Eurographics Symposium on Rendering (2004)
H. W. Jensen, A. Keller (Editors)

Realtime Caustics Using Distributed Photon Mapping

Johannes Günther, Ingo Wald†, and Philipp Slusallek
Computer Graphics Group, Saarland University, Saarbrücken, Germany
{guj,wald,slusallek}@graphics.cs.uni-sb.de

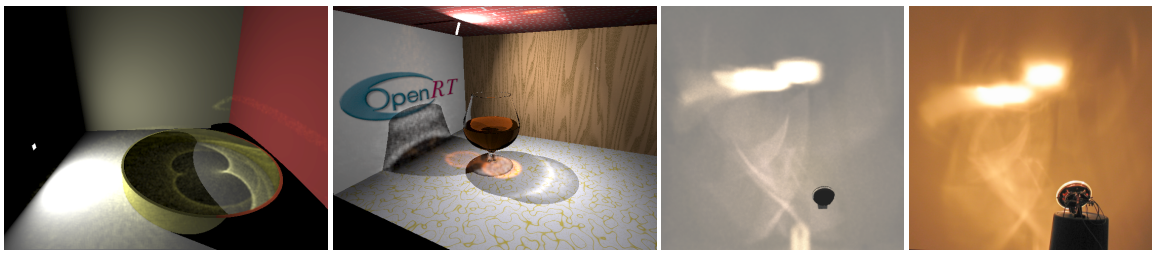

Figure 1: *Several examples of our distributed realtime photon mapping framework: a) Specular metal ring, including multiple reflections. b) Cognac glass with detailed caustics. c) Interactive visualization of the light emission characteristics of a car headlight. d) Reference photo of the same headlight. Using our framework, these scenes can be rendered interactively, running at 21, 12, and 11 frames per second, on 8, 13, and 18 dual-AMD AthlonMP 1800+ PCs, respectively.*

Abstract

With the advancements in realtime ray tracing and new global illumination algorithms we are now able to render the most important illumination effects at interactive rates. One of the major remaining issues is the fast and efficient simulation of caustic illumination, such as e.g. the illumination from a car headlight. The photon mapping algorithm is a simple and robust approach that generates high-quality results and is the preferred algorithm for computing caustic illumination. However, photon mapping has a number of properties that make it rather slow on today's processors. Photon mapping has also been notoriously difficult to parallelize efficiently.

In this paper, we present a detailed analysis of the performance issues of photon mapping together with significant performance improvements for all aspects of the photon mapping technique. The solution forms a complete framework for realtime photon mapping that efficiently combines realtime ray tracing, optimized and improved photon mapping algorithms, and efficient parallelization across commodity PCs. The presented system achieves realtime photon mapping performance of up to 22 frames per second on non-trivial scenes, while still allowing for interactively updating all aspects of the scene, including lighting, material properties, and geometry.

Keywords: Realtime rendering, lighting simulation, caustics, photon mapping, distributed computing, ray tracing

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

1. Introduction

Today, the repertoire of global illumination algorithms includes a large variety of techniques, such as (bidirectional) path tracing [Kaj86, LW93, VG94], different types of radiosity algorithms [HSA91, SAG94, Kel97], and photon mapping [Jen96, Jen01]. It even includes techniques for computing advanced issues of light transport, such as participating media, hair and fur, and subsurface scattering. Virtually all kinds of light transport can nowadays be simulated at any desired accuracy. However, these algorithms are still very compute intensive and are usually performed offline.

With the recent emergence of realtime ray tracing [PSL*99, WPS*03], it has become possible to simulate the most important aspects of global illumination at interactive frame rates [WKB*02, BWS03]. However, these systems have to impose some restrictions on the types of light transport in order to remain interactive.

One particularly problematic kind of light transport that is not supported well by these systems are caustics. For many applications, this in fact is only a minor limitation: For many engineering and architectural VR applications, the efficient and high-quality simulation of non-caustic illumination is much more important than caustic effects.

On the other hand, there are other applications that se-

† Now at MPII Saarbrücken, Germany, wald@mpi-sb.mpg.de

verely suffer from the lack of efficient support for caustics. For example, Benthin et al. [BWDS02] presented the use of realtime ray tracing for visualizing the complex reflection behavior of a car headlight. Without efficient support for caustics, however, this visualization was unable to also render the illumination patterns *caused* by the car headlight.

Today, the de-facto standard for generating high-quality caustics is photon mapping. However, the available photon mapping algorithms do not easily map to a distributed realtime ray tracing architecture, due to costly preprocessing, high computational complexity, and bad parallel scalability.

In this paper, we present a complete framework for realtime photon mapping in a distributed and interactive context. Our goal is to simulate caustic light effects for general scenes interactively (i.e. with several frames per second) at video resolution (640×480). While we are willing to tolerate a slightly reduced caustic quality during user interaction, we demand immediate feedback on interactions. In the absence of geometry or lighting changes we want to reach a view independent, high-quality result within a few seconds.

In the remainder of this paper, Section 2 first surveys existing methods that deal with the fast and efficient generation of caustics. Section 3 thoroughly analyzes the issues and constraints of photon mapping in a realtime context. This detailed analysis forms the basis for our improvements, which are discussed in Sections 4 and 5, followed by an evaluation of our techniques in Section 6. We conclude in Section 7 and provide a brief outlook on future work in Section 8.

2. Previous Work

We will concentrate our discussion on approaches that target the interactive generation of high-quality caustics. Caustics have always been a hard problem for global illumination: Pure radiosity-based algorithms are limited to light transport via diffuse surfaces only, and do not support caustics at all. There have been only very few and largely unsuccessful finite element approaches to compute non-diffuse effects.

Path-based approaches such as (bidirectional) path tracing [LW93, VG94] do support high-quality caustics, but usually require a prohibitively large number of samples to remove residual noise.

Higher performance – especially for caustics – can be achieved by density estimation methods: Instead of computing each pixel individually by an unbiased Monte Carlo method, these methods use a pre-computation step for approximating a biased but view-independent world space representation of the illumination that is then used for efficient lighting computations in the main rendering pass. Density estimation is usually performed by tracing light particles into the scene, recording their hit points, and then estimating the local energy density by suitable spatial filtering.

