Deducing Explicit from Implicit Visibility for Global Illumination with Antiradiance

Gregor Mückl
University of Stuttgart
Allmandring 19
70569 Stuttgart, Germany
Gregor.Mueckl@visus.uni-stuttgart.de

Carsten Dachsbacher
Karlsruhe Institute of Technology
Am Fasanengarten 5
76131 Karlsruhe, Germany
dachsbacher@kit.edu

Abstract

The antiradiance method, a variant of radiosity, allows the computation of global illumination solutions without determining visibility between surface patches explicitly, unlike the original radiosity method. However, this method creates excessively many links between patches, since virtually all elements exchange positive and negative energy whose interplay replaces the visibility tests. In this paper we study how and if explicit visibility information can be recovered only by analyzing the link mesh to identify chains of links whose light transport cancels out. We describe heuristics to reduce the number of links by extracting visibility information, still without resorting to explicit visibility tests, e.g. using ray casting, and use that in combination with the remaining implicit visibility information for rendering. Further, to prevent the link mesh from growing excessively in large scenes in the beginning, we also propose a simple means to let graphic artists define blocking planes as a way to support our algorithm with coarse explicit visibility information. Lastly, we propose a simple yet efficient image-space approach for displaying radiosity solutions without any meshing for interpolation.

Keywords: global illumination, radiosity, antiradiance

1 INTRODUCTION

In the 1990s, radiosity methods have been significantly improved, but after a period of large interest research essentially ceased for several years. Mainly three difficulties with radiosity caused the degrading interest: meshing of the input geometry and computing the (hierarchical) link mesh, the necessity of storing all patches and radiosity values in memory for computing a view-independent solution, and above all, the expensive visibility computation that typically consumes most of the computation time [8]. However, some of these problems have recently been successfully tackled: the antiradiance method reformulates the rendering equation such that visibility computation for form factors is no longer necessary, and by this enables a simple and fast GPU implementation of radiosity methods. Dong et al. [4] also demonstrated a GPU-radiosity algorithm by coarsely discretizing visibility that can be computed without ray casting. Motivated by this progress, Meyer et al. [15] introduced a data-parallel method for meshing and hierarchical linking, and demonstrate a CUDA implementation. In combination, these methods allow for interactive radiosity in dynamic scenes. Fig. 1 illustrates the fundamental differences of traditional radiosity, and the two aforementioned improvements. While Dong et al.'s [4] method produces small link meshes – actually smaller than traditional radiosity – it affects the global illumination result negatively due to the coarse discretization. In this paper, we focus our study on the antiradiance method which matches traditional radiosity in terms of quality. However, it replaces costly visibility computation for form factors by creating excessively many links to transport negative energy to compensate for missing occlusion (see Fig. 2).

That is, two solutions at the opposite ends of the spectrum exist for handling visibility in the radiosity method: either fully explicit or fully implicit. This paper bridges the gap in between by presenting a method to deduce explicit visibility information from the link mesh that is generated for the antiradiance (AR) method. By this we can reduce the number of links, however, still without computing form factors with explicit visibility, e.g. using ray casting. Obviously, the visibility information of a scene must be encoded in the AR link mesh: otherwise it would not be possible to compute the correct global illumination result with implicit visibility. Our method starts at exactly this point: we analyze the link mesh to identify chains of links whose light transport cancels out. Once such a chain has been identified, we can remove it without changing the result or reducing the rendering quality noticeably. However, reducing the number of links saves memory and computation time during light prop-

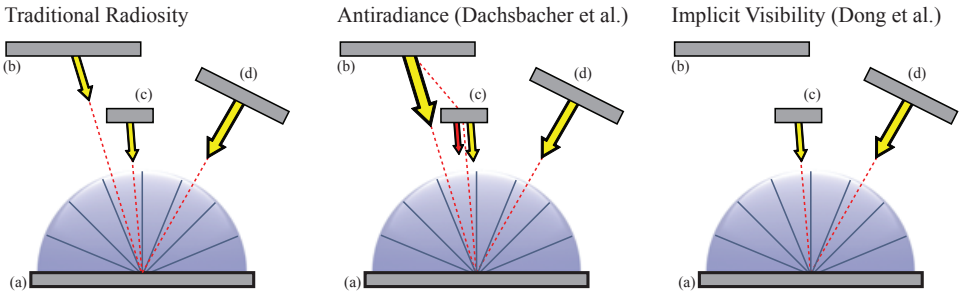


Figure 1: Light transport in different radiosity variants between a patch (a), and three other patches (transport links to (a) shown as dashed lines; sizes of yellow arrows indicate transported intensity): *Left:* radiosity computes visibility for each pair of patches; the transport from (b) is partly blocked. *Center:* the antiradiance method transports energy as if there is no blocking, but compensates for this by propagating negative light that originates from (b) via (c) to (a) (red arrow). *Right:* Dong et al. [4] discretize visibility to one link per direction bin eliminating transport from (b) to (a). In case of full visibility, as for patch (d), all three methods yield the same result.

agation. We consider our contribution being a principal study of deducing explicit from implicit visibility, and report statistical as well as visual results from our prototypal implementation. We also incorporate an effective way to let graphics artists define during the modeling stage where light transport cannot take place. For example, if there is no light exchange between two wings of a building, then we indicate this by simply placing a polygon somewhere in between. Our algorithm will use this coarse explicit visibility information to prevent the AR link mesh from growing large even prior to our reduction. Lastly, we describe a novel way for rendering high-quality antiradiance solutions without meshing or interpolating the final radiance across neighboring elements. At render time, we light the scene using every element as an area light source with an (anti)radiance distribution obtained from the global illumination solution. This allows the rendering of images with high quality interpolation and better contact shadows.

