SCI INSTITUTE
TECHNICAL REPORT

DIRmaps : Discretized Incident Radiance Maps for High-Quality Global Illumination Walkthroughs in Complex Environments

Ingo Wald

UUSCI-2005-010

Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA

December 6, 2005

Abstract:

Realtime ray tracing and interactive global illumination have recently made significant progress, allowing for ray traced rendering quality even in highly complex scenes. Nevertheless, high-quality global illumination is currently still limited to offline computations, as existing interactive techniques are applicable only to simple scenes, BRDFs, and/or lighting conditions. For realistically complex scenes, BRDFs, and lighting situations existing systems still have to compromise either performance or image quality, even if considerable compute power is invested. In this paper, we present a method that uses a precomputation approach to obtain a high-quality solution of the global illumination that is then used during interactive rendering. This solution is stored in a directionally dependent way, thus supporting complex BRDFs such as glossy surfaces or BTFs. Additionally, the illumination is stored independent of geometry, allowing for supporting arbitrary kinds of surface primitives as well as highly complex models. Using our method in a distributed interactive ray tracing system, we achieve interactive performance even for highly complex geometry, complex BRDFs and BTFs, and highly complex lighting situations, while maintaining offline-comparable image quality.

THE UNIVERSITY OF UTAH

DIRmaps : Discretized Incident Radiance Maps for High-Quality Global Illumination Walkthroughs in Complex Environments

Ingo Wald
SCI Institute, University of Utah, UT, 84112
wald@sci.utah.edu

Figure 1: *Several interactive examples of using Discretized Incident Radiance Maps: a) Stanford Buddha (1 million triangles) with a highly glossy BRDF, globally illuminated from an HDR environment map. b) Shirt with a measured wool BTF. The BTF's dependence on light and view direction is well supported by our method. c) Zoom onto a gearshift of a Mercedes Benz C-Class model, with measured real-world materials (BTFs), and under complex illumination through the refractive windows (actually a caustic). Note the smooth shadows, as well as the highlights and self-shadowing on the aluminium BTF. d) The entire car (1 million triangles), almost completely modelled with BTFs, and with complex lighting. Note the subtle effects in the windshields, and the correct illumination all over the model. Using a distributed interactive ray tracing system running on a cluster of PCs, all these images can be rendered at interactive rates.*

Abstract

Realtime ray tracing and interactive global illumination have recently made significant progress, allowing for ray traced rendering quality even in highly complex scenes. Nevertheless, high-quality *global illumination is currently still limited to offline computations, as existing interactive techniques are applicable only to simple scenes, BRDFs, and/or lighting conditions. For realistically complex scenes, BRDFs, and lighting situations existing systems still have to compromise either performance or image quality, even if considerable compute power is invested.*
In this paper, we present a method that uses a precomputation approach to obtain a high-quality solution of the global illumination that is then used during interactive rendering. This solution is stored in a directionally dependent way, thus supporting complex BRDFs such as glossy surfaces or BTFs. Additionally, the illumination is stored independent of geometry, allowing for supporting arbitrary kinds of surface primitives as well as highly complex models. Using our method in a distributed interactive ray tracing system, we achieve interactive performance even for highly complex geometry, complex BRDFs and BTFs, and highly complex lighting situations, while maintaining offline-comparable image quality.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Ray tracing I.3.2 [Computer Graphics]: Distributed/network graphics I.6.3 [Simulation and Modeling]: Applications

1. Introduction

Global illumination is one of the most important tools for computing photorealistic images. Unfortunately, it is also one of the greatest hurdles in the way of interactive photorealistic rendering, as computing global illumination at interactive rates is still a major challenge. Preliminary proof-of-concept systems are available, and have already achieved impressive results. Nonetheless, these systems are still limited to rather specialized applications.

In particular, existing approaches often support only suitable BRDF models (such as "mostly diffuse"), and/or relatively simple lighting situations. While mostly directly lit architectural scenes with diffuse surfaces can very well be handled already [WKB*02], for *really* realistic models such as the one shown in Figure 1c+d – including environmental illumination, specular illumination through refractive windows, strong direct illumination, complex BRDFs, and complex light transport paths – these interactive techniques fail completely, and often cannot produce reasonable images at all, even if considerable hardware is invested.

This inability to support complex models and lighting situations stands stark contrast to offline global illumination, where such models are commonplace. In particular, most *practical* applications of using global illumination for lighting simulation only make sense for such kinds of models.

In this paper, we are going to present "Discretized Incident Radiance Maps", a global illumination technique that combines a realtime ray tracing framework with a specially designed precomputation approach. In combination, this allows for offline-level rendering quality at interactive rates, even for as complex models as shown in Figure 1.

2. Previous Work

Our method will combine the advantages of several well-known global illumination techniques. To better understand our technique, it is important to discuss these in detail.

2.1. High-Quality Global Illumination

Global illumination techniques date back to the seminal work by Kajiya [Kaj86] and Cook [CPC84], which have who has proposed tracing paths to sample and compute the rendering equation

$$L(x,\omega_o) = \int_\Omega f_r(x,\omega_i,\omega_o) L'(x,\omega_i) \cos\theta_i d\omega_i.$$

This concept of tracing paths has later on be extended by Veach et al. [VG94], and Lafortune et al. [LW93], to bidirectional path tracing. Though being quite dated, these methods still set the quality standard for offline global illumination.

As path-based approaches are usually quite noisy, global illumination has long been dominated by radiosity-style techniques [HSA91] (see [Bek99] for an overview). Instead of computing each pixel individually, radiosity techniques project the illumination into a finite element basis, thus trading the pixel noise for discretization error. Unfortunately, this discretization error can be quite visible, and often requires a final gathering approach for high quality results. Additionally, radiosity techniques usually depend on a suitable surface mesh, often get problematic for real-world, "triangle soup" models, are are usually limited to diffuse surfaces only.

Today, state of the art for offline, high-quality global illumination is Photon mapping [Jen01]: Instead of connecting the camera with the light sources via uni- or bi-directional paths, photon mapping first generates a set of photons by tracing particles from the light sources, and later on these to estimate the irradiance via density estimation. Photon mapping is independent of geometry, can be combined with volumes and non-diffuse BRDFs, and can handle virtually all kinds of light transport paths. However, the density estimate is usually quite noisy, and in order to generate high-quality images has to be augmented by a final gathering pass as well.