This approach was first used by Shirley et al. and Walter et al. [SWZ*95, WHSG97] who recorded the particle hits, and uses density estimation on surface patches to produce a pre-illuminated triangle mesh for display. This unnecessary dependence on scene geometry can be successfully avoided by the Photon Map [Jen96]: The photon map stores particles in a *kd*-tree while density estimation is performed on the fly during rendering by querying the k nearest neighboring photons (kNN-query). As the *kd*-tree is independent of geometry, photon mapping does not suffer from the usual meshing artifacts, and can efficiently be used for highly complex scenes. Even non-polygonal scenes, such as fractal surfaces and volumetric effects [Jen01] can be handled well through photon mapping.

2.1. Interactive Caustics

The photon map has quickly become the preferred tool for computing caustic illumination. However, as we will see below its large computational cost and other properties complicate its use in an interactive context.

Granier et al. [GDW01] computed caustics by integrating a specular particle tracing step into a hierarchical radiosity (HR) system. This allowed for also reproducing caustics in their system, while still resulting in a readily illuminated triangle mesh that could be displayed via graphics hardware. By exploiting information from the HR data structure they were able to recompute a new solution at near-interactive frame rates. However, their method was still too slow for practical applications except for very simple scenes.

For generating realtime caustics Wand et al. [WS03] took a different approach: Sample points on the caustic object acted as virtual pinhole cameras, projecting the incoming light onto diffuse surfaces in the scene. They used environment maps and programmable graphics hardware to implement this projection via texture lookups. Unfortunately only a few hundred caustic samples were affordable in order to stay interactive, which lead to rather low quality caustics. This approach also assumed distant and non-occluded light sources, and is limited to reflective caustics only.

In their "Instant Global Illumination" (IGI) system, Wald et al. [WKB*02] proposed a simplified form of photon mapping (called "hashed photon mapping"), in which density estimation is not performed based on the k nearest neighbors, but rather with a fixed query radius. This allowed for much faster density estimation by storing the photons in a hashed grid data structure, but unfortunately also compromised on the quality of the caustics, as a fixed query radius destroys the photon map's automatic dependence on illumination density. Furthermore, the photon mapping step caused several performance limitations and scalability bottlenecks in their system. This only allowed for a relatively simple and low-quality caustic simulation during interaction, while nonetheless incurring a heavy impact on the maximum system performance. Because of these issues, the photon mapping stage was removed later [BWS03].

Yet another alternative for using approximative algorithms on special hardware was proposed by Ma et al. [MM02], who accelerated the nearest neighbor query using a block hashing scheme. However, their method is only approximative, too, and has mainly been designed for a hardware implementation. Finally, their algorithm can only accelerate the query process, but does not help with generating the photons nor in building the acceleration structure.

Purcell et al. [PDC*03] proposed an approach using programmable graphics hardware. Even though the computational power of modern GPUs allowed for interactive performance of several frames per second, they were limited to an approximative nearest neighbor query, only. More importantly, their method is problematic for dynamic scenes. The necessary update of the photon map due to light source or object movement takes several seconds, within which the caustic disappears. Another issue is that their method so far has been used only with relatively simple scenes, due to memory and architectural limitations of current GPUs.

3. Issues of Interactive Photon Mapping – An Analysis

Before we start to work on any specific improvements, it is essential to obtain a good understanding of the problems and issues of using photon mapping in an interactive context. The original photon mapping algorithm consists of three stages: photon shooting, *kd*-tree construction, and *k* nearest neighbor queries during rendering. As we will see below *all* three operations are equally problematic in an interactive setting. We will analyze these issues first in a single-CPU context before discussing issues of distributed processing.

3.1. *k*NN-queries During Rendering

One obvious cost factor for photon mapping is the cost for performing the *k* nearest neighbor queries used for density estimation. As we are only interested in caustic photon mapping here, we do not consider the cost for additional computations such as a local pass or irradiance gradients [WH92].

While a *k*NN-query is commonly considered to be rather cheap, it is in fact quite expensive when compared to a fast ray tracer used for rendering. As can be seen in Table 1, *k*NN queries are about 10 times as expensive as shooting a ray.

Scene (# triangles)	*N*/*k*	queries/sec	rays/sec
EGG	100k/40	976k	1.7M
(3852)	1M/100	382k	
RING	150k/40	262k	1.7M
(972)	2M/100	99k	
GLASS	75k/40	347k	0.98M
(22454)	1M/100	163k	
HEADLIGHT	100k/50	165k	0.56M
(167808)	750k/80	48k	

Table 1: *Number of kNN queries per second (as described in [Jen96], using N mapped photons) vs. number of rays per second (for individual, non-SSE rays during rendering) in our example scenes. Performing a kNN query is roughly one order of magnitude more expensive than shooting a ray.*

3.2. Preprocessing: Photon Map Generation

While the high query cost poses a severe performance problem, these queries are per-pixel operations working on read-only data. Thus they parallelize quite well, and could in principle be solved by adding enough processing power.

However, before the actual rendering phase can start, the photons must first be generated and organized into a *kd*-tree. This is less of a problem when performed as a separate preprocessing step, e.g. for an interactive walkthrough application of an otherwise static scene. For applications that include dynamic and unpredictable changes to material properties, light sources, and even scene geometry however this "pre"processing has to be performed for every frame.

3.2.1. Photon Shooting

High-quality and thus highly detailed caustic effects often require generating several hundred thousand photons, which easily exceeds the time slot for rendering a single frame in an interactive setting (e.g. 100 ms). This problem consists of two parts: First, the high cost for computing photon trajectories, and second, the low yield of caustic photons per ray, i.e. the high number of rays that have to be shot per caustic photon.

Slow Ray Tracing During Photon Shooting: Quite often, the availability of realtime ray tracing is believed to make the cost for photon shooting negligible: At peak rates of several million rays per second [WPS*03], shooting a few hundred thousand photons per frame should be well tolerable. This however is a misconception.

First of all, the rays shot in the photon shooting stage are quite incoherent. Such rays are usually several times as expensive as coherent rays [Wal04]. Furthermore, acceleration using SSE [Int02] instructions – as done with primary and shadow rays – would not be effective for incoherent rays. As a result, the performance of photon tracing is significantly lower than peak ray tracing performance for rendering, as shown in Table 2.

