EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

Global Illumination using Photon Ray Splatting

Robert Herzog[1], Vlastimil Havran[2], Shinichi Kinuwaki[1], Karol Myszkowski[1], Hans-Peter Seidel[1]

[1]MPI Informatik, Computer Graphics Group, Saarbrücken, Germany
[2]Czech Technical University in Prague, Czech Republic

Abstract

We present a novel framework for efficiently computing the indirect illumination in diffuse and moderately glossy scenes using density estimation techniques. Many existing global illumination approaches either quickly compute an overly approximate solution or perform an orders of magnitude slower computation to obtain high-quality results for the indirect illumination. The proposed method improves photon density estimation and leads to significantly better visual quality in particular for complex geometry, while only slightly increasing the computation time. We perform direct splatting of photon rays, which allows us to use simpler search data structures. Since our density estimation is carried out in ray space rather than on surfaces, as in the commonly used photon mapping algorithm, the results are more robust against geometrically incurred sources of bias. This holds also in combination with final gathering where photon mapping often overestimates the illumination near concave geometric features. In addition, we show that our photon splatting technique can be extended to handle moderately glossy surfaces and can be combined with traditional irradiance caching for sparse sampling and filtering in image space.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism - Rendering, Global Illumination

1. Introduction

Many rendering applications used in industrial design and special effects in movie productions require high quality global illumination solutions, which are costly for complex scenes with general reflectance models. A common choice in such applications is the photon mapping algorithm [Jen01], in which stochastic photon tracing is performed and the resulting photon hit points on the scene surfaces are registered in the photon map. The nearest-neighbor density-estimation method developed in statistics [Sil85] is then employed to reconstruct the lighting function based on the photon map. Since a finite neighborhood is needed to collect a sufficient number of photons and to reconstruct the lighting function with an acceptable noise level, all density estimation methods are prone to a systematic error, so-called *proximity bias* [Sil85, Sch03]. Photon mapping also suffers from other systematic errors: *the boundary bias* (i.e., underestimation of illumination near object boundaries) and *the topological bias* (i.e., overestimation of illumination for curved surfaces) [Sch03, HBHS05].

Recently, Lastra et al. [LURM02] and Havran et al. [HBHS05] have shown that a viable alternative for the density estimation of photon hit points on the scene surfaces is an analogous operation performed directly for photon paths traveling in the proximity of these surfaces. To compute the irradiance value at a given surface point, its neighborhood is searched for photon rays that intersect a disc in the tangent plane, which is centered at this point. The disc is extended until a minimum specified number of photon rays is found or a maximum disc radius is exceeded. This leads to elimination of boundary bias inherent to photon maps as well as a reduction of topological bias for convex surfaces, since density estimation is computed for a disc in the tangent plane and the real surface area does not need to be estimated. The disadvantage of these methods is that they need complex and memory demanding data structures for nearest neighbor searches of rays. Their algorithms rely heavily on the coherence in the search queries and therefore are mostly limited to primary-ray hit points shot from the camera, which we will refer to as *eye samples*. Another drawback of these methods is the heuristically computed tangent-disc area, which is incorrectly estimated for points on concave surfaces, for example, in corners where a disc is partially intersected.

Direct rendering of lighting computed using a visualiza-

Table 1: *Main symbols used in this paper.*

$x_0, x_1, \ldots$	Light path vertices (x_0 origin on light source)
ω', ω	Outgoing, incoming light directions, respectively
θ, ϕ	Polar, azimuthal angle relative to surface normal
$\cos\theta_x$	Cosine of the angle θ with normal at x
M	Total number of photon hits recorded in the scene
K	Number of nearest neighbors for density estimation
$\Phi(x,\omega)$	Incoming flux (photon) at x from direction ω
$L(x,\omega), E(x)$	Radiance at x from direction ω, irradiance at x
$\mathcal{K}_h(x,y)$	2D kernel function at x with bandwidth h
$\mathcal{K}_h(x,y,\omega)$	2D kernel with domain perpendicular to ω
$d\sigma(\omega)$	Differential solid angle for direction ω
$dA(x), \Delta A(x)$	Differential surface area, finite surface area at x
$dA^{\perp}_{\omega}(x)$	Projected differential surface area, $= dA(x) \cdot \cos\theta$
$V(x,y)$	Visibility predicate for points x and y
$f_s(x_i,x_j,x_k)$	BRDF at x_j for scattering from x_i to x_k
$p(x)$	Probability density function (pdf)
$p(y\|x)$	Conditional pdf for sampling path vertex y given x

tion of photon maps leads to poor image quality. This gives only a vague idea of the rendered scene appearance and to get high quality images costly *final gathering* (the numerical integration of incoming radiance over the whole hemisphere) has to be performed [Jen01]. These costs can be reduced by using the *irradiance cache* data structure to interpolate irradiance samples sparsely in object space both for diffuse [WRC88] and moderately glossy [KGPB05] surfaces.

The goal of this work is to provide a framework for quickly computing rendered previews of good quality, while also enabling the functionality of final gathering and irradiance caching if even higher quality and more robust results are needed. Our method shares all discussed benefits of ray density estimation, but neither requires the expensive *k-nearest neighbors* (K-NN) search as for the ray maps technique nor relies on the surface orientation. We replace the line density estimation in the tangent plane by energy splatting along photon rays to the eye samples. Our new density estimation metric is capable of handling illumination on complex geometric shapes (e.g. wrinkled and bump mapped surfaces), which is not working in a direct visualization of photon maps. It also reduces the low-frequency noise, which appears as speckles in the final image (Fig. 9). Compared to standard photon density estimation, we obtain better image quality with the same number of photons because more photons contribute to a pixel using the same kernel width, which reduces variance. Further combined with multi-stage filtering, our method leads to fast rendering of images with acceptable quality for low-frequency indirect illumination.

Additionally, we show how our method can be extended with state-of-the-art techniques in global illumination such as radiance caching and non-diffuse lighting on moderately glossy BRDFs. We only outline those extensions in this paper and provide more details in [HHK*07]. In the following section we review previous work concerning an operation central for our algorithm: photon energy splatting onto the image plane.

2. Previous Work

Photon mapping with direct photon splatting onto the image plane has already been investigated by Lavignotte et al. [LP03]. They focus on off-line rendering with a splatting method that avoids the boundary bias by precise computation of the area covered by the splat footprint over each mesh element. On the other hand, they do not solve this problem for arbitrary topology and their approach fails when it comes to estimation of the illumination on small surfaces. Furthermore, they allow only for diffuse scattering of light towards the camera.

Another interesting photon splatting approach was published in [BSC*03]. This work aims at high-quality rendering of wide range of BRDFs and illumination conditions using photon density estimation. This is achieved by correcting the error in the neighborhood of a photon hit point. However, their method has problems to estimate the solid angle subtended by the splatting footprint at the photon's origin, which is generally difficult.

