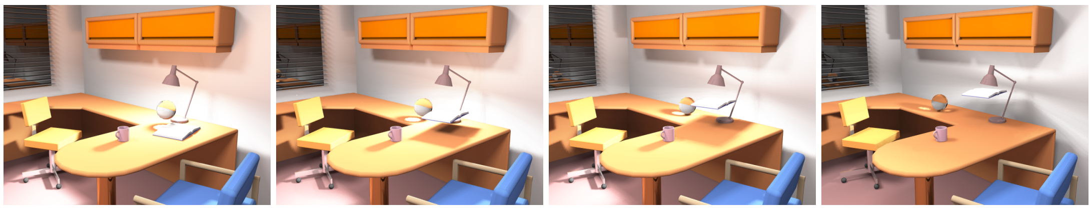
Thirteenth Eurographics Workshop on Rendering (2002)
P. Debevec and S. Gibson (Editors)

Interactive Global Illumination using Fast Ray Tracing



Ingo Wald[†] Thomas Kollig[‡] Carsten Benthin[†] Alexander Keller[‡] Philipp Slusallek[†]
[†]Saarland University [‡]Kaiserslautern University

Abstract
Rasterization hardware provides interactive frame rates for rendering dynamic scenes, but lacks the ability of ray tracing required for efficient global illumination simulation. Existing ray tracing based methods yield high quality renderings but are far too slow for interactive use. We present a new parallel global illumination algorithm that perfectly scales, has minimal preprocessing and communication overhead, applies highly efficient sampling techniques based on randomized quasi-Monte Carlo integration, and benefits from a fast parallel ray tracing implementation by shooting coherent groups of rays. Thus a performance is achieved that allows for applying arbitrary changes to the scene, while simulating global illumination including shadows from area light sources, indirect illumination, specular effects, and caustics at interactive frame rates. Ceasing interaction rapidly provides high quality renderings.

1. Introduction

Global illumination has a wide range of applications. It is instrumental in achieving realistic images of virtual objects in e.g. movie production or design processes in car and airplane industry as well as in architecture. Fast responses to interactive changes guarantee the efficiency of the design process and in addition production cost is reduced by faster rendering techniques.

With the availability of fast and inexpensive rasterization hardware, interactive 3D graphics has become a mainstream feature. Although the achievable realism significantly has been increased by techniques like e.g. multi-texturing, vertex programs, or pixel shaders [16, 21], due to the lack of ray tracing features global illumination is out of reach. Commonly, more complex lighting effects are precomputed with existing global illumination algorithms, which are expensive and relatively slow. Obviously, this only works for static illumination in static scenes and consequently does not satisfy the requirements of interactive applications in highly dynamic environments.

The research on fast and efficient ray tracing drastically has changed the environment in which global illumination operates. Even on commodity hardware, ray tracing now achieves interactive frame rates in dynamic settings [38, 39, 37]. In addition, novel techniques allow for an efficient and scalable distribution of the computations over a cluster of computers [36].

One could assume that global illumination algorithms would equally benefit from these developments. However, such fast and distributed ray tracing implementations impose constraints that are incompatible with most existing global illumination algorithms.

In Section 2 we review fast ray tracing systems and in particular discuss the constraints those systems impose on global illumination algorithms. In Section 3 we then discuss previous work with respect to these constraints. We derive our approach in Section 4 and present the results, strengths, and limitations of our system in Section 5.

2. Constraints imposed by Fast Ray Tracing

Ray tracing is one of the oldest and most fundamental techniques used in computer graphics [1, 43, 5]. Due to the high cost for tracing single rays, combined with the need to shoot millions of rays, computing one single image took minutes to hours. By exploiting the inherent parallelism of ray tracing, Muuss [19, 18] and Parker et al. [24, 25] have achieved interactive performance on shared memory supercomputer systems by massive parallelization.

Wald et al. [38] have shown that interactive ray tracing performance can also be obtained on inexpensive, commodity PCs. Their implementation is designed for good cache performance using reordering of computations, optimized intersection and traversal algorithms, and a careful layout and alignment of core data structures. These techniques increased the performance by more than an order of magnitude.

Fast ray tracing also scales also well in a distributed memory environment using commodity PCs and networks [39]. Distributed computing is implemented by a client/server model with tile-based load-balancing and hiding of network latencies by dynamic reordering of computations. With distributed ray tracing interactive rendering performance is achieved even for scenes with tens of millions of triangles. Extending the system to handle dynamic environments [36] allows for real user intervention as required by an interactive application.

It is an obvious next step to use the fast ray tracing engine to speed up existing global illumination algorithms that heavily rely on ray tracing. However, it turns out that this is not as simple as it seems at first, as most of these algorithms are incompatible with the constraints imposed by such a fast ray tracing system.

2.1. Performance Constraints

Even with a fast ray tracer, a global illumination system is limited to a rather small budget of rays per frame. At a target resolution of 640×480 pixels, each frame contains roughly 300,000 pixels. Assuming a network of PCs with 16 processors and a performance of 500,000 rays per second and processor, 27 rays per pixel remain for estimating the global illumination at one frame per second. Note that we are counting individual rays and not complete light paths. In practice, even less rays will be available due to other processing requirements like e.g. shading computations.

Given such extremely low sampling rates, we focus on the major contributions of global illumination, such as direct and multiple-bounce indirect illumination from point and area light sources, reflection and refraction, and direct caustics. We currently neglect more costly effects like glossy reflection or caustics of higher order.

An approach based on pure Monte Carlo techniques would be prone to noise at such low sampling rates. This noise becomes even more disturbing in dynamic environments. Such temporal artifacts should be carefully avoided.

A suitable algorithm must send rays in coherent groups to achieve best performance, because the speedup of the ray tracing engine significantly depends on efficient caching [38]. Then other costs, such as BRDF evaluations, sample selection, and even random number generation, can become new bottlenecks.

2.2. Parallel and Distributed Computing Constraints

Due to price and availability considerations, we are targeting networks of inexpensive but fast PCs with standard network components. Compared to shared-memory systems, the communication parameters of such low-cost equipment differ by several orders of magnitude; bandwidth is low and latencies are high. Thus, the desired algorithm must keep its bandwidth requirements low and must try to hide latencies.

