EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Anisotropic Radiance-Cache Splatting for Efficiently Computing High-Quality Global Illumination with Lightcuts

Robert Herzog[1], Karol Myszkowski[1], Hans-Peter Seidel[1]

[1]MPI Informatik, Computer Graphics Group, Saarbrücken, Germany

Abstract

Computing global illumination in complex scenes is even with todays computational power a demanding task. In this work we propose a novel irradiance caching scheme that combines the advantages of two state-of-the-art algorithms for high-quality global illumination rendering: lightcuts*, an adaptive and hierarchical instant-radiosity based algorithm and the widely used (ir)radiance caching algorithm for sparse sampling and interpolation of (ir)radiance in object space. Our adaptive radiance caching algorithm is based on anisotropic cache splatting, which adapts the cache footprints not only to the magnitude of the illumination gradient computed with lightcuts but also to its orientation allowing larger interpolation errors along the direction of coherent illumination while reducing the error along the illumination gradient. Since lightcuts computes the direct and indirect lighting seamlessly, we use a two-layer radiance cache, to store and control the interpolation of direct and indirect lighting individually with different error criteria. In multiple iterations our method detects cache interpolation errors above the visibility threshold of a pixel and reduces the anisotropic cache footprints accordingly. We achieve significantly better image quality while also speeding up the computation costs by one to two orders of magnitude with respect to the well-known photon mapping with (ir)radiance caching procedure.*

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

1. Introduction

A large amount of work has been devoted to the problem of computing the global illumination in synthetic 3D scenes. Unbiased algorithms converge very slowly, exhibit high-frequency noise and are therefore rarely used in industry. Biased techniques on the other hand are orders of magnitude more efficient and can often produce images hardly distinguishable from unbiased results. A common biased approach is to exploit the coherence in the lighting by caching the flux distribution in the scene by means of photon hits and reusing it for all pixels. Two well-known method of this variant are instant radiosity [Kel97] and photon mapping [Jen01]. Since computing the illumination on a pixel basis is still too costly in particular for indirect lighting, a common approach to speed up the computation of indirect light is to use irradiance caching [WRC88, TL04, KGPB05]. Irradiance caching is a well-established algorithm widely used in production rendering and lighting simulation software. The basic idea is to exploit the "smoothness" of the indirect light by sparse sampling and interpolating the irradiance in object space and thus reducing the intensive computation for the indirect light to only a few thousand point samples as opposed to millions. However, pure irradiance caching works only for diffuse surfaces. An extension to radiance caching on moderately glossy surfaces using (hemi-)spherical harmonics has been published in [KGPB05] and [KBPv06], which also proposes an adaptive, perceptually-inspired sampling scheme and an improved gradient-based interpolation of the cache samples. Nevertheless, those caching algorithms are difficult to control and producing artifact-free images often requires initial test-runs and user experience for setting the appropriate parameters. Little work has been devoted to the topic of how to adapt the computation of the cache records to the local lighting and scene complexity and also on how to reduce the space of user parameters.

In this work, we propose a method that aims in this direction. Our radiance and irradiance caching algorithm is not only one to two orders of magnitude more efficient than tra-

ditional irradiance caching but also more robust as it reduces the burden of choosing the "right" rendering parameters by the user. The performance gain is two-fold. First, we obtain a speedup by reversing the caching procedure and second, we use the lightcuts algorithm [WFA*05,WABG06] extended by our irradiance gradients [WH92] for more efficient computation of the cache records. The traditional gathering scheme for irradiance caching [WRC88], where a search for feasible cache records is initiated at each pixel sample, imposes restrictions on the traversal order of the pixels. It has been proposed at a time when main memory was scarce and is not appropriate nowadays. It also does not scale well with high-resolution images. We perform cache splatting in 3D object space, where each newly computed cache record directly extrapolates its (ir)radiance contribution to all neighboring eye samples in its ellipsoidal splatting footprint.

Using the lightcuts technique we are able to compute indirect and direct light in one pass without the need for user parameter settings like number of final gather rays (FGR) per pixel, k-nearest neighbor photons etc., as in the traditional photon mapping with (ir)radiance caching procedure. Lightcuts automatically adapts to the scene and lighting complexity and furthermore, when using our perceptually-derived visibility thresholds, to the perceptual cues in the image, which often counterbalances scene complexity (rendering errors in a complex scene with many geometric details and textures are less perceptible by the human eye than in simple scenes.) The latter case is particularly important as it allows to relax the rendering errors and reduce computation time in cluttered image regions.

In order to compute direct and indirect light faithfully using our caching technique, we maintain a "two-layer" irradiance cache, which caches and interpolates direct and indirect light contributions in parallel. However, both cache layers interact with each other on-the-fly in terms of luminance masking to steer the cache interpolation and error thresholds in the lightcuts computation, which is impossible if we compute direct and indirect lighting separately.

Finally, we improve the (ir)radiance interpolation by using anisotropic cache splatting, which reduces the overall number of cache records per frame while maintaining the same image quality.

2. Previous Work

The method proposed in this paper is most related to the irradiance [WRC88] or radiance caching [KGPB05] and the lightcuts algorithm [WFA*05].

Irradiance caching was introduced by [WRC88] to make global illumination practical and has been widely used ever since. In [WH92] the authors improved the cache interpolation by using irradiance gradients. The original irradiance cache sampling assumes purely diffuse environments when determining an upper bound for the cache sampling density. Smyk and Myszkowski [SM02] also exploits the irradiance gradients to increase the cache density where the actual gradient magnitude is larger. Tabellion et al. [TL04] propose several small practical modifications of the original irradiance caching scheme [WRC88], which we include into our framework. Křivánek et al. [KGPB05] generalized irradiance caching to radiance caching on moderately glossy surfaces. Essentially, the major difference to irradiance caching is that the full incident radiance field is captured and interpolated in the spherical or hemispherical harmonics basis [**?**] rather than just the directional independent irradiance. Since radiance caching is more sensitive to interpolation errors, the authors also propose an adaptive caching scheme [KBPv06] where in multiple iterations visible errors in the cache interpolation are removed by increasing the cache density where discontinuities are detected. We extend the adaptive caching by anisotropic cache interpolation.

The reconstruction quality of all those caching algorithms strongly depend on the pixel traversal order. In [GKBP05] a GPU-friendly method was presented for the widely used (ir)radiance caching algorithm that overcomes this limitation. The approach reverses the caching procedure by splatting cache samples to the image plane. For each cache sample a bounding quad is rasterized in the image plane that encloses all pixels in the footprint of the projected maximum sphere of influence of a cache sample. However, the method relies solely on screen space projection of cache samples and cannot naturally support splatting of indirect caches due to specular light transport from the eye. In our approach we perform "cache splatting" in 3D object space instead.

