Eurographics Conference on Visualization (EuroVis) 2014
H. Carr, P. Rheingans, and H. Schumann
(Guest Editors)

Volume 33 (2014), Number 3

Parallel Irradiance Caching for Interactive Monte-Carlo Direct Volume Rendering

Rostislav Khlebnikov[1], Philip Voglreiter[1], Markus Steinberger[1], Bernhard Kainz[2] and Dieter Schmalstieg[1]

[1]Institute for Computer Graphics and Vision, Graz University of Technology, Austria
[2]Department of Computing, Imperial College London, UK

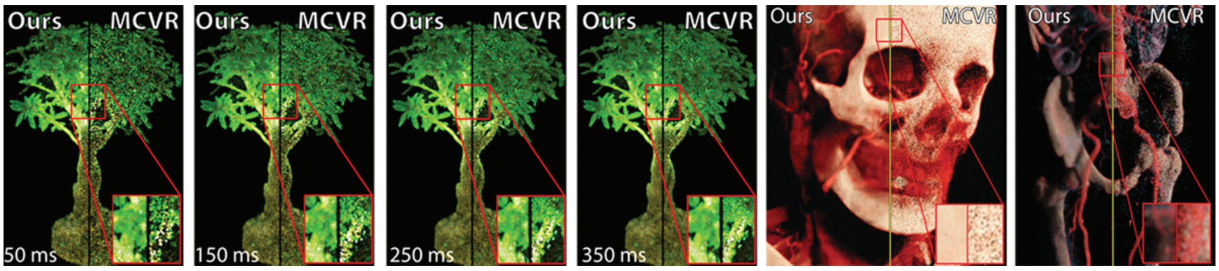

Figure 1: *Three volumes with enlarged areas rendered with Irradiance Caching (left half) and normal Monte Carlo Volume Rendering (right half) after the same time. The extracted areas show a significant reduction in noise with our approach.*

Abstract

We propose a technique to build the irradiance cache for isotropic scattering simultaneously with Monte Carlo progressive direct volume rendering on a single GPU, which allows us to achieve up to four times increased convergence rate for complex scenes with arbitrary sources of light. We use three procedures that run concurrently on a single GPU. The first is the main rendering procedure. The second procedure computes new cache entries, and the third one corrects the errors that may arise after creation of new cache entries. We propose two distinct approaches to allow massive parallelism of cache entry creation. In addition, we show a novel extrapolation approach which outputs high quality irradiance approximations and a suitable prioritization scheme to increase the convergence rate by dedicating more computational power to more complex rendering areas.

1. Introduction

Perception of volumetric data displayed with Direct Volume Rendering (DVR) approaches can be greatly improved if global illumination effects are taken into account. This includes both strong effects like global shadows, and more subtle ones, such as scattering. One of the approaches to compute costly global illumination effects is Monte Carlo Volume Rendering (MCVR). MCVR uses stochastic integration to solve the Radiative Transfer Equation (RTE) [Cha60], which describes the complex interaction of light with participating media. The progressive nature of MCVR is an advantage and allows showing intermediate results, thus enabling interactive data exploration. However, early results are very noisy, which may force the user to suspend interaction until higher levels of convergence are achieved.

Irradiance [WRC88] and radiance [KGPB05] caching techniques significantly improve the convergence rate by storing and reusing the illumination computation results for nearby pixels. This does not reduce the quality, because the illumination field varies smoothly in large parts of natural scenes. An additional advantage of such techniques is that the cache can be built during rendering – once a not cached position is encountered in the scene a new cache entry is computed before rendering proceeds. For interactive DVR, multiple problems prevent using traditional techniques.

First, the majority of modern DVR systems employ the

GPU to achieve interactive frame rates. The massive parallelism required to effectively utilize the power of the GPU is achieved by dedicating one thread per pixel and executing the threads in SIMD manner in thread blocks. Thus, if all the threads within a thread block compute a similar cache entry at the same time, a lot of computational effort is wasted since many redundant entries are created. Second, even if the cache density is controlled, displaying the results for the thread block will stall until all the necessary cache entries are created, which affects interactivity significantly.

In this paper, we propose an integrated approach that allows to build the irradiance cache for isotropically scattering materials in parallel to rendering on a single GPU, without leading to excessively dense caches or rendering stalls. To achieve compatibility with most DVR applications, we formulated the following goals:

- Irradiance caching should not interfere with the interaction and the visualization, allowing for efficient data exploration. This includes both virtual camera motion as well as interactive transfer function design.
- Caching should improve the convergence rate of the MCVR without sacrificing its quality.
- Memory footprint of the cache should be minimal.

These goals are contradictory to some extent. For instance, a very low memory footprint may be achieved at the expense of quality or interactivity. Achieving a good balance in a GPU-friendly way is not trivial, as multiple interlocking tasks compete for the GPU time and must be scheduled. In this regard, we present the following contributions:

- We demonstrate object- and screen-space approaches which allow the new cache requests to be handled in a massively parallel MCVR system without rendering stalls or creating an excessively dense cache on a single GPU.
- We show a prioritization scheme that allows to distribute the available computational power between building the irradiance cache and the actual rendering.
- We propose a modification to the exponential irradiance extrapolation approach, which is more accurate for scenes with high irradiance field gradients.

Overall, we show that our method improves the convergence rate and, consequently, the visualization quality of MCVR without reducing its interactivity.

2. Related work

In this section, we give a brief description of the core technologies building the foundation of our approach. In particular, we discuss MCVR and irradiance caching.

Global illumination in volume rendering: There are numerous approaches to computing global illumination for DVR. They differ concerning the range of effects they can achieve as well as concerning performance and memory requirements. An excellent overview of the existing approaches towards computing advanced volumetric illumination in interactive volume rendering is given by Jönsson *et al.* [JSYR13] in their state-of-the-art report.

The radiative transfer equation [Cha60] serves as a base for most methods, which compute global illumination in participating media. Kajiya and Von Herzen [KVH84] propose to use a two-pass approach to include global illumination effects for rendering of volumetric datasets. During the first pass, the radiance is estimated for each voxel, which is consecutively integrated along view rays during the second pass. However, the first pass is very time consuming and thus not applicable to interactive visualization of dynamic scenes. Many approximations for improving performance exist. Ambient occlusion is used for efficient shadow computation in a local neighborhood [SA07, RMSD*08, DVND10] and summed area tables provide an approximation including global shadowing [SMP11].

