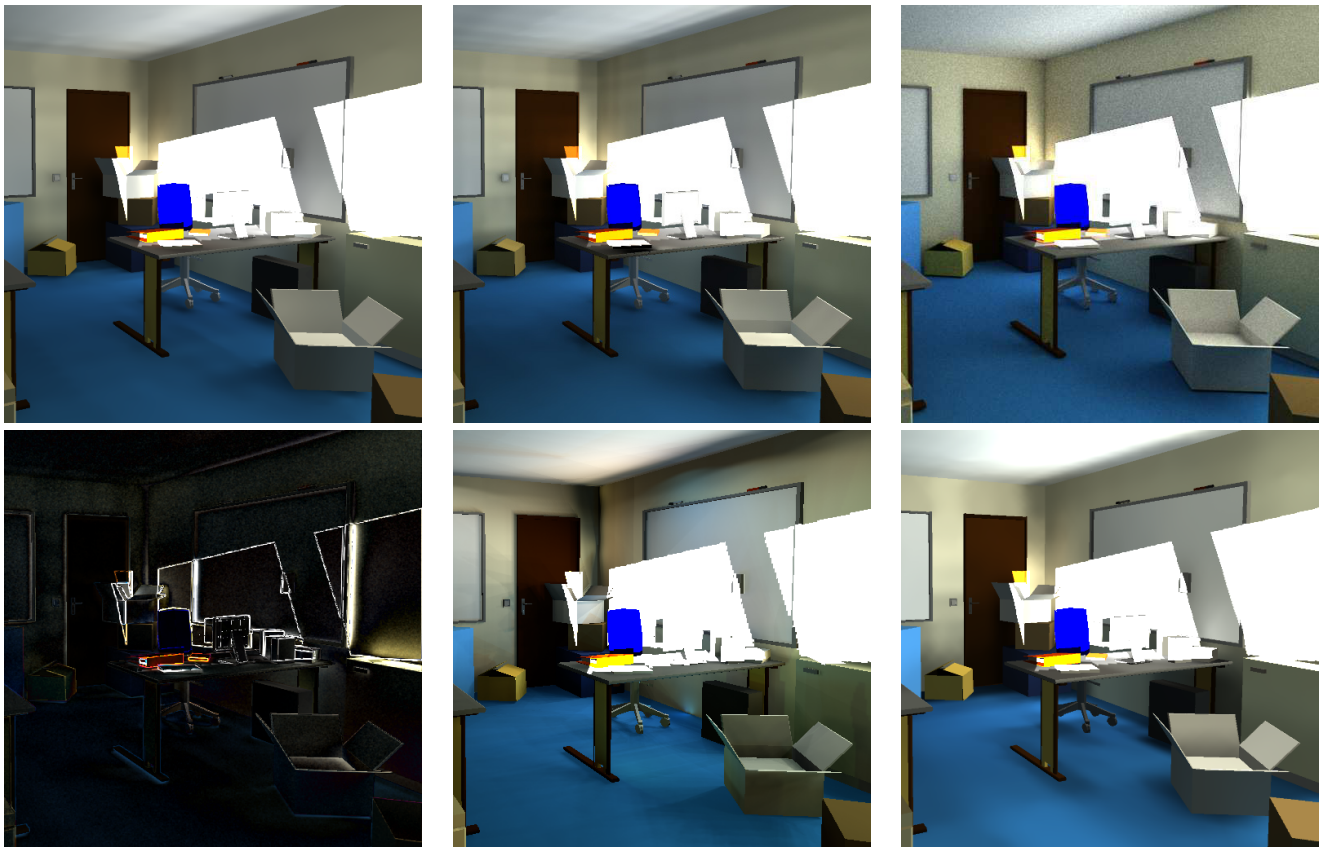

Figure 14: The untextured office scene for comparison (left to right, top to bottom): (a) The GPU solution at 9 fps. (b) A more refined solution computed on the CPU in 1m45s with 44,982 elements, 208 directions and 4.6 million links. (c) path-tracing reference, 2048 samples/pixel (path length 4). (d) absolute difference between GPU and path-tracing (variations at discontinuities mainly due to different anti-aliasing). (e) Indirect illumination computed with Instant Radiosity running at 9 fps. (f) Using the same link mesh as for Antiradiance, computing form-factors with visibility (141 rays per link) takes 80min, resulting in a solution akin to Hierarchical Radiosity with Clustering. Note that a visibility-based refinement would improve the quality.