All discussed (ir)radiance caching techniques use Monte Carlo integration and in particular the photon mapping algorithm [Jen01] to compute the incident radiance field or irradiance at cache samples. We propose to compute the cache samples using a deterministic and hierarchical instant-radiosity based algorithm [Kel97] called *lightcuts* [WFA*05, WABG06]. Lightcuts computes the pixel radiance by adaptively choosing a subset of "important" virtual point light sources (VPLs) for each pixel sample. To save computation in image regions with coherent lighting, the authors also propose an adaptive pixel-radiance caching scheme called reconstruction cuts [WFA*05]. Since reconstruction cuts operate in image space the interpolation of "cached pixels" is sensitive to object silhouettes and high-frequencies in the BRDF and textures, so that the initial sampling needs to be very dense (4×4 pixels). To our knowledge the combination of the lightcuts technique with the well-established (ir)radiance caching algorithm has not been considered yet.

3. Lightcut Computation

Lightcuts is a deterministic and hierarchical algorithm based on instant radiosity [Kel97]. It is one to two orders of magnitude more efficient than photon mapping because it neither requires a costly search for the nearest photon(s) nor does it

involve stochastic ray shooting for Monte Carlo final gathering. We omit a detailed introduction to the lightcuts computation here and refer the reader to [WFA*05, HKMS08]. Lightcuts are computed as in [WFA*05] except that we also compute gradients (see Section 4.1) for each cache record. Initially, the light distribution in a scene is hierarchically stored in a binary tree where each node clusters the intensity of all lights in its sub-tree. During rendering the light-tree is adaptively traversed in a top-down fashion and at each visited cluster node a conservative upper error bound is computed and the cluster is stored in a priority queue (the light-cut), which sorts the cluster according to its error. In each iteration we take the cluster with the maximum error from the queue and refine it by computing the radiance contributions and the error bounds of its children. The process repeats until all nodes in the priority queue have an error below a threshold ε_L relative to the pixel luminance or the size of lightcut N_L exceeds a maximum threshold. For our cache computation we set $\varepsilon_L = 1\%$ and $N_L = 1000$. Computing the upper error bounds for all traversed nodes is costly. We therefore refine a cluster immediately (add it to the top of the priority queue) if its energy contribution to the pixel luminance is above a threshold (we use 10%). This helps to skip needless computations of the upper error bounds for clusters in the upper part of the light-tree, which are later refined anyway.

Since the lightcuts algorithm scales well with the number of virtual point lights (VPLs), we can afford a huge number of VPLs (e.g. $N > 10^6$). Therefore, in contrast to instant radiosity [Kel97], the clamping of the energy contribution of close VPLs is much less objectionable in lightcuts. Such clamping also underestimates gradients near corners, which is undesirable as we adapt our cache record density and the cache splatting footprint to the gradient. Nevertheless, we clamp the maximum contribution of a VPL in the computed lightcut at 1% of the pixel's luminance value to suppress the otherwise visible noise in corners. The same applies also to the gradients of the VPL. This clamping introduces bias (missing energy) but a remedy is proposed in Section 6.3.

4. Radiance Caching and Radiance Gradients

The original lightcuts algorithm estimates the rendering equation for each pixel sample or adaptively for at least every fourth pixel where in-between pixel samples are interpolated in image space (reconstruction cuts [WFA*05]). Our method extends lightcuts with (ir)radiance caching in object space [WRC88, KGPB05]. As in [KGPB05] the incident radiance and BRDF/material term (M) are separately projected onto the hemispherical harmonics (HSH) basis $H_l^m(\theta,\phi)$ for the ease of interpolation on glossy surfaces. The radiance HSH coefficients λ_l^m capture the hemispherical radiance field for each color channel and are computed with lightcuts:

$$\lambda_l^m(\mathbf{x}) = \sum_{i=1}^{N_L} G(\mathbf{x},\mathbf{y}_i)\cdot V(\mathbf{x},\mathbf{y}_i)\cdot I_i \cdot H_l^m(\theta_i,\phi_i), \qquad (1)$$

where m is the degree and l is the band of the HSH coefficient, N_L is the size of the lightcut, $G(\mathbf{x},\mathbf{y})$ is the geometric term, $V(\mathbf{x},\mathbf{y})$ the binary visibility function, and I_i the intensity of the cluster i in the lightcut. Similarly, the BRDF is projected onto the HSH basis:

$$f_l^m(\mathbf{x},\omega_o) = \frac{2\pi}{N\cdot M}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1} M(\mathbf{x},\omega_o,\omega_{j,k})\cdot H_l^m(\theta_{j,k},\phi_{j,k}), \qquad (2)$$

where $f_l^m(\mathbf{x},\omega_o)$ are the precomputed BRDF HSH coefficients for a number of discrete outgoing directions ω_o for each BRDF in the scene model [KGPB05]. Since the HSH basis is orthonormal the outgoing radiance at any point on a surface is computed as the dot product of the radiance HSH coefficient vector $\Lambda(\mathbf{x})$ and the BRDF HSH coefficient vector $F(\mathbf{x},\omega_o)$.

$$L(\mathbf{x},\omega_o) = \sum_{l=0}^{n-1}\sum_{m=-l}^{l}\lambda_l^m(\mathbf{x})\cdot f_l^m(\mathbf{x},\omega_o), \qquad (3)$$

$$= \Lambda(\mathbf{x})\bullet F(\mathbf{x},\omega_o), \qquad (4)$$

where n is the number of bands (we use up to 8) depending on the "glossiness" of the surface [KGPB05].

With (ir)radiance caching, one cache sample contributes to many neighboring eye samples. Thus the pixel radiance L at a point $\mathbf{p}$ becomes the weighted average of the radiance HSH coefficients λ_l^m from the K neighboring cache samples modulated by the BRDF HSH coefficients at the point $\mathbf{p}$

$$L(\mathbf{p},\omega_o) \approx \sum_{k=1}^{K}\left(w_k(\mathbf{p})\cdot\tilde{\Lambda}_k(\mathbf{p})\right)\bullet F(\mathbf{p},\omega_o), \qquad (5)$$

with $\sum_{k=1}^{K} w_k(\mathbf{p}) = 1$. The extrapolated radiance HSH coefficients of a cache sample at $\mathbf{x}$ to a point $\mathbf{p}$ are computed as:

$$\tilde{\Lambda}(\mathbf{p}) = \Lambda(\mathbf{x}) + d_u\frac{\partial\Lambda(\mathbf{x})}{\partial u} + d_v\frac{\partial\Lambda(\mathbf{x})}{\partial v}, \qquad (6)$$

where $\frac{\partial\Lambda(\mathbf{x})}{\partial u}$ and $\frac{\partial\Lambda(\mathbf{x})}{\partial v}$ are the computed gradients in the local tangent plane of the cache sample and $d_u = p_u - x_u$, $d_v = p_v - x_v$ is the difference vector projected into the local coordinate system $(\vec{\mathbf{u}},\vec{\mathbf{v}},\vec{\mathbf{n}})$ at $\mathbf{x}$. As in [KGPB05] the projection along the surface normal direction $\vec{\mathbf{n}}$ is ignored.