Rezk-Salama [Sal07] propose to use a Monte-Carlo approach for physically-based volume rendering. However, displaying only a set of isosurfaces limits the quality of this method. Kroes *et al.* [KPB12] demonstrated that progressive Monte-Carlo rendering is feasible for interactive DVR. We use their method as basis for our method (Section 3).

Photon mapping approaches [JC98] have also been applied to DVR. Jönsson *et al.* [JKRY12] propose Historygrams to increase the speed of photon map re-computation for transfer function changes. However, the photon gathering step is still quite time consuming, leading to low frame rates when the camera moves. Precomputed radiance transfer approaches [SKS02] allow to overcome the high runtime cost. Zhang *et al.* [ZD13] propose to use precomputed photon maps for high quality interactive volume data visualization. While the rendering performance is very high, full recomputation of the photon map is still required upon a transfer function change.

Global illumination can also be approximated by a diffusion process [Sta95]. Zhang *et al.* [ZM13] propose to use a convection-diffusion partial differential equation. While the performance and range of effects achieved by this approach is high, it can only handle certain light types. Weber *et al.* [WKSD13] apply an approach using virtual point lights (VPL) in interactive volume rendering. Even though the resulting image is visually appealing, interactive visualization severely restricts the VPL number.

Irradiance caching: The irradiance caching technique [WRC88] takes advantage of the fact that the indirect irradiance field is mostly smooth. The irradiance and some additional quantities, such as the irradiance gradient [WH92], are computed for a sparse set of cache points, which are then used during rendering to interpolate the irradiance, avoiding costly integration over all directions. Křivánek *et al.* [KGPB05] propose radiance caching, which stores and interpolates direction-dependent radiance using spherical harmonics. Jarosz *et al.* [JDZJ08] extended the radiance caching approach for use with participating media.

The shape of the influence zone of cache entries affects the memory footprint necessary to achieve high quality results. We need to consider the total number of cache entries and the amount of data stored for each of them. The most widespread shape is spherical [JDZJ08]. The goal of storing very little information for each cache entry is affected, if the cache density increases significantly in the areas of high frequency illumination changes (Fig. 5). Furthermore, spherical influence zones cause excessive cache density in all directions, even if irradiance changes rapidly in only one direction. Therefore, an adaptive shape of the influence zone is more suitable for highly inhomogeneous media, which, for instance, can be observed in direct volume rendering of scientific data. Ribardière *et al.* proposed adaptive records for irradiance caching methods, applied to surface data [RCB11a] and volume data [RCB11b] rendering, where the influence zones are adapted according to geometrical features and irradiance changes.

There have been attempts at bringing irradiance caching to parallel systems, mostly for multiple nodes using MPI. Robertson *et al.* [RCLL99] propose distributing cache generation over several nodes and use frame-to-frame coherence for geometric scenes to reduce the computational demand. However, single nodes still compute entries for screen regions serially, which limits the overall parallelism of the approach. On top, cache synchronization among nodes is problematic. More recently, Debattista *et al.* [DSC06] propose separating rendering from irradiance calculations. Parts of the locally computed cache of single nodes at particular time intervals are shared, which leads to issues with latency and cache misses for entries computed on a different node. This has also been observed by Kohalka *et al.* [KMG99]. Gautron *et al.* [GKBP05] propose the radiance cache splatting approach to enable efficient use of the GPU. Debattista *et al.* [DDPdSC11] propose a wait-free data structure for populating irradiance cache on parallel systems with shared memory. However, both methods create the cache entries sequentially for the whole screen or a large tile and do not handle participating media.

3. Background: Exposure Render

We use the *Exposure Render* method presented by Kroes *et al.* [KPB12]. It applies the Monte Carlo approach to solve the radiance transfer equation for interactive progressive volume rendering. In this section, we describe the relevant techniques applied in the Exposure Render method that we further use to build our caching approach.

In its essence, the goal of volume rendering is to solve a radiative transfer equation for a heterogeneous participating medium. Using the notation employed by Jarosz *et al.* [JDZJ08], the radiative transfer equation for non-emissive participating media can be written as:

$$\begin{aligned}L(\mathbf{x},\vec{\omega}) = \int_0^s T_r(\mathbf{x}\leftrightarrow\mathbf{x}_t)\sigma_s(\mathbf{x}_t)L_i(\mathbf{x}_t,\vec{\omega})dt + \\ T_r(\mathbf{x}\leftrightarrow\mathbf{x}_s)L(\mathbf{x}_s,\vec{\omega}),\end{aligned} \tag{1}$$

where $L(\mathbf{x},\vec{\omega})$ is the radiance reaching $\mathbf{x}$ along a ray with direction $\vec{\omega}$, which is parametrized as $\mathbf{x}_t = \mathbf{x} - t\vec{\omega}, t \in (0,s)$, and $\mathbf{x}_s$ is the exit point from the volume in direction $-\vec{\omega}$. $L_i(\mathbf{x},\vec{\omega})$ is the in-scattered radiance at position $\mathbf{x}$ in direction $\vec{\omega}$. Finally, the transmittance, T_r, is computed as

$$T_r(\mathbf{x}'\leftrightarrow\mathbf{x}) = e^{-\tau(\mathbf{x}'\leftrightarrow\mathbf{x})}, \tag{2}$$

where τ is the optical thickness:

$$\tau(\mathbf{x}'\leftrightarrow\mathbf{x}) = \int_{\mathbf{x}'}^{\mathbf{x}} \sigma_t(\mathbf{x})dx. \tag{3}$$

Here, σ_t is the extinction coefficient, which is the sum of the scattering coefficient σ_s and the absorption coefficient σ_a. The solution of the radiative transfer equation may be obtained using a Monte-Carlo approach. In the Exposure Render framework, it is implemented as follows:

- For each pixel on the screen, a single ray is cast into the scene through a random position within this pixel.
- Each ray propagates through the volume to find a single tentative collision point (scatter event).
- For this event, the illumination is estimated by casting a ray towards a randomly selected sample located at one of the lights. Similarly to the previous step, if a collision point with the medium is found, the point is considered unlit. Otherwise, this light sample in conjunction with a phase function is used to compute the illumination and considered in the current estimate for this pixel.
- The current estimate is incorporated into the final pixel color using a running average approach.