2 RELATED WORK

The importance of global illumination (GI) for computer graphics can be seen from the vast body of research papers in this field. Two main directions have been studied intensively in the last decades: ray tracing and radiosity methods; we refer the reader to excellent text books on these topics, e.g. Dutré et al. [5].

With the increasing computational power of graphics hardware, there have been many attempts to use GPUs to speed up global illumination. In recent years, research in ray tracing made a great leap forwards and there exist algorithms for real-time, parallel kd-tree construction [24], BVH construction [12], and fully interactive GI based on photon mapping [22].

Ray tracing based methods often cache information about the lighting in a scene, e.g. irradiance caching [23], photon mapping [9, 7], or instant radiosity [11]. In particular instant radiosity gained much attraction as represent the lighting of a scene by a set of virtual point lights, and thus easily maps to GPUs [18]. The light cuts method [21] clusters point lights into a hierarchy to speed up rendering. Final gathering is an essential step for computing high-quality images and a parallel algorithm therefor has recently be demonstrated [17].

In the following, we discuss work which is more closely related to our approach. There have been several attempts to compute radiosity solutions on the GPU. The main cost factor is the evaluation of the mutual visibilities between surfaces patches. Either rasterization together with the hemicube method [2, 1], or ray tracing on the GPU [19] have been used to compute form factors. The antiradiance reformulation [3] of the rendering equation [10] replaces explicit visibility by a recursive computation with negative light ("antiradiance"). Dong et al. [4] use a directional discretization and store only one link to the respective closest patch per direction. Thus the visibility is constructed implicitly with the link hierarchy. Although both methods are fundamentally different, both rely on a hierarchical link structure which initially had to be generated sequentially on the CPU. Meyer et al. [15] present a

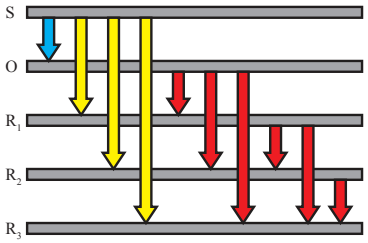


Figure 2: Radiance and antiradiance links for a 1D example. The normal of patch S is pointing downwards, the other surface normals are pointing towards S. There is only one unoccluded radiance link $S \xrightarrow{(+)} O$ (blue). The other radiance links $S \xrightarrow{(+)} R_n$ (yellow) and antiradiance links $R_n \xrightarrow{(-)} R_m$ (red) only exist for implicit occlusion handling.

data-parallel algorithm for link and patch creation, implemented in CUDA, which allows interactive radiosity methods with fully dynamic scenes.

Typically, the per-patch radiosity values that represent the GI solution are interpolated across adjacent patches. This can be done by generating an accordingly tessellated triangle mesh together with Gouraud shading [5] and improved using discontinuity meshing [14]. Dachsbacher et al. [3] used an image-space splatting approach that does not require a special mesh. Lehtinen et al. [13] directly compute global illumination using a meshless hierarchical representation and display the solution similarly.

3 ANTIRADIANCE

In this section we briefly review the antiradiance method [3], also following the notation of this work. The rendering equation describes the equilibrium of light transport in a scene and the radiance at a surface point x in direction ω_o is:

$$\begin{aligned} L(x,\omega_o) =& E(x,\omega_o) + \int_{\Omega^+} f(x,\omega_o,\omega_i) L_{in}(x,\omega_i) \cos\theta \, d\omega_i \\ =& E(x,\omega_o) + (\mathbf{K}L_{in})(x,\omega_o) \; . \end{aligned}$$

The incoming radiance at position x from direction ω_i originates from the closest surface in that direction. It is determined using the ray casting operator $ray(x,\omega_i)$ and part of the transport operator $\mathbf{G}$:

$$L_{in}(x,\omega_i) = L(ray(x,\omega_i),\omega_i) = (\mathbf{G}L)(x,\omega_i) \; . \quad (1)$$

As the computation of $\mathbf{G}$ is a very costly, the AR method strives to replace $\mathbf{G}$ by another transport operator that is cheaper to compute. Instead of resolving visibility explicitly by finding the nearest surface, the radiance is gathered from all surfaces along a ray yielding the transport operator $\mathbf{U}$. Extraneous light is then propagated and must be compensated. A pass-through operator $\mathbf{J}$ is defined that lets incoming radiance at a patch pass through without changing its magnitude or direction. The operators are related as follows:

$$\mathbf{G}L = \mathbf{U}L - \mathbf{U}\mathbf{J}\mathbf{G}L = \mathbf{U}(L-A) \quad (2)$$

with $A = \mathbf{J}\mathbf{G}L$ being the *antiradiance*. With this reformulation of the standard transport operator, the antiradiance rendering equation is obtained as:

$$L = E + \mathbf{K}\mathbf{U}(L-A) \quad (3)$$

$$A = \mathbf{J}\mathbf{U}(L-A) \; . \quad (4)$$

When L, A and E are projected into a suitable Hilbert base over the scene surface with finite dimensionality, these functions can be expressed as vectors of their components in that Hilbert space. Likewise, the operators $\mathbf{U}$, $\mathbf{K}$ and $\mathbf{J}$ become matrices:

$$\begin{pmatrix} L \\ A \end{pmatrix} = \begin{pmatrix} E \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{K}\mathbf{U} & 0 \\ 0 & \mathbf{J}\mathbf{U} \end{pmatrix} \begin{pmatrix} L-A \\ L-A \end{pmatrix} \; . \quad (5)$$