4.1. Gradient Computation

Radiance caching without gradients can result in highly visible artifacts (see Fig. 2). In contrast to final gathering with integration over solid angle, lightcuts solves the integral by sampling the scene's surface-area. Therefore, a stratified gradient computation scheme as proposed by [WH92, KGPB05] is difficult to achieve. We therefore focus on computing the radiance gradients by differentiating Eq. (1)

$$\frac{\partial\lambda_l^m}{\partial u}(\mathbf{x}) = \sum_{i=1}^{N_L} I_i\cdot\left[\frac{\partial}{\partial u}\{G(\mathbf{x},\mathbf{y}_i)\cdot V(\mathbf{x},\mathbf{y}_i)\}\cdot H_l^m(\theta_i,\phi_i) + G(\mathbf{x},\mathbf{y}_i)\cdot V(\mathbf{x},\mathbf{y}_i)\cdot\frac{\partial H_l^m}{\partial u}(\theta_i,\phi_i)\right], \qquad (7)$$

and in the same way for the gradient along direction $\vec{\mathbf{v}}$. For simplicity we only regard a single VPL with index i and a single HSH coefficient (we omit the indices of the HSH coefficients) in the equations below.

The gradient of the geometric term G expressed in the local coordinate frame $(\vec{\mathbf{u}},\vec{\mathbf{v}},\vec{\mathbf{n}})$ at $\mathbf{x}$ can be rearranged as follows [KGPB05] taking into account that the change of G with the displacement of the $\mathbf{x}$ is opposite to its change with the displacement of $\mathbf{y}$ (see Fig. 1):

$$\begin{aligned}\frac{\partial G}{\partial u}(\mathbf{x},\mathbf{y}) &= -\frac{\partial G}{\partial q_u}(\mathbf{x},\mathbf{y}) \\ &= -\frac{\partial}{\partial q_u}\left\{\frac{\cos\theta_x \cos\theta_y}{||\vec{\mathbf{q}}||^2}\right\}, \end{aligned} \tag{8}$$

where $\cos\theta_x = q_n/||\vec{\mathbf{q}}||$, and $\cos\theta_y = -(\vec{\mathbf{n}}_y \bullet \vec{\mathbf{q}})/||\vec{\mathbf{q}}||$ with the surface normal $\vec{\mathbf{n}}_y$ at $\mathbf{y}$ and $\vec{\mathbf{q}} = \mathbf{y} - \mathbf{x}$ expressed in the local coordinate frame of $\mathbf{x}$ (see Fig. 1).

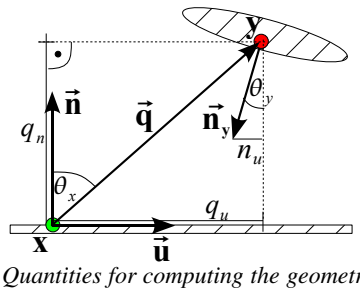

Figure 1: *Quantities for computing the geometric gradient at $\mathbf{x}$ for a VPL at point $\mathbf{y}$.*

Hence the gradient of G is computed as

$$\begin{aligned}\frac{\partial G}{\partial u}(\mathbf{x},\mathbf{y}) &= -\frac{\partial}{\partial q_u}\left\{\frac{-(\vec{\mathbf{n}}_y \bullet \vec{\mathbf{q}})\cdot q_n}{||\vec{\mathbf{q}}||^4}\right\} = q_n \cdot \frac{\partial}{\partial q_u}\left\{\frac{\vec{\mathbf{n}}_y \bullet \vec{\mathbf{q}}}{||\vec{\mathbf{q}}||^4}\right\} \\ &= q_n \cdot \left(\frac{n_u \cdot ||\vec{\mathbf{q}}|| + 4q_u \cdot \cos\theta_y}{||\vec{\mathbf{q}}||^5}\right). \end{aligned} \tag{9}$$

And similarly for tangent vector $\mathbf{v}$:

$$\frac{\partial G}{\partial v}(\mathbf{x},\mathbf{y}) = q_n \cdot \left(\frac{n_v \cdot ||\vec{\mathbf{q}}|| + 4q_v \cdot \cos\theta_y}{||\vec{\mathbf{q}}||^5}\right). \tag{10}$$

The derivatives of the HSH basis functions are given in [Kŏŏ5].

The gradients along $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ are computed during the lightcut computation. Whenever we evaluate the contribution of a visible light cluster, we also compute and store its gradient contribution in the cut according to Eq. (10), however, without multiplying the cluster intensity since this is refined during the cut computation. Once a cut is fully converged we loop over all cut entries and sum the gradients multiplied by the final (possibly clampled) cluster intensity of each entry. The geometric gradient computation requires clamping of the distance $||\vec{\mathbf{q}}||$ to prevent too small values for $||\vec{\mathbf{q}}||^5$ in the denominator of Eq. (10), which can lead to large numerical errors in the gradient computation particularly near corners.

The gradient computation only slightly influences the performance of the whole lightcut computation but significantly improves the cache interpolation quality (see Fig. 2). This is particularly important for the adaptive caching (Section 5.2) because interpolation without gradients creates many discontinuities resulting in an excessive cache density on flat, unoccluded surfaces.

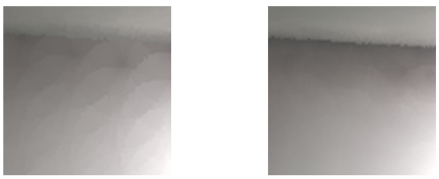

Figure 2: *Close-up of cache extrapolation without gradients (left), and with our gradients (right).*

5. Efficient Radiance Cache Splatting

In [GKBP05] an image space variant designed for the GPU of the widely used irradiance caching was presented. Besides being more suitable for a GPU implementation, it also bears several advantages, which we will discuss here. However, since the method operates in image space, it does not support splatting of indirect cache records arising from specular light transport along the eye path.