In this paper, we accelerate the illumination estimation using an irradiance cache. For isotropic phase functions, the irradiance can be computed as the integral of the incident radiance over the sphere of directions:

$$E(\mathbf{x}) = \int_\Omega L(\mathbf{x},\vec{\omega})d\vec{\omega} \tag{4}$$

4. Parallel irradiance cache management

The main objective of our approach is to generate an irradiance cache in a massively parallel way while concurrently performing DVR on a single GPU. As alternating between DVR and cache management would introduce a considerable latency, we want to *schedule* all involved tasks concurrently. In this way, we can provide immediate feedback to camera movements using progressive updates while building and adjusting the irradiance cache at the same time.

To enable this goal, we distinguish between three procedures which are executed on the GPU in parallel (Fig. 2):

The rendering procedure implements the basic Exposure Render approach with a modified illumination computation step. For each scattering event, we check whether cache information is available. If so, we extrapolate the irradiance and use it for shading. If not, we generate a cache entry request and use default MCVR shading instead.

The cache creation procedure handles incoming cache entry requests. It computes new cache entries by sampling the irradiance field in the local neighborhood of the cache request center.

The cache update procedure eliminates discontinuities in the estimated irradiance field. This discontinuities can arise in the areas of high frequency changes in the irradiance field, which are not captured by the values computed during cache entry creation.

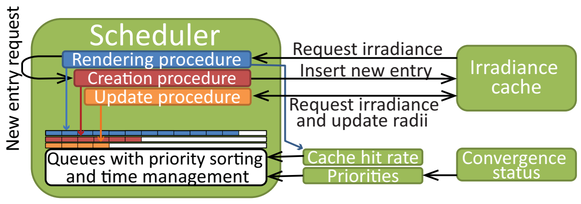

Figure 2: *The overview over our system. Three procedures run in parallel on a single GPU. The scheduler distributes time slots for each of them based on the cache hit rate. The rendering procedure is also prioritized based on screen-space convergence information.*

4.1. GPU Scheduling

Using the traditional execution model based on GPU shaders or SIMD languages like CUDA, a dynamic concurrent execution cannot be set up easily. Even with the most recent *dynamic parallelism*, which allows kernel launches from the GPU, we were unable to implement concurrent caching and rendering efficiently. The overhead of dynamic parallelism itself and finding a mapping between launched threads and actual work was too high in our experiments. Thus, we use *Softshell* [SKK*12], an open-source framework, which occupies the GPU to provide custom scheduling in software.

Softshell uses a *persistent megakernel* built on top of CUDA. In this approach, a single kernel continuously occupies the GPU, with each thread spinning in a loop. An idle thread draws a new task from a queue implemented in software. The queues support parallel insertion and removal using a fixed-size ring buffer with atomically operated front and back pointers with overflow/underflow protection. Individual slots are assigned to threads with atomic additions on these pointers. This design provides simultaneous access to an arbitrary number of threads without waiting.

To allow for different scheduling policies, we associate an individual queue with each procedure as shown in Figure 3. Depending on which queue is chosen for dequeue, different scheduling policy can be employed. For instance, different quotas can be assigned to different procedure types. Moreover, queues can be sorted to reflect priorities within one procedure type, enabling an out-of-order execution.

The priority queues offer the possibility to influence the execution order to, e.g., control the amount of resources invested into each individual procedure or focus processing power on more important image regions. As the persistent threads megakernel can be controlled from outside, it is possible to interrupt the GPU execution at a certain point in time. We use this feature to enable a frameless rendering approach aiming at a fixed refresh rate.

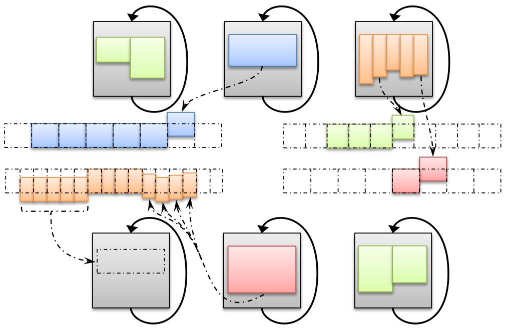

Figure 3: *Our approach consists of worker-blocks, continuously drawing tasks from queues. We keep one queue per procedure, which is essential for a divergence free execution of tasks with different granularities.*

4.2. Cache entry creation

We take the following steps to create a single cache entry:

- Compute the local coordinate frame.
- Estimate the incoming irradiance at the centre and at six points offset along the axes of the local coordinate frame.
- Compute validity radii for each of the six axis directions.

Below, we first show how we estimate the irradiance at the necessary positions. Next, we show how to extrapolate the irradiance to arbitrary positions. Then, we explain how we choose validity radii. Finally, we justify the selection of the local coordinate frame.

Estimation of irradiance values. The incoming irradiance is estimated with Monte Carlo integration using multiple importance sampling with power heuristics [Vea98]. We control the number of samples N for the Monte Carlo integration using the standard deviation based error estimate with 95% confidence interval [HGI08]:

$$1.96\sqrt{\frac{S^2(N)}{N}} < \frac{\gamma}{\gamma+1}E(N), \qquad (5)$$

where γ is the acceptable relative error. $E(N)$ and $S^2(N)$ are the average and the variance of irradiance estimates, which are computed incrementally [Fin09]. Note that we use the relative error estimate due to the fact that the human eye is most sensitive to relative and not absolute values of illumination change according to Weber's law of just noticeable differences [TFCRS11, p. 37]. Therefore, larger absolute errors are admissible for brighter regions.

Irradiance extrapolation and validity radius. We assume that the irradiance changes exponentially in the local neighborhood of the cache entry center $\mathbf{c}$, similarly to Jarosz *et al.* [JDZJ08]. Using the error metric shown in Eq. (5), we estimate the irradiance values at $\mathbf{c}$ and at positions $(\mathbf{c} + \xi \cdot \Delta)$ for $\xi \in (\pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k})$, where $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ are the unit vectors of the local coordinate frame at $\mathbf{c}$, and Δ is the computation offset (see Fig. 4, left). Then, for each of six directions, we fit a 1D exponential function of form $E(x) = E(\mathbf{c}) \cdot \exp(-\tilde{\sigma}_t x)$, where $\tilde{\sigma}_t$ is the tentative local extinction