We can then separate K out of this matrix and get:

$$\begin{pmatrix} L \\ A \end{pmatrix} = \begin{pmatrix} E \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{K} & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{U}^L & 0 \\ 0 & \mathbf{J}\mathbf{U}^A \end{pmatrix} \begin{pmatrix} L-A \\ L-A \end{pmatrix} \; . \quad (6)$$

The thus remaining matrix describes the occluded light transport by means of the transport operator $\mathbf{U}$. Replacing it with $\left(\begin{smallmatrix} \mathbf{G} & 0 \\ 0 & 0 \end{smallmatrix}\right)$ yields the discretized equation for occluded light transport (as in standard radiosity) again. This comparison shows that the upper left part $\mathbf{U}^L$ of the matrix describes radiance transport while the lower right part $\mathbf{U}^A$ describes antiradiance transport.

4 REMOVING OCCLUDED LINKS

Eq. 6 shows that once we separated out the transport operator matrix we can interpret antiradiance and traditional radiosity as two extremes of how light is propagated between elements in the scene. However, the transport matrix does not have to take the form of either the fully occluded or the fully unoccluded transport. Within limits that we will discuss in the following, it is possible to create intermediate matrices $\mathbf{U}'^L$ and $\mathbf{U}'^A$ that contain transport with and without explicit visibility, i.e. essentially a mixture of entries from $\mathbf{G}$ and $\mathbf{U}$.

Let us first assume the case where we replace one unoccluded transport by a transport with explicit visibility. For this we define the matrix $\mathbf{U}'^L$ which contains one entry of $\mathbf{G}$, i.e. $\mathbf{U}'^L_{kl} = \mathbf{G}'^A_{kl}$, and all other entries are equal to $\mathbf{U}^L$. If the resulting light transport is correct, then the solutions L_{ij} and L'_{ij} for both matrices are equal. The equation for the k-th patch, L_k, becomes:

$$L_k = \sum_{i \neq l} \mathbf{K}_{ki}(\mathbf{U}^L_{ki}L_i - \mathbf{U}'^A_{ki}A_i) + \mathbf{K}_{kl}(\mathbf{G}_{kl}L_{kl} - \mathbf{U}'^A_{kl}A_{kl}) \; . \quad (7)$$

An entry $\mathbf{G}_{ij}$ is always less or equal to the respective entry $\mathbf{U}_{ij}$, and thus the sum over all $\mathbf{U}'^A_{ki}A_i$ in this equation must either be equal or less than the sum over all $\mathbf{U}^A_{ki}A_i$. This means, that *at least* one of the entries in this particular row of $\mathbf{U}'^A$ must be decreased in value. In other words, if the radiance transport between two patches is performed with proper occlusion, the receiving patch must no longer receive the same amount of antiradiance that was previously transported to it (to account for the occlusion along this transport path that we now consider explicitly). Note that although this shows that unoccluded and occluded light transport can be performed at the same time, no rules for the adjustments to $\mathbf{U}'^A$ can be derived from these equations alone. If we assume for now that we can replace values of $\mathbf{U}^L_{ij}$ one after another, then we can repeat this until $\mathbf{U}'^L$ equals $\mathbf{G}$, and in this case $\mathbf{U}'^A$ vanishes.

4.1 Link Removal in 1D

We have shown that mixing occluded and unoccluded light transport operations is possible under the restriction that antiradiance must only be transported to

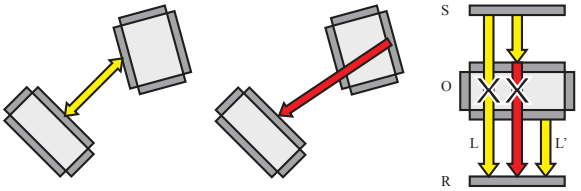


Figure 3: Convention for creating links: radiance links are created between patches facing each other (left); antiradiance links are in the negative hemisphere of a patch (center). Right: For every incoming link (here: L) we search if there are shorter incoming links (L'). Then we can remove L. Note that the antiradiance link shown in red is removed because of the same fact and thus the result is again correct at last.

patches that are the target of unoccluded transport operations. To introduce our algorithm we start with the instructional example of patches along one line. We will discuss rules for removing transport links for the case where a target patch is fully occluded from the source. Note that this information is solely based on the patch positions and extents, and the link mesh. Detecting partial occlusion, i.e. not only removing links for fully occluded patches, requires visibility computation, e.g. using ray casting that we still want to avoid. Also, we only consider opaque surfaces as potential occluders.