Splatting has also been used in the context of different global illumination algorithms such as path tracing [DLW93] and bidirectional path tracing [SW00], but there the main motivation was noise reduction. Suykens and Willems [SW00] propose an iterative procedure with adaptive kernel size control taking into account the density of samples and their energy contributions. This technique inspired us to control the splat size as a function of photon-path probability-density, BRDF sampling-density, and the number of photon bounces. A similar concept is *path differentials* [SW01], in which partial derivatives and a gradient for individual paths are computed to estimate a footprint. This footprint information gives an idea about the density of similar paths in the neighborhood. The footprint can be used for adaptive texture filtering or to steer the patch subdivision in a hierarchical radiosity algorithm.

Splatting is also commonly used in recent GPU-based global illumination techniques, which are often designed for games and are usually limited to a single bounce of indirect lighting for purely Lambertian environments. For example Dachsbacher and Stamminger [DS06] use an extended shadow map to deposit secondary light sources directly on lit scene surfaces and then splat energy from these sources to neighboring pixels without care of visibility in the indirect light. Splatting is also used to deposit lighting energy from reflections, refractions, and caustics [SKALP05, SKP06, WD06]. In all those solutions the main goal is maximizing the rendering speed even at the expense of reduced image quality.

Another class of algorithms are hybrid GPU-CPU off-line techniques that are designed for high quality rendering where the GPU role is merely to speed up some critical computation parts. Gautron et al. [GKBP05] use the GPU for the irradiance cache computation by rasterizing directly illuminated scene geometry (similar to [LC04]). In addition,

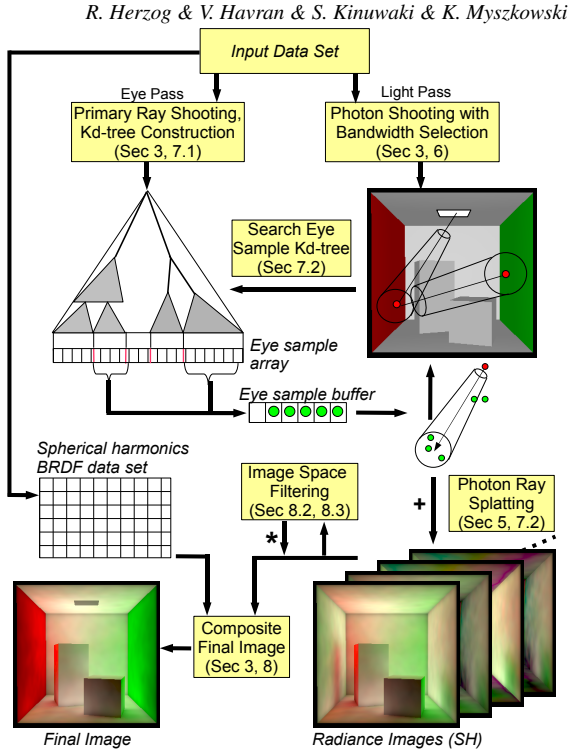


Figure 1: *The processing flow in our photon splatting architecture (spherical harmonics splatting).*

the GPU is employed for splatting irradiance caches into the image plane.

Our method differs from previous splatting approaches in the way how a photon splats its energy along whole photon paths. This avoids boundary bias and reduces topological bias inherent to photon maps [Jen01], which has been demonstrated in [HBHS05] and [LURM02]. In contrast to [HBHS05] and [LURM02] our photon ray splatting approach decouples the density estimation footprint entirely from scene surfaces by applying a new density estimation metric over photon rays.

3. Algorithm Overview

In this section we give an overview of the major processing steps in our photon-ray splatting architecture (refer also to Fig. 1).

The basis of our method is a bidirectional path tracing algorithm combined with density estimation that samples eye and light paths (photons) whereby the eye paths are kept short to avoid the expensive final gathering. No indirect eye paths are sampled except for deterministic reflections. Instead, the main computation is carried out for the light paths via photon splatting. The rendering algorithm consists of four passes:

Eye pass: Primary rays are shot from the eye (camera) and created *eye samples* are stored at the hit points on the scene surfaces. The eye samples store position, normal, BRDF index, direction, pixel index, and weight for RGB components. Next a spatial kd-tree over the eye samples is constructed (Section 7).

Light pass: After the eye pass the light pass starts with photon sampling. To speed up the subsequent search, the created photon paths are sorted by means of a 5D kd-tree in spatial and directional domain. Each photon ray is assigned a splatting kernel width, which is computed based on the entire photon path (Section 6).

Photon splatting: In the next phase all photons are splatted one by one to eye samples in the vicinity of the photon rays (Section 4) using a density estimation kernel as described in Section 6. For an efficient nearest eye sample search along a photon ray, the kd-tree over eye samples is traversed for a conical search domain (Section 7.2). In order to reconstruct and filter glossy light transport at eye sample hit points, photon energy contributions are accumulated in an intermediate 4D data structure, which we call the *radiance map*, before being rendered to a final pixel. The radiance map consists of a number of *radiance images* with the same resolution as the final image, which represent spatial and directional information of incoming light. In addition, a 2D image storing splat information per pixel is updated. This image records the harmonic mean distance of incident photon rays and the sum of density-estimation kernel weights for each pixel and is used for determining the radiance cache density (Section 8.3).

Final integration with BRDF evaluation: The final image is then composed from the radiance images either via BRDF sampling from eye samples or in the spherical harmonics basis. The main algorithm flow for the spherical harmonics splatting is illustrated in Fig. 1.

4. Photon Density Estimation in Ray Space

Standard photon density estimation methods gather photon energy in the neighborhood of a point inside a sphere using a normalized symmetric density estimation kernel [Sil85, Jen01]. This kernel ignores photon density changes (e.g. illumination gradients) and assumes an unbounded, planar surface in the neighborhood of estimates. In our method, we partially correct for the error introduced by the density estimation kernel and still keep performance high.

In order to accomplish these goals, let us first consider the probability density (pdf) for sampling the next photon hit point x_{i+1} from a particular photon location x_i. This pdf is given by

$$p_r(x_i \rightarrow x_{i+1}) = V(x_i, x_{i+1}) p_s^{\perp}(x_{i-1}, x_i, x_{i+1}) \frac{\cos\theta_{x_{i+1}}}{\|x_i - x_{i+1}\|^2}, \quad (1)$$

where $p_s^{\perp}(x_{i-1}, x_i, x_{i+1}) = p_s(x_{i-1}, x_i, x_{i+1}) \cdot \cos\theta'_{x_i}$ is the pdf for sampling the cosine weighted BRDF at point x_i. Refer to Table 1 for explanation of the symbols used in the paper.

Considering (1) we can draw some conclusions about the photon path density (i.e. flux density or irradiance) changes at point x_{i+1}. We have two assumptions. First, we assume that the density estimation footprint at x_{i+1} is relatively small with respect to the squared distance $\|x_i - x_{i+1}\|^2$ such that the visibility $V(x_i, x_{i+1})$ equals to one with a high probability. Second, we assume that the BRDF sampling density p_s at x_i is of low-frequency. Under these two assumptions the change in photon paths density in the neighborhood of x_{i+1} mainly depends on the change in surface orientation ($\cos\theta_{x_{i+1}}$).