Therefore, we propose to splat cache records in object space independent of the camera. We do this via searching in a kd-tree for all feasible eye samples in the bounding sphere of the cache record. As the number of those searches in the eye sample kd-tree is relatively small compared to the number of eye samples, the search is algorithmically superior [HHS05] to the traditional irradiance caching [WRC88] where the number of searches (e.g. in an octree) is equivalent to the number of pixels in the image. The number pixels is usually about one to two orders of magnitude greater than the number of cache records. Thus, cache splatting [GKBP05] is less dependent on image resolution than cache gathering. Consequently, the search is not the bottleneck in the whole cache splatting algorithm and we splat each cache record one after another. This way the order of the cache-record generation and the splatting is *cache-oblivious* since the eye samples casted from the camera are spatially sorted in a kd-tree such that during splatting successive cache records access the memory in a highly coherent manner.

5.1. Anisotropic Cache Splatting

In [KBPv06] a method for adaptive irradiance caching was presented. It proposes to use individual error thresholds a per cache rather than a global error ε as in [WRC88]. This has the advantage that cache records can adaptively reduce their footprint where discontinuities in the cache interpolation are detected. Through multiple iterations eye samples are

tested for visible discontinuities in the illumination extrapolated from neighboring cache records. Discontinuity causing cache-records are excluded from the interpolation and the cache record's error threshold a and hence its footprint is reduced accordingly until the solution converges (i.e. no more visible discontinuities). We follow this approach and omit the details here as they are well explained in [KBPv06].

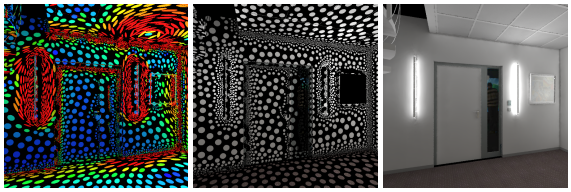

Figure 3: *Left: color-encoded anisotropic cache-record footprints in the* ICIDO *scene (red corresponds to maximum and blue to minimum anisotropy) (20881 records), middle: visualization of standard isotropic irradiance caching (23155 records), right: resulting indirect illumination ($\gamma = 2.5$, 8 f-Stops brighter), which is visually indistinguishable for both methods, although the numerical error is slightly smaller for the anisotropic reconstruction.*

We extend this method using anisotropic cache splatting. The adaptive caching of Křivánek et al. [KBPv06] increases the cache record density in regions with lighting changes, which the irradiance gradient can't compensate for. However, such strong lighting features are often of one-dimensional nature, which means that the changes in the direction orthogonal to the lighting gradient (the coherence direction) are often much smaller. This suggests to filter stronger in the coherence direction of a 2D signal and reduce interpolation along the gradient to avoid excessive "blurring" of lighting features. Such approach reduces the cache record density along edge features in the illumination (see Fig. 3). Anisotropic splatting and interpolation is not new but has been used frequently for example in image processing [Wei98, McC99] and point based rendering [ZPvBG01]. The proposed cache splatting imposes only small changes to the adaptive caching. Additionally, for each cache record we need the anisotropy and the direction of the gradient to perform anisotropic cache splatting.

The weights $w_k(\mathbf{p})$ are then computed similarly to traditional irradiance caching [TL04, WRC88]:

$$w_k(\mathbf{p}) = \left(\frac{||\mathbf{D} * (\mathbf{p} - \mathbf{x}_k)||}{max(R_-, min(R_+, R_k))} + \sqrt{1 - \vec{\mathbf{n}}_p \bullet \vec{\mathbf{n}}_k} \right)^{-1}, \tag{11}$$

where $\vec{\mathbf{n}}_p$ and $\vec{\mathbf{n}}_k$ are the surface normals at point $\mathbf{p}$ and cache position $\mathbf{x}_k$, respectively. Setting R_k to the harmonic mean distance of all visible surfaces [WRC88] computed with lightcuts does not yield a correct upper bound estimate for the indirect light gradient [WRC88] as we do not sample the hemisphere of directions uniformly (we essentially perform surface area integration). Therefore, we follow [TL04] and set R_k to the minimum distance instead. We use the distance to the nearest occluder computed from all traced shadow rays to the VPLs in the lightcut, which is clamped at a lower R_- and upper bound R_+. The bounds R_- and R_+ are computed from the projected pixel area at $\mathbf{x}_k$ [TL04], which depends on the distance of $\mathbf{x}_k$ to the camera.

Since lightcuts is a deterministic approach, which samples all "contributing" light-emitting surfaces, we do not suffer from under-sampling of small nearby emitters (provided that small surfaces were sampled with VPLs) or ray leaking [KBPv06] as in Monte Carlo final gathering. Nevertheless, additional nearest neighbor clamping [KBPv06] should be applied since R_k is chosen as the distance to the nearest occluder rather than the distance to the nearest point light. And a small occluder, in particular when seen at a grazing angle, might not always be intersected by a shadow ray. Choosing the distance to the nearest occluder hardly changes R_k for the indirect light caches but is particularly important for direct light caches where we would like to sample denser in shadowed regions (see Section 5.3) in order to avoid under-sampling of the visibility function. Although such approach does not avoid light bleeding into the shadowed region, it can be detected later by the discontinuity checking of overlapping cache records (see Section 5.2).

The 3×3 matrix $\mathbf{D} = \mathbf{G} * \mathbf{A}^T$ transforms $(\mathbf{p} - \mathbf{x}_k)$ into the gradient-spanned coordinate system given by the orthogonal matrix $\mathbf{A} = \{v_{||}, v_\perp, n_k\}$, where $v_{||}$ is the normalized gradient (in the local tangent plane) and $v_\perp$ is the coherence direction orthogonal to $v_{||}$, and then scales it by the diagonal matrix

$$\mathbf{G} = \begin{pmatrix} g(||\nabla(x_k)||^2)^{-1} & 0 & 0 \\ 0 & g(||\nabla(x_k)||^2) & 0 \\ 0 & 0 & 1/c_n \end{pmatrix}.$$

The matrix $\mathbf{G}$ suppresses the spherical splatting footprint along the illumination-gradient direction (see Fig. 4) and along the surface normal direction since those directions are likely to incur larger errors due to discontinuities. $c_n < 1$ is a constant scaling factor, which basically flattens the ellipsoidal splatting footprint in surface normal direction. We set it to 0.2. Note that we do not change the area of a cache footprint in the tangent plane but just change its shape and orientation (see Fig. 4).