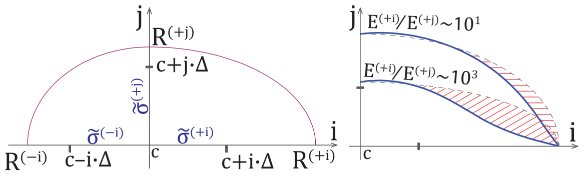

Figure 4: *(Left) We compute the tentative extinction coefficients $\tilde{\sigma}$ as well as radii R for six directions $(\pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k})$. (Right) When interpolating local extinction coefficients, the ellipsoidal shape may be inconsistent with the actual validity radius if the ratio of irradiances at different axes is very high (inner graph). Therefore, to avoid visual artefacts, we discard the influence of a cache entry in areas beyond the interpolated validity radius (red hatching). However, for lower ratios (outer graph), the ellipsoidal shape is a good estimate for the actual validity radius.*

coefficient, using two computed data points $\{0; E(\mathbf{c})\}$ and $\{\Delta; E(\mathbf{c}+\xi\cdot\Delta)\}$:

$$\tilde{\sigma}_t^{\xi} = -\ln\left(\frac{E(\mathbf{c}+\xi\cdot\Delta)}{E(\mathbf{c})}\right)\cdot\frac{1}{\Delta}. \tag{6}$$

To perform the extrapolation to an arbitrary position $\mathbf{p}$, we first compute its coordinates $\mathbf{p}' = (p'_{\mathbf{i}}, p'_{\mathbf{j}}, p'_{\mathbf{k}})$ in the local coordinate frame. Then the tentative local extinction coefficient in the direction $\mathbf{p}'$ is computed using barycentric coordinates:

$$\tilde{\sigma}_t^{(\mathbf{p}')} = \frac{|p'_{\mathbf{i}}|\cdot\tilde{\sigma}_t^{(\pm\mathbf{i})} + |p'_{\mathbf{j}}|\cdot\tilde{\sigma}_t^{(\pm\mathbf{j})} + |p'_{\mathbf{k}}|\cdot\tilde{\sigma}_t^{(\pm\mathbf{k})}}{|p'_{\mathbf{i}}| + |p'_{\mathbf{j}}| + |p'_{\mathbf{k}}|}, \tag{7}$$

where the sign of $(\pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k})$ corresponds to the sign of $(p'_{\mathbf{i}}, p'_{\mathbf{j}}, p'_{\mathbf{k}})$. Finally, the extrapolated irradiance value is computed as $E(\mathbf{p}') = E(\mathbf{c})\cdot\exp(-\tilde{\sigma}_t^{(\mathbf{p}')}|\mathbf{p}'|)$.

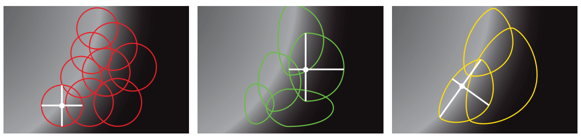

Figure 5: *Influence of the entry shape on the cache density. The white lines denote the radii corresponding to the local coordinate frame of an entry. If the entry is represented as a union of elliptical sections, (middle) less cache entries are required compared to the spherical ones (left). Orienting these shapes along the opacity gradient (right) reduces the cache density even more because in many cases, the largest irradiance gradient coincides with the opacity gradient. Orienting the entries allows for larger validity zones, since the radius has to be small only along one axis.*

Given this extrapolation approach, we compute the validity radius for each half-axis such that the extrapolated irradiance at the edge of the influence zone differs by not more than one ε relative to the estimated irradiance at position $(\mathbf{c}+\xi\cdot\Delta)$:

$$R^{\xi} = \Delta\cdot\frac{\ln(E_{\varepsilon}/E(\mathbf{c}))}{\ln(E(\mathbf{c}+\xi\cdot\Delta)/E(\mathbf{c}))}, \tag{8}$$

where

$$E_{\varepsilon} = \begin{cases} (1+\varepsilon)E(\mathbf{c}+\xi\cdot\Delta) & \text{if } E(\mathbf{c}) \le E(\mathbf{c}+\xi\cdot\Delta) \\ (1-\varepsilon)E(\mathbf{c}+\xi\cdot\Delta) & \text{if } E(\mathbf{c}) > E(\mathbf{c}+\xi\cdot\Delta). \end{cases} \tag{9}$$

We define the shape of the cache entries as a union of elliptical sections with distinct radius along each of half-axes of the local coordinate frame (Fig. 4, left). While an ellipsoid is a good estimate, the actual validity radius for the interpolated local tentative extinction coefficient may be smaller than the corresponding ellipsoid radius. With our exponential extrapolation approach, this may lead to errors in case the ratio of the estimated irradiance along the half-axes is large (Fig. 4, right). Therefore, we discard the influence of a cache entry if the point lies outside the validity radius of the interpolated local tentative extinction coefficient.

If a point lies within the validity zone of multiple cache entries, we compute the log-space weighted average of the per-entry extrapolation results [JDZJ08]:

$$E(\mathbf{p}) \approx \exp\left(\frac{\sum_{k\in C}\ln(E_k)w(d_k)}{\sum_{k\in C}w(d_k)}\right) \tag{10}$$

where the weight is computed as $w(d) = 3d^2 - 2d^3$ with $d_k = 1 - \|\mathbf{p}'_k\|/R_k(\mathbf{p}'_k)$, $\mathbf{p}'_k$ are the coordinates of the point in the local coordinate frame of the cache entry k, and E_k is the irradiance extrapolated using the cache entry k. Since the shape of cache entry in each octant is an ellipsoid section, the radius is computed as:

$$R(\mathbf{p}') = \|\mathbf{p}'\|\cdot\left((p'_{\mathbf{i}}/R_{\mathbf{i}})^2 + (p'_{\mathbf{j}}/R_{\mathbf{j}})^2 + (p'_{\mathbf{k}}/R_{\mathbf{k}})^2\right)^{-0.5}$$

Local coordinate frame. To minimize the effect of errors introduced by the 3D interpolation of the local tentative extinction coefficient, it would be preferable to orient one of the axes of a cache entry's local coordinate frame along the irradiance gradient. The exact orientation would require knowing the gradient in advance. This is a very expensive computation, requiring to store the local coordinate frame with each cache entry. However, we have observed that in many cases the largest change in irradiance is related to the variation of the extinction coefficient. Therefore, orienting the local coordinate frame along the gradient of the extinction coefficient is a good approximation of the optimal orientation. Furthermore, as the computation of extinction coefficients is deterministic and can be done using a simple central differences approach, we can easily recompute it on demand rather than expend additional memory for its storage.