In photon mapping [Jen01] this factor is neglected, assuming the domain is planar and continuous resulting in visible bias in corners and on curved surfaces, see Fig. 9(a). To avoid this problem, we propose to estimate the photon density in ray space from the nearest photon rays and project the result onto the local surface to obtain the irradiance contribution.

Recall that radiance is a 5-dimensional quantity independent of surface orientation. However, its measure is defined on the surface since radiant energy (photons) is measured on surfaces (our sensors). Therefore, in order to compute radiance, the density of photons incident on a surface with area $dA(x)$ is projected in the direction ω. This gives us the number of photons passing through a differential area $dA^{\perp}_{\omega}$ perpendicular to ω (see Fig. 2). Equivalently, if we know

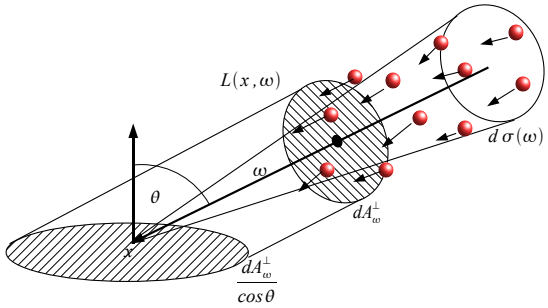


Figure 2: *Radiance is defined as the incoming photon flux density per unit time in a projected differential area $dA^{\perp}_{\omega}$ and differential solid angle $d\sigma(\omega)$. Intuitively this can be understood as all the photons with direction in a certain solid angle crossing $dA^{\perp}_{\omega}$ per time unit.*

the radiance $L(x,\omega)$ for the differential area $dA^{\perp}_{\omega}$, we can compute its contribution to the irradiance of an arbitrarily oriented surface with area dA by multiplying $L(x,\omega)$ with the cosine of the angle θ between surface normal and ω (see Fig. 2).

To compute the irradiance $E(x)$ at a point x on a surface, the incoming radiance is integrated over the hemisphere at x

$$E(x) = \int_{\Omega^+} L(x,\omega)\cos\theta\, d\sigma(\omega) = \int_{\Omega^+} \frac{d\Phi(x,\omega)}{dA(x)} \quad (2)$$

$$= \int_{\Omega^+} \frac{d\Phi(x,\omega)\cos\theta}{dA^{\perp}_{\omega}(x)} \quad (3)$$

In photon density estimation we replace the differential quantities by finite ones and compute an approximation to the real irradiance

$$E(x) \approx \sum_i^{K_1} \mathcal{K}_h(x,x_i)\frac{\Delta\Phi_i(x_i,\omega_i)}{\Delta A(x)} \quad (4)$$

$$\approx \sum_i^{K_2} \mathcal{K}_h(x,x_i,\omega_i)\frac{\Delta\Phi_i(x_i,\omega_i)}{\Delta A^{\perp}_{\omega_i}(x)}\cos\theta_i, \quad (5)$$

where $\mathcal{K}_h(x,x_i)$ is a density estimation kernel that satisfies $\int_S \mathcal{K}_h(x,y)dy = 1, \forall x$ and $\mathcal{K}_h(x,x_i,\omega_i)$ is the same kernel whose domain is oriented perpendicular to the direction ω_i. Hence the kernel evaluates the distance of x to the ray (x_i,ω_i) (Fig. 3) and includes more photon samples in the density estimation (i. e. $K_1 \leq K_2$). Combining (5) with a BRDF function f_s to compute reflected radiance $L(x,\omega')$ towards the viewer leads to:

$$L(x,\omega') \approx \sum_i^{K} f_s(\omega_i,x,\omega')\mathcal{K}_h(x,x_i,\omega_i)\frac{\Delta\Phi_i(x_i,\omega_i)\cos\theta_i}{\Delta A^{\perp}_{\omega_i}(x)}. \quad (6)$$

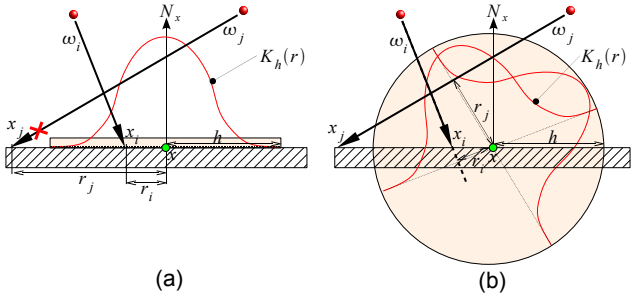


Figure 3: *Photon density estimation via* ray gathering *visualized in 1D for two incoming photon rays: (a) when gathering in the tangent plane (ray disc intersection) only ray i contributes; (b) with our new metric, all rays intersecting the sphere with radius h centered around x contribute to the irradiance at x.*

5. Photon Splatting Instead of Gathering

Instead of computing the photon density at an eye sample point x by *gathering* all neighboring photon rays, we utilize a *splatting* approach. In splatting methods a photon computes its contribution weighted by a normalized kernel to a number of eye samples at once. This corresponds to kernel density estimation (KDE) in statistics [Sil85]. *Variable KDE* with kernel width adaptive to the local photon density is preferable over *k-nearest neighbors* (K-NN) density estimation because it does not excessively blur high density gradients (e.g. shadow boundary), see [Sil85] for details. Adaptive splatting also simplifies the rendering algorithm since photons do not need to be classified to caustics and global photons as in photon mapping where two photon map queries are performed each one with its own K-NN gather parameters.

If we consider now a constant bandwidth for kernel $\mathcal{K}$, then searching at each eye sample point all photons crossing

the bounding sphere (Fig. 3(b)) is equivalent to searching for each photon all eye sample points in the cylinder centered along the photon's ray (x_i, ω_i). Instead of a cylinder we use a conical frustum as search domain because it is better suited for our bandwidth selection scheme (Section 6). Each photon splats its energy to the eye samples found in a conical frustum (Fig. 4). The energy is weighted by a 2D kernel perpendicular to the direction of its photon path.

In practice the difficulty of this approach is to define the scope of the splatting at the ends of the cone, in particular for rays arriving at a grazing angle. We propose a simple heuristic: for eye samples located beyond the photon hit point the splatting footprint is reduced to a hemisphere (Fig. 4). This heuristic simplifies the search and prevents excessive light leakage as we do not evaluate the visibility for eye samples within the splatting footprint. It can increase the noise since the splatting radius reduces gradually at the end of the cone. In practice however, the noise is hardly visible as it only affects rays arriving at a grazing angle, which have low contribution.

Naturally, a photon can only contribute to eye samples on surfaces that are oriented towards the photon ray. A splatting example is shown in Fig. 4.