2.2. Interactive Global Illumination

Due to its practical significance, researchers have long been looking into using global illumination also for interactive applications. For that purpose, radiosity was well suited: Once the radiosity for each patch path is computed, this solution can be visualized interactively by rendering the patches as colored triangles on graphics hardware. In fact, for many years this has been the only way of rendering globally illuminated scenes interactively at all. Unfortunately, using graphics hardware no longer allows for final gathering, and discretization artifacts are directly visible. Also, specular effects like reflections and refraction are not supported at all. As a result, the image quality of interactive radiosity techniques is usually far from offline quality.

With the advent of realtime ray tracing [Wal04], interactive ray traced global illumination now also has come within reach. Using a realtime ray tracing system running on a cluster of PCs, Wald et al. [WKB*02] have demonstrated interactive, ray-traced global illumination using the "Instant Global Illumination" (IGI) technique, basically a variant of Instant Radiosity [Kel97]: Virtual Point Lights (VPLs) are generated by a random walk from the light sources, which are then used for illuminating the scene. In contrast to radiosity, IGI can also simulate specular effects, and does not depend on a surface mesh. In general, the image quality is usually superior to plain radiosity.

Unfortunately, the efficiency of IGI depends to a large degree on how few VPLs suffice for good image quality: For highly occluded scenes and/or complex BRDFs, most of the costly VPLs have but a very small influence, and getting good image quality requires generating lots of VPLs, and tracing many shadow rays. Similarly, scenes with complex light transport paths often require huge numbers of VPLs to generate enough VPLs to sufficiently illuminate each of the scene'sregions. Thus, scenes with complex BRDFs and/or complex lighting cannot be handled well by this technique at all. Though solutions for special cases have been devised [WBS03, Wal04], in general this statement is true nonetheless.

In the context of interactive global illumination, one should also mention approaches like, e.g. Instant Radiosity [Kel97], the Shading Cache [TPWG02], and Selective Photon Tracing [DBMS02] (also see [Wal04]). These approaches however did not target high-quality global illumination in complex scenes and lighting situations, and thus will not be discussed in more detail.

2.3. Precomputed Radiance Transfer (PRT) et al.

Yet another approach towards interactive, high-quality rendering – in particular for environmental illumination – has been proposed in the form of Precomputed Radiance transfer [KSS02, SKS02, NRH04]. In that approach, the environmental illumination is transformed to a spherical harmon-

ics (SH) basis, and for each of the mesh vertices the relation between environmental illumination and irradiance is computed and stored in yet another SH basis. During rendering, the irradiance at any vertex can then be computed by a dot product with the environment's SH coefficients, which can be done in real time on current graphics hardware. However, PRT depends on a suitable surface mesh, and in its basic form is only applicable to diffuse reflections and low-frequency lighting. Extensions to high-frequency lighting and glossy BRDFs exist, but are have never been shown to be able to handle as complex settings as described above.

A framework that uses PRT in the context of a VR framework has recently been presented by Dmitriev et al. [DAK*04]. Though this yields global illumination at realtime performance in a CAVE, it cannot handle complex materials, somewhat suffers from meshing problems, cannot handle specular reflections and refractions (i.e., glass), and in general is still far from offline rendering quality.

3. Discretized Incident Irradiance Maps

As the previous discussion has shown, all of these techniques have different advantages and disadvantages, but none can handle our eventual goal of interactive, offline-quality rendering for the combination of complex models, complex BRDFs, and complex lighting. To fill this gap, we need a method that combines the advantages of several of the known techniques. In particular, our method should be ...

able to deliver offline quality, which basically implies building it on ray tracing.

suitable for interactivity, i.e., high quality should be delivered with a single ray per pixel, without final gathering.

able to handle complex light transport paths, which with the previous item implies using precomputed data.

independent of triangular mesh, i.e., not assume a suitable mesh were available.

applicable to complex geometries. To be able to support real-world scenes, we would like to support complex geometries of up to several million triangles.

applicable to arbitrary BRDFs. In particular, this requires to not store a derived value like radiosity or irradiance, but to reconstruct the directionally dependent incident radiance field around any given point.

able to handle complex lighting, including area lights and environmental lighting, highly indirect illumination, complex light transport paths, and specular illumination.

3.1. Method Overview

Eventually targeting offline-level quality, we need to base our method on a realtime ray tracer. To maintain interactivity even for our high-quality demands, we are willing to sacrifice the ability to *modify* the model, and (for now) constrain ourselves to static scenes with static lighting only, for which the illumination can be precomputed and stored. As we eventually want to support arbitrary BRDFs, we cannot store derived values like radiosity or irradiance, but rather also need the directional information of the lighting. In particular, we do not want to precompute the *reflected* radiance across the surface, but only the radiance *incident* on the surface.

To this end, we discretize phase space (x,ω) into finite elements $(x_i,\omega_i)_{i=1}^N$, and project the incident radiance field $L'(x,\omega) = L(rt(x,\omega),-\omega)$, into this basis, yielding a discretization $(L'_i)_{i=1}^N$ of L'. During rendering, we can then reconstruct an approximation of the radiance field around any surface point in question, and can use this for shading the surface. For the discretization, we will use a set of J discrete surface points x_j, and equip each of those with a set of K discrete directions ω_k. During rendering, we will then query the k nearest sample points to the surface point in question, and will interpolate the radiance field from these surface points. In particular, we will use the *same* set of directions for each of these points, which allows for smoothly combining the information from multiple samples via filtering.

4. Precomputation

To make our method work, we have to implement two operations: Choosing discrete samples (x_j,ω_k), and computing a suitable representation L_{jk} at these sample points.

Surface samples x_j are simply chosen by randomly sampling the geometry of the model. As – in contrast to Photon mapping – the density of the sample points will *not* directly influence our radiance estimate, we do not have to take any care at all on maintaining any point distribution properties. Overlapping or multiply defined surfaces – which often are problematic for meshing approaches – only lead to a locally higher sampling resolution, but do not pose any problem.

Discrete directions ω_k are computed by simply sampling a sphere with a Halton sequence. Of course, generating a set of directions by triangulating the sphere would work just as well, but is more complicated to implement. Using a Halton sequence allows for using any number K of directions, is trivial to implement, and yields nicely distributed directions.