Furthermore, photon shooting requires costly computations for sampling and computing BRDFs. These computations can easily dominate the cost for tracing the rays (see Table 3).

Bad Yield of Caustic Photons: Additional to this high cost for tracing incoherent rays, each photon on average requires

Scene	peak rays/sec (primary rays, SSE)	rays/sec (during photon shooting)	photons/sec (during photon shooting)
EGG	4.3M	870k	355k
RING	3.4M	890k	375k
GLASS	3.3M	535k	240k
HEADLIGHT	3.9M	255k	29k

Table 2: *Raw ray tracing performance during photon shooting (excluding sampling and surface interaction), as compared to peak ray tracing performance in the same scene with coherent primary rays and fast SSE code. As can be seen, incoherent rays during photon shooting are 4 to 15 times more costly. The need for shooting several rays per photon path, as well as other cost factors (e.g. sampling and surface interaction) further reduce photon generation performance by a factor of 2 to 9.*

Scene	sampling	ray casting	BRDF evaluation
EGG	34 %	39 %	27 %
RING	36 %	50 %	14 %
GLASS	22 %	42 %	36 %
HEADLIGHT	5 %	62 %	35 %

Table 3: *Relative cost for light source sampling, ray casting and BRDF evaluation during photon shooting. Sampling and BRDF evaluation – which cannot be reduced by faster ray tracing performance – consume a significant part of total photon generation time.*

several rays to be generated, due to multiple reflections and refractions of caustic photons. Even worse, only few photon trajectories actually contribute to caustics at all. Small caustic generators usually cover only a small solid angle with respect to a light source. As a result only a small fraction of all emitted photons will actually hit such objects and produce a caustic. Without further arrangements photon yields are often less than 10 percent (see Table 4).

To raise the caustic photon yield, Jensen proposed to shoot photons directly towards caustic generators [Jen01] by first projecting and rasterizing them onto the (discretized) hemisphere of directions around the light source, and shooting photons only into relevant directions. Obviously however this approach only works for *direct* caustics, and cannot capture indirect caustics. Even worse, it is quite problematic for an interactive setting, as the projection and rasterization of the caustic generators would potentially have to be done for each light in every frame. This is obviously too costly in an interactive context with realistically complex scenes and many specular primitives.

Instead of projecting individual specular triangles, this could also be achieved by identifying "groups" of specular primitives, and projecting only a bounding object, as e.g. done in [WKB*02]. This however is problematic for realistically shaped objects, and would require user intervention, which is undesirable for an automatic, interactive system.

3.2.2. *kd*-tree Construction

Even if we were able to generate photons fast enough we have to address the problem how to quickly organize these photons for fast retrieval. In order to efficiently perform the nearest neighbor search it is necessary to build a spatial index, usually a *kd*-tree. While the cost to create this index is negligible for small numbers of photons it becomes a serious problem as the number of photons increases for realistic scenes. In particular, the cost for building the *kd*-tree has complexity $\mathcal{O}(n \log n)$ [Jen01], i.e. it increases more than linearly with the number of photons.

Also problematic is the fact that the construction process cannot easily be parallelized or accelerated, e.g. by the use of SSE instructions of today's processors. Finally, building the *kd*-tree causes several related problems: For example, an obvious approach to reduce the photon shooting time is to split the photon generation step into several independent jobs – e.g. by updating only every *M*-th photon in each frame or by having *M* independent machines generate one *M*-th of all photons each – and combine these *M* parts later. Similarly, one could imagine only updating photons that are affected by a change to the scene, using an approach similar to Selective Photon Tracing [DBMS02]. These approaches can significantly reduce the number of photons to be (re)generated every frame.

However, *all* of these approaches must still regenerate the *entire kd*-tree affecting *all* photons in every frame (i.e. not only the newly generated ones), even if only a small part of the photons have actually been updated. This could obviously be fixed by not using *one kd*-tree, but rather querying from several *kd*-trees in parallel. This, however, is quite inefficient, as the same region of space would have to be traversed for each of the *M kd*-trees.

3.3. Distribution and Parallelization Issues

While all the previously discussed issues are problematic for interactive photon mapping on a single machine, they get even more problematic in a distributed setup. Even though ray tracing performance has been improved significantly, virtually all of todays interactive ray tracing system (see e.g. [WPS*03] for an overview) still rely on massive parallelization to achieve realtime performance on non-trivial scenes. This is particularly true for applications where the cost for each individual pixel is as high as for photon mapping (see Section 3.1).

One option in a distributed system is to have each rendering client compute the same photon map. This however performs the costly preprocessing step redundantly on every machine and unnecessarily limits scalability. Note that this

problem is not limited to a distributed setting such as a networked cluster of PCs, as the hierarchical *kd*-tree generation is difficult to parallelize even on a shared-memory system.

An alternative option is to have each client compute different parts of the photon map that are combined later. This would require to broadcast the individual parts to all clients, which would be too slow for interactive rates on current network technology. Furthermore, this approach creates timing dependencies between all clients, limiting the system performance to that of the "weakest link in the chain". Thus this approach is unlikely to scale well beyond a few clients. Similarly, one could shoot all photons on a single machine, build the photon map there, and distribute them to all clients afterwards. This however would make this machine the bottleneck and is unlikely to work well, either.

Finally, a third alternative, which was used by Wald et al. [WKB*02], is based on a combination of interleaved sampling [KH01] and discontinuity buffering [Kel98]: In that approach, different clients generate different photon maps of a smaller size each. The illumination information is later combined by interleaving pixels from different clients and performing an image-based filtering stage (i.e. discontinuity buffering) on the server. Even though this approach avoids many of the above problems, it poses a set of different issues.

Most importantly, it creates a severe server bottleneck (in both compute performance and network bandwidth), reducing the maximum system performance to at most 5 frames per second at 640×480 pixels in the original implementation [WKB*02]. Furthermore, dynamic load balancing between the clients leads to cases where clients must compute additional photon maps because they start working on pixels from other clients. These photon maps can then often be used for only a few pixels.