We have compared the cache size for spherical, as well as non-oriented and oriented pseudo-ellipsoidal influence zones. The results have shown that spherical zones indeed lead to very high cache densities (more than four times larger). Oriented and non-oriented zones in many cases produce very similar cache sizes. However, we observed that the

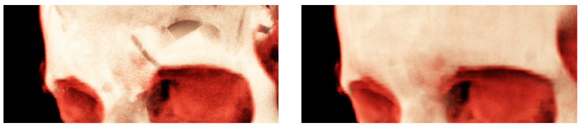

Figure 6: *(left) Visible artefacts can be produced if extrapolated irradiance does not capture the actual irradiance field. (right) The update procedure removes such artefacts.*

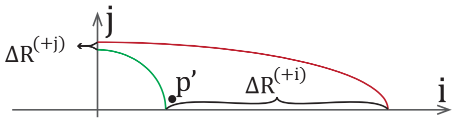

Figure 7: *Reduction of radii in the local coordinate frame of the cache entry to exclude point* $\mathbf{p}'$ *from the validity area in the corresponding octant of the cache entry. The radii are reduced proportionally to the coordinates of point* $\mathbf{p}'$ *in the local coordinate frame. In this case,* $|p'_\mathbf{i}|/\Delta R^{(+\mathbf{i})} = |p'_\mathbf{j}|/\Delta R^{(+\mathbf{j})}$.

cache size for oriented zones was 5-10% smaller than that of the non-oriented ones for optically dense materials. Because using oriented zones does not require additional storage and little computational effort, we use oriented cache entries throughout the remainder of this paper.

4.3. Cache update

While computing the validity radius gives a good approximation for the cases where the irradiance varies smoothly, it can still be erroneous to a certain degree (Fig. 6, left). Deviations result from considering global illumination information only partially during the estimation of incoming irradiance in the local neighborhood of the cache entry centre. To avoid the resulting artefacts, we incorporate a cache update procedure into our system. The cache update is done in a similar way as proposed by Křivánek *et al.* [KBPv08]. The update procedure searches for scatter events in the same way as the main rendering procedure. Whenever the predictions of overlapping cache entries conflict at the scatter event position, the contribution of the cache entry with the lowest weight is removed from the computation by reducing its radii in the corresponding octant. The radii are reduced proportionally to the event's coordinates in the local coordinate frame (see Fig. 7), i.e. $|p'_\mathbf{i}|/\Delta R_\mathbf{i} = |p'_\mathbf{j}|/\Delta R_\mathbf{j} = |p'_\mathbf{k}|/\Delta R_\mathbf{k}$. We repeat this process, until all conflicts are resolved, with the special case of only one remaining influence.

4.4. Cache entry storage

We store the cache entries in a multi-reference octree, similarly to Křivánek and Gautron [KG09]. Even though the multi-reference octree requires additional storage space, its performance is significantly higher than that of its single-reference counterpart [Kar11]. Since we only need to store additional references rather than full entries, the impact on the memory footprint is small.

We use oriented bounding boxes enclosing the cache entries to find the nodes for which to store a reference. Additionally, as the update procedure may change the extent of a cache entry influence zone and reduce the number of influenced octree nodes, we rebuild the octree from scratch at particular time intervals (Table 1). This overall reduces the total number of references (Section 7).

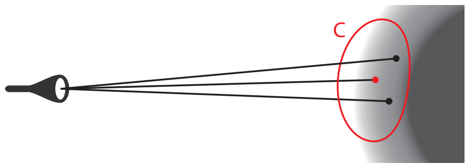

Figure 8: *Neighboring rays may issue cache entry creation requests at very similar positions. We avoid redundant cache entries by allowing only one cache entry to be created per octree node. Once a cache entry C is created, creation requests within the influence of C will be ignored.*

5. Parallelization of cache entry computations

Neighboring rays are likely to issue a scattering event at similar positions, especially if they encounter optically dense material (Fig. 8). This may lead to too dense and redundant cache regions despite low frequency irradiance variation.

To solve this issue, we evaluated two different approaches. Method 1 forbids simultaneous creation of more than one entry per octree node. Since the octree resides in the object-space of the volume, we call this approach *object-space locking*. Method 2 is based on the fact that the creation of new cache entries is driven by image-based ray generation. Hence, locking parts of the screen in form of superpixels is an alternative. We call this scheme *screen-space locking*.

For both versions, upon requesting a new entry, we first try to lock the corresponding primitive (node or superpixel) using atomic operations, similar to Debattista *et al.* [DDPdSC11]. Upon success, we issue the request for creating a new cache entry, and, as soon as its estimation has finished, we insert it into the octree and release the lock. Requests for already locked primitives are discarded and the sample of the scattering event is shaded with MCVR.

5.1. Object-space locking

In this technique, each octree node can be separately locked. If we cannot extrapolate the irradiance from the cache, we check whether we can lock the node furthest down in the branch enclosing the event position. Improving over the method of Debattista *et al.* [DDPdSC11], however, our approach also allows parallel, asynchronous determination of the positions where new cache entries need to be created.

The effects of this approach are twofold. First, it allows parallel creation of multiple cache entries in distant parts

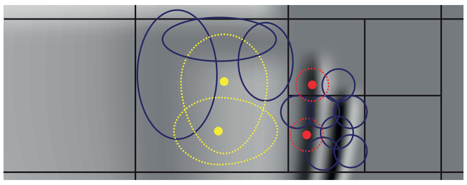

Figure 9: *As the octree nodes are small in the areas of high cache density, more cache entries can be created in these areas in parallel. This does not lead to the creation of redundant cache entries, because the new entries are likely to have small radii as a high cache density suggests high frequency of irradiance change in this area. In this illustration, the simultaneous creation of two entries, shown in red, will be allowed, while it will be forbidden for the two yellow ones, even though the distance between them is the same.*

along a single ray. Second, the octree adapts its local resolution to cache density. (Fig. 9).

5.2. Screen-space locking

In contrast to the aforementioned method, *screen-space locking* operates on rectangular regions of the screen (superpixels). For an optimal distribution of requests, we use two restrictions. First, each superpixel may only issue one request at a time. We limit the total number of cache entry requests throughout the whole screen to balance workload.

Only a fraction of all rays in a superpixel are actually in need of a new entry, since some may have encountered available cache information in a particular iteration. The ratio of requests to total rays bears information on the projected cache distribution in the current region. We only issue creation of a new cache entry, if a randomized rejection test based on the ratio passes. This method allows for accessing the same node of the octree from different superpixels on the screen. Consecutively, for preventing conflicts, we use atomic operations on the reference list per node.