The motivation for the link removal is that the implicit handling of visibility in antiradiance generates a number of light transporting links that rises disproportionally with the depth complexity of the scene.

A simple set of surfaces in a 1D example as seen in Fig. 2 motivates the removal of unnecessary links. With unoccluded light transport, the topmost surface S illuminates all other surfaces. Therefore, it is linked to all of these surfaces, although the nearest surface O already fully occludes the light, and the light transported across the links to the surfaces further away needs to be compensated for. This is achieved by the antiradiance links from O to all other patches R_n below, and the pattern repeats analogously for every surface. When explicit visibility is taken into account, only the link $S \xrightarrow{(+)} O$ is required to illuminate this example scene correctly. With the implicit handling of occlusion as described above, $n+n!-1$ excess links are generated for n patches.

The first observation here is that all occluded radiance links in this example intersect the sole unoccluded patch O. Removing them along with all antiradiance links that originate from O itself effectively results in the occluded transport again. However, the remaining antiradiance links do not transport energy anymore since the patches where they start from do not receive any. To optimally reduce the link mesh, we also want to remove those from the transport matrix.

It is now possible to formulate two rules to find

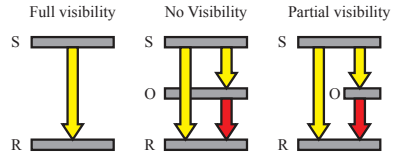


Figure 4: Classification of visibility in 3D used for our heuristics. A sender patch S emits radiance towards a receiving patch R, which may be blocked by a potential occluding patch O. Three situations need to be distinguished: full (left), none/fully occluded (middle) and none (right).

groups of links that can be removed from the link mesh without changing the result:

Rule 1 Search for a pattern of three links: from the sender S to the occluding receiver O, from S to the occluded receiver R, and from O to R. When such a pattern is found, both links to the occluded receiver $S \xrightarrow{(+)} R$ and $O \xrightarrow{(-)} R$ may be removed. We distinguish radiance and antiradiance links by the sign over the arrow.

Rule 2 Check for every *incoming link* of a patch L if there is another, shorter incoming link L'. If such a link is found, we can remove L as it is occluded (Fig. 3). Note that this rule is only valid if the objects in the scenes are manifolds with closed surfaces as also assumed in [3].

4.2 Heuristics for Link Removal in 3D

Obviously, the aforementioned removal in 1D retains validity in two or three dimensions only if it is applied to infinitely small patches along a single ray through the scene. For patches with finite size and only a finite number of discrete directions (*directional bins*), as used in the antiradiance method [3], we can derive heuristics from these rules that still work well when the scene discretization is reasonably fine. In Section 6 we evaluate the validity of both heuristics described in this section.

The modification compared to the 1D case is necessary due to the fact that the visibility function V between two patches is no longer either 0 or 1, but can take any value in-between (see Fig. 4). Furthermore, a special treatment is required to respect peculiarities in a hierarchical link mesh: to determine if the light transport between two patches is blocked, we might have to search across different levels of the hierarchy.

Heuristic 1 The first rule from Section 4.1 transforms into Algorithm 1. Again we find a combination of sender S, occluder O, and receiver R. However, we only remove the links, if O or one of its parents (in the patch hierarchy) subtends a large enough solid angle such that the light transport from a sender to another surface is completely blocked (see Fig. 5). In addition, the radiance link $S \xrightarrow{(+)} R$ must have the same (discrete) direction as the antiradiance link $O \xrightarrow{(-)} R$. To test this,

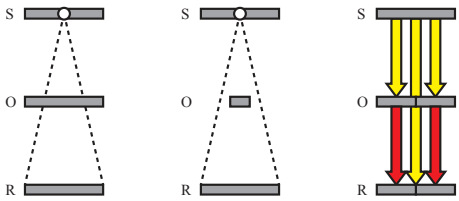


Figure 5: Left: the occluder O, as seen from S (note that in the antiradiance method links are established between patch centers), subtends a larger solid angle than the receiver patch R. Center: the subtended solid angle of O is too small and the link from S to R cannot be removed. Right: the radiance link $S \xrightarrow{(+)} R$ is on a different level of the hierarchical link mesh than the antiradiance links $O \xrightarrow{(-)} R$. Thus we also consider parents in the patch hierarchy when searching for blockers.

we construct two infinite circular cones with their tips in S: the first one, C_{Link} connects the centers of S and O with its axis and has an opening angle so that the solid angle subtended by this cone equals that of O as seen from S. The second cone, C_{bin} has the discrete direction of the link's source bin as its axis and its opening angle is chosen so that C_{bin} covers a solid angle of ω_{bin}. Then, C_{Link} must enclose C_{bin} (cf. Alg. 1 lines 6-10). This ensures that all patches lie on one line, analogous to the 1D example before. When using hierarchical link meshes, an occluder can be linked to the receiver on a finer level than the sender-receiver link, in which case the patch at the finer level takes on the role of R (Fig. 5, right, Alg. 1 line 5). Also, the full extent of the occluding geometry can be better estimated by checking the solid angles subtended by any parents of the suspected occluding patch.