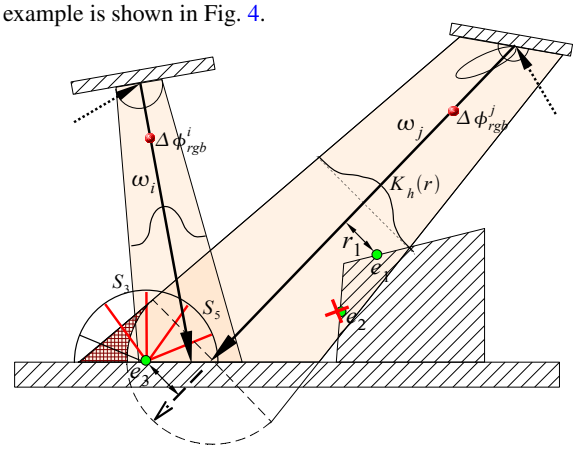


Figure 4: *Density estimation via photon ray splatting to eye samples (e_1, e_2, e_3). Photon j splats its energy within its kernel support (shaded area) of the function $\mathcal{K}(r)$ to e_1 and e_3, but not to back-facing e_2. The resulting energy contribution is either added to a corresponding radiance image or directly projected onto the spherical harmonics basis. Note that the splatting footprint (conical frustum) is extended to a hemisphere (dashed line) for eye samples beyond the photon hit point culling a part of the frustum (red cross-hatched region).*

6. Choice of Splat Kernel and Bandwidth Selection

According to statistical studies [Sil85], the shape of the kernel $\mathcal{K}$ is rather unimportant for the bias reduction in density estimation. Therefore, we have used the computationally efficient *Epanechnikov* kernel [Sch03, Sil85]. A more important issue in density estimation is the choice of the *kernel width* or *bandwidth* h [Sil85], which is difficult to choose and often replaced by various heuristics. Most photon mapping algorithms are based on k-nearest neighbors (K-NN) density estimation where the bandwidth is directly related to the local density of the samples. K-NN density estimation reduces variance uniformly careless of introducing bias, which may result in strongly "blurred" images especially in dark regions.

For photon splatting, it is difficult to have a bandwidth selection proportional to the local sample density, which is not explicitly known during photon tracing. What we know is the path density and the contribution of individual photon paths. In the case of perfect BRDF importance sampling ($p_s^{\perp} \sim f_s \cdot \cos\theta'$) and light source sampling proportional to its energy contribution ($p_e \sim L_e$), the photons are distributed according to the irradiance function and all the photons have the same power.

Based on (1), we relate the bandwidth $h(x_i)$ to the inverse path probability density of a photon, which has some desirable bias reduction properties. For example, photons from a small number of bounces obtain a smaller bandwidth better preserving shadow boundaries and high illumination gradients while photons of multiple bounces spread their energy in a larger area, which reduces low-frequency noise. Also caustic photon paths yield a relatively small bandwidth since BRDF sampling density is high.

Since we do splatting in ray space with projected area measure, the path density is independent of the surface orientation at x_{i+1} and the cosine term $\cos\theta_{x_{i+1}}$ cancels out. Moreover, the path density $p(x_i|x_{i-1})$ can be arbitrarily small and arbitrarily large due to the distance term $\|x_i - x_{i+1}\|^2$ and we need to clamp it before using it in the bandwidth selection:

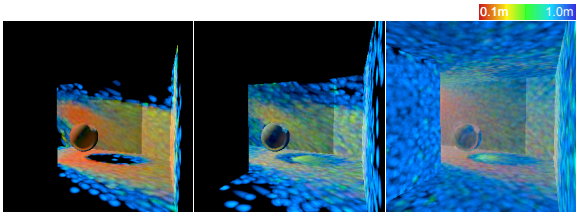
$$\tilde{p}(x_{i+1}|x_i) = \begin{cases} \frac{p_e(x_0,x_1)}{D(x_0,x_1)} & i = 0 \\ \tilde{p}(x_i|x_{i-1}) \cdot \frac{p_s^{\perp}(x_{i-1},x_i,x_{i+1})}{D(x_i,x_{i+1})} & i > 0, \end{cases} \tag{7}$$

where $D(x,y) = \max(\tilde{D}^2, \|x-y\|^2)$ is the squared length of the photon ray clamped at a scene dependent distance threshold $\tilde{D}^2$. Bounding the geometry term is also commonly applied, in a different context, to instant radiosity algorithms [Kel97].

Using the bounded path probability density defined in (7), we compute the bandwidth $h(x_i)$ per photon ray by the following heuristic

$$h(x_i) = \begin{cases} 0 & i = 0 \\ \frac{C}{\sqrt[6]{M}} \frac{w}{\sqrt{\tilde{p}(x_i|x_{i-1})^S}} & \forall i > 0, \end{cases} \tag{8}$$

where C is a user defined "smoothness" parameter and $S \in (0 \ldots 1]$ is a user defined bandwidth sensitivity controlling the variance of $h(x_i)$. According to the optimal bandwidth for minimizing the mean integrated square error, $h(x_i)$ should be inversely proportional to the sixth root of the total number of samples M [Sil85]. This ensures that the



(a) (b) (c)

Figure 5: *Color-coded splatting size (bandwidth) per photon in our test scene, red corresponds to minimum, blue to maximum bandwidth. (a) indirect photon-ray splats from 1 discrete direction for second bounce, (b) fourth bounce, and (c) all photon bounces from all directions. For visualization purposes the color-coded bandwidth is splatted in a constant radius directly to the pixels. Note the smaller bandwidth associated to the indirect caustics photons passing through the glass sphere, which have a higher path probability density because of the two specular refractions.*

level of noise stays approximately the same when increasing or decreasing the number of photons. In our implementation the number of stored photons M is set for direct, indirect diffuse, and caustics photons separately. The square root comes from the fact that $\tilde{p}$ is related to the area rather than the radius of the splat footprint. The normalization coefficient w is automatically precomputed in an initial pilot shooting phase, which estimates the mean $\langle r \rangle$ of the term $r = 1/\sqrt{\tilde{p}(x_i|x_{i-1})^S}$. Parameter w is then computed as $w = m_0 \frac{\mu_C}{\langle r \rangle}$, where the scene size dependent parameter $\mu_C = a \cdot \langle D(x,y) \rangle$ is computed from the average path segment lengths $\langle D(x,y) \rangle$ also estimated in the pilot shooting phase. We set the constant $a = 0.2$ and $m_0 = \sqrt[6]{10^5} \approx 6.8$ is a calibration factor normalizing the bandwidth dependency on M for $M = 10^5$ photons. For the sake of robustness, the resulting bandwidth $h(x_i)$ is clamped at a minimum and maximum boundary value derived from μ_C and a user defined maximum bandwidth scaling R.