6. Priorization

To control the assignment of available processing power to procedures, we use a two-level priorization. First, we dynamically adjust the ratio of processing time spent on each of the three procedures based on the cache hit rate. Second, we use different priorities for the individual image regions during the rendering procedure to direct processing power towards parts further from convergence.

Cache status directed scheduling. If the number of cache entries ensures that the majority of lighting computation requests can be extrapolated from the cache, it is not necessary to allocate time for creating additional cache entries. To achieve this goal, we dynamically adjust the time frame assigned to the individual procedures based on the cache hit rate (h). We estimate h over a small time interval to consider multiple depths for each pixel. The time frames are then computed as: $t_c = t_{c,min} + (1 - h^{e_c}) \cdot (t_{c,max} - t_{c,min})$, where t_c corresponds to the relative time frame assigned to the cache creation procedure, $t_{c,min}$ and $t_{c,max}$ are the minimum and maximum time that should be used for cache creation, respectively. e_c allows to control a non-linear translation from hit rate to assigned time. Based on the intuition that the number of required cache updates should be proportional to the number of newly created cache entries, we use the same formula with different minimum and maximum times for the cache update procedure. The relative time spent on rendering simply corresponds to the remaining time (parameters given in Table 1).

Rendering image priorities. Based on the geometric relations of the scene and lighting, the convergence rate of individual parts of the image may differ strongly. For instance, a region completely in shadow may converge to black within a single iteration, while complex light interactions from multiple light sources will result in very slow convergence rates. Thus, we want to provide a more uniform convergence rate across the entire image. We do that by estimating the expected gain from running another iteration per region. If we assume that the sample variance does not change after an additional iteration, then the error estimate will be reduced by

$$\Delta E = 1.96 \left(\sqrt{\frac{S^2(N)}{N}} - \sqrt{\frac{S^2(N)}{N+1}} \right) \quad (11)$$

Using ΔE, we compute the average expected error reduction for each block and use it directly rendering priority.

7. Implementation

As mentioned in section 4, an essential part of our approach is the concurrent execution of cache creation, cache update and rendering. Using the Softshell [SKK*12] scheduling framework, we generate and manage work descriptors for all three procedures on the GPU. Each procedure is run in blocks of 128 threads.

The *rendering procedure* divides the screen into blocks of 16×8 pixels and performs ray casting with one pixel per thread. A sufficient number of blocks are created initially to cover the entire screen. For each scattering event, we query information from the irradiance cache. In case no entry exists we use either of the locking methods described in Section 5. In case a new request should be generated, we generate a work descriptor for the cache creation procedure and hand it over to the scheduling framework. After all threads of the rendering procedure have added their contribution to each pixel estimate, we hand the work descriptor for this screen block back to the scheduling framework. In this way, rays for the same screen region will again be traced at a later point in time, implementing the progressive rendering method.

All 128 threads in the *cache creation procedure* work on a single cache entry. Each thread casts seven rays towards randomly selected lights – one ray for the central position and one for each of the six positions offset in both directions of the axes of local coordinate frame. Then, the results are com-

Table 1: *The parameter values used in our experiments.*

Cache entry creation and update		
Estimation accuracy	γ	0.1
Validity radius threshold	ε	0.1
Position offset	Δ	2 voxel
Minimum influence zone radius	-	0.01 voxel
Maximum influence zone radius	-	16 voxel
Cache entry update threshold	-	0.1
Procedure time distribution		
Max. creation procedure fraction	$t_{c,max}$	15%
Min. creation procedure fraction	$t_{c,min}$	5%
Max. update procedure fraction	$t_{u,max}$	2%
Min. update procedure fraction	$t_{u,min}$	1%
Exponent for fractions update	e_c	3
Screen-space locking		
Max.# simult. created cache entries	-	1000
Max.# simult. created entries/superpixel	-	1
Miscellaneous		
Octree rebuild interval	-	1000 ms
Cache hit rate update interval	-	100 ms

bined using shared memory. If the stopping criterion is met (Eq. (5)), the newly created cache entry is inserted into the octree, which uses atomic operations to avoid potential race conditions. Otherwise, the process is repeated. The work descriptor is simply removed after execution.

Similar to the rendering procedure, the *cache update procedure* also starts with thread blocks covering the entire screen. Each thread randomly selects one of the covered pixels and casts a ray into the scene. At the scatter event, it checks whether cache entries conflict with each other and reduces radii if necessary. The random selection of a pixel avoids updating the cache in neighboring object-space positions. When the entire block has finished, the block is reemitted for scheduling. In this way, cache entries for the entire image are consistently checked and improved.

To incorporate execution time into the priority scheduling process, we measure the number of cycles of each individual procedure execution using the clock cycle counter provided by CUDA. We then update the current estimate of the overall time spent on each of the three procedures using atomic addition. Using these measurements we update the procedure priorities, and steer the scheduling in such a way that they receive predefined portions of the overall available processing time. For instance, we could use 20% for cache creation and 80% for rendering.

8. Results

We evaluate several conditions of our proposed method in comparison to plain MCVR. In the following, we discuss cache creation in static scenes and during interaction, and also evaluate convergence rates for a fully built cache. Furthermore, we provide measurements for comparing the behavior of object- vs. screen-space locking. For our experiments, we used the following datasets: Bonsai ($512 \times 512 \times 182$), Manix ($512 \times 512 \times 460$), Macoessix ($512 \times 512 \times 460$) in full, half and quarter resolution each.

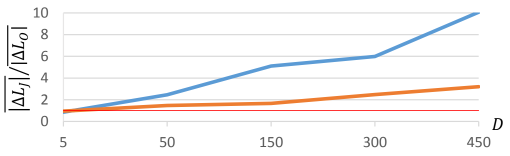

Figure 10: *The ratio of average relative error of extrapolated radiance values computed using the approach presented in [JDZJ08] ($\overline{|\Delta L_J|}$) to the ones computed with our method ($\overline{|\Delta L_O|}$) for scene with a single light source (blue) and with additional background illumination (orange). The red line shows the value of* 1 *which means no difference in average errors of two methods. The horizontal axis shows the global density factor D which is used to convert transfer function values in range* $[0,1]$ *to the density of participating medium. Our method is significantly more accurate for higher gradient magnitudes, which are caused by larger global volume density factor D or by the light setup. Lower estimation errors for our method lead to the lower number of cache entries required to represent the irradiance field.*

In Table 3, we list both the total number of cache entries for our three tested volumes, as well as the total number of references stored in the multi-reference octree for object-space locking. During all our experiments, we measured an average of six to seven references for each cache entry. In fact, these additional five to six references per entry, stored as indices, are a good trade-off memory-wise. Single referencing induces the necessity of searching through neighbor nodes on each level along an octree branch leading to significantly higher number of costly global memory accesses, decreasing the performance.