Example Scenes We show three more involved example scenes. The first is an office scene, which receives a directional light coming through the window, illuminating the right wall. The interior is illuminated almost exclusively by indirect light. The second scene is a simple non-diffuse environment, with a glossy object. The third scene is a oriental living room (Fig. 16). We use shadow maps for direct light in the office and living room. Recordings of interactive sessions for all three sequences are shown in the video.

In Fig. 1 (left), we show the office scene. This scene contains 5,552 input polygons and is subdivided to 18,253 elements and 632,098 links. We use 128 bins for directions. We can move the position of the light, or any object in the scene at 7-10 fps for 4 iterations. For example, we can include an animated character with full interactive update of global illumination in the scene (center). This scene has 19,716 hierarchy elements, 660,783 links and 8,072 input polygons.

To compare the lighting solution against other methods, we chose to remove textures which tend to hide artifacts. A texture-free rendition of the office is shown in Fig. 14(a) (running at about 9.0fps), while Fig. 14(b) shows a solution on the CPU with improved lighting simulation; Fig. 14(c) is a path-traced solution computed in PBRT, which we use as a reference. This solution uses 2048 rays per pixel, a path length of 4, and took 8h 25min to compute and is still noisy. We also show the absolute difference between the GPU solution and the reference (Fig. 14(d)). An Instant Radiosity [Keller 1997] solution running at 9 fps with 190 virtual light sources (Fig. 14(e)) suffers from artifacts like bright spots, incorrect lighting levels, shadows which are too sharp, and banding. To generate Fig. 14(f) we used our link structure to compute a Hierarchical Radiosity with Clustering solution using explicit visibility which takes 80min.

The second scene includes a glossy object (3072 triangles) lit by a spot light. We use a Lafortune BRDF as implemented in PBRT. For low-frequency glossy lighting, our approach can achieve a good approximation (Fig. 15, left) compared to a reference solution computed by PBRT with path-tracing (Fig. 15, right). This example at interactive quality with 4550 elements and 177,000 links running at about 12 fps for 3 iterations is shown in the video. A more complex object (12,940 input polygons for the bust) is shown in Fig. 1 right. The glossy bust is lit indirectly, and we can move the spot light, change view point, and modify material parameters at about 6 fps.

We show statistics for all scenes in Tables 1 and 2. Table 2 shows the breakdown of computation time and memory usage. Unsurprisingly, the local pass dominates for glossy scenes, while it is the other way round for diffuse scenes. Overall GPU memory usage ($<$ 140Mb for all scenes, including optional look-up textures for speed-up) is reasonable for modern high-end graphics cards. On the GPU we store intensities and radiances as 16-bit floats and link data as 32-floats; no precision problems were observed.

Scene	T	E	L	N_i	fps	fps_s
Office	5.6K	18.2K	632K	4	10.1	9.0
Character	8.1K	19.7K	661K	4	8.3	7.1
Oriental	11.9k	15.6K	776K	4	7.5	6.8
Glossy box	3.5k	4.6K	177K	3	14.9	11.9
Glossy bust	13.4k	11.5K	412K	3	6.4	5.6

Table 1: Scene data and timings. T: number of input triangles, E: number of elements, L: number of links, N_i: number of iterations; 128 bins are used. fps_s: with splatting and 4x anti-aliasing. Shared vertex mesh elements are created at vertices yielding fewer samples.

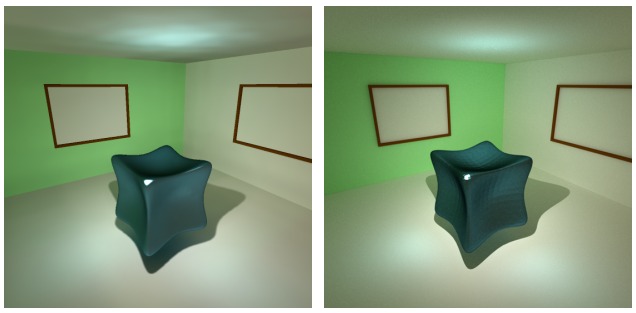

Figure 15: (a) A scene with a glossy object rendered with high quality settings (12.5K elements, 480K links, 8 iterations, 1.1 fps). Note the bluish directional indirect illumination from the glossy cube onto the floor and ceiling. (b) A PBRT reference solution using path-tracing with 8K rays/pixel computed in more than 40h.

6 Discussion

As illustrated in the results and in the accompanying video, our method provides fast interactive indirect lighting for interesting scenes. The main approximation in our approach is the directional discretization. For indirect light, the shadows and the overall lighting effects are generally smooth anyway, and thus this does not create too many visible artifacts. We consider that the comparison in overall lighting levels with our reference path-traced solutions is satisfactory for the interactive application context we target.