The monotonically decreasing function $g(x^2)$ maps the squared gradient magnitude to a value between 0 and 1 determining the anisotropy. We have chosen g as the Perona-Malik diffusivity function [PM90, Wei98] since it is efficiently computed as

$$g(x^2) = \frac{1}{1 + x^2/\mu^2}, \tag{12}$$

with user defined contrast parameter μ: the smaller μ the higher is the sensitivity to the gradient magnitude yielding in more anisotropy along coherent lighting features. In order to prevent too elongated, slim footprints, we clamp $g(x^2)$ at a minimum of 0.3. One has to be aware that this diffusivity

function was developed for image processing and considers absolute values of image-pixel gradients. In our case the illumination gradients depend on the scene measurement unit (e.g. inches) and lighting. Therefore, μ should be normalized to correspond to the actually perceived gradients after scaling the computed image to the desired image brightness (tonemapping).

5.2. Multi-Pass Adaptive Caching

In the final pass after all cache records have been created and splatted their contribution to all eye samples, we check at each eye sample e for discontinuities in the illumination as in [KBPv06] and reduce the footprint of the discontinuity-causing cache record such that the eye sample at which the discontinuity was detected is just excluded. Instead of resizing the footprint by reducing just the error a of a cache record (Fig. 4a), we reduce the footprint only in one dimension, either along the gradient $v_{||}$ or along the coherence direction $v_{\perp}$ while still keeping the symmetry and orientation of the footprint. We choose the dimension that maximizes the resulting footprint area when just excluding the eye sample e and recompute the resulting anisotropy and the new cache error a of the record (Fig. 4b). Note that this approach also compensates for wrongly oriented cache records due to the missing visibility gradients. One major difference to the *discernability metric* used to detect discontinuities in [KBPv06] is that our cache records are computed deterministically via lightcuts as opposed to Monte Carlo final gathering. Therefore, high-frequency noise is not present in the computed cache records and we do not need to estimate the standard deviation in the illumination to compensate for noise [KBPv06].

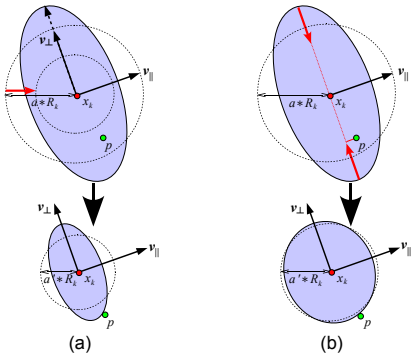

Figure 4: *(a) excluding the discontinuous eye sample at p from the cache footprint by reducing the cache error a while keeping the same anisotropy, (b) by shrinking the footprint only in the dimension with maximum elliptical distance to x and recomputing cache error a and the new anisotropy.*

We tolerate discontinuities as long as they are invisible to a standard human observer, i.e. the lighting discontinuities are below the visibility thresholds [†]

[†] A contribution of a cache record to an eye sample is said to be continuous if the interpolated pixel radiance including this cache record is indistinguishable from the pixel radiance without the cache record. This requires that we have at least two cache contributions per eye sample.

5.3. Two-level Radiance Caching

Irradiance caching has been designed for indirect lighting and the direct light is commonly computed using a different method. However, the advantage of lightcuts is that we can seamlessly compute the direct and indirect lighting. When computing the direct and indirect light one after another in different rendering passes, we cannot obtain the full lighting information for a pixel. Since lightcuts adapts to the pixel luminance (Weber law), such an approach would result in larger cut sizes and therefore higher computation costs for the first pass rendering. This is particularly problematic when computing the direct light in fully occluded regions where only indirect light is contributing since lightcuts will refine the cut to the maximum size. Therefore, we compute direct and indirect light in parallel with our (ir)radiance caching such that the indirect light can mask the direct light and vice versa. We maintain two cache layers and two light trees, one for direct light and one for indirect light. The main rendering loop for the cache splatting described in Section 5 needs only small modifications to test both cache layers and to weight the direct and indirect light contributions of cache records individually. A rendering pass is finished if the accumulated weights of both cache layers in each pixel are above the required user-thresholds, which can be set individually for direct and indirect light.

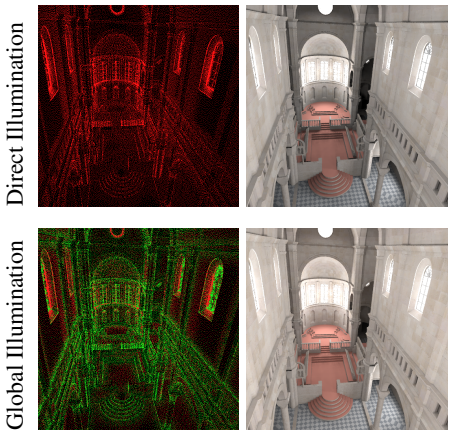

Figure 5: *left: distribution of the cache records in the* SIBENIK *scene, red dots represent direct light and green dots indirect light caches, right: rendered results.*

6. Implementation Details

6.1. Light-tree Construction

A light-tree is initially constructed over the sampled VPLs in a greedy bottom-up fashion as in [WFA*05]. We also

tested a simple recursive top-down clustering, which can be very efficiently computed (<1 sec for 500,000 VPLs). However, it results in poorer performance during rendering due to larger lightcut sizes arising from less precise error bounds. The naive greedy bottom-up construction has $\mathcal{O}(N^3)$ time complexity and may take a few hours for clustering half a million VPLs. To achieve a tree construction time-complexity of $\mathcal{O}(N \log N)$ is non-trivial and requires several optimizations, which are analyzed and well-explained in [WBKP08]. The clustering is conservative in a greedy sense and is computed in 5 to 30 seconds for 500,000 VPLs depending on the scene.

6.2. VPL Visibility Computation

Our cache computation and thus also the lightcut computation is very coherent since we iterate over the spatially sorted eye samples rather than over pixels in the image. This is particularly important as we access a large amount of data (light-tree and VPLs, eye samples and cache records, the frame buffer and cache-splat buffer), which might not always fit into the CPU caches. Furthermore, we can also exploit this coherence during ray-shadow testing of clusters using a shadow cache since neighboring cache records evaluate similar light clusters. Mapping the index of the cluster's representative VPL to a hash index, we can cache the last occluder of a VPL in a small hash-table. We set the size of the hash-table to a power of two (e.g. 4096) in order to use a fast modulo hash-function via bit masking. Each successive shadow test for this VPL first checks the last occluder stored in the hash-table. This results in more than 20% savings of expensive shadow ray-traversals in the raytracing acceleration data structure for all tested scenes.