Despite these issues, we consider the third approach superior to the former two alternatives. We follow the same approach and concentrate on reducing the impact of the mentioned disadvantages.

Note that distributed photon mapping is also possible in a data-parallel approach as used in e.g. the Kilauea renderer (see [CDR02]). This approach however mostly targets the distributed storage of both photon map and scene data base in order to support complex scenes and highly detailed photon maps in offline rendering. It has neither been designed for, nor is it applicable to an interactive setting, and will therefore not be discussed here in more detail.

4. Distributed Realtime Photon Mapping

The previous section clarified that in order to achieve realtime photon mapping it is important not to work on individual problems but to consider the *entire* set of issues in combination. In the following we discuss our improvements to the server related issues while the next section covers our contribution to the photon map itself.

4.1. Faster Filtering to Reduce Server Load

A number of observations allow us to improve the filtering problem dramatically. Caustic illumination is affected much less by geometric discontinuities because it is much more localized (i.e. focussed) than diffuse illumination. Another observation is that potential artifacts will often be masked by radiance resulting from direct illumination.

Thus, the sanity tests for filtering caustics can be relaxed significantly. In fact, we do not perform any continuity tests at all and always filter caustic illumination in image space. Similarly, we no longer separate irradiance and diffuse material properties for caustic illumination and simply filter the direct RGB pixel contribution using a box filter, which can be implemented very efficiently using SSE instructions.

In fact, the filtering step can be implemented using incremental computations, such that it essentially runs in constant time per pixel instead of the naive $\mathcal{O}(n^2)$ for a filter of size $n \times n$. This is possible because we use a simple box filter without per pixel weighting. We maintain a buffer for the currently processed row of pixels holding the vertical sums of pixel values within the filter kernel size. The horizontal filter value is then incrementally updated along the row and the row buffer is updated when switching to the next row.

This approach also leads to a very efficient and cache-friendly implementation and reduces memory bandwidth. In summary, we achieve a filtering performance of 300 frames per second (640×480 pixels) on a single 2.2GHz Pentium-4 CPU, which is more than sufficient for our needs. Being independent of filter size also allows us to use much larger filter sizes than the original system (see below).

4.2. Reducing the Server Network Bottleneck

Filtering on the server was not only limited by filtering performance but also suffered from the additional data that had to be transferred to the server for proper filtering. This data included normal and distance information for the hit point, in order to limit the filtering to geometrical smooth regions. Additionally, the filtering was only performed on the *ir*radiances in order to avoid blurring across material boundaries or textures. Therefore, the irradiance and diffuse material properties had to be sent separately.

With our relaxed filtering stage, the information sent to the server is drastically reduced as normals and distances are no longer needed. This leaves only two RGB values per pixel, one for the radiance resulting from diffuse illumination and one for the radiance resulting from caustics that will be filtered. This reduction in network bandwidth significantly relaxes the server network bottleneck: Instead of a maximum of only 5 frames per second, the network bandwidth of the

server (measured at a rather low 350 Mbit/sec for GigaBit Ethernet due to older hardware) now allows for up to 22 frames per second (fps). With better network adapters this should scale well to much higher frame rates.

Unfortunately this relaxed filtering can be problematic in some cases. In the absence of strong direct illumination textures are slightly blurred, as can be seen under the glass in Figure 1b . Also, partially occluded caustics (e.g. by the rim of the ring in Figure 3a) are smeared one or two pixels over the object border, although this is hardly visible. We believe this potential image-space blurring to be a reasonable price for the significantly increased scalability.

4.3. Avoiding Redundant Photon Computations

The dynamic load balancing scheme used by the original system allows for efficiently combining essentially arbitrary numbers of heterogeneous machines with different performance levels. Note that dynamic load balancing works best if the average setup cost per job is minimal. However, this is not the case for our combination of photon mapping and interleaved sampling. For example, if a client wants to "help" another, slower client by taking over some of its pixels, it may easily end up spending more time in generating the newly required photon map than just waiting for that client to finish its pixels.

As a result we now use a static load balancing scheme. Of course, static load balancing only works well if all clients have similar performance and receive roughly the same work load. While the first constraint is easy to guarantee, the second one is more problematic. However, with the finely interleaved sampling of the image all clients effectively compute the same downscaled image, except for a small shift and different sets of photons. This leads to a well-balanced work load and thus works well even with static load balancing. On the downside this approach limits the number of possible clients to the number of interleaving sets (or multiples thereof).

5. Improving Photon Mapping on the Client

Having removed the main limitations of the former system allows the server to handle up to 22 fps at video resolution. Obviously however this performance can only be maintained once the clients can actually deliver this performance. The high frame rate evidently imposes a strict upper bound on the time available for preprocessing, i.e. for photon generation and *kd*-tree construction.

Due to the use of interleaved sampling together with filtering we can reduce the number of photons per client proportional to the filter size. As mentioned above, the faster filtering now also allows for larger filter sizes, and thus for less photons per client. In practice we usually use filter sizes in the range of 3×3 to 6×6. Since the box filter eventually combines the results obtained by the individual photon maps to one smooth image, the illumination obtained by $M \times M$ interleaved photon maps of N photons each is similar to using one larger map of M^2N photons [WKB*02]. Obviously, this reduction in the number of photons per map (by a factor of 9 to 36) directly translates to significant savings in photon shooting and *kd*-tree construction time on the clients. However, this saving is insufficient and must be reinforced by other ideas.

5.1. Faster Photon Generation

The most obvious candidate for optimization is the time required for generating caustic photons. One simple approach would be to exploit the second CPU of each client using multithreading to shoot the photons. However, this can only give a factor of two in the best case, which is far from sufficient.

Looking at the above analysis (Section 3.2) reveals that some of the cost factors for photon generation cannot be improved on. For example, rays *will* be incoherent during photon generation, and each light path *will* require several surface interactions (for reflection and refraction) in order to generate a caustic photon. However, the number of paths that actually yield caustic photons can be influenced, and should be maximized.

As discussed before, Jensen's approach of projecting caustic generators onto the light source [Jen01] is inefficient for complex-shaped caustic generators and does not scale to complex and interactive environments.