Each photon ray $(x_i \rightarrow x_{i+1})$ stores the initial bandwidth $h_0(x_i) = min\{h(x_i), h(x_{i+1})\}$ and the differential bandwidth per ray length $dh(x_i) = max\{0, h(x_{i+1}) - h(x_i)\}/\|x_{i+1} - x_i\|$, which determines the angle of the conical frustum. In Fig. 5(a - c), the precomputed bandwidth per photon splat is shown for 2 to 4 photon bounces in a false-color mapping.

7. Efficient Nearest Neighbor Search Using a KD-Tree

All kernel density estimation methods require a search for the nearest neighbors at the sample points. Previous approaches to photon-ray density estimation in the tangent plane are based on gathering the K-NN photon rays and need complex search data structures to manage the increased dimensionality (5D) of the ray data [LURM02, HBHS05]. Since we utilize a splatting approach, we can still restrict our method to the well-researched problem of searching point data.

We use a kd-tree over 3D points (eye samples), which can be constructed very efficiently. However, we need to modify the search for finding the nearest neighbor points in a conical frustum associated with a photon ray with parameters depending on the bandwidth as explained in Section 6.

7.1. The Splat KD-Tree Layout

We have chosen an axis-aligned kd-tree built with sliding midpoint rule [HHS05]. Splitting planes are positioned at the spatial median of a node's associated bounding box or at the sample point nearest to the spatial median if either half space is empty. The kd-tree is constructed recursively from top to bottom until the number of samples per node is smaller than 16 or the diagonal of the node's bounding box is smaller than a minimum threshold (0.1% of scene diagonal).

The kd-tree consists of four node types: *interior nodes*, *leaf nodes*, *empty nodes*, and *backface-culling nodes*. Each node uses 8 Bytes. The interior node encodes 1D splitting plane, offset to the right child node, node type and splitting axis. The left child node is always found at the next position $(index + 1)$ in the array of kd-tree nodes. In a leaf node the reference to the first eye sample and their number is stored. Empty nodes (stopping nodes) are stored whenever the sub-tree does not contain any eye samples. Additionally, we insert special nodes similar to [HBHS05] that we call backface-culling nodes. Such nodes allow for early culling of entire sub-trees containing infeasible backfacing eye samples with coherent normals. In addition to the reference normal [HBHS05], we also store the maximum angular deviation from the reference normal. This conservative approach yields higher efficiency of successfully culling rays in particular on planar surfaces where the angular deviation of the normals is zero. More details can be found in [HHK*07].

7.2. Photon Ray KD-Tree Traversal

The tree is traversed from top to bottom as in standard ray tracing algorithms with node traversal in 1D ray space. The difference is that we need to consider a volume associated with the ray. A kd-tree traversal algorithm for a similar problem, however in a different context, has been proposed by Dahmen [Dah04]. He uses a kd-tree for accelerating the ray tracing of point data represented as oriented discs.

We start computing the minimum and maximum ray distance t_0 and t_1 by clipping the ray at the bounding box of the kd-tree extended by the ray's splat radius. Then we test t_0 and t_1 with the splitting plane of the current tree node. This plane is virtually moved to its left and right by the maximum splat radius R_1 of the photon ray. A child node needs to be traversed if a ray intersection with its virtual plane lies between t_0 and t_1. If the front-facing child node needs to be traversed, t_1 and R_1 are updated. If the back-facing child

node is scheduled for traversal, t_0 and R_0 are updated. The pseudo code in 1 describes the basic recursive version of the algorithm. The complete traversal step of one interior node

Algorithm 1 KD-Tree Traversal

```
TraverseTree(ray, i_n, t_0, t_1)
  node := nodes[i_n]
  nearChild := i_n + 1
  farChild := node.offset()
  if (node.type = EMPTY) then
    return
  else if (node.type = LEAF) then
    test all samples in leaf and add to candidate list
    return
  else if (node.type = BACKFACE_CULL) then
    if (ray.dir • node.normal > node.C_max) then
      return
    end if
    TraverseTree(ray, nearChild, t_0, t_1)
  else if (node.type = INTERIOR) then
    a := node.axis
    R_1 := ray.h_0 + ray.dh · t_1  /* compute splat radius at t_1 */
    /* compute ray length Δt between virtual plane and splitting plane */
    Δt := R_1 / ray.dir[a]
    t := (node.plane1D − ray.org[a]) / ray.dir[a]
    /* compute ray lengths t_α and t_β to virtual planes α and β */
    t_α := t − Δt; t_β := t + Δt
    if (t_0 < t_β) then
      /* traverse front */ TraverseTree(ray, nearChild, t_0, min(t_β, t_1))
    end if
    if (t_1 > t_α) then
      /* traverse back */ TraverseTree(ray, farChild, max(t_α, t0), t_1)
    end if
  end if
```

is shown in Fig. 6, where the ray needs to visit both left and right child nodes. Once the ray traverses a leaf node all eye samples associated with the node are tested for feasibility, i.e. distance to ray is smaller than splat radius, the normal of the eye sample is front facing, and its position is in front of the surface at the ray's origin. If feasible, the eye sample's weight is computed by the 2D kernel multiplied with the cosine between normal and ray direction. In the proceeding splatting phase the photon ray splats its energy contribution to all pixels corresponding to the eye samples gathered during kd-tree traversal.

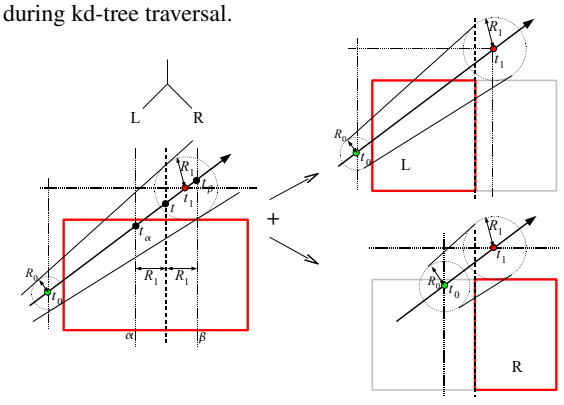


Figure 6: *Traversal of a kd-tree node given the photon ray and its associated splat radius h. The dashed vertical line represents the 1D splitting plane and the thin dotted lines the virtual extensions of the node's corresponding voxel. The red frame depicts the currently traversed node. If $t_1 > t_\alpha$, we descend to the right child node. If $t_0 < t_\beta$, we traverse to the left child node.*

8. Algorithmic Extensions

Like photon mapping [Jen01], ray splatting is a very general method. It can be extended with many state-of-the-art techniques. Next we present a few examples that we have implemented and tested. Further details can be found in [HHK*07].

8.1. Rendering Glossy Surfaces

In order to handle moderately glossy surfaces, we have tested different methods for representing the directional information of incoming light. First method is based on a histogram approach. This means that incoming photon flux from a global discrete direction arriving in the neighborhood of the corresponding density estimation point is recorded in directional strata (Fig. 4). Since we only account for moderately glossy BRDFs, high angular frequencies in the incoming radiance are filtered by the BRDF [RH04]. We discretize the sphere of incident directions to $N = 20$ strata defined by the vertices of a regular dodecahedron. A directional filter kernel can be applied [Stü98] such that each photon splats its energy to several neighboring strata with precomputed filter weights for N_D discrete photon directions.