Although classical ray tracing parallelizes trivially, the same does not automatically hold for global illumination algorithms even if they are based on ray tracing. All the global illumination computations must also be decomposable into independent jobs in order to run in parallel across a number of client machines. Furthermore, load balancing requires that the number of jobs is significantly larger than the number of processors so that jobs can be dynamically scheduled.

In addition, the algorithm must minimize synchronization across the slow network, such as updating shared data structures, which would result in costly round trip delays. In this respect many existing global illumination algorithms are very problematic as they heavily depend on global data structures such as e.g. photon maps.

Ideally, communication with clients is based on a pipeline model, where all non-scene data is part of the input job description and output is then piped back to the receiver. This model allows one to simply and efficiently hide communication latencies.

2.3. Interactivity Constraints

Many existing algorithms have to perform lengthy precomputations before first results become available. This amortization strategy is inadequate for interactive applications, where the user should receive immediate feedback. Since a whole frame has to be finished in a fraction of a second, preprocessing must be limited to at most a few milliseconds per frame. Furthermore, it can be amortized over only a few frames, as it might otherwise become obsolete due to interactive changes in the dynamic environment. In static situations, however, accumulation can be used to improve the quality of the global illumination solution.

3. Previous Work

Interactive global illumination using fast ray tracing simultaneously must handle all constraints from the previous section. However, most existing global illumination techniques focus on only some of the constraints while violating others.

Using finite element methods, increasingly complex algorithms have been developed to approximate the global solution of the radiance equation. In diffuse environments, radiosity methods [4] were the first to allow for interactive walk-throughs using rasterization hardware, but required extensive preprocessing and thus were available only for static scenes. Accounting for interactive changes by incremental updates [6, 8] forces the expensive manipulation of global data structures and is difficult to parallelize. While rasterization hardware enables the interactive display of finite element solutions, glossy and specular effects can only be approximated or must be added by a separate ray tracing pass [30].

Instant radiosity [11] allows for interactive radiosity without finite element discretization of the solution. The lighting in a scene is approximated by point light sources generated by a quasi-random walk, and rasterization hardware is used for shadow computation. Although arbitrary interactive changes to the environment were possible, the large number of rendering passes required for a single frame limited interactivity to relatively simple environments. Udeshi and Hansen [32] modified the approach and complemented reflection and refraction effects by ray tracing on a shared memory super-computer with multiple graphics pipes. Although they obtained interactive frame rates, the main drawbacks are the relatively poor scalability even on a shared memory system, missing illumination effects like e.g. caustics, and limited image quality.

Path tracing based algorithms [5, 10, 3, 34, 35] correctly handle glossy and specular effects, but the view dependency requires to recompute the solution for every frame. The typical discretization artifacts of finite element methods are replaced by less objectionable noise [27], which however is difficult to handle over time. Reducing the noise to acceptable levels usually is obtained by increasing the sampling rate and results in frame rates that are far from interactive.

In order to efficiently render effects like caustics that may be difficult to generate with the previous path tracing based approaches, photon mapping [9] can be supplemented. This simple method of direct light simulation results in biased solutions. While the bias is less visible in high density regions such as e.g. caustics, artifacts in low density regions have to be removed by a local smoothing or final gather pass that, however, leads to an excessive amount of rays to be traced.

Exploiting the local smoothness of the irradiance, an extrapolation scheme [42] has been developed that considerably reduces the rendering time required for a local pass. However, far too many of the expensive samples are concentrated around corners as illustrated in [9] and the position of the samples is hardly predictable. This requires either dense initial sampling and consequently is expensive or results in tremendous popping artifacts when changing geometry during interaction. In addition a parallel implementation requires extensive synchronization and communication for maintaining the global data structure.

The principle of caching expensive illumination data allows one to achieve interactive frame rates. This has been exploited by the render cache [40] and the tapestry data structure [28] that reuse results from previous frames by image space interpolation of reprojected cached data. The shading cache [31] performs object space interpolation requiring a local parametrization. These caching schemes perform interactively for high temporal coherence, however fail if caches run cold resulting in non-interactive updates of global illumination effects. In order to minimize interpolation artifacts only converged and therefore expensive samples can be used. This implies a long setup time for filling the caches.

4. Algorithm

In the following we present a global illumination algorithm that meets the constraints of Section 2. It achieves interactive performance at a slightly reduced quality and rapidly generates a high quality solution. For brevity we assume familiarity with the radiance integral equation and refer to standard texts like e.g. [4].

The radiance $L(x,\omega)$ (for the selected symbols see Figure 1) at a point x in direction ω is approximated by

$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + \int_S V(y,x) f_r(\omega_{yx},x,\omega) L_{in}(y,x) G(y,x) dA(y) \\ &\approx L_e(x,\omega) + \sum_{j=1}^{M} V(y_j,x) f_r(\omega_{y_j x},x,\omega) L_j G(y_j,x) \\ &\quad + \frac{1}{\pi r^2} \sum_{j=1}^{N} B_r(z_j,x) f_r(\omega_j,x,\omega)\Phi_j \qquad (1) \end{aligned}$$

with $G(y,x) := \frac{\cos\theta_y \cos\theta_x}{|y-x|^2}$, where $P := (y_j, L_j)_{j=1}^{M}$ is the set of point lights in y_j with radiance L_j [11] and $C := (z_j, \omega_j, \Phi_j)_{j=1}^{N}$ is the set of caustic photons that are incident from direction ω_j in z_j with the flux Φ_j [9]. These sets have to be generated at least once per frame by a random walk with fixed maximum path length. After this preprocessing step the scattered radiance is determined only by visibility tests $V(y_j,x)$ and photon queries, where $B_r(z_j,x)$ is 1 if $|z_j - x| \le r$ and 0 else.