Sampling Caustics using Selective Photon Tracing

As an alternative, we use a method similar to Selective Photon Tracing (SPT) [DBMS02], except that we do not consider the temporal domain, but rather use it for adaptively sampling path space: In a first step, a set of "pilot photons" is traced into the scene in order to detect paths that generate caustics. For those pilot paths, periodicity properties of the Halton sequence [Nie92] are exploited to generate similar photons (see Figure 2).

By using Selective Photon Tracing we increase the yield of caustic photons by roughly a factor of four (see Table 4). Essentially, this means that the same number of caustic photons can be generated with only one fourth of all rays. As the improvement depends significantly on the (projected) size of the caustic generator, the results for smaller caustic generators than those used in our examples are likely to be even more significant.

However, using Halton numbers for adaptive sampling is also not free from sampling artifacts. The non-uniform distribution of refinement photons around the pilot photon leads to structured noise patterns when sampling with too few pilot photons.

Nevertheless, the new approach also handles indirect

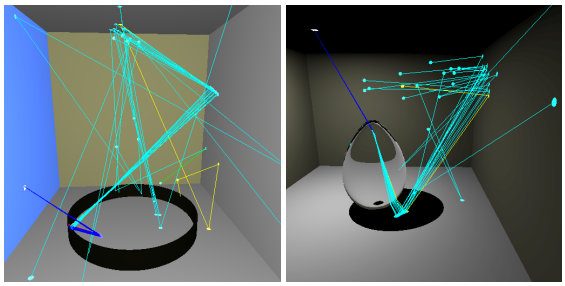

Figure 2: *Automatic sampling of caustic generators using Selective Photon Tracing [DBMS02]: First, some pilot photons (yellow) are shot into the scene. Those pilot photons contributing to a caustic are then refined (cyan) using the similarity property of the Halton sequence [Nie92].*

caustics (see Figure 3), as the photons of one group also stay together after diffuse bounces. Most importantly, however, this method does not require any preprocessing and maintains the photon map's property of being independent of scene geometry. This it is well suited for both complex scenes and interactive setups.

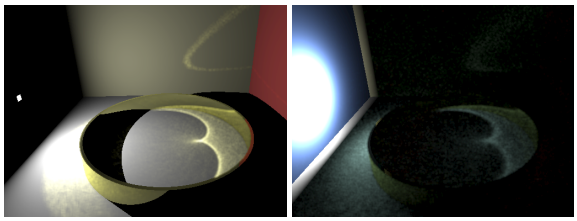

Figure 3: *Our approach of using Selective Photon Tracing for generating the photons also handles indirect caustics. Left:* RING *scene with a direct caustic cast by a small area light source. Right: Turning the light towards the wall generates an indirect caustic caused by the brightly illuminated spot on the wall. Directly shooting towards the caustic generators would have resulted in no caustic at all.*

Scene	default sampling	SPT	improvement
EGG	6.6 %	29.5 %	4.5
RING	10.4 %	40.5 %	3.9
GLASS	9.4 %	40.8 %	4.3

Table 4: *Percentage of photon paths that contribute to the caustic. Using Selective Photon Tracing (SPT) improves the yield by roughly a factor of four.*

5.2. Faster *kd*-tree Construction and Query

Being able to quickly generate photons we now have to concentrate on indexing and querying the list of photons in an efficient manner. In particular for non-trivial numbers of photons the time to build the *kd*-tree can easily dominate overall rendering time, even after a careful and optimized implementation of the construction algorithm.

Building the *kd*-tree requires to determine the median of all photons in a subtree in every splitting step, which turns out to be quite costly. In order to avoid this cost we instead split in the middle of the largest dimension of the current voxel. As can be seen in Table 5, the construction time for the "off-balanced" *kd*-tree improves by roughly a factor of 1.5 for small numbers of photons. This improvement increases with the number of photons and reaches a factor of 4 for the case of the headlight.

Even though these results are encouraging we also have to regard the query times. The new "unbalanced" build strategy is only viable if it is not offset by an increase in query time. However, using the new construction strategy we improve the query time by a factor of 1.5 to 2.5 compared to the common median approach. These experiments confirm similar results in [WGS04], which shows that balancing the *kd*-tree is not always optimal with respect to query times, and that higher performance can be reached with unbalanced kd-trees.

scene	photons		photons	
method	build	query	build	query
RING	24934		100844	
median	21.4	388	132.9	660
middle	16.3	151	89.5	330
grid lo	3.9	254	23.4	362
grid hi	14.5	86	34.3	212
GLASS	15548		322574	
median	11.4	399	900.4	1706
middle	11.0	241	459.9	1116
grid lo	4.3	173	146.6	2384
grid hi	12.8	146	223.7	1802
HEADLIGHT	8655		43332	
median	9.0	520	332.6	1123
middle	6.3	305	76.3	743
grid lo	10.9	313	31.8	3409
grid hi	69.9	151	93.8	837

Table 5: *Comparison of different acceleration structures. "median": kd-tree, split at median of largest dimension; "middle": kd-tree, split at spatial center of largest dimension; "grid lo": low resolution grid; "grid hi": high resolution grid; Times are in milliseconds on a single Pentium-4 2.2 GHz CPU. The left two columns show the data for a moderate number of photons while the right two columns present the results for a larger photon map. The query was done for a complete image of* 640×480 *with* $k = 64$*. For a discussion of the results see the Sections 5.2 and 5.3.*

5.3. The Uniform Grid

Although our results in improving the build time for *kd*-trees are encouraging we still need to traverse many *kd*-nodes for each query. Thus it is worth to investigate a uniform grid as an alternative data structure to accelerate the nearest neighbor searches.

In order to achieve a cache-friendly layout of the photons in memory we sort them into an array by a modified quicksort, first regarding the *x*-dimension. After binning the photons in this direction according to the grid resolution, this procedure is repeated for each bin recursively in the other dimensions. The grid itself is then implemented as a lookup table that stores in each cell the position of the first photon in the array segment belonging to that cell. The position of the last photon is determined by the start position of the next cell. With this implementation the grid itself takes only 4 bytes per cell allowing for a grid resolution of up to 256^3. If finer resolutions are required the grid could be hashed.