6.3. Dealing with Weak Singularities

The lightcuts algorithm based on instant radiosity estimates the rendering equation by integrating the radiant intensity over the surface area using point samples (VPLs). This naturally leads to the problem of weak singularities, which results in low-frequency noise in corners seen as "blotchy" artifacts. Although lightcuts is very scalable – we can use hundreds of thousands to millions of VPLs – it still requires clamping of very close VPLs resulting in darkening in corners. As in [KK04] we can compensate for the clamping of the VPL contributions by using a second estimator that integrates over the solid angle. The luminance contribution Y_c of a VPL is clamped at 2% of the cache's luminance $Y_{E(\mathbf{x})}$:

$$Y_c = \min\left\{0.02 \cdot Y_{E(\mathbf{x})}, G(\mathbf{x},\mathbf{y}) \cdot Y_{I(\mathbf{y})}\right\} \qquad (13)$$

Assuming that all VPLs have a similar intensity $I(\mathbf{y})$ we can compute the upper bound $b(\mathbf{x})$ for the geometric term $G(\mathbf{x},\mathbf{y})$ at each point $\mathbf{x}$ as $b(\mathbf{x}) = \frac{0.02}{\bar{Y}_I} \cdot Y_{E(\mathbf{x})}$, where $\bar{Y}_I$ is the mean luminance of the VPL intensity. The cache irradiance is then computed as follows:

$$\begin{aligned} E(\mathbf{x},\omega_o) &= E_{cut} + E_{clamp} \\ &= \int_A \min\{b(\mathbf{x}), V(\mathbf{x},\mathbf{y})G(\mathbf{x},\mathbf{y})\} L_e(\mathbf{y},\omega(\mathbf{x},\mathbf{y}))d\mathbf{y} \\ &+ \int_\Omega \frac{\max\{0, G(\mathbf{x},\mathbf{y}) - b(\mathbf{x})\}}{G(\mathbf{x},\mathbf{y})} L(\mathbf{x},\omega)\cos\theta_x d\omega, \end{aligned}$$

where $L(\mathbf{x},\omega)$ is the incident radiance from direction ω. The irradiance contribution E_{clamp} from the second integral compensates for the clamped VPL contributions in the first integral, which we estimated by lightcuts. E_{clamp} is solved via Monte Carlo final gathering with uniform sampling of the hemisphere of directions Ω. $L(x,\omega)$ is computed by querying the radiance cache. If no feasible cache record is found, we use lightcuts however with a larger error threshold ($\varepsilon_L = 5\%$).

Figure 6: *Left: clamped indirect lighting in the* SIBENIK *scene using* 500,000 *VPLs, right: visualization of the clamped irradiance estimate (1 f-stop brighter), which is added to the left image. Note that this contribution automatically decreases with the total number of VPLs in the scene.*

6.4. Walk-through Animations

Radiance caching is well-suited for walk-through animations since the previously computed cache records can be reused in successive frames and only few cache records need to be added from frame to frame. However, a few changes need to be considered to prepare the algorithm for walk-throughs. First of all, to avoid exhaustion of memory all cache records need to maintain an age indicating the last frame the record contributed to. All records that did not contribute to the last n frames are deleted at the end of each frame to make space for new caches. Second, we can further exploit temporal coherence between frames to compute perceptual visibility thresholds for each pixel. These can then be reprojected and used in the next frame to detect discontinuities and steer the adaptive cache splatting (see Section 5.2). For all feasible pixel reprojections we use the computed visibility thresholds instead of the discontinuity thresholds based on the Weber law [KBPv06].

The computation of the perceptual visibility thresholds is borrowed from the well-established JPEG/MPEG compression standards [RW96]. Such a threshold computation based on the discrete cosine transform (DCT) of (8×8) image blocks can be very efficiently computed (ca. 60 ms for

a 500×500 pixels on a single PC). We use the approach in [HKMS08]. The user is able to set a scaling parameter q_{scale} (originally used in MPEG to adapt the quantization level to the bandwidth of a channel) to define the quality of the overall cache interpolation. Higher q_{scale} values raise the visibility thresholds and increase the sensitivity to luminance and contrast masking [HKMS08, RW96]. Setting $q_{scale} <= 4$ computes thresholds below the just-noticeable-difference thresholds and does not generate noticeable artifacts in our tests. In contrast to [HKMS08] we do not use the thresholds for the lightcut computation but only for the discontinuity detection in the cache interpolation. An example for a frame is shown in Fig. 7.

Figure 7: *(left:) one frame from a camera walkthrough in the apartment scene computed with lightcut caching for direct and indirect light, (middle) computed thresholds for $q_{scale} = 5$, (right) reprojected thresholds from previous frame (red color means no correspondance). For visualization the thresholds are scaled by 32.*

7. Results

We compared our method with the lightcuts algorithm [WFA*05], which computes the radiance for each pixel and with photon mapping combined with radiance caching [KBPv06]. To be fair all methods were integrated in the same renderer using the same basic data structures and algorithms. The photon mapping implementation uses the same adaptive cache splatting procedure as for our lightcuts caching, only the computation of the caches differs. The gradients are computed numerically in a stratified manner [KŏŠ05]. For precomputing the BRDF hemispherical harmonics (HSH) coefficients and rotating the radiance HSH coefficients we included the open source code provided by J. Křivánek [Web]. All results were obtained on an AMD Athlon 64 (2.2 GHz) with 3 GByte of RAM. In order to compare with the lightcuts algorithm we provide the numerical results in Table 1 without pixel super-sampling while we show antialiased images in Fig. 8 using 4 super-samples per pixel. Since the eye-pass (raycasting the frame-buffer and storing eye samples) is the same for all methods it is not explicitly shown in Table 1. Although there are still small differences to the reference computed with lightcuts [WFA*05], the overall quality is high while the rendering time is in the order of 20 to 100 seconds for computing a single image in high quality. Interestingly, one can observe in Table 1 how the direct light masks the indirect light during the cache computation. A good example is the diffuse sibenik scene where the computation of direct plus indirect light is even more efficient than solely computing the indirect lighting with our caching. The discontinuity detection and cache adaptation described in Section 5.2 takes about 0.2 to 0.5 seconds for 5 cache iterations and 512^2 pixels for the tested scenes.

The knot scene is lit by high-frequency direct and indirect light casting sharp shadows on glossy and diffuse surfaces. The kitchen and apartment scene ©INRIA 2005 is challenging because it contains many light sources and small objects and most surfaces are slightly glossy. Also the indirect light is very strong and localized (e.g. above the chandelier) and photon mapping requires about 4000 final gather rays per cache record to avoid the otherwise distractable noise. Compared to a path-traced reference, the photon mapping result however is slightly better than our result mainly due to the VPL clamping in corners. We also computed two walkthrough animations, one only with indirect the other with global illumination, in the apartment scene consisting of 281 frames. Using the discernibility metric described in Section 6.4 with $q_{scale} = 4$ instead of the Weber law with 2% luminance thresholds [KBPv06] reduces the total number of indirect light records by 19% in this walk-through animation. The icido scene is an example of difficult indirect lighting. Even with 5000 final gather rays per cache record the photon mapping results are too noisy and prone to artifacts. For this scene anisotropic caching pays off most and less cache records are created around the light sources and in corners than for isotropic caching (see Fig. 3).