Discretizing Incident Radiance: Given the discrete set of surface points $(x_j)_{j=1}^J$ and directions $(\omega_k)_{k=1}^K$, we need to project the continuous radiance field $L(x,\omega)$ into this discrete basis. The most trivial way of doing so would be to define L_{jk} as $L_{jk} = L'(x_j,\omega_k)$. However, sampling only the fixed directions ω_k is likely to lead to strong aliasing. Though this could also be treated by choosing more directions, we use the approach of partially pre-integrating the illumination: We equip each ω_k with a (normalized) filter $w_k(\omega)$, and define

$$L_{jk} = \int_\Omega w_k(\omega) L'(x_j,-\omega) d\omega, \qquad (1)$$

i.e., each ω_k actually represents an entire cone of directions around ω_k. What filter is used exactly seems to have only a minor impact, as long as all directions are eventually covered. In practice, we simply assign each direction ω to its closest ω_j. Note that this pre-integration does *not* include the BRDF at x_i, nor the cosine to the surface normal; we only pre-integrate the incident radiance around (x_j, ω_k).

4.1. Precomputing L_{ij}

In its most simple form, computing L_{ij} is as simple as just evaluating Equation 1 via (Quasi) Monte Carlo Integration using, e.g., a path tracer. In fact, this is exactly what we do: Albeit being slow and noisy, path tracing is usually the method of choice for most "master" images, as it is unbiased, makes no assumptions at all on scene, BRDFs, or lighting, and – in particular – its simplicity allows for verifying its correctness much easier than more complex techniques. Targeting an offline precomputation anyway, being slow is not really a problem.

In practice, we usually specify the (average) number of paths P per (x_j, ω_k) sample. Then, for each surface point x_j we start $K \times P$ paths, with the initial direction sampled via a Halton sequence. To avoid the sample directions directly coinciding with the ω_k, the Halton pattern is slightly shifted. For each of these directions, we then trace a path, and distribute it's radiance to the L_{jk} depending on $w_k(x)$.

4.2. Parallel and Progressive Precomputation

For complex illumination paths, getting a noise-free result can be computationally quite expensive, and can easily take several hours, up to a few (compute-)days. Additionally, choosing the correct number of samples is quite problematic, and a wrong choice easily leads to either a noisy solution, or to the simulation not being ready once it is needed.

For this reason, we have implemented a very simple parallelization scheme: As in a path tracer all samples are independent of each other, we can just as well compute several independent solutions – using the same set of surface points and discrete directions, but different random numbers – and average these later on. Thus, we choose the surface and direction samples deterministically, and afterwards use the process ID as a random seed for computing the L_{jk}.

We then let each cluster node generate such solutions (with relatively low sample rates of $\sim 8-16$ paths per direction) in an infinite loop. The current status of the simulation can then be checked any time by simply averaging all of the already computed partial results. Apart from an effective parallelization tool, this additionally provides a simple way of progressively getting intermediate results.

4.3. Separate Direct Illumination Computation

An additional method for reducing noise and thus reduce precomputation time is to compute the direct lighting component of L_{jk} separately, and use the path tracer only for the indirect component. For that purpose, we have used an implementation of structured importance sampling [ARBJ03] for environmental illumination, and can use Instant Radiosity-like virtual point lights for area lights. Using deterministic illumination samples guarantees to yield noise-free results (see Figure 2).

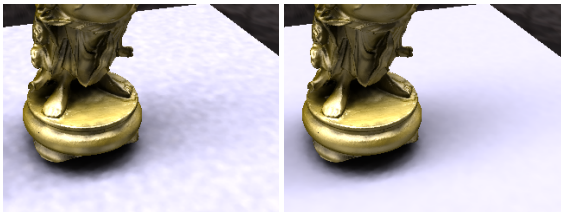

Figure 2: *Impact on computing the direct component of the illumination using Structured Importance Sampling (right) vs a plain path tracer (right). As expected, the results are entirely smooth, except for minor discretization artifacts. Note that the separate computation of direct illumination in* only *performed during preprocessing, and* not *during rendering.*

Once the direct illumination is known, we can also use an iterative scheme, in which – starting with the direct illumination only – the DIRMap data from the previous pass is used to compute a new DIRMap by final gathering data. This in fact corresponds directly to typical iteration schemes in radiosity, and works quite well, except that two DIRMaps are required during precomputation.

4.4. Compression

Once all of the L_{jk} are computed, depending on J and K one can easily end up with up to several hundred Megabytes of data. Though on current hardware we consider this quite practicable, it might be a problem for extremely large scenes.

Fortunately, there is huge redundancy in the data computed, as nearby samples are likely to have similar – if not the same – incident radiance fields. In fact, most regions – except shadow boundaries – often show only minor variations. Additionally, we can live quite well with a lossy compression, as any kind of decompression artifacts in individual samples will be smoothed away by the interpolation during reconstruction, as well as by the convolution with the BRDF.

As a result, lossy compression techniques work extremely well. For example, one can see the entire L_{jk} as a $3K$ dimensional dataset with J entries, and can apply vector quantization (VQ) [Gra84] to it. Though one has to take care about the huge dynamic range in the data, using VQ is straightforward, and very efficient. Initial results have shown to yield compression rates of 100:1 without visible artifacts. Decompression cost is negligible, as VQ decompression is simply

an indirection. However, we usually do not need to use this technique, as we are usually not limited by memory, anyway.

Note that we could also just simply decimate the DIRMap by throwing away all samples whose data is closely enough represented by its neighbors – as we perform interpolation instead of *density* estimation, we can arbitrarily change the sampling density without introducing artifacts.

5. Rendering

Once the DIRMap solution is available, using it during rendering is very simple: We simply (interactively) ray trace the model, and – for each surface intersection x – reconstruct an approximation of the incident radiance field from the DIRMap data, which we then feed into the BRDF.

5.1. Reconstructing the Incident Radiance Field

For reconstructing the incident radiance field around a point x we first perform a k nearest neighbor (kNN) query around x, yielding the k nearest DIRMap samples, each consisting of K discrete directional samples. As all samples use the same set of world-space directions, we can easily combine the resulting information via filtering, yielding a filtered discrete approximation of the incident radiance field $(L_k(x))_{k=1}^K$ with

$$L_k(x) = \frac{\sum_{j \in kNN(x)} w(x,x_j) L_{jk}}{\sum_{j \in kNN(x)} w(x,x_j)}.$$

As filter $w(x,x_j)$ we have experimented with several kinds of filters: For a box filter, the voronoi-like regions of the surface discretization are usually quite visible, in particular in regions with a steep illumination gradient, but evan a simply hat filter already reduces them significantly (see Figure 3). Though these have not (yet) been implemented, in particular edge- and gradient-preserving filters – such as the bi-lateran and tri-lateral filter [CT03] – could further reduce discretization artifacts. Due to the highly regular organization of our sample, these should be simple to apply.