To perform a *k*NN-query all grid cells within the specified maximum radius are visited and each of all their photons are put into the priority queue to get the *k* nearest ones. As can be seen in Table 5, this uniform grid approach can lead to even higher performance than the kd-tree. Unfortunately, its performance depends on the actual parameter settings, and requires manual tuning to reach optimal performance. We therefore usually use the unbalanced kd-tree, which performs similarly well, but additionally adapts better to different scenes, photon distributions, and parameter settings.

5.4. Accumulation of Photons Across Frames

High-quality caustics require many photons and reduce the rendering speed accordingly. However, during interaction we prefer fast response to high-quality images. We can accommodate a trade-off between these contradicting requirements by shooting only a part of all photons during scene modifications and accumulate caustic photons across consecutive frames once interaction has stopped.

Shooting a new batch of *P* photons each frame is relatively easy. However, as the photons available after each accumulated frame increases, so does the time for kd-tree construction: The number of photons to be sorted into a kd-tree rises linearly with the number of accumulated frames, and the construction time rises even superlinearly.

Fortunately, moving to an unbalanced *kd*-tree also makes it possible to incrementally insert new photons into an existing *kd*-tree. We use this approach to build the *kd*-tree in the above situation by incrementally adding the newly generated photons in each frame. Obviously resulting accumulated *kd*-tree is not as optimal (with respect to query time) as a *kd*-tree rebuilt from scratch. The slightly reduced query performance however is more than offset by the significantly increased construction performance, in particular after accumulating photons from several frames – for which rebuilding a complete kd-tree simply would be impossible at a tenth of a second.

Accumulating photons across frames allows for maintaining interactivity even for highly complex lighting situations that require millions of photons for sufficient quality. Once accumulation stops, the resulting high-quality photon map can still be viewed interactively in walkthrough-mode, as camera movements do not require to rebuild the photon map. Usually the performance during high-quality walkthroughs is even *higher* than during accumulation (see Figure 5), as no photon shooting has to be performed any more during the walkthrough.

6. Results

After analyzing the issues of photon mapping in an interactive context and discussing the improvements for the individual steps of the algorithm, we now discuss the overall performance improvements of the new realtime photon mapping framework. In particular, we compare our new system to the IGI system by Wald et al. [WKB*02] regarding image quality and performance and demonstrate the improved scalability and features.

For experiments we use the same test environment as Wald et al. [WKB*02] consisting of several AthlonMP 1800+ PCs connected to a fully switched 100Mbit network with a GigaBit uplink to the master server.

6.1. Comparison in Image Quality

In order to directly and fairly compare both systems we turned off the sampling of indirect illumination in the original IGI system. In addition we *restrict* the new system to not accumulate photons across frames and to use only a 3×3 interleaving pattern as used by the original system. Both systems use 9 CPUs. We then increased the number of photons in the new system to match the framerate of the original IGI system.

Even though the EGG scene (Figure 4) can be considered a best case scenario for the original IGI system, the new system is capable of sampling the caustics with more than 3 times as many photons.

The RING scene clearly demonstrates the improvements of the Selective Photon Tracing approach. The approximation of the caustic generator by a bounding object from the original system fails to properly direct the photon shooting. The new system samples the caustic much more efficiently with roughly 6.4 times as many photons as the IGI system.

By being able to afford more photons at the same performance – together with the fact that our system performs accurate nearest neighbor queries as opposed to a fixed filter radius – we achieve much higher image quality than the original system, as shown in Figure 4.

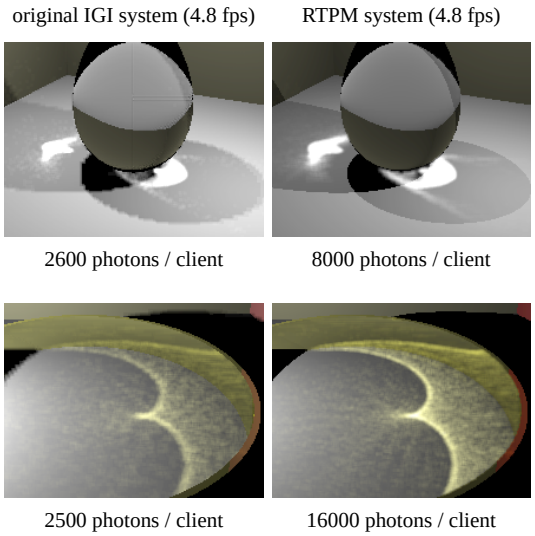

Figure 4: *Comparison of our new RTPM system to IGI system of Wald et al. [WKB*02], with a zoomed-in image of the Egg and the Ring scene. Although both systems run on the same hardware (9 AthlonMP 1800+ CPUs) and network with 4.8 fps, our new system ends up with more than 3 resp. 6 times as many photons at the same time resulting in substantially better image quality.*

6.2. Performance and Scalability

Apart from higher performance, our new framework also achieves much better scalability. As shown in Table 6 the new system scales linearly up to 22 fps at which point our current network bandwidth becomes saturated. Note that we provide two frame rates; one during photon generation (corresponding to interactive sessions that modify the scene's content) and one after enough photons have been accumulated. In the latter case no more photons are generated and the system is in a kind of of walkthrough mode that only queries the photon map during rendering.

Scene	RING		GLASS	
# photons	81k	483k	98k	393k
# CPUs	interaction	walkthrough	interaction	walkthrough
1	0.45	0.81	0.16	0.34
4	2.1	4.7	0.66	1.9
9	5.0	11.4	1.5	4.6
16	8.7	21.0	2.7	8.0
25	13.7	22.0	4.3	12.3

Table 6: *Our framework scales roughly linear in the number of clients up to a rate of 22 frames per second, being mainly limited by the available network bandwidth. "interaction": during photon shooting and accumulation; "walkthrough": after accumulation (no photon shooting any more);*

Resolution	RING	GLASS
640×480	21.0	11.9
800×600	13.7	7.8
1024×768	8.7	4.8
1280×960	5.4	3.2

Table 7: *The frame rates achieved by our framework also scale nearly linear in the number of pixels.*

Utilizing only commodity CPUs our software implementation can compete well with hardware accellerated implementations such as the one proposed by Purcell et al. [PDC*03]. Using only one single CPU and the same parameter settings ($N = 16000$, $k = 64$) with a similar (RING) scene, we are able to render each frame of 512×384 pixels in 2.2 seconds, which is 3.7 times faster than their published rendering time of 8.1 seconds.