For each pixel a primary ray is shot and, if the material has a diffuse component, the hitpoint is illuminated by (1). From there for each specular and transparent component a path is generated by randomly following further singular scattering events. The paths are terminated if the random decisions select the diffuse component of f_r. The resulting endpoints are

S	scene surface
A	area measure
L	radiance
L_e	emitted radiance
L_{in}	incident radiance
f_r	bidirectional scattering distribution function
V	mutual visibility
θ	angle between incident direction and normal
G	geometry term
ω_{yx}	direction from y to x

Figure 1: *Selected Symbols.*

illuminated by (1) and attenuated by the transfer function along the path. Splitting the path at the first hitpoint reduces the variance of the estimate at an affordable cost (at most 3 evaluations of (1)) and reduces material flicker. During times of no interaction anti-aliasing is performed by accumulating the images over time.

In comparison to bidirectional path tracing [34] the first sum in (1) uses only one technique to generate path space samples. For the majority of all path space samples that are taken into account by our algorithm this technique is best or at least sufficient to generate them. An exception are path space samples with a small distance $|y_j - x|$ or which belong to caustics. In order to avoid overmodulation the first group is handled in a biased way by just clipping the distance to a minimal value. Since samples of the second group cannot be generated by this technique their contribution is approximated by the second sum using caustic photon mapping.

In order to obtain interactive frame rates with the above algorithm on a cluster of PCs, the preprocessing must not block the clients and must avoid repeated computation of identical results. For good interactive performance further variance reduction is needed to reduce noise artifacts and increase efficiency. These improvements are discussed below.

4.1. Fast Caustics

Shooting a sufficient number of photons is affordable in an interactive application, since the random walk simulations require only a small fraction of the total number of rays to be shot. However, the photon map algorithms [9, 33] for storing and querying photons are far too slow for interactive purposes: Rebuilding the kd-tree for the photon map for every frame does not amortize, and the nearest neighbor queries are as costly as shooting several rays.

Therefore photon mapping is applied only to visualize caustics, where usually the photon density is rather high and density estimation can be applied with a fixed filter radius r. Assuming the photons to be stored in a 3-dimensional regular grid of resolution $2r$, only 8 voxels have to be looked up for a query. Since in practice only a few voxels will actually be occupied by caustic photons, a simple hashing scheme is used in order to avoid storing the complete grid. Then storing and hashing the photons is almost negligible as compared to left-balancing and traversing a kd-tree.

4.2. Interleaved Sampling

Generating a different set of point lights for each pixel is too costly, while using the same set causes aliasing artifacts (see Figure 2a). The same argument holds for the caustic photon map. In addition computing a sufficiently large photon map in parallel and merging the results would block the clients before actually rendering the frame and decreases available network bandwidth.

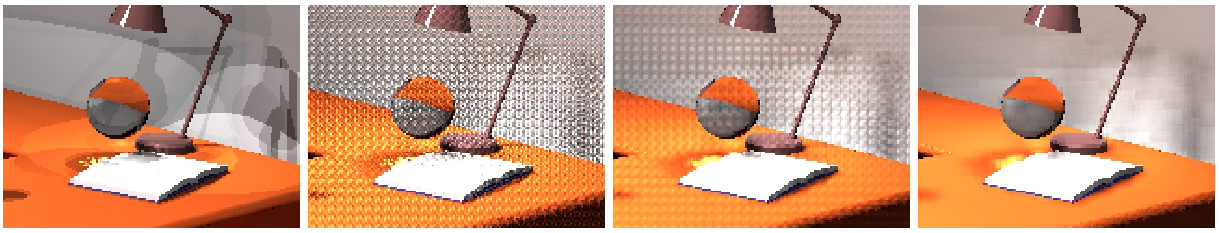
Generalizing interleaved sampling [17, 13] allows for controlling the ratio of preprocessing cost and aliasing: Each pixel of a small $n \times m$ tile is assigned a different set P_k of point lights and C_k of caustic photons ($1 \leq k \leq n \cdot m$). Padding this tile over the whole image replaces aliasing artifacts by structured noise (see Figure 2b) while only a small number $n \cdot m$ of sets of point lights and caustic photons has to be generated. Since this interleaved sampling achieves a much better visual quality, the sets P_k and C_k can be chosen smaller than P and C.

Each client computes the sets P_k of point lights and C_k of caustic photons by itself and on demand. Parallel tasks are assigned such that a client predominantly processes pixels with equal k in order to allow for caching of the P_k and C_k. Due to interleaved sampling synchronizing for global sets P and C is obsolete and in fact no network communication between the clients is required. Different clients only need the set number k, which serves as a seed value for generating the whole set of point lights and caustic photons.

4.3. The Discontinuity Buffer

The constraints of interactivity allow for only a small budget of rays to be shot, resulting in sets P_k and C_k of moderate size. Consequently the variance is rather high and has to be reduced in order to remove the noise artifacts. Taking into account that the irradiance is a piecewise smooth function the variance can be reduced efficiently by the discontinuity buffer [12].

For each pixel the server buffers the reflectance function accumulated up to the endpoint of an eye path, the distance to that point, the normal in that point, and the incident irradiance. The irradiance value consists of both the contribution by the point lights P_k and the caustic photons C_k. Instead of just multiplying irradiance and reflectance function, the irradiance of the 8 neighboring pixels is considered, too: Local smoothness is detected by thresholding the difference of distances and the scalar product of the normals of the center pixel and each neighbor. If geometric continuity is detected, the irradiance of the neighboring pixel is added to the center pixel's irradiance. The final pixel color is determined by



(a) no interleaved sampling no discontinuity buffer (b) 5×5 interleaved sampling no discontinuity buffer (c) 5×5 interleaved sampling 3×3 discontinuity buffer (d) 5×5 interleaved sampling 5×5 discontinuity buffer

Figure 2: *Interleaved sampling and the discontinuity buffer: All close-ups have been rendered with the same number of rays apart from preprocessing. In a) only one set of point light sources and caustic photons is generated, while for b)-d) 25 independent such sets have been interleaved. Choosing the filter size appropriate to the interleaving factor completely removes the structured noise artifacts.*

multiplying the accumulated irradiance with the reflectance function divided by the number of total irradiances included.