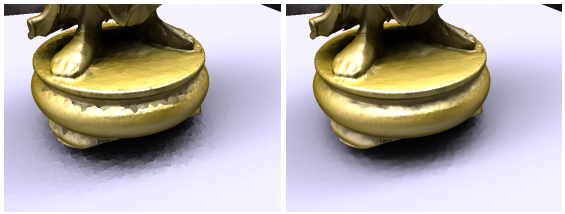

Figure 3: *Using a hat filter (right) vs. a box filter (left) for reconstruction. Edge- and gradient-preserving filters should work even better, but a hat-filter is usually sufficient.*

5.2. Computing the Reflected Radiance

Once the filtered incident radiance samples are available, computing the reflected radiance is as simple as feeding the $(L_k(x), \omega_k)_{k=1}^K$ samples into a shader, yielding

$$L(x,\omega_o) \approx \sum_{k=1}^{K} f_r(x,\omega_o,\omega_k) L_k(x) cos(n_x,\omega_k)$$

Note that this is exactly the same as shaders in Monte-Carlo based renderers usually work, except that the light samples (L_x,ω_k) are not generated by sampling the light sources, but by querying them from the DIRMap. Thus, we do not need any special BRDF representation (such as an SH representation), but can apply our method to any sample based shader.

5.3. Performance Optimizations

In the way just described, the method can be quite costly, as both the filtering and the kNN queries consume considerable time. Thus, we have applied several optimizations.

SIMD Filtering: The filtering operations are computationally expensive, and have very regular execution patterns, thus lend well to a SIMD implementation. As expected, implementing the filtering operations in SIMD gave a good speedup, and allowed for cutting filtering time roughly in half. Note however that this implementation is not yet fully optimized: So far we used only a straightforward approach, and have not yet rearranged the code for further speedups.

Precomputing the kNNs: Even with a highly optimized kNN implementation, a kNN query is quite costly. In fact, in the initial implementation the kNN query took roughly as much time as ray tracing, filtering, and shading together. To improve on that, for each sample point x_j we pre-compute and store the kNN set. During rendering, for any point x we then take the kNN set of the nearest x_j. Finding only *one* nearest neighbor is considerably faster, as the kNN set does not have to be kept in a heap during traversal, and because the search radius drops much faster.

Obviously, x might not have *exactly* the same kNN set as its closest x_j. As however we use the kNNs *only* for interpolation – and not for density estimation – this in fact is no problem at all. Thus, precomputing the kNN set does not introduce any artifacts at all, but could significantly increase performance: For the Buddha scene (see below), precomputing the kNN sets improved performance by roughly 50%.

Nearest Neighbor Caching: kNN queries can be significantly accelerated if a good upper bound on the maximum search radius is known, as large parts of the kd-tree can then be quickly culled during traversal. As we are only looking for *one* NN sample, *any* of the DIRMap points yields a conservative estimate. In particular, taking the nearest sample of the previous ray is a very good estimate – as we trace (almost) only secondary rays, neighboring pixels will have nearby hitpoints, and the previous pixel's NN sample will be a good guess for the current pixel. Using this caching technique can further reduce the query times of the kNN queries.

5.4. Extensions

Apart from several performance optimizations (see above), there are also some more algorithmic (potential) extensions of our method that we would like to briefly mention.

5.4.1. BRDF Sampling for Extremely Glossy BRDFs

Obviously, the more glossy the surface gets, the more likely is it that a highlight may be missed due to the glossy lobe falling exactly in-between the discrete direction samples. Though this in practice has not led to any problems at all, it could – apart from taking more samples – be solved by *not* evaluating the BRDF with the precomputed incident radiance samples, but to instead sample the BRDF. For the sampled direction, the incident radiance could then be interpolated from the DIRMap data. Eventually, both ways of sampling – inserting the ω_k directions into the BRDF on one side, and sampling the BRDF on the other side – could be combined via Multiple Importance Sampling [VG95].

5.4.2. Lighting from Dynamic Environments

Another extension of our method would be extend it such as to allow for dynamically changing the environmental illumination similar to Precomputed Radiance Transfer (PRT) [SKS02]: Instead of storing the precomputed illumination at each sample point as an RGB triple, one could just as well store N SH coefficients that describe L_{jk} as a function of the environment map. The only modification required to make that work would be to first reconstruct the L_{jk} via a dot product before filtering. The scheme should be trivial to implement, and would immediately allow for dynamic (environmental) lighting as available in PRT.

6. Experiments and Results

In the previous Sections, we have already discussed the individual ingredients of our technique. Now, we are going to evaluate the complete technique. In particular, we are going to demonstrate that

1. it can well handle geometric complexity,
2. it can be applied to arbitrary kinds of geometry,
3. it can handle glossy BRDFs and even measured BTFs,
4. it supports all kinds of global effects,
5. it yields quality comparable to offline path traced images, but with a single ray per pixel, without final gathering.
6. it yields interactive performance.

If not noted otherwise all frame rates correspond to 640×480 pixels. Path traced master images are computed with 1,000-65,000 paths per pixel, usually over night. All experiments are performed on a dual-2.4GHz Opteron PC with 4GB RAM, or on a cluster thereof with 8GB RAM each.

6.1. Glossy Buddha

The Buddha model (see Figure 4) consists of roughly one million triangles, and is rendered with 100,000 randomly chosen surface points distributed over model and ground plane, as well as usually 8 kNNs for reconstruction. The DIRMap is precomputed with a highly glossy Phong BRDF mimicking the appearance of gold. Using 18 directions, this results in only 30 MB in total, for 160 MB of geometry data (kd-tree and triangles). Though the surface discretization is still somewhat visible on the base plane, the overall rendering quality is already quite high, as can particularly be seen by comparing to the master image.

