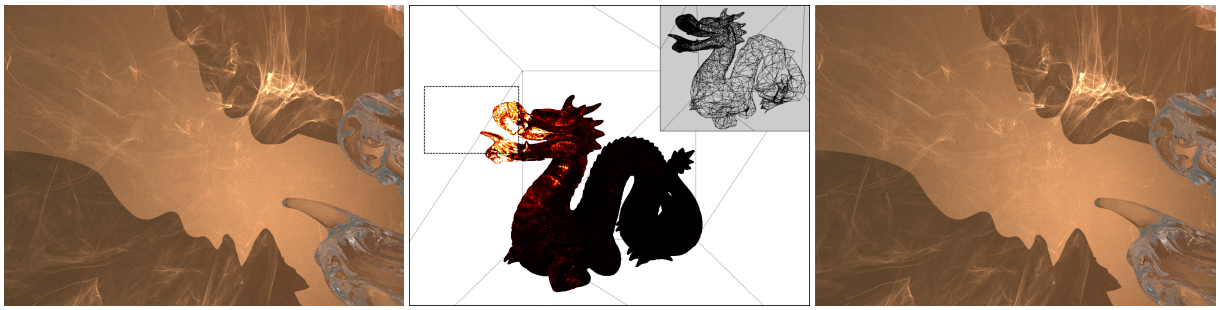
Eurographics Symposium on Rendering 2015
J. Lehtinen and D. Nowrouzezahrai
(Guest Editors)

Volume 34 (2015), Number 4

Illumination-driven Mesh Reduction for Accelerating Light Transport Simulations

Andreas Reich[1], Tobias Günther[2] and Thorsten Grosch[1]

[1]Computational Visualistics, University of Magdeburg, Germany
[2]Visual Computing, University of Magdeburg, Germany



(a) Reference (b) Importance and reduced mesh (c) 10% of original mesh data

Figure 1: Our illumination-driven mesh reduction is able to preserve complex indirect lighting.

Abstract

Progressive light transport simulations aspire a physically-based, consistent rendering to obtain visually appealing illumination effects, depth and realism. Thereby, the handling of large scenes is a difficult problem, as in typical scene subdivision approaches the parallel processing requires frequent synchronization due to the bouncing of light throughout the scene. In practice, however, only few object parts noticeably contribute to the radiance observable in the image, whereas large areas play only a minor role. In fact, a mesh simplification of the latter can go unnoticed by the human eye. This particular importance to the visible radiance in the image calls for an output-sensitive mesh reduction that allows to render originally out-of-core scenes on a single machine without swapping of memory. Thus, in this paper, we present a preprocessing step that reduces the scene size under the constraint of radiance preservation with focus on high-frequency effects such as caustics. For this, we perform a small number of preliminary light transport simulation iterations. Thereby, we identify mesh parts that contribute significantly to the visible radiance in the scene, and which we thus preserve during mesh reduction.

1. Introduction

Global illumination effects allow for striking visual experiences, depth and realism. Inspired by such, progressive light transport became a vibrant and active field of research, focusing on physically-based, accurate lighting [DKHS14]. While much research was dedicated to the specialization—and later generalization—of lighting effects across various kinds of materials and media [KGH*14], the visual effects industry made the growing need for complexity and scene size ever more pressing. Recent approaches took wide strides to handle progressive light transport simulations on large scenes [BBS*09,GG14], typically by subdividing them

into parts, manageable by a single computation node and processed the parts sequentially or in parallel on a cluster. The nature of light, however, makes this problem very challenging, as individual light paths frequently visit many scene parts, making it necessary to forward a significant amount of rays between scene parts. While it was shown that swapping of geometry data becomes highly inefficient in complex lighting scenarios [GG14], as photons might visit a single scene part multiple times, other strategies are desirable that operate on fixed memory bounds without the need for swapping. Mesh simplification is a straight-forward possibility to reduce memory consumption and computational complexity. To minimize resulting shape errors, simplification is usually applied adaptively, based on surface properties. This, however, does not account for the possibly significant impact on light paths in the visible part of the scene.

In this paper, we present a preprocessing step that reduces the scene under the constraint of radiance preservation. In contrast to previous mesh reduction heuristics that consider geometric errors to favor shape and volume preservation, our heuristic is based on the radiance visible by the camera. As such, our technique is view-dependent and able to significantly reduce the complexity of off-screen objects that barely contribute to the visible light in the image. To evaluate the contribution of meshes we perform a number of light transport iterations. Triangles that were less frequently visited by visible photons thereby receive lower collapse cost in the subsequent mesh reduction. The mesh reduction trades correctness for a certain desired scene size, which enables the rendering of an originally out-of-core scene on a single machine without swapping and minimal error.