Heuristic 2 The second removal rule transfers to the 3D case analogous to the previous one (see also Algorithm 2). The aforementioned restriction to scenes consisting entirely of closed manifolds stays valid. When

Algorithm 1 Find and disable all links between senders, occluders and receivers.

```
1  for each radiance link L
2    required[L] = true
3
4  for each radiance link L
5    for all target patch P_j of link L and its parents
6      C_Link ← cone(center(P_i), actualdir(L), ω_{i←j})
7      // sourcedir(L) is discreet source direction of Link L
8      C_bin ← cone(center(P_i), sourcedir(L), ω_bin)
9      if ω_{i←j} > ω_bin and C_Link encloses C_bin
10       binOccluded ← true
11
12   if binOccluded
13     for all antriadiance links L' starting from P_j
14       P_k ← target_patch(L)
15       if link P_i → P_k exists
16         if sourcedir(L)=sourcedir(L')
17           if targetdir(L)=targetdir(L')
18             required[L] ← false
19             required[L'] ← false
20
21 remove all links where required[L] is false
```

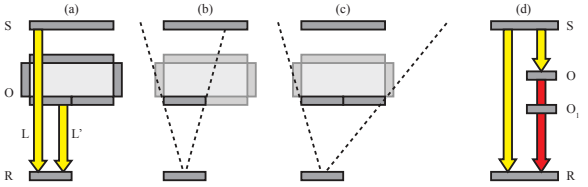


Figure 6: (a) Two incoming links L and L' at R have the same direction. To determine if we can remove L, we need to test if the transport from S to R is blocked. Testing this blocking with the patch O where L' emanates from does not reveal this information. Ascending the hierarchy and testing with the parent of O determines full blocking (c) and L can be faithfully removed. (d) A situation where a fully occluded antiradiance link must not be removed: The link $S \xrightarrow{(+)} R$ is partially occluded by O and O_1 and the link $O \xrightarrow{(-)} R$ is fully occluded by O_1, but the antiradiance produced at O must arrive at R to correctly reproduce the partial occlusion. Therefore, the link $O \xrightarrow{(-)} R$ must not be removed.

we test if a link $S \xrightarrow{(+)} R$ can be removed, we need to find a shorter link from a potential occluder O to R. However, these two links must have the same (discretized) direction (Alg. 2 line 4). If we detect such a situation, then we need to determine if O, or the surface to which is belongs, is large enough to block the transport from S to R by ascending the hierarchy from O to its largest parent. Next, we project S onto the plane of this parent patch and only if the projection is fully on that surface the link can be removed (Fig. 6, Alg. 2 lines 9-15). Note that patches that share a common parent with S will never be considered as occluders (Alg. 2 line 8).

However, not all links that are detected as fully occluded may actually be removed. As illustrated in Fig. 6 antiradiance links might be necessary to capture partial occlusion although the above heuristic marks them as occluded. Algorithm 3 detects these links and prevents them from being removed. For each partially visible link $S \rightarrow R$ it searches all other antiradiance links

Algorithm 2 Classify Link visibility as seen from target patch.

```
1  for each radiance link L
2    visibility[L] = full
3    for each link L' ending at target patch of L
4      if target_bin(L)=target_bin(L')
5        R ← target_patch(L)
6        S ← source_patch(L)
7        O ← source_patch(L')
8        if S and O do not have same parent patch
9          // If S or O are clusters, the following is
10         // checked on every pair of patches in these clusters
11         O' ← biggest parent of O
12         S' ← perspective projection of S onto O'
13         if S on far side of O' and S' fully inside O'
14           visibility[L] ← none
15           continue
16         else
17           visibility[L] ← partial
```

Algorithm 3 Finding required occluded links.

```
1  for each link L
2    if visibility[L] = none
3      required[L] = false
4    else
5      required[L] = true
6
7  for each link L with visibility[L]=partial
8    S ← source_patch(L)
9    R ← target_patch(L)
10   for R and each child patch of R
11     for each antiradiance link L'
12       S' ← source_patch(L')
13       R' ← target_patch(L')
14       if R=R'
15         for each link L'' connecting S to S'
16           if visibility[L''] != none and L'' is radiance link
17             required[L'] ← true
```

$S' \xrightarrow{(-)} R$ to the same target patch and checks if a non-occluded link $S \rightarrow S'$ exists. If so, $S' \xrightarrow{(-)} R$ is required to keep the partial occlusion of S' from R by S intact and is marked as required. Note that this heuristic cannot be used after heuristic 1 because links $S' \xrightarrow{(-)} R$ may already have been removed, resulting in required links getting missed and removed.

Patches without incoming links In case of static direct lighting in the scene we can also remove all links emanating from patches that are neither light sources nor have any incoming links, as these links will never transport energy. However, similar to the original antiradiance implementation [3] we typically use shadow maps for computing direct illumination and thus potentially every patch can receive energy that has to be propagated further.

4.3 User-Defined Link Removal

In addition to the link removal heuristics described above, we can optionally perform link removal based on user-defined (invisible) blocking geometry which strictly cuts all links that intersect it. This additional geometry can be a simple polygon generated by the user along with the scene to separate parts of the scene that obviously do not directly exchange light with each other, e.g. two rooms separated by solid walls (see Fig. 8). Since this geometry consists of few polygons only, we test for every link if it intersects the blocking geometry without generating high cost, and remove the link if this is the case. Note that this is similar to Fradin et. al. [6] who used manually placed portals to section large scenes.