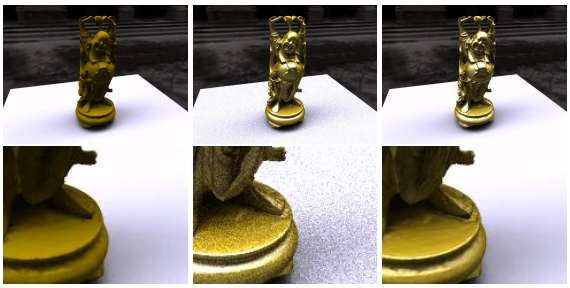

Figure 4: *Glossy reflections on the Stanford Buddha (1 million triangles). Left: Diffuse only. Center, with a glossy component as well. Right: Path traced reference image (1,000 paths per pixel). Note that the left and middle image use the* same *DIR-map from the middle image.*

6.1.1. Performance

Although the scene is already quite complex, we achieve $2.5 fps@640 \times 480$ pixel on a single 2.4GHz dual-Opteron PC, and higher frame rates can be achieved by adding more PCs. Most of the time is actually spent in reconstructing the incident radiance field and shading: Roughly 10–15% of the time is spent on the nearest neighbor query (already using the precomputed kNN set), another ∼25–30% on the filtering operations, and yet another ∼30% on evaluating the shader with the filtered samples. Less than one third of total rendering time is actually spent on "typical" ray tracing, already including overhead like ray generation and frame buffer handling. This is a severe cost from a realtime ray tracing point of view, but is quite reasonable for getting complete global illumination including glossy effects.

6.1.2. A-posteriori BRDF editing

An interesting side effect of our method can be seen on the left of Figure 4. While the Buddha there is completely diffuse, it is actually illuminated by the DIRMap from the center, glossy Buddha. Though this is actually wrong in a physical sense, it looks totally plausible: As we do not store the *reflected*, but rather the *incident* radiance, the object itself is still correctly lit even after changing its BRDF – the only error is that changing an objects BRDF should have indirect effects on the incident illumination on other surfaces, which

is not correctly accounted for. These indirect effects however are usually quite small, and get further hidden by the BRDF of the second surface where they actually take effect. Though extreme examples can be constructed, in practice the indirect effects of changing the BRDF of one surface have shown to be hardly measurable.

This is a powerful feature of our method, as to a certain degree it allows for at least evaluating/experimenting with different material variants (also see Figure 7). Though this is physically incorrect, the error is very small (probably less than approximations and bias in other techniques), and the rendered images look totally plausible.

6.2. Application to Non-Polygonal Scenes

Not using any surface mesh, our method is applicable to any kinds of surface primitives. As an example, Figure 5 shows our technique applied to the point-based Iphigenia model (one million points), rendered with a point-based ray tracing module [WS05]. As the sample distribution does not matter in our technique, we have simply used the model's input points for placing the DIRMap samples. As the point-based surface only approximates (not interpolates) these points, we had to manually specify a large enough ray offset to avoid self-intersection with the model. In all other respects, there is no difference at all to the triangular models. Other primitive types such as freeform surfaces [BWS05] and isosurfaces [WFM*05] are also supported in our framework, and can be used just as well. Using our technique on the Iphigenia, we achieve around 4 frames per second, on a single PC (see Figure 5). Note in particular to totally different appearance with global illumination, due to color bleeding from the reddish environment.

Figure 5: *DIRMaps applied to a ray traced one-million-point model of an Iphigenia statue. Left: Ambient shading, as usually done for point-based models. Center: Direct illumination from hand-placed light sources. Right: Globally lighted in St.Peters, running at* $\sim 4fps@512\times512$ *pixels, on a single 2.4GHz dual-Opteron PC.*

6.3. Application to Complex BRDFs (BTFs)

As already demonstrated in Section 6.1) our method allows for also producing directionally dependant effects like highlights. In particular, this also allows for using our method with real-world measured BRDFs such as bidirectional texture functions [DvGNK99, MMS*04]. Figure 6 shows a model with a measured wool BTF, computed both with direct illumination from a few hand-placed point lights, as well as globally illuminated using our technique (100,000 surface samples, 16 discrete directions). As expected global effects significantly enhance the realism of such models. The impact of the directional component becomes even more pronounced during interaction, once the highlights start to move around. In particular, Figure 6 shows that also for BTFs the rendering quality of our method is similar to an offline path traced image. At the given quality, the model can be rendered at a few frames per second on a single PCs.

6.4. A Highly Complex Real-world Example

In the previous examples, we have already shown that our method can be applied to complex BRDFs and complex geometries (one million triangles and one million points, respectively). So far however, all scenes have been topologically simple, contained only simple, mostly direct lighting, and could in principle also be handled by techniques such as e.g. PRT. To demonstrate that our method is not limited to such simple topologies and lighting situations, we have also applied it to a real stress test, a model of a complete Mercedes Benz C-Class model. Apart from being modelled almost exclusively with BTFs, the model completely lacks a reasonably surface mesh (also see [DAK*04]) that makes PRT- or Radiosity techniques hard to apply.

Probably most challenging, the illumination is quite complex, being highly directed in some areas (e.g., below the windshields), and quite diffuse in others. In particular, *all* light entering the model has to penetrate through the refractive glass shaders of the windshields. Note that we have *not* taken the usual trick of simply making the glass transparent for the global illumination computations, but really simulate refractive and reflective glass also in the precomputation step. In particular, this means that techniques like instant global illumination cannot be applied to this model, and that the separate computation of direct illumination – a usual trick in e.g., Photon Mapping – cannot be applied, either. Also note that many areas (such as, e.g., the leg-rest) are lighted highly indirectly, which would be hard to cap-

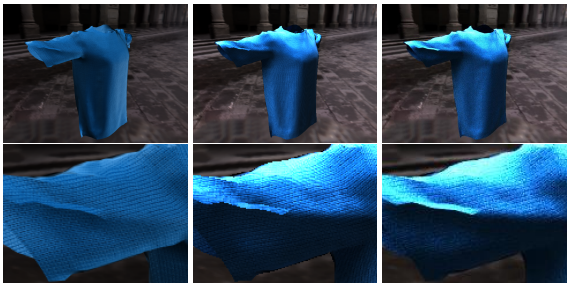

Figure 6: *BTF shirt, showing a model of a shirt with a measured Wool BTF. Note the glossy effects at grazing angles. Left: Direct illumination from two hand-placed point lights. Center: Our method, globally lighted from the Uffizi environment map. Right: Path traced reference image.*