We examine our heuristic in several small scenes under controlled lighting conditions, and discuss both a bottom-up and top-down reduction/refinement strategy. Finally, we study an out-of-core scenario in which we reduce the mesh to 10% of its original vertex count, achieving a rendering speed-up of factor $8.56\times$, compared to a memory swapping out-of-core implementation. As our method is a preprocessing step, it is orthogonal to the choice of light transport simulation algorithm. For our GPU implementation, we chose an Optix-based [PBD*10] stochastic progressive photon mapping [HJ09] implementation using a rectified hash grid [DKHS14] as underlying light transport algorithm.

2. Related Work

2.1. Light Transport Simulation

The rendering equation [Kaj86] describes the radiance at a certain location and direction. In his work, Kajiya introduced path tracing as a possible solution. It uses Monte Carlo integration by generating random paths originating from the camera. Bidirectional path tracing [LW93] connects paths from both light and camera to improve convergence. Metropolis light transport [VG97] extends bidirectional path tracing with Metropolis-Hastings sampling to explore paths nearby of already found ones with high contribution.

Classic photon mapping [Jen96] is a two-pass algorithm. First, photons are distributed from light sources and bounce on surfaces. For diffuse hits, photons are stored in the photon map (usually represented by a kd-tree). In the second pass, camera rays are traced for each pixel. At each diffuse scene intersection (*hit point*) a radiance estimate is performed, which gathers all photons within a certain search radius. Since the search radius has fixed size, the technique is biased. Progressive photon mapping [HOJ08] (PPM) overcomes this by using progressively shrinking search radii. Stochastic progressive photon mapping [HJ09] (SPPM) is an extension of this technique. While PPM computes hit points once, SPPM traces new hit point rays every iteration, which enables a range of different effects that profit from multiple view rays, e.g., glossy materials or lens effects. Recently, Georgiev et al. [GKDS12] and Hachisuka et al. [HPJ12] independently described vertex connection merging, which combines SPPM with bidirectional path tracing via multiple importance sampling in a consistent framework.

Davidovič et al. [DKHS14] compares different GPU implementations of (stochastic progressive) photon mapping and proposes several improvements. A practical and fast approach is the use of stochastic spatial hashing [HJ10]. Each grid cells saves only the last photon with an adjusted flux to account for previously rejected photons. This technique was further improved by Davidovič et al. [DKHS14] to make the selection probability independent of the path length.

2.2. Triangle Mesh Decimation

Botsch et al. [BKP*10] give a practical overview of different triangle mesh decimation algorithms. The iterative, quadric error metric algorithm by Garland and Heckbert [GH97] balances speed, fidelity and robustness. A symmetric 4×4 matrix is assigned to every vertex, representing the sum of squared distances to the supporting plane of neighboring triangles. Mesh reduction is steered by vertex collapse costs.

Rushmeier et al. [RV93] used a simplified scene to compute approximate radiosity, and the original scene for direct lighting and visibility. Similar, Christensen et al. [CLF*03] use multi-resolution geometry caches for secondary rays. Here, ray differentials steer the geometry tessellation level. Their technique is suited for out-of-core scenes, too, as the highest tessellation level does not reside in memory, but in contrast to our approach, they do not consider the light paths.

In a related thread of research, view-dependent simplification was studied, e.g., by Klein et al. [KSS98] for maintaining local illumination and by Guthe et al. [GBBK04], and Menzel and Guthe [MG10] for improving perception. In contrast to our work, their methods consider only visible geometry and direct illumination, which is insufficient for preservation of global illumination effects.

2.3. Out-of-Core Photon Mapping

By now, most global illumination techniques have been extended to handle out-of-core scenes. Christensen and

Batali [CB04] describe an approximation that was deployed for out-of-core production rendering. They build a multi-resolution irradiance atlas for efficient caching. Lighting is done by final gathering and irradiance interpolation in the atlas. Günther and Grosch [GG14] present a method for distributed out-of-core SPPM. Using a spatial subdivision of the input geometry into blocks, photons and view rays are traced in a network. The blocks are connected via portals that forward rays between machines. Contrary to earlier out-of-core photon mapping methods, like the technique for large buildings by Fradin et al. [FMH05], this method is consistent. Our approach requires only a single machine and is consistent within the resulting simplified scene.

2.4. Importance

Christensen's survey [Chr03] gives an overview of a large spectrum of techniques that used an importance concept. Each type of importance is basically an adjoint to one or more representations of light (exiting/incoming radiance, radiosity, irradiance or exiting/incoming power). Techniques based on importance use knowledge to make meaningful approximations or shortcuts in the lighting computation.