8. Discussion

Although our method seems superior in terms of robustness and efficiency of the global illumination computation, it also has a few disadvantages when comparing it with traditional photon mapping. While photon mapping is able to deal with low-frequency caustics, yet on a high price, our method cannot handle any caustics because photons stored on moderately glossy surfaces are treated like diffuse VPLs. Extending the lightcuts algorithm to cluster VPLs on moderately glossy surfaces and computing the upper error bounds by taking also into account the BRDF at the clusters locations during the lightcut computation is left as future work.

Second, the clamping of close VPLs results in bias in particular on glossy surfaces near corners where only a few VPLs (in the BRDF lobe) have a very high contribution to the pixel radiance. The clamping correction described in Section 6.3 can always be applied but is computationally expensive. However, also photon mapping suffers from bias in corners due to noise and overestimation of photon density since almost the same set of photons is queried by many final gather rays (see Fig. 9 left). Consequently, to produce high quality images with photon mapping also requires secondary final gathering.

Third, along sharp shadow boundaries cache records do

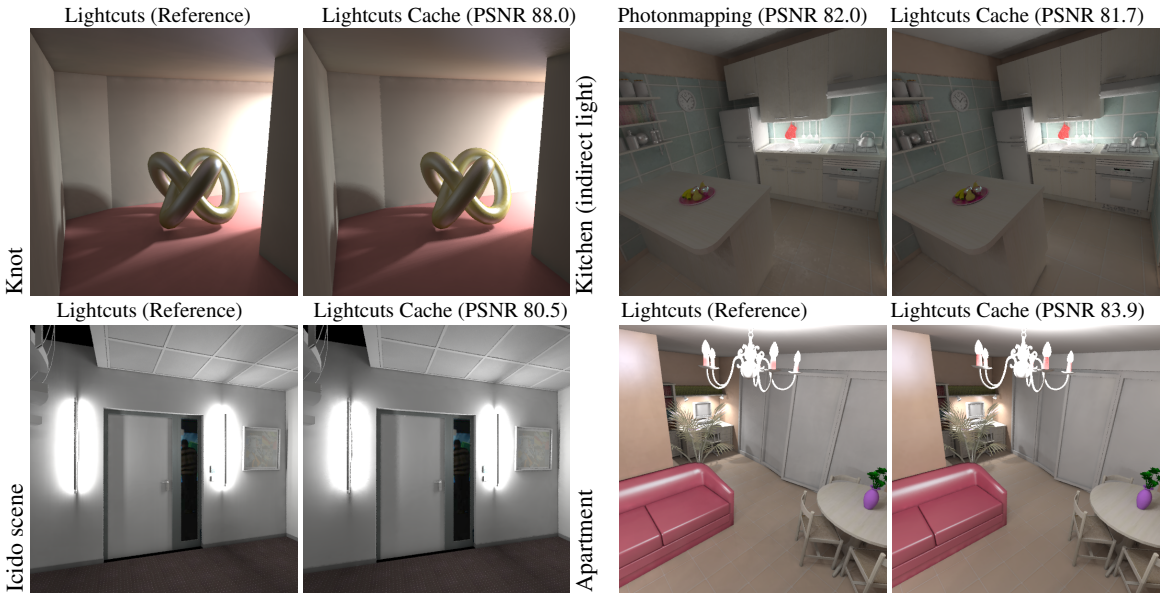

Figure 8: *Rendered images and peak-signal-to-noise-ratio (PSNR) for our method (lightcuts cache) compared with lightcuts and photonmapping with radiance caching. The kitchen-view results are compared against a path-traced image (not shown).*

Figure 9: *Zoom into kitchen scene result of Fig. 8, left: photon mapping, (middle) lightcut caching, (right) pathtracing.*

not always reduce to a pixel-sized footprint. Hence, shadow edges may appear slightly jaggy or washed out. For high-quality direct lighting a smart algorithm should detect and separate single point light and directional light sources and compute their contribution for each pixel.

One drawback of the proposed irradiance-gradient computation in Section 4.1 is the absence of visibility gradients (we simply assume the visibility gradients are zero since we only know the point to point visibility). This results in errors in particular for the direct light gradients (see Fig. 10). While

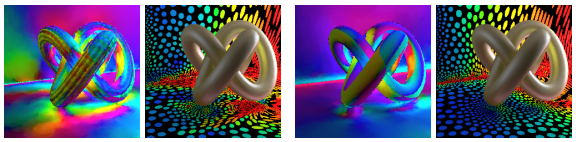

Figure 10: *color-encoded gradient direction and corresponding anisotropic cache footprints where color shows anisotropy, (left) stratified gradients can handle visibility, (right) our gradients fail in the presence of large occlusions.*

a stratified gradient computation as in [KŮ05] should be feasible, it is non-trivial and less efficient for irregular surface area sampling of VPLs than for uniform solid-angle based sampling.

9. Conclusion

We presented a high-quality global illumination algorithm with its strength in robustness and automatic adaptation to scene and lighting complexity with relatively little user intervention (setting the standard parameters works well for most scenes) in contrast to traditional photon mapping with (ir)radiance caching. The method extends the original lightcuts algorithm [WFA*05] to be used for (ir)radiance caching and improves the adaptive cache interpolation [KBPv06] by anisotropic cache splatting driven by perceptual visibility thresholds, which could also replace the traditional (ir)radiance caching for photon mapping. We further proposed several optimizations and implementation details to make the algorithm efficient and practical for various scene and light settings. We achieve computation times for a single image in the order of seconds rather than minutes on a standard PC. As future work we would like to extend the algorithm to dynamic scenes in the spirit of [GBP06].

References

[GBP06] GAUTRON P., BOUATOUCH K., PATTANAIK S.: *Temporal Radiance Caching*. Tech. Rep. 1796, IRISA, Rennes, France, 2006.