5 FINAL SHOOTING

Dachsbacher et al. [3] used a splatting approach, similar to point-based rendering, to render an image with interpolated patch colors. Instead we propose to use a "final shooting" approach: we treat each patch in the scene as a *patch light source* (PLS) with a directional intensity distribution according to the total exitant intensity determined by the antiradiance solver. For rendering the final solution, we simply light the scene only using the PLSs. This approach can be seen as a variant of instant radiosity [11], where light sources emit radiance and antiradiance (computed using the antiradiance method) and thus account for shadowing implicitly. Note that the resulting number of virtual light sources in our approach is typically orders of magnitudes higher. Furthermore, every PLS is an area light source from which lighting computation is more intricate than from point lights. To this end, when computing the lighting of a fragment due to a PLS, we replace the PLS by 8 point light sources randomly placed on the PLS. This can be seen as a Monte-Carlo sampling of the area light sources; no noise is visible in the images, as the number of PLS is very high (it equals the number of patches in the scene). Note that by lighting the scene with all PLSs we obtain not only a smooth interpolation, but also one (additional) indirect bounce at the same time with no additional cost.

When using a full link mesh, we efficiently accumulate the contributions of all PLSs using deferred shading and interleaved sampling [20]. However, care has to be taken when using final shooting together with the link removal heuristics: in this case, only those patches are to be lit by a PLS that are still linked to it after reducing the link mesh. To account for this, instead of using deferred shading, we have to render every receiver patch for every PLS, compute per-pixel lighting, and accumulate the contributions. Note that this process benefits from the GPU's early-z culling automatically omitting occluded receivers. For lighting computing from a PLS, we look up the interpolated intensity towards a fragment using precomputed interpolation weights in a cube map (6×512^2 resolution in our examples).

6 RESULTS AND DISCUSSION

In this section we compare the heuristics and the user-defined link removal. To determine the impact of the heuristics' approximation regarding the link removal, we modified both to perform explicit visibility checks using ray casting and Monte Carlo sampling instead of the link mesh based checks. Note that no heuristic *should* remove more links than those removed with explicit visibility testing. We have implemented an Antiradiance solver similar to the one described in [3], using the same algorithm for building the hierarchy. The preprocessing, including the previously discussed link removal heuristics, is initially implemented and executed on the CPU (running on multiple cores), while OpenGL is used for the simulation of the light transport and for displaying the result. While our current implementation can be seen as an experimental prototype, we plan to integrate our heuristics into the data-parallel link generation method by Meyer et al. [15] in the future.

			heuristic 1				heuristic 2			
scene	patches	links	explicit test	heuristic removal	incorrect	not removed	explicit test	heuristic removal	incorrect	not removed
Japan	12745	629665	157449	129313	54872 (42%)	83058 (52%)	71007	34408	11732 (34%)	48331 (68%)
Office	14246	1470440	581768	395592	124232 (31%)	310408 (53%)	260475	85075	28739 (34%)	204139 (78%)
Desks	14396	1465632	759669	280705	7752 (10%)	486716 (40%)	690145	300316	11577 (4%)	401406 (58%)
Soda Hall	25023	2774452	2076609	1621749	73998 (5%)	528858 (25%)	1888014	1016923	21026 (2%)	892117 (47%)

Table 1: Results of applying our heuristics to the test scenes with: the total number of radiance and antiradiance links, the number of links removed by the explicit test, the links removed by the heuristic, the number of incorrectly removed links compared to the explicit test, and the number of links that have not been removed by the heuristic but by the explicit test.

scene	patches	links	heuristic 1	heuristic 2
Japan	12745	629665	22.1s	26.0s
Office	14246	1470440	248.2s	224.9s
Desks	14396	1465632	80.5s	127.0s
Soda Hall	25023	2774452	87.9s	280.0s

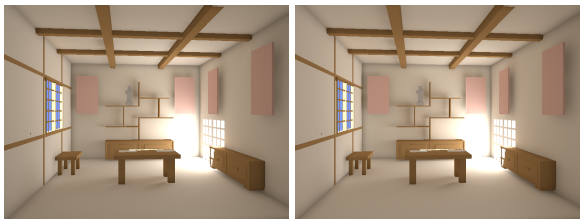
Table 2: Measured run time for our multithreaded CPU implementation of the heuristics running 8 concurrent threads (averaged over 10 runs)

6.1 Link Removal Heuristics

We tested our link removal heuristics against the explicit visibility oracle on various scenes: the Japanese room and office from [3], the "desks" shown in Fig. 8 and the model of the fifth floor of the Soda Hall (see Fig. 12). The results are summarized in Table 1 and the run times are given in Table 2. All measurements were taken on an Intel Core i7 CPU with 2.66MHz, 6GB RAM, and an NVIDIA GeForce GTX 465 with 1024MB RAM, running Linux.