Using uniformly distributed strata over the hemisphere is efficient for splatting, but is not adaptive and can lead to aliasing artifacts if a BRDF contains too high frequencies. In our second approach we use spherical harmonics (SH) [Mac48, RH04], since they are well suited for low-frequency signals over the sphere and simplify interpolation of radiance for non-diffuse surfaces. We represent illumination as well as BRDFs by a small number of coefficients. All BRDF data is initially precomputed for a number of discrete outgoing directions mapped to SH coefficients, which we store in a table. As opposed to [RH04,KGPB05], we encode the term $\cos\theta$ in the radiance SH coefficients rather than in the BRDF coefficients and represent Lambertian BRDFs by only one SH coefficient.

8.2. Radiance Filtering in 2D Image Space

The ray splatting complexity depends linearly on the search neighborhood and therefore on the size of the splat footprint. Instead of increasing the splat footprint, effectively the radius of the cone, we can also employ a second pass filter in 2D image space to filter noise in the radiance images. To preserve discontinuities during filtering of the radiance images we only filter over geometrically continuous image regions, which we quickly identify by segmentation of a *discontinuity buffer* [WKB*02, McC99]. In addition, we keep the density estimation kernel weights per pixel accumulated during ray splatting, from which we can estimate the variance of each pixel and decide upon the filter size. We do not filter visible radiance but indirect radiance for each direction or spherical harmonics band separately, which is only visible through BRDF modulation [RH04]. The image space

filtering makes use of summed area tables to speed up the computation and also allows for more advanced bandwidth selection in spatial and directional domain. Details can be found in [HHK*07].

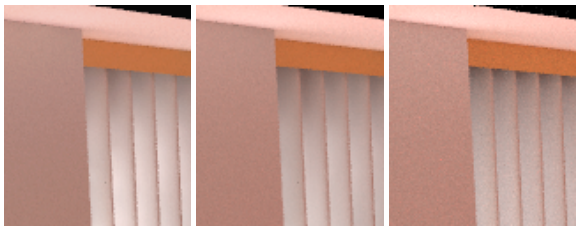
8.3. Radiance Caching

It is well known that (ir)radiance caching significantly speeds up computation of diffuse (glossy) indirect illumination computed with final gathering because the image screen is adaptively sampled [WRC88, KGPB05]. Motivated by the radiance caching technique of Křivánek et al. [KGPB05], we also perform the caching in the spherical harmonics basis. However, the difference lies in the illumination computation. While the approach in [KGPB05] computes a high-quality solution of the incoming radiance by Monte Carlo final gathering, we estimate the incoming radiance directly from neighboring photon-ray splats. Our caching algorithm, initially intended to speed up the computation, inherently filters noisy photon splats as a by-product. In contrast to traditional (ir)radiance caching, reducing the cache error ε below a certain minimum error will not give any quality improvements since the illumination at computed eye samples is already low-pass filtered due to density estimation. This, on the other hand, makes the algorithm well-suited for sparse sampling in the image plane.

We utilize a multi-pass radiance caching algorithm [GKBP05]. The radiance caching and extrapolation is carried out in image space via cache splatting similar to [GKBP05]. The filter radius of a cache record is computed in world space from the harmonic mean distance [WRC88] and projected to image space as visualized in Fig. 8(b). We estimate the harmonic mean distance from the neighboring photon rays during the ray splatting.

8.4. High Quality Rendering with Final Gathering

In photon density estimation the visibility within the density estimation footprint is neglected and high frequency indirect lighting due to occlusion cannot be reproduced and may lead to energy leaking for complex scenes. To address this problem, the global photon map is only queried for rays after the first bounce from the camera, referred to as final gather rays (FGRs), generated using Monte Carlo sampling of the BRDF (final gathering) [Jen01]. Final gathering balances the local error in the photon map estimate across all pixels and produces high quality indirect illumination (except for caustics, which are computed from a direct visualization of the caustic photon map). Nevertheless, final gathering with photon mapping has its shortcoming for concave surfaces where many FGRs hit the local neighborhood resulting in overestimated illumination, see Fig. 7(a). In such cases secondary final gathering is initiated drastically increasing the computation cost of the corresponding pixel. This is especially problematic for (ir)radiance caching [WRC88] where the cache samples are concentrated near concave features such as corners. Combining our method with final gathering, we can mostly avoid secondary final gathering and also speed up the nearest neighbor search compared to photon mapping. This requires only a small change in the algorithm described in Section 3. Instead of primary ray hit points, we need to store all FGR hit points. To handle the increased memory demands, the image plane is rendered in tiles utilizing multiple splatting passes with the same photon ray distribution as proposed in [HHS05]. Further, we do not use the radiance map (Section 8.1), which would be too memory consuming, but directly splat photon energy to the corresponding pixels weighted by the FGR contribution and the BRDF at the FGR hit point.



(a) (b) (c)

Figure 7: *Difficult indirect lighting in the conference scene rendered with (a) photon mapping with 600 final gather rays per pixel, (b) ray splatting with 600 final gather rays per pixel, and (c) path tracing with 2000 paths per pixel. The full images are shown in Fig. 9(d).*

9. Results

We have evaluated our method for the scenes shown in Figs. 9 and 10 using standard photon mapping with K-NN density estimation and our ray splatting. The rendering times are given in Table 2. All results were computed on a single PC (AMD Opteron 2.4 GHz) with a Linux operating system. The first scene, CORNELL BOX I, contains high frequency geometry and glossy objects rendered in full global illumination using a direct visualization of the photon/ray density. The illumination on the bumpy floor is "washed out" in photon mapping due to the high normal variation and area underestimation. Also note the slightly glossy icosahedron that does not record enough photon hits on its bottom to accurately reconstruct the glossy reflection towards camera. The second test scene, CORNELL BOX II, shows direct light, indirect and direct caustics rendered with 50,000 direct photons and 250,000 caustics photons, where the direct, caustics illumination is computed separately from 400, 200 K-NN photons respectively. The APPARTMENT scene exhibits strong indirect diffuse and glossy light that is difficult to compute using a direct visualization of photon maps. The CONFERENCE scene was rendered with 600 final gather rays per pixel for photon mapping as well as for ray splatting.

For a fair comparison we have used the same data structures and algorithms for sampling and searching for photon

mapping as for ray splatting. To achieve the same level of noise in a direct photon map visualization, we had to set the number of k-nearest neighbors (K-NN) from at least 400 up to 1,200 photons. In this setup K-NN search becomes inefficient as reflected in the results in Table 2.

The rendering time is sub-linear in the number of photons even though the photon rays are splatted sequentially. This is because the splat radius reduces automatically with increasing number of photons (Section 6) and photon rays are splatted coherently. Ray splatting can be faster than photon mapping if the number of photons is much smaller than the number of eye samples as common in final gathering (see Table 2). It becomes less efficient if the number of photons increases [HHS05].