Figure 7: *Our technique applied to a real stress-test, a Mercedes C-Class Model (1M triangles) modelled almost completely with BTFs, and illuminated from an HDR environment map through refractive windshields. Note that we have* not *used the usual trick of making the glass transparent for shadow rays, and that all interior illumination in principle is an (unfocussed) caustic. All global illumination effects are accounted for. Note in particular the highlights and self-shadowing on the BTFs, the smooth shadows of the gearstick, the smooth lighting even in the indirectly lit legroom and the subtle reflections in the windshields. On a cluster of 20 PCs, this scene can be rendered at around 5–10 frames per second at* 640×480 *pixels. From a) to b) and c), the gearbox' material has been changed from wood to aluminium, both measured BTFs of the real world car's materials.*

ture at all with techniques like Instant Global Illumination, as huge numbers of VPLs would have to be generated on the environment map to eventually illuminate these regions at sufficient quality. In fact, this model was the main motivation of our work, as we did not believe existing techniques would be able to handle such a configuration of geometry, lighting, and BRDFs at interactive rates.

As can be seen from Figures 7 and 8, the overall rendering quality of using our method in this model is quite high, and is comparable to offline renderings. For this example, only 1 million sample points with 18 directions have been used, totalling a mere 293MB of precomputed data (roughly the same as for the geometry), even without any compression at all. Vector quantization by a factor of 1:100 (3MB DIRMap plus 12MB sample coordinates) yields comparable results.

Some artifacts are still visible, but are mostly due to missing or buggy texture coordinates in the model, and are also present in the path traced master images. In this model, a single PC does not suffice to achieve interactive performance. However, using a cluster of 20 dual-2.4GHz Opteron PCs, all of the images in Figures 8 and 7 can be rendered interactively at 5-10fps@640×480 pixels. Though we cannot interactively generate full screen resolutions yet, we can use the usual trick of zooming the 640×480 image to full screen during motion, and switch to full-resolution images as soon as the user stops for inspection – seeing the above rendering quality at full screen has shown to be particularly impressive to users, and a reduced rendering resolution during navigation is usually tolerable in exchange for the faster feedback. Additionally, we can also use the afore-mentioned feature of a-posteriori BRDF editing (see Section 6.1.2) in order to investigate how the model would look if, e.g., the wooden parts are replaced by an aluminium material (see Figure 7). Though this is not entirely correct in the physical sense, the subtle errors introduced by this are quite well acceptable in practice, making the feature an important tool in evaluating different furnishing variants.

Figure 8: *Qualitative validation of our method (left) to a path traced reference image (right). In print, the difference is hard to see at all. Except for some residual noise in the path traced images, both methods yield the same results.*

6.5. Application to Massively Complex Geometries

Since our method is independent of geometry, in principle it can also be applied to massively complex scenes. In fact, the density of the surface samples only has to match the frequency of the (incident) illumination, and is otherwise completely independent of the model's actual triangle count.

As a proof of concept, we have applied our method to the Boeing 777 airliner model, which consists of 350 million triangles (40GB of data), rendered using an out-of-core (OOC) ray tracing framework [WDS04]. Figure 9 shows this model with global illumination from an HDR environment map (realistic materials and light sources unfortunately are not available), using only 4 million surface samples. Though

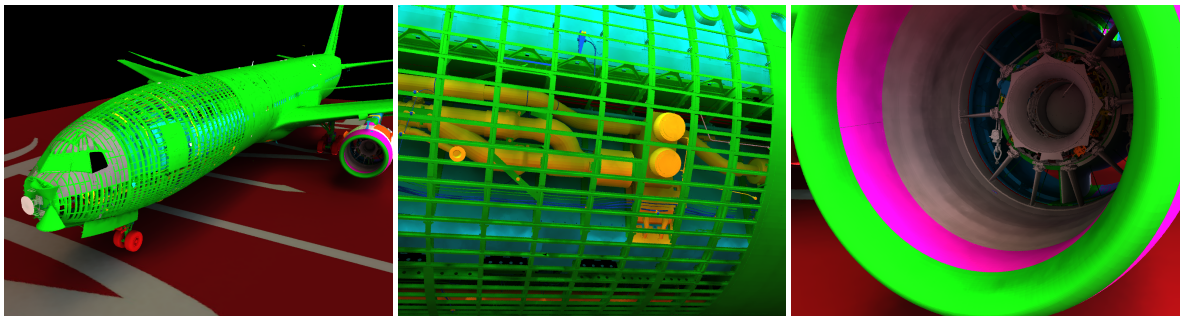

Figure 9: *Our technique applied to the 350 million triangle "Boeing 777" model, using only 4 million DIR-map samples. Left: overview. Center: Zoom onto the belly. Right: The engine. This is still work in progress, as the preprocessing for an out-of-core model are still far too large (several days). However, it already demonstrates the potential of using our method for such models as well. Using OOC ray tracing on 6 dual-Opteron PCs, these views render at 3.7, 3.5, and 3.2 frames per second, respectively.*

some artifacts are still visible, even at that small sampling rate the overall quality on the outside is quite OK. Using 6 cluster nodes, this example runs at roughly $2-4$ frames per second, at 640×480 pixels.

Note however that these results are quite preliminary. In particular, the precomputation in this particular model is *extremely* costly, as sampling a 40GB model with random paths on an 8GB (minus the DIRMap data) PC leads to excessive disk thrashing. For the interior, using only 4 million points for a 350 million triangle points is far too coarse. Higher resolutions unfortunately are not yet available, due to the excessive precomputation times in that particular model. Nevertheless, these preliminary results already show the potential of using our method even for such kinds of models.

7. Summary and Discussion

In this paper, we have proposed Discretized Incident Radiance Maps, a novel technique that uses a precomputed discretized representation L_{jk} of the incident radiance field $L'(x,\omega)$ in combination with nearest-neighbor interpolation during rendering. For any surface point x our method yields an approximation of the incident radiance field at x, which can be used for (globally) illuminating even highly glossy BRDFs and BTFs with a single ray per pixel. Being independent of any surface mesh, the method can also be applied to highly complex models, as well as to non-triangular surfaces like Bézier patches, iso-surfaces, and point-based models. In particular, it does not suffer from any meshing artifacts at all.