Our results show that the heuristics also remove links that are not totally occluded. In the Japanese room and office scene, which both consist of a single room with many objects inside, the error rate is particularly high while it is low for the "desks" scene and the Soda Hall with their walls as main occluders. This shows that while both heuristics perform well with big solid occluders, they are prone to inaccuracy with correctly estimating visiblity along silhouettes of objects. The first heuristic relies on the subtended solid angle for testing blocking. However, this gives no indication about the actual patch shapes: for instance, elongated patches can be visible although nearly quadratic patches with larger solid angles are in front of them. The second heuristic erroneously removes links for which unblocked paths between the patches may still exist although the projection test succeeds. The runtime for heuristic 1 depends on the number of occluders in the scene: the algorithm only needs one occluder per link and thus has to search less links and terminates quicker when many large occluders are present. Heuristic 2 spends most time in algorithm 3, whose complexity is dominated by the number of links that have to be searched.

Although the resulting error due to the link removal heuristics seems significant for the affected scenes, the impact on the rendered images is hardly perceivable (see Fig. 7 and Fig. 10a). Obviously, errors are only introduced for links whose contribution (either radiance or antiradiance) is negligible. The first heuristic creates generally brighter shadow regions. It discards $O \xrightarrow{(-)} R$



(a) heuristic 1 (b) heuristic 2

(c) 4× difference heuristic 1 (d) 4× difference heuristic 2

Figure 7: The Japan scene rendered with reduced link meshes using final shooting (top), and then luminance differences to an image computing with the full link mesh (red is less, green is greater than original). The difference images have been scaled by a factor of 4. Compare Fig. 9, 10a for references.

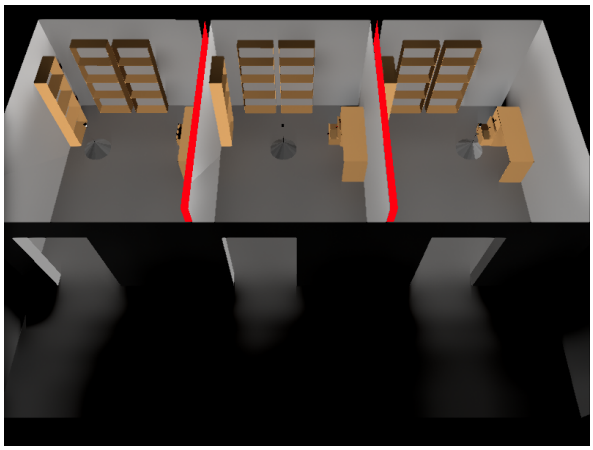


Figure 8: Desk scene with blocking geometry (red) after 6 antiradiance iterations, rendered with splatting.

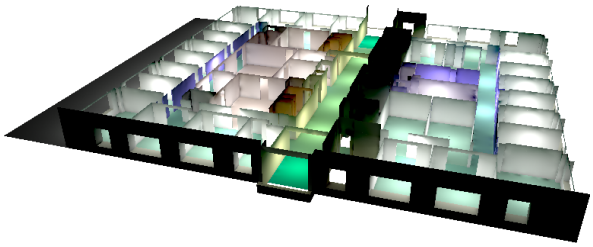


Figure 12: Soda Hall level 5 lit by diffuse emitting area light sources in the rooms and on the hallways. It was rendered using final shooting and the reduced link mesh produced using heursitic 1 (see Table 1).

links even if there exists another link $S' \xrightarrow{(+)} R$ that is only partially occluded by O and requires the antiradiance generated at O to correctly light R. The second algorithm of heuristic 2, which is run to preserve partial occlusion, results in a stronger tendency to keep antiradiance links. Thus, the corresponding difference image shows lesser brightening of shadow regions. Applying heuristic 2 followed by heuristic 1 on the Japanese room and Soda Hall scenes results in 144983 removed links (45.5% error) and 1628987 removed links (4.7% error), respectively, and visual quality comparable to Fig. 7.

We tested the blocking planes by adding two such polygons between the rooms in the "desks" scene (see Fig. 8). These two quads alone caused a removal of almost 76% of all occluded links in the scene, demonstrating that this mechanism is not only cheap to compute, but also highly efficient.

6.2 Rendering Quality

Fig. 9 shows resulting images for the Japanese room without interpolation, with splatting, and final shooting. The global illumination solution has been computed with 4 antiradiance iterations and 128 direction bins for all of our test scenes.

At a resolution of 800×600 pixels the rendering speed was 8.5 frames per second with no interpolation, 7.7 fps with splatting, and 0.04 fps with shooting. Although final shooting has a considerable impact on performance due to the high number PLSs, the resulting images capture finer details, e.g. contact shadows, due to patches emitting antiradiance. Interleaved sampling for final shooting with 4×4 interleaved sampling runs at 0.44 fps, and the performance increases further when using larger interleaving patterns yielding 1.22 fps at 8×8, and 2.88 fps at 16×16. However, at some point the wider Gaussian blur filter again removes details (see Fig. 10). Nichols et al. [16] report tremendous speedups with multi-resolution splatting compared to interleaved sampling, however, there is another way to speed up shooting. So far, we used the leaf nodes in the hierarchy as PLSs, but we also can use interior nodes therefor. Using the leaves' parents yields about 75% less PLSs, but only slightly reduces the amount of detail in the lighting as we treat them as area lights (see Fig. 11), but already yields a significant speedup: shooting at full resolution runs at 0.14 fps, at 1.29 fps with 4×4, at 2.99 fps with 8×8, and at 5.09 fps with 16×16 interleaving, respectively. A further optimization, and direction for future work, is to exploit the patch hierarchy also for selecting the PLSs, in spirit of [21].