There are limitations in our current implementation: Increased memory usage for the directional data structures and excessive subdivision for dynamic scenes. At the numerical level, the convergence of the symmetric scheme cannot be ensured; however, the asymmetric one converges, albeit slower. The excessive subdivision for dynamic links is a design choice. In the context of a content creation workflow, we could imagine a tool which allows the designer to specify regions in which this should happen, based on knowledge of the usage scenarios. Update of the patch subdivisions and links can of course be made at runtime, but would involve CPU computations and upload to the GPU. Finally, in all the examples we have shown, the computation of 3-4 iterations is sufficient both in quality and in speed to avoid artifacts. Clearly, for more complex scenes, this could be a problem: hierarchical or space-subdivision schemes could potentially provide a solution to this issue.

Our algorithm shares some of the limitations of hierarchical radiosity methods in what concerns refinement and meshing. Both are hard problems; we hope that the availability of interactive global illumination using our approach will accelerate research in this area and hopefully open the way to practical solutions.

We could also handle transparent surfaces, however the operators J and K would interfere and could not share a single data structure thus increasing the memory required.

Our method is related to the idea of "negative light" (Sect. 1.1) recently used by Bunnell [2005] on the GPU. However we present a proper definition of "negative light" (antiradiance) and the appropriate theoretical basis showing that we solve the rendering equation. In addition, our algorithmic solution is completely different, since we store a directional representation for radiance/antiradiance, and we have reorganized light-transfer into global and local computation. Our hierarchical solution also permits the treatment of more complex scenes.

Figure 16: Oriental room scene; indirect light is updated at 6.8 fps.

7 Conclusion

We have presented a reformulation of the rendering equation which uses *implicit* visibility. This is achieved by using a new quantity, called *antiradiance*, which is propagated with radiance, and is stored in directional elements in the scene. This results in a reorganization of the computation and data structures which are local, and much easier to parallelize. As a result, we can achieve interactive updates of global illumination for moderately complex scenes, with moving objects and lights, and glossy reflectors.

In future work, we believe that the reformulation provides a different way of thinking about visibility, and could also lead to results in other applications in computer graphics such as occlusion culling.

Scene	IL(s)	LP(ms)	GP(ms)	PP(ms)	M_T(Mb)	M_L(Mb)
Office	8.7s	7.1	63.4	14.0	71.3	29.5
Character	13.1s	8.2	72.0	17.0	77.0	31.1
Oriental	13.0s	10.8	81.4	18.9	60.9	36.0
Glossy box	2.4s	43.4	2.4	2.8	17.8	3.8
Glossy bust	13.0s	100.9	5.6	6.6	44.9	8.8

Table 2: Timing and memory breakdown. IL is the time for initial linking. LP is the time (total, for all iterations) for the local pass, GP for the global pass, PP for push pull, M_T is texture memory for I_{ex}/L_{in} tables and ping-pong textures and M_L is the memory required for links (vertex buffers). Element attributes require less than 500Kb in all scenes.

A more theoretical analysis of the sampling and filtering issues would be interesting and should lead to more efficient and high-quality solutions. We are also considering more efficient rendering, extensions to more involved lighting phenomena and resolving the meshing/refinement issues discussed above.

Acknowledgements The first author initiated this work at the University of Erlangen, and was partially funded by the DFG project "Interaktive Visualisierung Prozeduraler Modelle"; he also acknowledges support of the Marie-Curie Fellowship "Scalable-GlobIllum" (MEIF-CT-2006-041306). F. Durand acknowledges a Microsoft Research New Faculty Fellowship, a Sloan fellowship and an NSF CAREER award 0447561 "Transient Signal Processing for Realistic Imagery." We thank Autodesk for the generous donation of Maya, A. Olivier, F. Firsching and P. Richard for modeling help, and P. Green for suggesting the term Antiradiance.

References

ARVO, J., TORRANCE, K., AND SMITS, B. 1994. A framework for the analysis of error in global illumination algorithms. In *SIGGRAPH '94*, 75–84.

BUCKALEW, C., AND FUSSELL, D. 1989. Illumination networks: Fast realistic rendering with general reflectance functions. In *SIGGRAPH '89*, 89–98.

BUNNELL, M. 2005. Dynamic ambient occlusion and indirect lighting. *GPU Gems 2:Programming Techniques for High Performance Graphics and General-Purpose Computation*, 223–233.

CHEN, S. E. 1990. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. In *SIGGRAPH'90*, 135–144.