Scene	Method	T_E	T_L	T_S	$\sum T$
CORNELL BOX I	PM (254/400 K-NN)	1.8	2.0	64.3	69.1
(98 % diffuse)	RS ($C = 1.0, S = 0.4$)	1.8	2.3	58.3	63.4
$N_{Prim} = 19635$	RC ($C = 0.6, S = 0.3, \varepsilon = 0.7$)	2.3	1.9	11.9	16.1
Scene settings	$500 \times 500 \times 4$; $M = 200,000$; $(0.3, 0.6, 0.1)$; $R = 0.2$				
CORNELL BOX II	PM (310/400, 206/250 K-NN)	2.2	2.9	87.1	92.2
(98 % diffuse)	RS ($C = 1.1, S = 0.5$)	2.2	3.5	47.1	52.8
$N_{Prim} = 2265$					
Scene settings	$500 \times 500 \times 4$; $M = 300,000$; $(0.2, 0.0, 0.8)$; $R = 0.2$				
SIBENIK	PM (342/500 K-NN)	1.2	3.8	26.2	31.2
(99 % diffuse)	RS ($C = 0.9, S = 0.3$)	1.3	5.3	26.3	32.9
$N_{Prim} = 78362$	RC ($C = 0.5, S = 0.2, \varepsilon = 0.9$)	1.4	5.3	14.0	20.7
Scene settings	$500 \times 500 \times 1$; $M = 500,000$; $(0.0, 1.0, 0.0)$; $R = 0.2$				
APPARTMENT	PM (482/600 K-NN)	1.0	4.1	61.6	66.7
(47 % diffuse)	RS ($C = 0.9, S = 0.4$)	1.1	4.2	59.2	64.5
$N_{Prim} = 73668$	RC ($C = 0.5, S = 0.3, \varepsilon = 0.4$)	1.1	4.2	15.5	20.8
Scene settings	$500 \times 500 \times 1$; $M = 500,000$; $(0.0, 0.9, 0.1)$; $R = 0.2$				
CONFERENCE	PM (68/70 K-NN)	3.5	2.5	35.7	41.7
(86 % diffuse)	RS ($C = 0.6, S = 0.2$)	3.6	2.7	25.7	32.0
$N_{Prim} = 265880$	PM-FG (600 FGRs)	991	2.5	4114	5108
	RS-FG (600 FGRs, 5×5 tiles)	1109	2.6	2714	3825
Scene settings	$700 \times 700 \times 4$; $M = 160,000$; $(0.3, 0.6, 0.1)$; $R = 0.4$				

Table 2: *Computation times for the rendering phases of our algorithm using either a direct visualization of photon maps with K-NN density estimation (PM), ray splatting (RS), ray splatting with radiance caching (RC), or photon mapping/ray splatting with final gathering (PM-FG)/(RS-FG) respectively. The computed images are shown in Fig. 9. N_{Prim} is the number of geometric primitives, T_E, T_L, T_S, $\sum T$ are the times in seconds for eye pass, light pass, photon splatting, and total time respectively. For photon mapping T_S is the time for K-NN search and density estimation. The photon map parameters are (average/maximum K-NN global photons, average/maximum K-NN caustics photons). The scene settings are: image resolution times the number of samples per pixel; total number of stored photons (M); and the fraction of direct, indirect diffuse, caustics photons; and the bandwidth clamping parameter R.*

For 500×500 pixels and 500,000 photons the memory requirements for splatting and radiance map are about 100 to 160 MBytes independent of the scene complexity. Note that the memory requirements can be reduced significantly if photon rays are not explicitly stored during photon tracing but are progressively splatted to screen pixels. The ray splatting and search in a conical frustum (Section 7) is approximately 1.5 to 2 times slower than for photon splatting in a spherical footprint with the same precomputed radius (i. e. without K-NN search). This holds also for a larger number of eye samples, for example when final gathering is used storing several hundred eye samples per pixel.

In order to compute glossy light transport, we have implemented and tested two different approaches: the histogram splatting, and the splatting in the SH basis. For the histogram method we used 20 strata and for the SH method 16 coefficients per pixel, which yields similar results. The splatting to the histogram is more efficient. However, the final BRDF evaluation is more expensive since we apply BRDF sampling for all glossy eye samples.

The additional radiance caching scheme further reduces the rendering time by one order of magnitude but requires low-frequency illumination. For filtering purposes, the minimum number of cache records contributing to a pixel was set to at least 4 to 7. Refer to Fig. 8 as an example of radiance caching.

In Fig. 10 we compare our method quality-wise with photon mapping using k-nearest neighbor (K-NN) density estimation. The smoothness parameter C (see Section 6) was chosen to have on average a similar density-estimation bandwidth as in the photon map solution. The third column (c) shows the filtered radiance cache solution that is based on a noisy photon ray splatting input (see for example Fig. 8(a)). For all solutions we have used the same photon sampling algorithm with 500,000 stored photon samples in total.

10. Conclusions

We proposed an algorithm that improves photon density estimation. Our method solves some of the problems inherent to photon density estimation and brings the quality closer to the expensive final gathering approaches.

First, we eliminate boundary bias thanks to the volumetric search along photon paths. This is especially noticeable on small unconnected surfaces where all hit point density estimation techniques fail. Since we do this via splatting instead of gathering, we avoid the use of complex and memory demanding data structures as in [HBHS05]. Second, our method does not suffer from discontinuities in surface orientation. Since we estimate the density in photon ray space, we decouple the density estimation and bandwidth selection from the surface area and obtain the convolved irradiance in ray space. Therefore, we can compute illumination on surfaces of complex shapes where the actual surface area is too small or difficult to estimate.

Although our algorithm often yields satisfying results, it also has some limitations. Like in all density estimation methods there are occasionally problems with light leakage due to the neglected visibility in the splat footprint. Only low-frequency lighting can be reconstructed with our method, but final gathering can be performed as in [HHS05]. We are currently working on incorporating partial visibility information gathered during the photon ray traversal to the

ray splatting, which should further improve image quality. At present the method also requires tuning several parameters for the bandwidth selection: the smoothness coefficient C, bandwidth sensitivity S, and the maximum bandwidth deviation R, which is one parameter more than for photon mapping. Also the number for stored direct, indirect diffuse, and caustic photons as well as the parameters for the optional radiance caching and filtering must be set by the user.

We conclude that our algorithm has a potential in fast rendering of low-frequency illumination for previewing purposes in production rendering.

Acknowledgement

We would like to thank the anonymous reviewers and Jaroslav Křivánek for their comments. This work has been partially supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program MSM 6840770014 and LC-06008 (Center for Computer Graphics).

References

[BSC*03] BEKAERT P., SLUSALLEK P., COOLS R., HAVRAN V., SEIDEL H.-P.: *A custom designed density estimation method for light transport*. Research Report MPI-I-2003-4-004, MPI Informatik, September 2003.