Using a path tracer for computing the L_{jk}'s all kinds of global illumination effects are accounted for: Direct and indirect illumination from area lights or HDR environment maps, smooth shadows, color bleeding, specular illumination (caustics), glossiness, etc. are all supported. As usually only a single ray is required per pixel interactive performance can be achieved even for offline-standard rendering quality. Obviously, reflections and refraction (such as glass) are trivially supported via shooting secondary rays.

7.1. Limitations

Even with all these advantages, there are still several limitations: First, some advanced physical effects – as for example polarization and wavelength-dependent effects – are currently not supported. Similarly, tone mapping is not yet supported in our framework, but would be particularly interesting to use as well. Also, discretization artifacts can appear if the surface sample density is too small. This dependence on sampling rate also poses a problem for really large models, as both memory consumption and precomputation time rise linearly with the number of surface samples.

Finally, a sufficiently large number of PCs is required to maintain interactivity except for very simple models. Nevertheless, using the same hardware setup and frame rate, competing techniques like Instant Radiosity cannot even come close to the rendering quality achieved by our method.

7.2. Potential artifacts

Quite obviously, using a discretization based approach can also lead to artifacts, including discretization error, shadow bleeding, energy bleeding, noise, blurring, etc. Though we originally feared discretization artifacts to be worst, for most scenes these can be handled quite well using the kNN filtering described above. Shadow initially posed a problem, in particular for "badly" modelled scenes where additional, completely occluded geometry is available slightly behind the actually visible surface. In that case, the black samples of the occluded surface are likely to shine through. For most parts, this can be handled by slightly shifting the sample rays' origins away from the surface point, which additionally allows for letting samples slightly across a corner still contribute to the current sample point. In particularly in the C-class model, this allowed for completely getting rid of any shadow bleeding. Note however that too much offsetting can obviously result in the opposite effect, energy bleeding. The latter in practice has shown to be much less of a problem.

Of course, even with filtering and ray offsetting, the sample density should be high enough to capture the frequency

of the (incident) lighting, as no filtering or heuristic can replace lighting information that simply is not available. Finally, noise can be a problem, in particular for extremely unlikely light transport paths such as in some interior regions of the 777 model. While the technique described in Section 4.3 helps a lot, getting rid of residual noise eventually requires taking enough samples during preprocessing.

In summary, yes, our technique *can* suffer from artifacts like noise, discretization artifacts, and shadow and energy bleeding. In particular with too few samples, too few preprocessing, and badly chosen parameters, these artifacts can be quite visible. Note however that these artifacts are "common" problems in most discretization based techniques, and appear similarly in, e.g., Radiosity and Photon mapping. Fortunately, such artifacts only appear in highly complex scenes at all, and finding reasonable parameters to make them go away in these is rather simple, eventually leading to the high-quality, artifact free results as shown earlier on.

7.3. Comparison to Existing Techniques

By design, our technique shares properties of several existing global illumination techniques (also see Table 1):

Like *Photon Mapping* [Jen01], we use an approach that is independent of geometry, and thus avoids any of the meshing problem of typical finite element approaches like Radiosity or PRT. Unlike Photon Mapping, we use nearest neighbor *interpolation* – not density estimation – thereby getting rid of the density noise in the estimate, and avoiding the need for final gathering and separate direct illumination computation.

Like *Instant Global Illumination* [WKB*02], we build on interactive ray tracing, allowing for reflections, refractions, etc. Unlike IGI, our method allows for supporting complex illumination and BRDFs, and in general yields better rendering quality.

Like *PRT*, we precompute a high-quality global illumination solution for each finite element that can directly be used during rendering. Unlike PRT, we are independent of any surface mesh, can handle arbitrary BRDFs and lighting, and support specular effects like reflection and refraction.

Like *Radiosity*, we use a discretization into finite elements to store the precomputed illumination, thereby being independent of the complexity of the lighting. Unlike radiosity, we are not limited to diffuse surfaces only, and in particular get independent of the usual meshing problems.

A cross-comparison of the features of these methods is shown in Table 1: Except for not (yet) being able to handle dynamic lighting and dynamic scenes, our method nicely balances the advantages and disadvantages of the respective methods, resulting in a framework that combines their positive aspects while avoiding their respective drawbacks.

	PRT	Radio sity	RT+ Rad.	Pht. Map	IGI	DIR maps
dyn. geom./ill.	(+)	(-)	-	-	++	-
dynamic illum.	(+)	-	-	-	++	(-)
interactive	+	(+)	(+)	–	+	+
mesh indep.	(–)	–	–	+	+	+
single ray/pix	(+)	(+)	(+)	(-)	–	+
arbit.BRDFs	(+)	–	–	(+)	(-)	+
reflections etc	(-)	-	+	+	+	+
complex illum.	(+)	(+)	(+)	+	–	+

Table 1: *Features of DIRMaps wrt. existing techniques like PRT, Radiosity, Photon Mapping, and Instant Global Illumination (IGI). Except for not supporting dynamic scenes, DIRMaps nicely combine the advantages of existing methods, while avoiding their limitations.*

8. Conclusions and Future Work

In conclusion, we have presented a novel method that allows for high-quality, near-production-quality global illumination walkthroughs even for scenes with complex geometry, complex BRDFs, and highly complex lighting, which cannot be handled by existing interactive techniques. Though our method method is (still) limited to static scenes and static lighting only, we believe it to be a major step towards real-time photorealistic rendering. The advantages of supporting global illumination in interactive applications become particularly obvious when comparing the images rendered with out technique to those of rendering the same scene with a non-global illumination package [WBE*05] (see Figure 10): Even though the non-global image was already rendered with a ray tracer that supports Glas, BTF, etc. [WBE*05], the impact of global effects is quite pronounced.

As next steps, we would like to investigate the aforementioned scheme of storing PRT-coefficients instead of an RGB triple per L_{jk} sample (Section 5.4.2), thereby allowing for interactively modifying the environment map. Also, the application to massively complex scenes seems highly interesting (Section 6.5). Real-world validations would also be interesting but requires cooperation from industrial partners. Finally, it seems interesting to investigate hierarchical techniques for reducing the precomputation times. There is

Figure 10: *Comparison of our technique (right) to a non-global illumination rendering (left). Though the non-global image already used a ray tracer with BTF support, the increased level of realism is quite noticeable.*

plenty of literature of such techniques in global illumination, and many should be applicable to our setting as well.