In the locally smooth case this procedure implicitly increases the irradiance sampling rate by a factor of 9, while at the same time reducing its variance by the same factor. Note that no additional rays have to be shot in order to obtain this huge reduction of noise, and that a generalization to larger than 3×3 filter kernels is straightforward. In the discontinuous case no smoothing is possible, however the remaining noise is superimposed on the discontinuities and such less perceivable [27]. Since only the irradiance is blurred, texture details on the surface are perfectly reconstructed. Note that blurring is done in image space and limited by the filter size.

Including the direct illumination calculations into the discontinuity buffer averaging process allows one to drastically reduce the number of shadow rays to be shot, but slightly blurs the direct shadows. Similar to the irradiance caching method [42], the detection of geometric discontinuities can fail. Then the same blurring artifacts become visible for example at shadow boundaries or slightly offset parallel planar objects [42].

Interleaved sampling and the discontinuity buffer perfectly complement each other (see Figure 2d) but require the filtering to be done on the display server. As a consequence an increased amount of data has to be sent across the network to the server. Quantization of normals and distances (16 bits each) and color values (RGBE format with 32 bits) and compression using the LZO library [22] helps to reduce bandwidth.

Compared to irradiance caching the discontinuity buffer samples the space much more evenly and avoids the typical flickering artifacts encountered in dynamic scenes. In addition no communication is required to broadcast irradiance samples before rendering.

4.4. Minimal Randomization

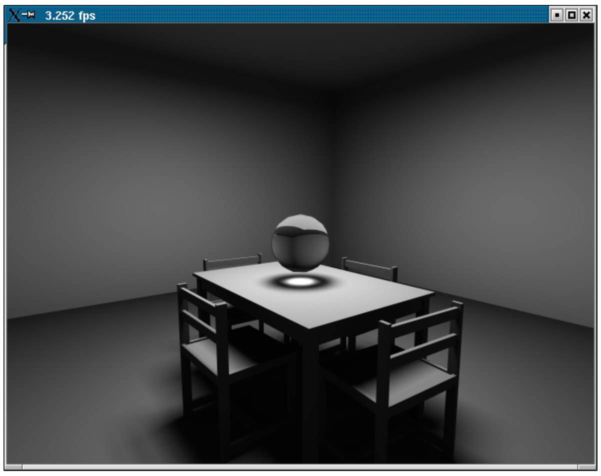
The integrands in computer graphics are square-integrable, usually of high dimension and contain unknown discontinuities. Consequently the Monte Carlo method is appropriate for numerical integration. Since the pure Monte Carlo method is rather slow, we use the much more efficient randomized quasi-Monte Carlo integration [23]. This method of integration saves around 30% of computation time [15] as compared to stratified sampling and exposes much less variance, i.e. noise. The idea consists of using the estimator

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{r}\sum_{i=1}^{r}\frac{1}{n}\sum_{j=0}^{n-1} f(x_{i,j}),$$

where the samples $x_{i,j}$ for fixed i are of low discrepancy (for definitions see [20]) and for fixed j are independent sets of random realizations. Low discrepancy guarantees a much better uniform distribution of the samples than independent random samples can obtain. This implies good stratification properties that guarantee for faster convergence. On the other hand the independence makes the estimator a real Monte Carlo estimate that is valid for all square-integrable functions and in addition allows for estimating the variance of the estimate.

The deterministic low discrepancy sequence of Sobol' can be generated in a few lines of code [26] in integer arithmetic. The points are randomized by just `xor`-ing them with a random bit vector before floating point conversion [7]. Taking the first n points a_j of the Sobol' sequence, r independent random realizations for the above scheme are obtained by $x_{i,j} := ((2^{32} \cdot a_j)\ \texttt{xor}\ b_i) \cdot 2^{-32}$, where b_i are r independent random bit vectors. Note that this randomization scheme preserves the good uniformity properties of the points.

In fact choosing only $r = 1$ randomized instances is sufficient to obtain a valid Monte Carlo estimator, while $r = 2$ already enables estimating the error (see [29]). Variance and noise inherent with Monte Carlo methods can be almost avoided by this simple and minimal scheme of randomiza-



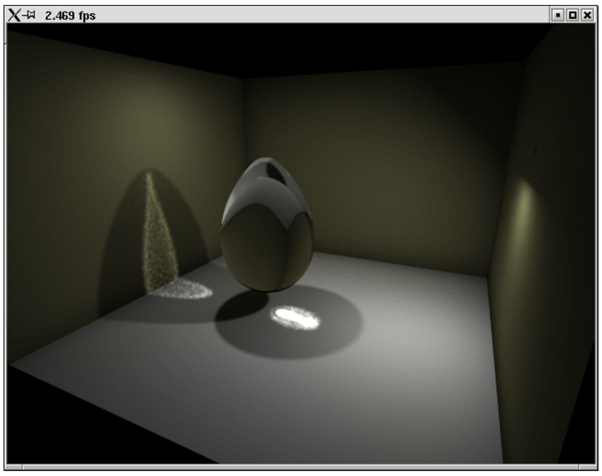


Figure 3: *Two simple test scenes with a glass ball and a glass egg consisting of 800 and 4,000 triangles. These scenes render at 3.3 and 2.5 frames per second on 8 clients.*

tion. Tabulating the Sobol' sequence provides stratified sample generation at rates much faster than can be achieved with a pseudo-random number generator in combination with stratified sampling.

In the algorithm each identification k is assigned a subsequent subsequence of one instance of a randomized low discrepancy sequence. These samples are used for generating the sets P_k of point lights and C_k of caustic photons. In case of geometric continuity the discontinuity buffer assembles the samples of neighboring pixels such joining different subsequences. Since these subsequences are part of the large sequence, the samples almost perfectly complement each other resulting in a superior convergence. In order to avoid the costly computation of high-dimensional low discrepancy sequences, padded replications sampling is used [15].

For the randomization each client needs an identical stream of only a few pseudo-random numbers per frame, which are created by the same pseudo-random number generator on each client. Thus any parallelization problems inherent with pseudo-random number generation are avoided and no communication is required during rendering. In order to avoid flickering due to changes of the sets P_k of point lights and C_k of caustic photons the same random numbers and low discrepancy points are used for each frame during interaction.