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

	Method	N_c	$\frac{a_\parallel}{a_\perp}$	a[%]	FGRs	T_{light}	$T_\int$	T_{cache}	$\sum T$
Knot	PM	12479(0%)	–/0.9	–/35	3000(48)	3.4	328	1.0	331
	LCache2	10177(0%)	–/1.0	–/33	–/678	3.1+8.3	26	0.7	28
	LCache1	34309(69%)	1.0/1.0	8.5/32	77/679	3.2+8.4	33	1.5	36
	LCache2	33231(70%)	0.86/1.0	8.4/31	76/658	3.2+8.4	30	1.2	32
	LC	–	–	–	193	3.2+8.4	–	–	235
	N_{prim} = 2936, N_V = 405,000, N_{glossy} = 43%, ε = 0.1/0.4, μ = 2.0								
Sibenik	PM	34467(0%)	–/1.0	–/24.5	3000(30)	17.5	4697	2.0	4701
	LCache2	35944(0%)	–/0.93	–/25	–/886	17.5+49	107	2.2	111
	LCache2	109518(71%)	0.74/0.65	8.8/24	123/600	7.5+50	94	3.3	98
	LC	–	–	–	279	7.4+50	–	–	522
	N_{prim} = 80394, N_V = 560,000, N_{glossy} = 0%, ε = 0.1/0.25, μ = 1.3								
Kitchen	PM	15452(0%)	–/1.2	–/37	3000(29)	7.8	2686	5.0	2695
	LCache2	15113(0%)	–/1.1	–/36	–/679	15.1+93	33	0.8	35
	LCache2	39510(67%)	1.12/1.02	9/35	179/595	7.5+96	42	0.9	45
	LC	–	–	–	279	7.4+96	–	–	608
	N_{prim} = 72365, N_V = 600,000, N_{glossy} = 28%, ε = 0.1/0.4, μ = 1.3								
Icido	PM	14613(0%)	–/1.3	–/26	5000(35)	8.9	2001	7.6	2010
	LCache2	18063(0%)	–/1.3	–/29	–/493	8.6+34	19	4.2	24
	LCache1	38991(69%)	1.0/1.0	8/30	149/385	4.1+34	21	8.9	31
	LCache2	36520(69%)	0.65/1.1	8/30	150/381	4.1+34	23	4.7	29
	LC	–	–	–	260	4.1+34	–	–	291
	N_{prim} = 199835, N_V = 515,000, N_{glossy} = 10%, ε = 0.1/0.4, μ = 1.3								
Apartment	PM	20996(0%)	–/1.08	–/38	4000(25)	7.8	1189	1.1	1193
	LCache2	18063(0%)	–/1.09	–/37	–/851	15+92	61	0.8	63
	LCache2	45661(67%)	0.92/1.02	10/37	263/798	7.5+96	79	1.5	82
	LC	–	–	–	260	7.4+96	–	–	467
	LCwalk2	3325(65%)	1.1/1.5	8/27	148/738	(7.3+95)	12	4.7	18
	LCwalk2	1237(0%)	–/1.6	–/26	–/826	(15+92)	10	3.0	14
	N_{prim} = 72365, N_V = 600,000, N_{glossy} = 62%, ε = 0.1/0.4, μ = 1.3								

Table 1: *Statistics and computation times (in seconds) for the rendering with our algorithm and isotropic cache splatting (LCache1), anisotropic cache splatting (LCache2) , lightcuts (LC) with 2% error threshold, and photonmapping with radiance caching (PM) for 5 scenes shown in Fig. 8 and Fig. 5. For the walk-through (281 frames) computed with LCache2 the statistics for the average frame are shown. Image resolution is* 512×512 *pixels.* N_c *is the number of computed cache records, the fraction of direct light records is shown in percent.* $\frac{a_\parallel}{a_\perp}$ *is the mean anisotropy of the cache records for direct/indirect light, column FGR shows the average lightcut size for direct/indirect light or in the case of photon mapping the number of final gather rays and average found nearest neighbor photons, a is the average cache error for direct/indirect light in percent,* T_{light} *is the time for the light pass including photon shooting plus light-tree construction,* $T_\int$ *is the cache computation time,* T_{cache} *is the time for cache splatting,* T_Σ *is the total rendering time without the initial* T_{light}*.* N_{prim} *is the number of scene primitives,* N_V *is the number of VPLs/photons,* N_{glossy} *is the fraction of glossy cache records,* ε *is the global cache error for direct/indirect light, and* μ *is the initial anisotropy scaling parameter.*

[GKPB04] GAUTRON P., KŘIVÁNEK J., PATTANAIK S. N., BOUATOUCH K.: A novel hemispherical basis for accurate and efficient rendering. In *Eurographics Symposium on Rendering* (2004), pp. 321–330.

[HHS05] HAVRAN V., HERZOG R., SEIDEL H.-P.: Fast final gathering via reverse photon mapping. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3 (2005), 323–333.

[HKMS08] HERZOG R., KINUWAKI S., MYSZKOWSKI K., SEIDEL H.-P.: Render2mpeg: A perception-based framework towards integrating rendering and video compression. In *Proceedings of Eurographics* (2008).

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

[KBPZ06] KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S. N., ŽÁRA J.: Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Eurographics Symposium on Rendering* (2006), pp. 127–138.

[Kel97] KELLER A.: Instant radiosity. In *Proceedings of SIGGRAPH* (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).

[KK04] KOLLIG T., KELLER A.: Illumination in the presence of weak singularities. *Monte Carlo and Quasi-Monte Carlo Methods* (2004), 245–257.

[Kri05] KŘIVÁNEK J.: *Radiance Caching for Global Illumination Computation on Glossy Surfaces*. Ph.d. thesis, Université de Rennes 1 and Czech Technical University in Prague, 2005.

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

[PM90] PERONA P., MALIK J.: Scale-space and edge detection using anisotropic diffusion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1990), vol. 12, pp. 629–639.

[RW96] ROSENHOLTZ R., WATSON A. B.: Perceptual adaptive JPEG coding. In *IEEE International Conference on Image Processing* (1996), pp. 901–904.

[SM02] SMYK M., MYSZKOWSKI K.: Quality improvements for indirect illumination interpolation. *Proceedings of the International Conference on Computer Vision and Graphics* (2002), 685–692.

[TL04] TABELLION E., LAMORLETTE A.: An approximate global illumination system for computer generated films. In *Proceedings of SIGGRAPH* (2004), pp. 469–476.

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. *ACM Transactions on Graphics 25*, 3 (2006), 1081–1088.

[WBKP08] WALTER B., BALA K., KULKARNI M., PINGALI K.: Fast agglomerative clustering for rendering. In *IEEE Symposium on Interactive Ray Tracing (IRT)* (2008).

[Web] WEBRESOURCE: Hemispherical harmonics source code. http://www.cgg.cvut.cz/.

[Wei98] WEICKERT J.: *Anisotropic diffusion in image processing*. Teubner, Stuttgart, 1998.

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D.: Lightcuts: A scalable approach to illumination. In *Proceedings of SIGGRAPH* (2005), pp. 1098 – 1107.

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

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

[ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *Proceedings of SIGGRAPH* (2001), pp. 371–378.