From the applications point of view, it would be integrate the proposed technique into the framework described in [WBE*05]: As the DIRmap technique is independent of triangular geometry, it should work well also with the freeform car model used in that paper, and adding it as a special "shader" for the car interior should be straightforward.

Acknowledgements

We would like to thank the RealReflect project – in particular Daimler-Chrysler AG and the Computer Graphics Group of Bonn University - for making the DaimlerChrysler model and the BTF data available for our experiments. The Iphigenia model has been provided by Leif Kobbelt (RWTH Aachen), and the Boeing 3D data was provided by and used with permission of Boeing Corp. All Software has been implemented inside inTrace's inView/OpenRT software, to which numerous people have contributed. Johannes Günther from the Max Planck Institute (MPI) in Saarbrücken has also provided significant help in coding, in particular for the path tracer reference comparison. Most of the compute infrastructure has been provided by the MPI Saarbrücken.

References

[ARBJ03] AGARWAL S., RAMAMOORTHI R., BELONGIE S., JENSEN H. W.: Structured Importance Sampling of Environment Maps. *Computer Graphics (Proceedings of ACM SIGGRAPH) 22*, 3 (2003), 605–612.

[Bek99] BEKAERT P.: *Hierarchical and Stochastic Algorithms for Radiosity*. PhD thesis, Katholieke Universitiet Leuven, Belgium, 1999.

[BWS05] BENTHIN C., WALD I., SLUSALLEK P.: Techniques for Interactive Ray Tracing of Bézier Surfaces. *Journal of Graphics Tools*, to appear (2005).

[CPC84] COOK R., PORTER T., CARPENTER L.: Distributed Ray Tracing. *Computer Graphics (Proceeding of SIGGRAPH 84) 18*, 3 (1984), 137–144.

[CT03] CHOUDHURY P., TUMBLIN J.: The trilateral filter for high contrast images and meshes. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), pp. 186–196.

[DAK*04] DMITRIEV K., ANNEN T., KRAWCZYK G., MYSZKOWSKI K. O., SEIDEL H.-P.: A CAVE System for Interactive Modeling of Global Illumination in Car Interior. In *ACM Symposium on Virtual Reality Software and Technology* (2004), pp. 137–145.

[DBMS02] DMITRIEV K., BRABEC S., MYSZKOWSKI K., SEIDEL H.-P.: Interactive Global Illumination using Selective Photon Tracing. In *Proceedings of the 13th Eurographics Workshop on Rendering* (2002), pp. 21–34.

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KONDERINK J. J.: Reflectance and Texture of Real-World Surfaces. *ACM Transactions on Graphics 18*, 1 (1999), 1–34.

[Gra84] GRAY R. M.: Vector Quantization. *IEEE ASSP Magazine 1*, 2 (1984), 4–29.

[HSA91] HANRAHAN P., SALZMAN D., AUPPERLE L.: A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (Proc. of ACM SIGGRAPH)* (1991), pp. 197–206.

[Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*. A K Peters, 2001.

[Kaj86] KAJIYA J. T.: The Rendering Equation. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1986), Evans D. C., Athay R. J., (Eds.), vol. 20, pp. 143–150.

[Kel97] KELLER A.: Instant Radiosity. *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1997), 49–56.

[KSS02] KAUTZ J., SLOAN P.-P., SNYDER J.: Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics. In *Rendering Techniques* (2002), pp. 301–308. (Proceedings of Eurographics Workshop on Rendering).

[LW93] LAFORTUNE E., WILLEMS Y.: Bidirectional Path Tracing. In *Proc. 3rd International Conference on Computational Graphics and Visualization Techniques (Compugraphics)* (1993), pp. 145–153.

[MMS*04] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, Synthesis and Rendering of Bidirectional Texture Functions. In *Eurographics 2004, State of the Art Reports* (2004), pp. 69–94.

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple Product Wavelet Integrals for All-Frequency Relighting. *Computer Graphics (Proceedings of ACM SIGGRAPH) 23*, 3 (2004), 477–487.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *Proceedings of ACM SIGGRAPH* (2002), pp. 527–536.

[TPWG02] TOLE P., PELLACINI F., WALTER B., GREENBERG D. P.: Interactive Global Illumination in Dynamic Scenes. *ACM Transactions on Graphics 21*, 3 (2002), 537–546. (Proceedings of ACM SIGGRAPH 2002).

[VG94] VEACH E., GUIBAS L.: Bidirectional Estimators for Light Transport. In *Proceedings of the 5th Eurographics Workshop on Rendering* (1994), pp. 147 – 161.

[VG95] VEACH E., GUIBAS L.: Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *SIGGRAPH 95 Conference Proceedings* (1995), pp. 419–428.

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004. Available at http://www.mpi-sb.mpg.de/~wald/PhD/.

[WBE*05] WALD I., BENTHIN C., EFREMOV A., DAHMEN T., GUENTHER J., DIETRICH A., HAVRAN V., SLUSALLEK P., SEIDEL H.-P.: *A Ray Tracing based Framework for High-Quality Virtual Reality in Industrial Design Applications.* Tech. Rep. UUSCI-2005-009, SCI Institute, University of Utah, 2005. available at http://www.sci.utah.edu/~wald.

[WBS03] WALD I., BENTHIN C., SLUSALLEK P.: Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proceedings of the 2003 Eurographics Symposium on Rendering* (Leuven, Belgium, June 2003), Christensen P. H., Cohen-Or D., (Eds.), pp. 74–81.

[WDS04] WALD I., DIETRICH A., SLUSALLEK P.: An Interactive Out-of-Core Rendering Framework for Visualizing Massively Complex Models. In *Rendering Techniques 2004, Proceedings of the Eurographics Symposium on Rendering* (2004), pp. 81–92.

[WFM*05] WALD I., FRIEDRICH H., MARMITT G., SLUSALLEK P., SEIDEL H.-P.: Faster Isosurface Ray Tracing using Implicit KD-Trees. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 562–573.

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive Global Illumination using Fast Ray Tracing. *Rendering Techniques* (2002), 15–24. (Proceedings of the 13th Eurographics Workshop on Rendering).

[WS05] WALD I., SEIDEL H.-P.: Interactive Ray Tracing of Point Based Models. In *Proceedings of 2005 Symposium on Point Based Graphics (PGB)* (2005), p. to appear.