5. Results and Discussion

For our experiments we have used a cluster of dual processor machines each equipped with two AMD AthlonMP 1800+ CPUs and 512 MB of RAM. All machines are connected to a fully switched 100 Mbit Ethernet. In order to handle the amount of pixel data sent to the server, a single Gigabit uplink from the switch was connected to the master machine that otherwise was identical to the clients.

As the underlying ray tracing engine handles changing geometry transparently to the global illumination application, we can interactively manipulate all of the following examples, which are rendered at video resolution of 640×480 pixels with 3×3 interleaved sampling in combination with a 3×3 discontinuity buffer. During interaction all illumination is recomputed every frame. If interaction stops, a converged high quality image is obtained in $1-2$ seconds by accumulating successive frames.

The left image in Figure 3 shows a simple room lit by a single area light source located underneath the ceiling above the table, where a glass sphere casts a caustic. Global illumination is computed at 3.3 fps on only 8 clients, while for each pixel 22 shadow rays (4 samples for direct and 18 samples for indirect illumination) are cast and 500 caustic photons are generated for each frame. The scene in the right image of Figure 3 contains two light sources with 5 direct light samples in addition to 20 indirect light samples. Roughly 1,500 photons per light source were used to generate the two caustics.

The "Invisible Date" scene shown in Figure 4 contains 9,000 triangles. It is lit mostly indirectly from the two lamps pointing towards the ceiling. Without indirect illumination (see the left image in Figure 4) this scene would be almost entirely dark as no direct illumination reaches the furniture and the reflective floor. The right image in Figure 4 shows a flying glass sphere below the ceiling casting caustics on the wall. This scene nicely demonstrates the combination of specular illumination effects due to ray traced reflections and smooth indirect illumination computed with the new algorithm. It uses 4 direct and 9 indirect light samples and 500

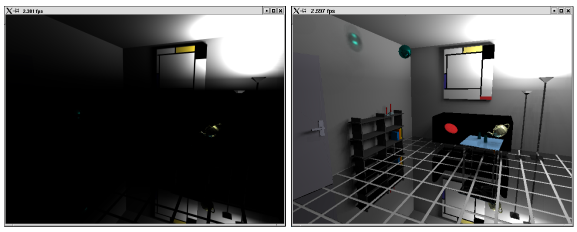


Figure 4: *The "Invisible Date" scene with only direct illumination on the left and global illumination on the right. Indirect lighting is dominant in this scene and is well handled by our algorithm. In this scene the depicted quality can even be achieved in interactive mode and is hardly distinguishable from the converged images.*

caustic photons per light source. The scene renders at 2.6 fps on 8 clients. Indirect shadows are smooth and detailed even during interaction. The dynamic and converged images can only be distinguished by the slightly better caustics and anti-aliasing due to accumulation.

The office scene in Figure 5 contains 34,000 triangles. Global illumination is computed with 4 direct and 18 indirect light samples and 1,000 caustic photons per light source, which results in a frame rate of 2.2 fps on 8 clients for both views. Although geometrically rather simple, the lighting in this scene is considerably complex. The desk light is extremely bright and highly occluded, translating to large variance of the estimate. Furthermore, the complex illumination patterns visible in the bottom row of Figure 5 are mostly due to indirect lighting from the strongly illuminated book that acts as a secondary area light source. Moving the book causes flicker due to the indirect light samples located on and reflected off the book.

The conference room in Figure 6 contains illumination from 104 area light sources and consists of 290,000 triangles. Due to its geometrical complexity the scene renders only at 1.7 fps using 12 clients. Using only 5 direct and 20 indirect light samples results in severe undersampling, however the well distributed location of the light samples and the filtering by the discontinuity buffer allow one to compute a solution with rather good quality even during interaction. As soon as interaction stops, the solution refines to high quality within 2 seconds.

5.1. Simulation Quality

The algorithm can be controlled by a small set of intuitive parameters: The number of direct and indirect light samples, the number of caustic photons, and the filter size of the discontinuity buffer that automatically determines an interleaved sampling pattern of the same size. The user can interactively trade off rendering speed for image quality by adjusting these parameters. The interactive system then provides immediate feedback for any changes to these parameters and to the scene itself.

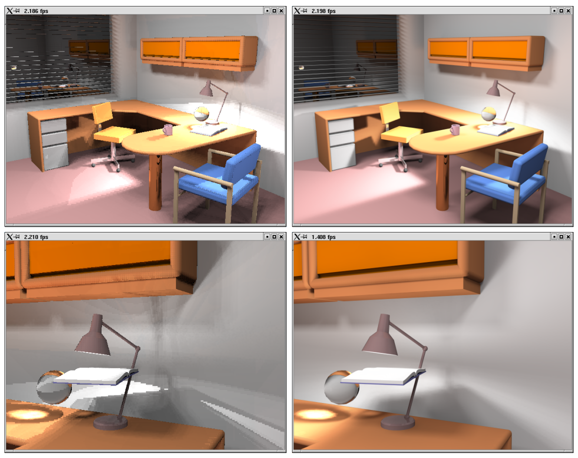


Figure 5: *The office scene is illuminated by two area lights at the ceiling and a desk lamp. The images in the bottom row show a close-up of the detailed illumination patterns caused by the book under the desk lamp. Both views render at 2.2 fps on 8 clients. Flicker is caused by moving the book, however the accumulation process quickly yields the converged images on the right.*

As expected, the low sampling rates used during interaction produce rendering artifacts that mostly appear in the form of shadow banding. Although the interactive rendering quality does not reach production quality, it still gives a very good impression of the global illumination in a scene and high quality still images are achieved after 1 – 2 seconds.

The discontinuity buffer can cause two kinds of artifacts. If no continuity is detected undersampling artifacts can occur during interaction (see e.g. the bookshelf in Figure 5 or the contours of the chairs in Figure 6). These artifacts are less perceivable during interaction and rapidly averaged out during accumulation. On the other hand the simple heuristic can fail resulting in falsely detected continuity. These artifacts now appear smoothly blurred and thus are hardly perceivable. Due to the failure of the heuristic, however, they are not averaged out during accumulation. A trivial way to get rid of this source of bias is to switch off the discontinuity buffer after a certain time interval of no interaction.