[Dah04] DAHMEN T.: *Multi Resolution Ray Tracing of Point Data*. Master's thesis, Universität des Saarlandes, Computer Graphics Group, 2004.

[DLW93] DUTRÉ P., LAFORTUNE E., WILLEMS Y.: Monte Carlo light tracing with direct computation of pixel intensities. In *Proceedings of Compugraphics* (1993), pp. 128–137.

[DS06] DACHSBACHER C., STAMMINGER M.: Splatting indirect illumination. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2006), pp. 93 – 100.

[GKBP05] GAUTRON P., KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S.: Radiance cache splatting: A GPU-friendly global illumination algorithm. Eurographics Symposium on Rendering (2005), pp. 55–64.

[HBHS05] HAVRAN V., BITTNER J., HERZOG R., SEIDEL H.-P.: Ray maps for global illumination. In *Eurographics Symposium on Rendering* (2005), pp. 43–54.

[HHK*07] HERZOG R., HAVRAN V., KINUWAKI S., MYSZKOWSKI K., SEIDEL H.-P.: *Global Illumination using Photon Ray Splatting*. Research Report MPI-I-2007-4-003, MPI Informatik, May 2007.

[HHS05] HAVRAN V., HERZOG R., SEIDEL H.-P.: Fast final gathering via reverse photon mapping. In *Computer Graphics Forum* (2005), vol. 24, pp. 323–333.

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

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 49–56.

[KGPB05] KŘIVÁNEK J., GAUTRON P., PATTANAIK S., BOUATOUCH K.: Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 550 – 561.

[LC04] LARSEN B. D., CHRISTENSEN N. J.: Simulating photon mapping for real-time applications. In *15th Eurographics Symposium on Rendering* (2004), pp. 123–132.

[LP03] LAVIGNOTTE F., PAULIN M.: Scalable photon splatting for global illumination. In *GRAPHITE* (2003), ACM SIGGRAPH, pp. 1–11.

[LURM02] LASTRA M., URENA C., REVELLES J., MONTES R.: A particle-path based method for Monte Carlo density estimation. In *Poster Papers Proceeding of the 13th Eurographics Workshop on Rendering* (2002), pp. 33–40.

[Mac48] MACROBERT T.: *Spherical Harmonics; An Elementary Treatise on Harmonic Functions, with Applications*. Dover Publications, 1948.

[McC99] MCCOOL M. D.: Anisotropic diffusion for Monte Carlo noise reduction. *ACM Transactions on Graphics 18*, 2 (1999), 171–194.

[RH04] RAMAMOORTHI R., HANRAHAN P.: A signal-processing framework for reflection. *ACM Transactions on Graphics 23*, 4 (2004), 1004–1042.

[Sch03] SCHREGLE R.: Bias compensation for photon maps. *Computer Graphics Forum 22*, 4 (2003), 729–742.

[Sil85] SILVERMAN B.: *Density Estimation for Statistics and Data Analysis*. Chapmann and Hall, London, 1985.

[SKALP05] SZIRMAY-KALOS L., ASZÓDI B., LAZÁNYI I., PREMECZ M.: Approximate ray-tracing on the GPU with distance impostors. *Computer Graphics Forum 24*, 3 (Sept. 2005), 695–704.

[SKP06] SHAH M. A., KONTTINEN J., PATTANAIK S. N.: Caustics mapping: An image-space technique for real-time. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2006), 272–280.

[Stü98] STÜRZLINGER W.: Calculating global illumination for glossy surfaces. *Computers & Graphics 22*, 2-3 (1998), 175–180.

[SW00] SUYKENS F., WILLEMS Y. D.: Adaptive filtering of progressive Monte Carlo image rendering. In *Proceedings of WSCG* (2000).

[SW01] SUYKENS F., WILLEMS Y. D.: Path differentials and applications. In *Eurographics Workshop on Rendering* (2001), pp. 257–268.

[WD06] WYMAN C., DAVIS S.: Interactive image-space techniques for approximating caustics. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games* (2006), pp. 153–160.

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive global illumination using fast ray tracing. In *Eurographics Workshop on Rendering* (2002), pp. 9 – 19.

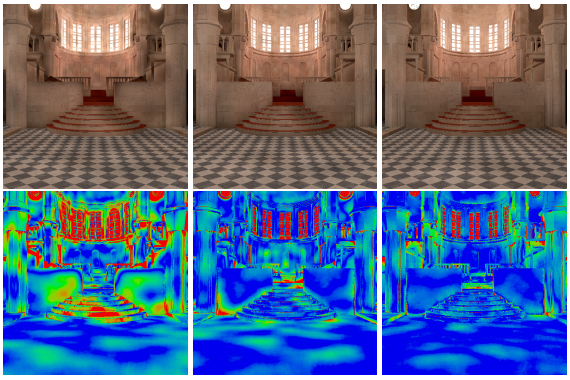
[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. Proceedings of SIGGRAPH (1988), pp. 85–92.

(a) (b) (c) (d) (FG)

Figure 9: *Comparing photon mapping using k-NN density estimation (1. row) with our method (2. row) in 4 scenes with different illumination conditions (the rendering times are given in Table 2). The number of k-NN photons was chosen to have on average a gather radius similar to the splat radius used for ray splatting. Column 1 to 3 show a direct visualization of the density estimation, while column 4 is computed with final gathering (FG) using 600 rays per pixel. From left to right column: (a) diffuse/glossy* CORNELL BOX I *with full global illumination, (b)* CORNELL BOX II *with direct light and indirect/direct caustics (c) the* APPARTMENT *scene* © *INRIA 2005, which exhibits indirect diffuse and glossy light transport, and (d) the indirect diffuse* CONFERENCE *scene.*

(a) (b)

Figure 8: *Results of photon ray splatting (direct and indirect light) with radiance caching in spherical harmonics basis for the* APPARTMENT *scene, (a) shows the noisy direct visualization resulting from a small splat size on the right half, which is given as initial input to the filtered radiance cache solution shown on its left half, (b) the radiance cache splats after first pass (for visualization purposes a smaller cache error was chosen). The time for computing photon ray splatting and cache splatting in 3 passes took 15 seconds for* $500,000$ *photons and* 500×500 *pixels.*



(a) 29 s (b) 35 s (c) 20 s

Figure 10: *Comparing ray splatting with photon maps K-NN density estimation relative to a reference solution (1200 final gather rays per pixel) for indirect light in the* SIBENIK *scene. 1. row: (a) direct visualization of photon maps with 500 nearest neighbors, (b) our ray splatting in spherical harmonics basis, (c) with additional radiance caching and filtering. 2. row: the relative error color-coded from blue (*$<$ *5% error) to green (*15% *error) to red (*$\geq$ 30% *error) with respect to the reference image. Note the reduced bias near boundaries and on curved surfaces for the ray splatting. Also the filtering due to radiance caching (c) can further reduce low-frequency noise leading to better visual quality.*