COOMBE, G., HARRIS, M. J., AND LASTRA, A. 2004. Radiosity on graphics hardware. In *Proc. Graphics Interface'04*, 161–168.

DRETTAKIS, G., AND SILLION, F. 1997. Interactive update of global illumination using a line-space hierarchy. In *SIGGRAPH '97*, 57–64.

DUTRÉ, P., BEKAERT, P., AND BALA, K. 2006. *Advanced Global Illumination*. A K Peters, Natick, USA.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2006. Real-time soft shadow mapping by backprojection. In *Proc. EG Symposium on Rendering 2006*, 227–234.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2006. Direct-to-indirect transfer for cinematic relighting. *ACM Trans. on Graphics (SIGGRAPH'06) 25*, 3 (July), 1089–1097.

JENSEN, H. W., AND CHRISTENSEN, N. J. 1995. Efficiently Rendering Shadows Using the Photon Map. In *Compugraphics '95*, 285–291.

KAJIYA, J. T. 1986. The rendering equation. *SIGGRAPH '86 20*, 3, 143–150.

KAUTZ, J., SLOAN, P.-P., AND SNYDER, J. 2002. Fast, arbitrary BRDF shading for low-frequency lighting using Spherical Harmonics. In *Proc. EG Workshop on Rendering*, 291–296.

KELLER, A. 1997. Instant radiosity. In *SIGGRAPH '97*, 49–56.

KONTKANEN, J., TURQUIN, E., HOLZSCHUCH, N., AND SILLION, F. 2006. Wavelet radiance transport for interactive indirect lighting. In *Proc. EG Symposium on Rendering 2006*.

KRISTENSEN, A. W., AKENINE-MÖLLER, T., AND JENSEN, H. W. 2005. Precomputed local radiance transfer for real-time lighting design. *ACM Trans. on Graphics (SIGGRAPH'05) 24*, 3, 1208.

LAVIGNOTTE, F., AND PAULIN, M. 2003. Scalable photon splatting for global illumination. In *GRAPHITE '03*.

PELLEGRINI, M. 1999. Rendering equation revisited: How to avoid explicit visibility computations. In *SODA*, 725–733.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering from Theory to Implementation*. Morgan Kaufmann.

PUECH, C., SILLION, F., AND VEDEL, C. 1990. Improving interaction with radiosity-based lighting simulation programs. 51–57. Proc. SIGGRAPH Symposium on Interactive 3D Graphics'90.

PURCELL, T. J., DONNER, C., CAMMARANO, M., JENSEN, H. W., AND HANRAHAN, P. 2003. Photon mapping on programmable graphics hardware. In *Proc. of the SIGGRAPH/EG Conf. on Graphics Hardware*, 41–50.

REN, L., PFISTER, H., AND ZWICKER, M. 2002. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. *Comp. Graphics Forum (Proc. Eurographics'02) 21*, 3.

REN, Z., WANG, R., SNYDER, J., ZHOU, K., LIU, X., SUN, B., SLOAN, P.-P., BAO, H., PENG, Q., AND GUO, B. 2006. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Trans. on Graphics (SIGGRAPH'06) 25*, 3 (July), 977–986.

SHAW, E. 1997. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum 16*, 2, 107–118.

SILLION, F. X., AND PUECH, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, USA.

SILLION, F. X., DRETTAKIS, G., AND SOLER, C. 1995. A clustering algorithm for radiance calculation in general environments. In *Proc. EG Workshop on Rendering*, 196–205.

SILLION, F. X. 1995. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Trans. on Visualization and Computer Graphics 1*, 3, 240–254.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. on Graphics (SIGGRAPH'02) 21*, 3, 527–536.

SLOAN, P.-P., LUNA, B., AND SNYDER, J. 2005. Local, deformable precomputed radiance transfer. *ACM Trans. on Graphics (SIGGRAPH'05) 24*, 3, 1216–1224.

SMITS, B. E., ARVO, J., AND GREENBERG, D. P. 1994. A clustering algorithm for radiosity in complex environments. In *SIGGRAPH '94*, 435–442.

STAMMINGER, M., SCHEEL, A., GRANIER, X., PEREZ-CAZORLA, F., DRETTAKIS, G., AND SILLION, F. X. 2000. Efficient glossy global illumination with interactive viewing. *Computer Graphics Forum 19*, 1, 13–25.

TOLE, P., PELLACINI, F., WALTER, B., AND GREENBERG, D. P. 2002. Interactive global illumination in dynamic scenes. *ACM Trans. on Graphics (SIGGRAPH'02) 21*, 3, 537–546.