Extrapolation accuracy for single cache entries. We compared the radiance extrapolation accuracy computed using the method by Jarosz *et al.* [JDZJ08] with our method (Section 4.2). The radiance for our method is obtained by multiplying the irradiance value by the isotropic phase function value and scattering coefficient. Fig. 10 shows that with increasing global density, i. e., growing gradient magnitudes, the Jarosz method, which is relying on only a single gradient value, becomes insufficient to capture the high frequency changes in the irradiance field. By comparison, our method produces drastically lower error, quickly amortizing the memory and computation overhead (a factor of 2-4$\times$) required for the more complex cache entry creation. Higher gradients induced by the lack of background illumination cause similar effects.

Behavior during and after interaction. During interaction, new regions of the dataset may appear which are not covered by the cache yet. However, we implicitly exploit

Table 2: *Comparison of plain MCVR error estimates with our irradiance caching method. The numbers express the ratio of plain MCVR error over irradiance caching error, averaged over 200 frames. The higher the number, the faster our approach converges. The values in bold show the ratio with "expected gain" priorities enabled.*

	512x512x* ; 1280x720			512x512x* ; 1920x1080			256x256x* ; 1920x1080		
Lights	Bonsai	Manix	Macoessix	Bonsai	Manix	Macoessix	Bonsai	Manix	Macoessix
1	1.58 **1.62**	1.49 **1.55**	1.65 **1.70**	1.42 **1.65**	1.44 **1.57**	1.64 **1.74**	1.94 **1.99**	1.89 **2.29**	1.42 **1.58**
3	1.38 **1.34**	1.23 **1.28**	2.67 **2.56**	1.28 **1.38**	1.24 **1.31**	2.72 **2.78**	3.36 **3.51**	1.71 **2.40**	2.55 **2.76**
4	1.20 **1.29**	1.23 **1.25**	2.56 **2.46**	1.23 **1.34**	1.12 **1.22**	2.42 **2.57**	4.03 **3.89**	2.03 **2.05**	2.86 **2.95**

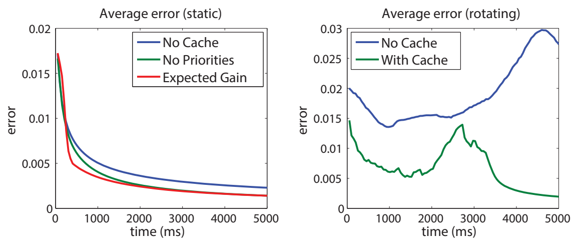

Figure 11: *(Left) Average error progression observed for the static Bonsai with initially empty cache. Both rendering with (red) and without priorities (green) achieve faster convergence than plain MCVR (blue). (Right) Average error observed during rotating Bonsai by 360 degrees. Even during interaction, our method (green) has lower error compared to plain MCVR (blue). Stopping interaction (3300 ms) leads to rapid convergence due to present cache information.*

frame-to-frame coherence, since previous regions rarely vanish instantly, and we generate new entries on-the-fly. As expected, stopping interaction leads to quickly and drastic decrease in error, whereas plain MCVR starts from scratch. The right panel of Figure 11 shows the average error rates for the Bonsai during 360 rotation and then stopping interaction.

Convergence rates in static scenes. Static scenes starting with an empty cache and without user interaction are a hard case for a cache-based algorithm, since a lot of noise is accumulated initially. We consider three different lighting setups. First, we use a single small, distant area light. Second, we use a setup with three very intense, large area lights close to the volume. Third, we use four rather dim, large area lights surrounding the volume.

For each of these light setups, we have measured the convergence rate of our approach compared to plain MCVR using the error estimate shown on the left side of Eq. (5). This definition of the error per pixel in screen-space allows us to locally estimate the difference to an optimal solution. Since we observed very similar outcomes for all scene experiments, we provide the average ratio of plain MCVR versus our irradiance cache error over 200 frames in Table 2 and left panel of Figure 11 (plots in supplemental material). Note that despite the difficult setting, our method consistently outperforms plain MCVR event without priorities.

Object-space vs screen-space locking. As discussed in Section 5, we evaluate two different methods for preventing creation of excessive amount of redundant cache entries at once. In our first measurement, we rotate the volume by 360 degrees and record the number of cache entries and tree references. We show the results for the detailed structures of the Bonsai dataset. In a second, more extensive test, we focus on static scenes showing Manix and Bonsai with varying parameters. These include volume resolution (512^3, 256^3, 128^3), screen resolution (1920×1080, 1280×720, 640×480) as well as background illumination (on, off), while the viewpoint is fixed. Our tests show very marginal differences in convergence rates. For low screen resolutions, screen-space locking seems to create entries in more beneficial locations and achieves the desired cache hit rates and convergence values slightly faster. For high screen resolutions, object-space locking creates less redundant entries and achieves high cache hit rates faster with comparable cache sizes. For detailed outcomes of all parameter combinations, please refer to the supplemental material.

Table 3: *Cache sizes for object-space locking. The "References" column shows the total number of references to the cache entries in the multi-reference octree.*

Volume	Cache Entries	References
Bonsai	32421	193939
Manix	30755	206807
Macoessix	24774	150660

9. Conclusion and future work

We have presented a method for DVR irradiance caching in parallel, concurrently to MCVR. Two techniques have been proposed to keep the memory footprint small by preventing the creation of redundant cache entries. We have discussed a new irradiance extrapolation method, which outperforms previous approaches. Our experiments illustrate even for the most difficult situations a drastic increase in convergence rate compared to standard MCVR. Especially during and after interaction, our method achieves a significant improvement, outperforming MCVR by up to four times.

In future work, we will incorporate multiple scattering, as well as spherical harmonics to remove the current limitation to isotropic phase functions. We will also investigate alternative data structures for the cache for further performance.