7 CONCLUSIONS

In this paper we studied heuristics operating directly on the hierarchical link mesh of the antiradiance method to deduce explicit visibility information and thus to reduce the number of links. The energy propagation is the most time consuming step in antiradiance methods and greatly benefits from link removal as its cost is proportional to the number of links. Our heuristics remove links more faithfully than Dong et al.'s method [4] yielding results of similar quality as the antiradiance method. The user-defined blocker geometry can further be used to remove links with little computation. The final shooting simplifies the rendering of a high-quality final solution and yields better results and requires less tweaking than Dachsbacher et al.'s splatting approach.

Obviously, our heuristics will have to be incorporated into a data-parallel link generation method, such as Meyer et al.'s work [15] to actually speed up antiradiance in fully dynamic scenes. This data-parallel algorithm is about two orders of magnitude faster than our (and Dachsbacher et al.'s) CPU-based link generation algorithm (45 million links/s versus 140 thousand links/s), and we would expect a similar speedup for the link removal. Another challenge is to create blocking geometry automatically by analyzing the scene geometry. We believe that these two steps will enable our link removal in fully dynamic scenes at interactive speed.

ACKNOWLEDGEMENTS

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

REFERENCES

[1] Attila Barsi, László Szirmay-Kalos, and Gábor Szijártó. Stochastic glossy global illumination on the gpu. In *Spring Conference on Computer Graphics 2005*, pages 187–193, 2005.

[2] Greg Coombe, Mark J. Harris, and Anselmo Lastra. Radiosity on graphics hardware. In *Graphics Interface 2004*, pages 161–168, 2004.

[3] Carsten Dachsbacher, Marc Stamminger, George Drettakis, and Frédo Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 26(3):61, 2007.

Figure 9: The Japanese room with different rendering methods. The room is lit from outside the window on the left. The images were rendered with 4 iterations at a resolution of 800×600 pixels with a full link mesh.

(a) PBRT reference (b) 4×4 interleaved sampling (c) 16×16 interleaved sampling

Figure 10: Results with different interleaved sampling patterns: larger patterns result in stronger blurring and loss of small scale features.

(a) final shooting (b) 4×4 interleaved sampling (c) 16×16 interleaved sampling

Figure 11: Final shooting using not the leafs of the patch hierarchy, but its parents as PLSs. This results in significantly faster rendering speed, sacrificing only little details in the lighting (e.g. visible in shadowed areas around the bookshelf).

[4] Zhao Dong, Jan Kautz, Christian Theobalt, and Hans-Peter Seidel. Interactive global illumination using implicit visibility. In *Proc. of Pacific Graphics*, pages 77–86, 2007.

[5] P. Dutré, K. Bala, and P. Bekaert. *Advanced Global Illumination*. AK Peters, 2006.

[6] D. Fradin, D. Meneveaux, and S. Horna. Out-of-core photon-mapping for large buldings. In *Proceedings of Eurographics symposium on Rendering*, June 2005.

[7] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–8, 2008.

[8] Nicolas Holzschuch, François X. Sillion, and George Drettakis. An efficient progressive refinement strategy for hierarchical radiosity. In *Photorealistic Rendering Techniques (Eurographics Workshop on Rendering)*, pages 357–372, 1994.

[9] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001.

[10] James T. Kajiya. The rendering equation. *Computer Graphics (Proc. of SIGGRAPH '86)*, 20(4):143–150, 1986.

[11] Alexander Keller. Instant radiosity. In *SIGGRAPH '97*, pages 49–56, 1997.

[12] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. Fast bvh construction on gpus. *Computer Graphics Forum*, 28(2), 2009.

[13] Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François X. Sillion, and Timo Aila. A meshless hierarchical representation for light transport. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 27(3):1–9, 2008.

[14] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, 1992.

[15] Quirin Meyer, Christian Eisenacher, Marc Stamminger, and Carsten Dachsbacher. Data-Parallel, Hierarchical Link Creation. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization*, 2009.

[16] Greg Nichols, Jeremy Shopf, and Chris Wyman. Hierarchical image-space radiosity for interactive global illumination. In *Computer Graphics Forum 28(4)*, pages 1141–1149, 2009.

[17] Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, Hans-Peter Seidel, Jan Kautz, and Carsten Dachsbacher. Micro-rendering for scalable, parallel final gathering. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 28(5), 2009.

[18] Tobias Ritschel, Thorsten Grosch, Min H. Kim, Hans-Peter Seidel, Carsten Dachsbacher, and Jan Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 27(5), 2008.

[19] Arne Schmitz, Markus Tavenrath, and Leif Kobbelt. Interactive global illumination for deformable geometry in cuda. *Computer Graphics Forum (Proc. of Pacific Graphics 2008)*, 27(7), 2008.

[20] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Non-interleaved deferred shading of interleaved sample patterns. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics Hardware*, pages 53–60, 2006.

[21] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 24(3):1098–1107, 2005.

[22] Rui Wang, Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. An efficient gpu-based approach for interactive global illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 28(3):1–8, 2009.

[23] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In *Eurographics Workshop on Rendering*, pages 85–98, 1992.

[24] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 27(5):1–11, 2008.