6.3. Practical Applications

Using the photon map together with Selective Photon Tracing as an adaptive sampling strategy our system poses few if any limitations regarding scene properties and complexity. In particular, we can also handle indirect caustics (as shown in Figure 3) without considerable performance loss. The ability to handle multiple reflections also includes the correct simulation of colored liquid as shown in the GLASS scene (see Figure 1b).

We have also evaluated the practical applicability of our system by rendering a real world model from the automotive industry. In particular it is not only important to look at car headlights directly [BWDS02] but to also simulate and visualize its light distribution and the illumination cast onto other surfaces. The HEADLIGHT scene shows exactly this application and compares the interactive simulation to real data taken from a photograph of the illumination pattern (Figure 5).

With this scene it is not neccessary to sample caustic generators adaptively using SPT as each photon emitted by the filament hits a specular object (the glass of the light bulb). In this scene each photon is particularly costly as at least 6 rays must be cast casts before a photon hits the wall (2 intersection and interactions with the glass of the bulb, one with the reflector, plus two more with the front glass). In addition a large number of photons are required to accurately simulate the very complex light pattern of the headlight.

Even though we only approximated the filament with a simple isotropically emitting box and put little effort into optimizing the other parameters, the accuracy and the match of the caustic patterns between the simulation and the photograph is remarkable, and even subtle details of the illumination pattern are faithfully reproduced. Even the low quality image with only 250k photons already provides a good estimate of the final simulation. Using 36 CPUs we achive a

frame rate of about 3 fps during the photon shooting and accumulation stage while providing the user with immediate response to his modifications. After the accumulation is finished, the detailed, high-quality caustic (Figure 5c) can be examined at about 11 fps.

7. Conclusions

In this paper, we have presented a complete framework for realtime distributed photon mapping. Our approach builds on the combination of realtime ray tracing, a highly optimized photon mapping algorithms, and sophisticated parallelization on PCs. Using 9 to 36 AthlonMP 1800+ CPUs, we achieve frame rates of up to 22 fps at video resolution (640×480 pixels). On the same hardware platform we obtain a performance improvement of roughly a factor of 2-6 compared to previous results [WKB*02] while simultaneously achieving better image quality and scalability to higher frame rates. Table 7 demonstrates that the new system scales essentially linear also in the image resolution.

To accelerate the photon generation process we presented an elegant adaptive sampling strategy which also allows for the efficient generation of indirect caustics. Unfortunately this sampling method tends to introduce structured artifacts when using only few photons.

Our approach is fully automatic, and does not require any manual user intervention except for specifying the photon mapping parameters N and k, which can be changed interactively. Additionally, it supports realistically complex scenes (such as the HEADLIGHT with 168k triangles) and highly complex light transport patterns.

The new system is an important step towards fully interactive realtime lighting simulations that has the potential to change the way lighting engineers work – similar to the way CAD has changed the environment of designers. Such systems must provide robust visual feedback during interaction and must converge to a high-quality solution – comparable to an offline rendering – within a few seconds.

8. Future Work

In this paper we concentrated on the generation of caustics and intentionally neglected non-caustic illumination. It is an obvious next step to combine our new caustic approach with the scalable interactive global illumination system by Benthin et al. [BWS03].

Additionally, there is significant room for further improving the sampling methods given the current photon yield of 30% to 40% with Selective Photon Tracing. This could also help in reducing sampling artifacts.

Better acceleration structures for photon maps also seem to be a promising research area as even simple ideas did already have a significant effect on the performance. In particular a hybrid of a (hashed) grid with *kd*-trees in the grid cells could be interesting in both query performance and build time. Furthermore, this approach might help in performing the individual *kd*-tree constructions 'on demand' thereby further increasing scalability. Data structures that allow for the efficient use of SSE extensions during traversal are also worth further investigation.

Finally, the recent emergence of medium-sized shared-memory multiprocessor PCs might provide another interesting platform for realtime photon mapping. While the basic framework would probably stay the same, it would be interesting to see whether a shared-memory environment would allow for even higher performance than the current setup.

Acknowledgements

We would like to thank Kirill A. Dmitriev for helpful discussions on selective photon tracing, as well as Georg Demme and Andreas Pomi for helping with the comparison to the physical lamp. We would also like to thank Hella Corp for providing the physical and the VRML headlight model. This work has been supported by AMD and Intel Corp.

References

[BWDS02] BENTHIN C., WALD I., DAHMEN T., SLUSALLEK P.: Interactive Headlight Simulation – A Case Study of Distributed Interactive Ray Tracing. In *Proceedings of the 4th Eurographics Workshop on Parallel Graphics and Visualization (PGV)* (2002), pp. 81–88.

[BWS03] BENTHIN C., WALD I., SLUSALLEK P.: A Scalable Approach to Interactive Global Illumination. *Computer Graphics Forum 22*, 3 (2003), 621–630. (Proceedings of Eurographics).

[CDR02] CHALMERS A., DAVIS T., REINHARD E. (Eds.): *Practical Parallel Rendering*. A K Peters, 2002.

[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.

[GDW01] GRANIER X., DRETTAKIS G., WALTER B.: Fast Global Illumination Including Specular Effects. In *Rendering Techniques* (2001), pp. 47–58. (Proceedings of the 12th Eurographics Workshop on Rendering).

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

[Int02] INTEL CORP.: Intel Pentium III Streaming SIMD Extensions. http://developer.intel.com/vtune/cbts/-simd.htm, 2002.

[Jen96] JENSEN H. W.: Global Illumination using Photon Maps. *Rendering Techniques* (1996), 21–30. (Proceedings of the 7th Eurographics Workshop on Rendering).