5.2. Scalability

The scalability of a parallel system depends on the ratio of overhead including idle times and work done by the clients, on the ability of the server to schedule tasks and to process their results, and on hardware constraints such as the maximum network bandwidth.

Our concept of interleaved sampling allows one to ef-

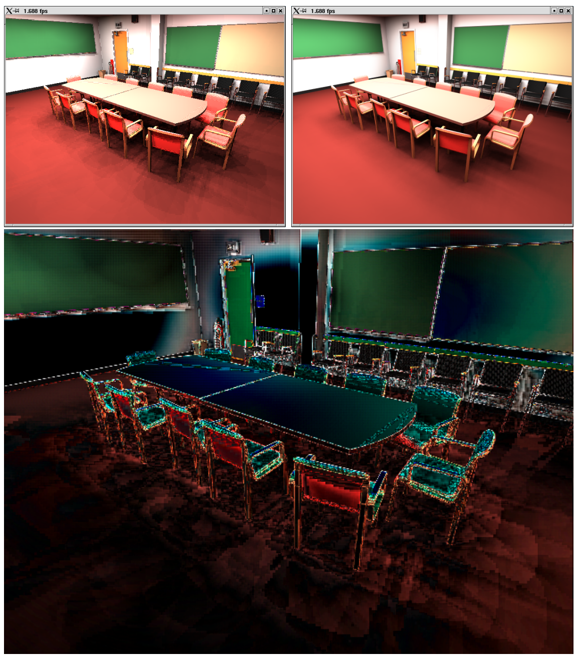


Figure 6: *Conference room scene with 104 area light sources and detailed shadows cast by the chairs. The bottom image shows the difference (scaled by a factor of 3) between the interactive rendering on the left and the converged image on the right. The scene renders at 1.7 fps on 12 clients.*

ficiently distribute the computation over a number of machines with a minimum overhead. Scheduling tiles with the same identification k primarily to the same client avoids redundant computations and increases cache efficiency. However due to dynamic load balancing, this ideal distribution cannot always be maintained. Our experiments show that on the average the clients have to preprocess data for roughly two different identifications k. The number of rays traced for a preprocess is mainly determined by the number of caustic photons and is small compared to the total number of shadow rays. There is no need to communicate preprocessing results to either the master or other clients avoiding idle times caused by synchronization. Summing up, the ratio of overhead and work is small and hardly influences scalability.

On the server side, the main workload consists of handling large amounts of pixel data and performing the filtering by the discontinuity buffer. Since these computations require information from adjacent pixels computed by different clients, they have to be performed on the server. Consequently frame rate and resolution are restricted by the performance of the server. Similarly, the limited network bandwidth restricts the maximum achievable frame rate and resolution.

As expected, the algorithm scales almost perfectly as

Number of clients	1	2	4	8	16
Room with table	0.4	0.8	1.6	3.2	5.3
Room with egg	0.3	0.7	1.4	2.7	5.4
Office	0.2	0.3	0.6	1.2	2.4

Table 1: *The algorithm almost perfectly scales over the range of available clients. For frame rates above 5 fps the server workload currently limits performance.*

shown in Table 1. The maximum performance of our server is currently limited to processing around 1.5 million pixels per second. This corresponds to roughly 5 frames per second at a resolution of 640×480 pixels. Reducing the image resolution allows one to easily scale the maximum frame rate, reaching more than 10 frames per second at 400×300 pixels. Other than waiting for faster hardware this bottleneck could be avoided by distributing the server computations, which at the same time splits up the required network bandwidth to each server.

However there are no restrictions imposed on the scalability with respect to image quality: Increasing the number of point lights obviously improves image quality. Increasing the number of clients by the same factor then results in exactly the same frame rate and server load.

6. Conclusion

By designing a global illumination system with respect to the constraints imposed by a fast, scalable ray tracing engine, we realized interactive global illumination in complex dynamic environments with multiple area light sources on a low cost cluster of consumer PC hardware. All lighting is recomputed every frame and the image quality scales with the number of available clients. In our current implementation frame rate and image size are limited by server performance, however this bottleneck can be removed by also distributing the server tasks.

The high performance of our system stems from excellent cache performance, non-blocking parallelization by interleaved sampling, and efficient variance reduction by randomized quasi-Monte Carlo integration and the discontinuity buffer.

The very good cache performance is achieved, because the majority of rays traced by our system are highly coherent shadow rays towards point lights. This coherence also allows one to further increase efficiency by exploiting a streaming SIMD architecture as introduced in [38] and perfectly fits the concept of ray classification [2].

Future work will concentrate on the efficient evaluation of the point lights along the lines of [41, 14] in order to avoid a severe performance loss in highly occluded environments. Besides including arbitrary physical surface properties, temporal coherence will be considered.

Acknowledgments

Thomas Kollig has been supported by the Stiftung Rheinland-Pfalz für Innovation. This work was supported by Advanced Micro Devices (AMD).

References

1. A. Appel. Some Techniques for Shading Machine Renderings of Solids. *SJCC Proceedings, Thompson Books*, pages 37–45, 1968. 2

2. J. Arvo and D. Kirk. Fast Ray Tracing by Ray Classification. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):55–64, 1987. 6

3. S. Chen, H. Rushmeier, G. Miller, and D. Turner. A progressive Multi-Pass Method for Global Illumination. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):165 – 174, 1991. 3

4. M. Cohen and J. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Cambridge, 1993. 3, 4

5. R. Cook, T. Porter, and L. Carpenter. Distributed Ray Tracing. *Computer Graphics (Proceedings of SIGGRAPH 84)*, 18(3):137–145, 1984. 2, 3

6. G. Drettakis and F. Sillion. Interactive Update of Global Illumination Using a Line-Space Hierarchy. In *Proceedings of SIGGRAPH 97*, Annual Conference Series, pages 57–64, 1997. 3

7. I. Friedel and A. Keller. Fast generation of randomized low-discrepancy point sets. In H. Niederreiter, K. Fang, and F. Hickernell, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 257–273. Springer, 2001. 4.4