References

[Cha60] CHANDRASEKHAR S.: *Radiative Transfer*. Books on Intermediate and Advanced Mathematics. Dover Publications, 1960. 1, 2

[DDPdSC11] DEBATTISTA K., DUBLA P., PEIXOTO DOS SANTOS L., CHALMERS A.: Wait-free shared-memory irradiance caching. *IEEE Comput. Graph. Appl. 31*, 5 (2011), 66–78. 3, 6

[DSC06] DEBATTISTA K., SANTOS L. P., CHALMERS A.: Accelerating the Irradiance Cache through Parallel Component-Based Rendering . Heirich A., Raffin B., dos Santos L. P., (Eds.), Eurographics Association, pp. 27–34. 3

[DVND10] DIAZ J., VAZQUEZ P.-P., NAVAZO I., DUGUET F.: Real-time ambient occlusion and halos with summed area tables. *Computers and Graphics 34*, 4 (2010), 337 – 350. 2

[Fin09] FINCH T.: Incremental calculation of weighted mean and variance. *University of Cambridge* (2009). 4

[GKBP05] GAUTRON P., KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S.: Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2005), EGSR'05, Eurographics Association, pp. 55–64. 3

[HGI08] HOLMES M., GRAY A., ISBELL C.: Ultrafast monte carlo for statistical summations. In *Advances in Neural Information Processing Systems 20*, Platt J., Koller D., Singer Y., Roweis S., (Eds.). MIT Press, Cambridge, MA, 2008, pp. 673–680. 4

[JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of SIGGRAPH '98* (New York, NY, USA, 1998), ACM, pp. 311–320. 2

[JDZJ08] JAROSZ W., DONNER C., ZWICKER M., JENSEN H. W.: Radiance caching for participating media. *ACM Trans. Graph. 27*, 1 (Mar. 2008), 7:1–7:11. 2, 3, 4, 5, 8

[JKRY12] JÖNSSON D., KRONANDER J., ROPINSKI T., YNNERMAN A.: Historygrams: Enabling interactive global illumination in direct volume rendering using photon mapping. *IEEE TVCG 18*, 12 (2012), 2364–2371. 2

[JSYR13] JÖNSSON D., SUNDÉN E., YNNERMAN A., ROPINSKI T.: A survey of volumetric illumination techniques for interactive volume rendering. *Computer Graphics Forum (conditionally accepted)* (2013). 2

[Kar11] KARLIK O.: *Data Structures for Interpolation of Illumination with Radiance and Irradiance Caching*. Master's thesis, Czech Technical University in Prague, 2011. 6

[KBPv08] KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S., ŽÁRA J.: Making radiance and irradiance caching practical: adaptive caching and neighbor clamping. In *ACM SIGGRAPH 2008 classes* (New York, NY, USA, 2008), ACM, pp. 77:1–77:12. 6

[KG09] KŘIVÁNEK J., GAUTRON P.: Practical global illumination with irradiance caching. *Synthesis Lectures on Computer Graphics and Animation 4*, 1 (2009), 1–148. 6

[KGPB05] KŘIVÁNEK J., GAUTRON P., PATTANAIK S., BOUATOUCH K.: Radiance caching for efficient global illumination computation. *IEEE TVCG 11*, 5 (2005), 550–561. 1, 2

[KMG99] KOHOLKA R., MAYER H., GOLLER A.: Mpi-parallelized radiance on sgi cow and smp. In *Parallel Computation*, Zinterhof P., VajterÃĚÂąic M., Uhl A., (Eds.), vol. 1557 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1999, pp. 549–558. 3

[KPB12] KROES T., POST F. H., BOTHA C. P.: Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE 7*, 7 (07 2012), e38586. 2, 3

[KVH84] KAJIYA J. T., VON HERZEN B. P.: Ray tracing volume densities. *SIGGRAPH Comput. Graph. 18*, 3 (Jan. 1984), 165–174. 2

[RCB11a] RIBARDIÈRE M., CARRÉ S., BOUATOUCH K.: Adaptive records for irradiance caching. *Comput. Graph. Forum 30*, 6 (2011), 1603–1616. 3

[RCB11b] RIBARDIÈRE M., CARRÉ S., BOUATOUCH K.: Adaptive records for volume irradiance caching. *The Visual Computer 27*, 6-8 (2011), 655–664. 3

[RCLL99] ROBERTSON D., CAMPBELL K., LAU S., LIGOCKI T.: Parallelization of radiance for real time interactive lighting visualization walkthroughs. In *Supercomputing, ACM/IEEE 1999 Conference* (1999), p. 61. 3

[RMSD*08] ROPINSKI T., MEYER-SPRADOW J., DIEPENBROCK S., MENSMANN J., HINRICHS K. H.: Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Eurographics 2008) 27*, 2 (2008), 567–576. 2

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of the I3D, ACM* (2007). 2

[Sal07] SALAMA C. R.: GPU-based Monte-Carlo volume raycasting. In *Proceedings of Pacific Vis* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 411–414. 2

[SKK*12] STEINBERGER M., KAINZ B., KERBL B., HAUSWIESNER S., KENZEL M., SCHMALSTIEG D.: Softshell: dynamic scheduling on GPUs. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 161:1–161:11. 4, 7

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (July 2002), 527–536. 2

[SMP11] SCHLEGEL P., MAKHINYA M., PAJAROLA R.: Extinction-based shading and illumination in GPU volume raycasting. *IEEE TVCG 17*, 12 (2011), 1795–1802. 2

[Sta95] STAM J.: Multiple scattering as a diffusion process. In *Rendering Techniques 1995*, Hanrahan P. M., Purgathofer W., (Eds.), Eurographics. Springer Vienna, 1995, pp. 41–50. 2

[TFCRS11] THOMPSON W., FLEMING R., CREEM-REGEHR S., STEFANUCCI J. K.: *Visual Perception from a Computer Graphics Perspective*. A K Peters/CRC Press, 2011. 4

[Vea98] VEACH E.: *Robust monte carlo methods for light transport simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998. AAI9837162. 4

[WH92] WARD G. J., HECKBERT P. S.: Irradiance Gradients. *1992 Eurographics Workshop on Rendering* (1992), 85–98. 2

[WKSD13] WEBER C., KAPLANYAN A. S., STAMMINGER M., DACHSBACHER C.: Interactive direct volume rendering with many-light methods and transmittance caching. *Proceedings of the Vision, Modeling, and Visualization Workshop* (2013). 2

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 85–92. 1, 2

[ZD13] ZHANG Y., DONG ZHAO MA K.-L.: Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. *IEEE TVCG 19*, 8 (aug 2013), 1317–1330. 2

[ZM13] ZHANG Y., MA K.-L.: Fast global illumination for interactive volume visualization. In *Proceedings I3D, ACM* (New York, NY, USA, 2013), ACM, pp. 55–62. 2