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

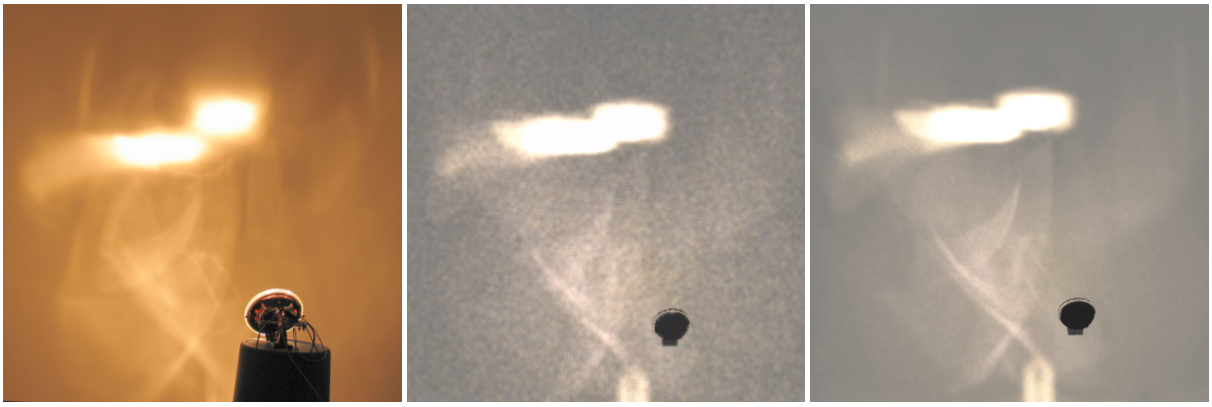

Figure 5: *Headlight simulation as an example for a complex caustic. The only light source is a simple isotropically emitting box inside the headlight. Left: photo of real headlight. Center: interactive simulation using 250k photons and 36 CPUs at 3 fps during manipulation. Right: high-quality simulation using 25M photons at 11 fps after accumulation. Note that after photon accumulation the high-quality photon map on the right can be inspected at even higher frame rate than in the center image.*

[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.

[Kel98] KELLER A.: *Quasi-Monte Carlo Methods for Realistic Image Synthesis*. PhD thesis, University of Kaiserslautern, 1998.

[KH01] KELLER A., HEIDRICH W.: Interleaved Sampling. *Rendering Techniques* (2001), 269–276. (Proceedings of the 12th 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.

[MM02] MA V. C. H., MCCOOL M. D.: Low Latency Photon Mapping Using Block Hashing. In *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware* (2002), pp. 89–99.

[Nie92] NIEDERREITER H.: *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992.

[PDC*03] PURCELL T. J., DONNER C., CAMMARANO M., JENSEN H. W., HANRAHAN P.: Photon Mapping on Programmable Graphics Hardware. In *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware* (2003), pp. 41–50.

[PSL*99] PARKER S., SHIRLEY P., LIVNAT Y., HANSEN C., SLOAN P.-P.: Interactive Ray Tracing. In *Proceedings of Interactive 3D Graphics* (1999), pp. 119–126.

[SAG94] SMITS B., ARVO J., GREENBERG D.: A Clustering Algorithm for Radiosity in Complex Environments. In *Proceedings of the 21st annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)* (1994), pp. 435–442.

[SWZ*95] SHIRLEY P., WADE B., ZARESKI D., HUBBARD P., WALTER B., GREENBERG D. P.: Global Illumination via Density Estimation. In *Proc. of the 6th Eurographics Workshop on Rendering* (1995), pp. 187–199.

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

[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/.

[WGS04] WALD I., GÜNTHER J., SLUSALLEK P.: Balancing Considered Harmful – Faster Photon Mapping using the Voxel Volume Heuristic. *Computer Graphics Forum 22*, 3 (2004). (Proceedings of Eurographics, to appear).

[WH92] WARD G. J., HECKBERT P.: Irradiance Gradients. In *Third Eurographics Workshop on Rendering* (1992), pp. 85–98.

[WHSG97] WALTER B., HUBBARD P. M., SHIRLEY P., GREENBERG D. P.: Global Illumination Using Local Linear Density Estimation. *ACM Transactions on Graphics 16*, 3 (1997), 217–259.

[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).

[WPS*03] WALD I., PURCELL T. J., SCHMITTLER J., BENTHIN C., SLUSALLEK P.: Realtime Ray Tracing and its use for Interactive Global Illumination. In *Eurographics State of the Art Reports* (2003), Eurographics.

[WS03] WAND M., STRASSER W.: Real-Time Caustics. *Computer Graphics Forum 22*, 3 (2003), 611. (Proceedings of Eurographics).

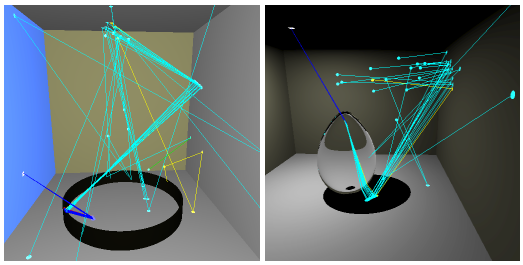

Figure 2: *Automatic sampling of caustic generators using Selective Photon Tracing [DBMS02]: First, some pilot photons (yellow) are shot into the scene. Those pilot photons contributing to a caustic are then refined (cyan) using the similarity property of the Halton sequence [Nie92].*

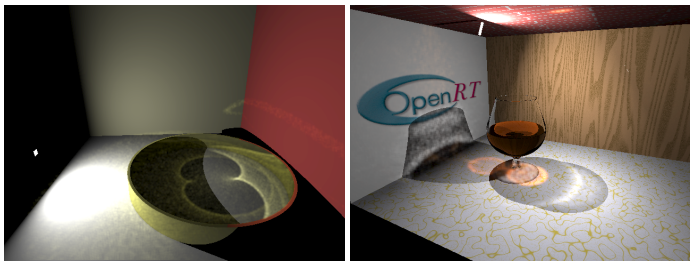

Figure 1: *Several examples of our distributed realtime photon mapping framework: a) Specular metal ring, including multiple reflections. b) Cognac glass with detailed caustics. Using our framework, these scenes can be rendered interactively, running at 21 and 12 frames per second, on 8 and 13 dual-AMD AthlonMP 1800+ PCs, respectively.*