8. X. Granier and G. Drettakis. Incremental Updates for Rapid Glossy Global Illumination. *Computer Graphics Forum*, 20(3):268–277, 2001. 3

9. H. Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001. 3, 4, 4.1

10. J. Kajiya. The Rendering Equation. *Computer Graphics (Proceedings of SIGGRAPH 86)*, 20(4):143–150, 1986. 3

11. A. Keller. Instant Radiosity. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 49–56, 1997. 3, 4

12. A. Keller. *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis*. Ph.D. thesis, Shaker Verlag Aachen, 1998. 4.3

13. A. Keller and W. Heidrich. Interleaved Sampling. In K. Myszkowski and S. Gortler, editors, *Rendering Techniques 2001 (Proc. 12th Eurographics Workshop on Rendering)*, pages 269–276. Springer, 2001. 4.2

14. A. Keller and I. Wald. Efficient Importance Sampling Techniques for the Photon Map. In *Proceedings of VISION, MODELING and VISUALIZATION)*, pages 271–279. IOS Press, 2000. 6

15. T. Kollig and A. Keller. Efficient bidirectional path tracing by randomized quasi-Monte Carlo integration. In H. Niederreiter, K. Fang, and F. Hickernell, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 290–305. Springer, 2001. 4.4, 4.4

16. E. Lindholm, M. Kilgard, and H. Moreton. A User-Programmable Vertex Engine. In *Proceedings of SIGGRAPH 2001*, Annual Conference Series, pages 149–158, 2001. 1

17. S. Molnar. Efficient Supersampling Antialiasing for High-Performance Architectures. Technical Report TR91-023, The University of Noth Carolina at Chapel Hill, April 1991. 4.2

18. M. Muuss. Towards Real-Time Ray-Tracing of Combinatorial Solid Geometric Models. In *Proceedings of BRL-CAD Symposium '95, MD, 5-9 June*, 1995. 2

19. M. Muuss and M. Lorenzo. High-Resolution Interactive Multispectral Missile Sensor Simulation for ATR and DIS. In *Proceedings of BRL-CAD Symposium '95, MD, 5-9 June*, 1995. 2

20. H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Pennsylvania, 1992. 4.4

21. *n*VIDIA. *nVIDIA OpenGL Extension Specifications*. March 2001. 1

22. M. Oberhumer. LZO-Compression Library. Available at http://www.oberhumer.com/. 4.3

23. A. Owen. Monte Carlo Extension of Quasi-Monte Carlo. In *Winter Simulation Conference*, pages 571–577. IEEE Press, 1998. 4.4

24. S. Parker, W. Martin, P. Sloan, P. Shirley, B. Smits, and C. Hansen. Interactive Ray Tracing. In *Symposium on Interactive 3D Graphics*, pages 119–126. ACM SIGGRAPH, 1999. 2

25. S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P. Sloan. Interactive Ray Tracing for Isosurface Rendering. In *IEEE Visualization '98*, pages 233–238, 1998. 2

26. H. Press, S. Teukolsky, T. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992. 4.4

27. M. Ramasubramanian, S. Pattanaik, and D. Greenberg. A Perceptually Based Physical Error Metric for Realistic Image Synthesis. In *Proceedings of SIGGRAPH 99*, Annual Conference Series, pages 73–82, 1999. 3, 4.3

28. M. Simmons and C. Séquin. Tapestry: A Dynamic Mesh-based Display Representation for Interactive Rendering. In B. Péroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proc. 11th Eurographics Workshop on Rendering)*, pages 329–340. Springer, 2000. 3

29. I. Sobol'. *A Primer for the Monte Carlo Method*. CRC Press, 1994. 4.4

30. M. Stamminger, J. Haber, H. Schirmacher, and H.-P. Seidel. Walkthroughs with Corrective Texturing. In *Rendering Techniques 2000 (Proc. 11th Eurographics Workshop on Rendering)*, pages 377–390. Springer, 2000. 3

31. P. Tole, F. Pellacini, B. Walter, and D. Greenberg. Interactive Global Illumination in Dynamic Scenes. In *Proceedings of SIGGRAPH 2002*, Annual Conference Series, 2002, to appear. 3

32. T. Udeshi and C. Hansen. Towards Interactive Photorealistic Rendering of Indoor Scenes: A Hybrid Approach. In D. Lischinski and G. W. Larson, editors, *Rendering Techniques '99 (Proc. 10th Eurographics Workshop on Rendering)*, pages 63–76. Springer, 1999. 3

33. M. Vanco, G. Brunnett, and T. Schreiber. A hashing strategy for efficient k-nearest neighbors computation. In *Proceedings of Computer Graphics International*, pages 120–128, 1999. 4.1

34. E. Veach and L. Guibas. Bidirectional Estimators for Light Transport. In *Proc. 5th Eurographics Worshop on Rendering*, pages 147 – 162, Darmstadt, Germany, June 1994. 3, 4

35. E. Veach and L. Guibas. Metropolis Light Transport. In *Proceedings of SIGGRAPH 97*, Annual Conference Series, pages 65–76, 1997. 3

36. I. Wald, C. Benthin, and P. Slusallek. OpenRT – a Flexible and Scalable Rendering Engine for Interactive 3D Graphics. Technical report, Saarland University, 2002. 1, 2

37. I. Wald, C. Benthin, and P. Slusallek. A simple and practical Method for Interactive Ray Tracing of Dynamic Scenes. Technical report, Saarland University, 2002. 1

38. I. Wald, C. Benthin, M. Wagner, and P. Slusallek. Interactive Rendering with Coherent Ray Tracing. *Computer Graphics Forum*, 20(3), 2001. 1, 2, 2.1, 6

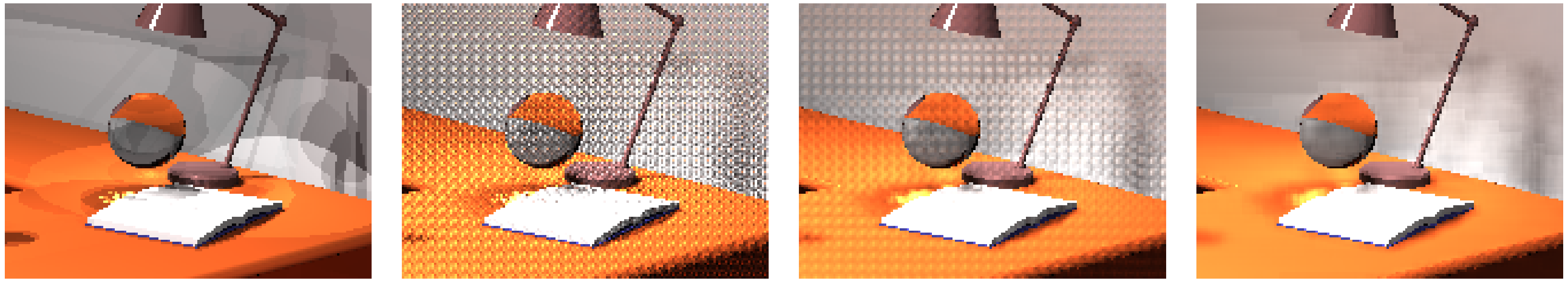
39. I. Wald, P. Slusallek, and C. Benthin. Interactive Distributed Ray Tracing of Highly Complex Models. In K. Myszkowski and S. Gortler, editors, *Rendering Techniques 2001 (Proc. 12th Eurographics Workshop on Rendering)*, 2001. 1, 2

40. B. Walter, G. Drettakis, and S. Parker. Interactive Rendering using the Render Cache. In D. Lischinski and G. Larson, editors, *Rendering Techniques '99 (Proc. 10th Eurographics Workshop on Rendering)*, 1999. 3

41. G. Ward. Adaptive Shadow Testing for Ray Tracing. In *Photorealistic Rendering in Computer Graphics (Proceedings of the 2nd Eurographics Workshop on Rendering*, pages 11–20. Springer, 1994. 6

42. G. Ward and P. Heckbert. Irradiance Gradients. In *3rd Eurographics Workshop on Rendering*, Bristol, United Kingdom, May 1992. 3, 4.3

43. T. Whitted. An improved illumination model for shaded display. *CACM*, 23(6):343–349, June 1980. 2



(a) no interleaved sampling
no discontinuity buffer

(b) 5×5 interleaved sampling
no discontinuity buffer

(c) 5×5 interleaved sampling
3×3 discontinuity buffer

(d) 5×5 interleaved sampling
5×5 discontinuity buffer

Figure 2: *Interleaved sampling and the discontinuity buffer: Using corresponding interleaving and filter sizes reduces sampling artifacts and noise.*

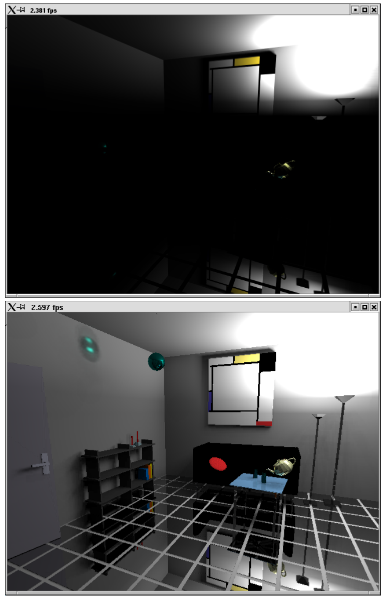


Figure 4: *The comparison of the "Invisible Date" scene rendered without (upper image) and with indirect illumination (lower image) demonstrates how important indirect illumination can be.*

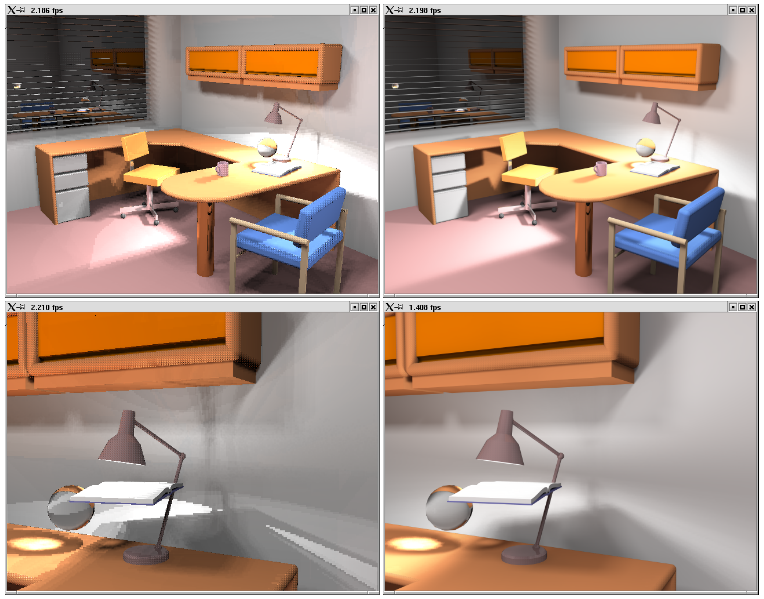


Figure 5: *The difficult lighting situation of the office scene: Moving the book under the desk lamp causes flicker. However, the images rapidly converge to the smooth images on the right.*

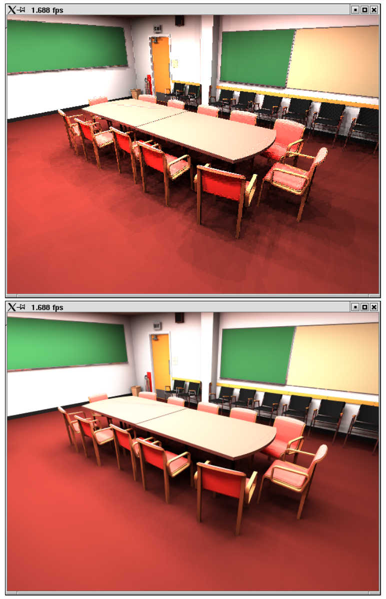


Figure 6: *Conference room scene with 104 area light sources and detailed shadows cast by the chairs. The scene renders at 1.7 fps on 12 clients.*