In photon mapping for example, importance can determine the photons to preferably compute and store. This is done by tracing *importons* from the camera, to determine areas contributing more to the visible radiance. This idea was mentioned by Jensen [Jen96] and later examined in detail by Keller and Wald [KW00], and Suykens and Willems [SW00]. Peter and Pietrek [PP98] focus on limiting or guiding the distribution and reflection of photons. Jensen [Jen95] demonstrated that photon maps can serve as importance for Monte Carlo raytracing.

Likewise, our approach involves a photon tracing pass to acquire photons as importance indicator. In our approach importance is used to determine decimation factors for geometry, rather than for image pixels or dominant light directions. Due to this difference, we do not use the term *importon*.

3. Illumination-Driven Mesh Reduction

3.1. Overview

Our goal is to simplify the geometry of a scene in a way that preserves visible features for a given camera. We strive for a minimal image error for both directly visible geometry and indirect lighting effects even for very high reduction ratios. To solve this problem we introduce an iterative preprocess that computes an importance heuristic for every vertex in the scene. Using this heuristic the scene is decimated with standard techniques [GH97].

In each iteration of our iterative preprocess, we trace camera and photon paths and connect them via density estimation. The importance of a triangle, i.e., its contribution to the visible radiance, is determined by the number of visiting photons and camera rays. Thereby, a visiting camera path always contributes to the triangle importance. Photon paths on the other hand contribute only if they were connected with at least one camera path. Fig. 2 gives a simplified example of an importance path iteration. Eventually, the importance values are used to steer a scene simplification that preserves triangles that contribute strongly to the visible radiance.

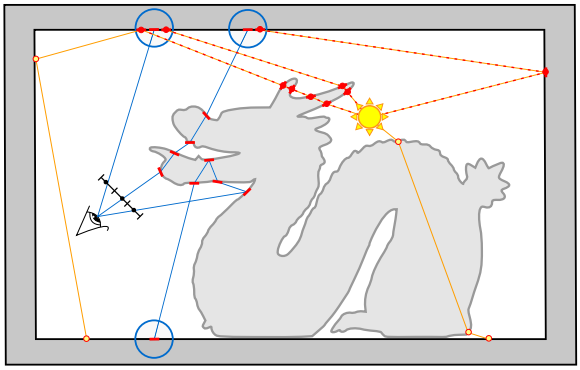


Figure 2: All camera path (blue) vertices increase importance on the mesh (red bars). Photon path (orange) vertices store photons (small circles) but increase importance only if they are gathered (blue circle) by a hit point.

The importance can either be obtained from the original scene or an already decimated one. The latter enables rendering of out-of-core scenes since the memory footprint is reduced up-front. However, this additionally involves a remapping of the importance to the original scene which will be explained in Section 3.5.

3.2. Mesh-based Importance

To steer the mesh reduction, we define an importance I per triangle $\mathbf{T}$ as a sum of weights $\hat{I}$ over all paths that hit the triangle. We assume a total of $i = 1...N$ paths, where each path $\bar{\mathbf{x}}_i = (\mathbf{z}_{i0}, \mathbf{z}_{i1}\ \ldots\ \mathbf{y}_{i1}, \mathbf{y}_{i0})$ consists of a camera sub-path $\bar{\mathbf{z}}_i$ and (optionally) a light sub-path $\bar{\mathbf{y}}_i$.

$$I(\mathbf{T}) = \frac{1}{A_\mathbf{T}} \sum_{i=1}^{N} \left(\gamma \sum_{j=1}^{|\bar{\mathbf{z}}_i|} \hat{I}(\mathbf{z}_{ij}, \mathbf{T}) + \sum_{j=1}^{|\bar{\mathbf{y}}_i|} \hat{I}(\mathbf{y}_{ij}, \mathbf{T}) \right) \quad (1)$$

The importance is normalized by the triangle area $A_\mathbf{T}$. The user parameter γ controls the preservation of directly visible objects in comparison to objects that contribute only indirectly, e.g., by indirect illumination or caustics. If not mentioned otherwise, we set $\gamma = 1$, giving camera paths the same weight as light paths. Taking camera paths into account that are not connected with a light is important, since otherwise light blocking, yet directly visible objects would not receive importance and would thus be heavily decimated.

Let $\mathbf{x}_{ij}$ be the j-th vertex of the i-th path. The influence $\hat{I}$ of a path segment $(\mathbf{x}_{ij-1}, \mathbf{x}_{ij})$ depends on the angle between the incident direction and the triangle normal $n_\mathbf{T}$:

$$\hat{I}(\mathbf{x}_{ij}, \mathbf{T}) = \begin{cases} 1 - \left| n_\mathbf{T} \circ \frac{\mathbf{x}_{ij} - \mathbf{x}_{ij-1}}{|\mathbf{x}_{ij} - \mathbf{x}_{ij-1}|} \right| & \text{if } \mathbf{x}_{ij} \in \mathbf{T} \\ 0 & \text{else} \end{cases} \quad (2)$$

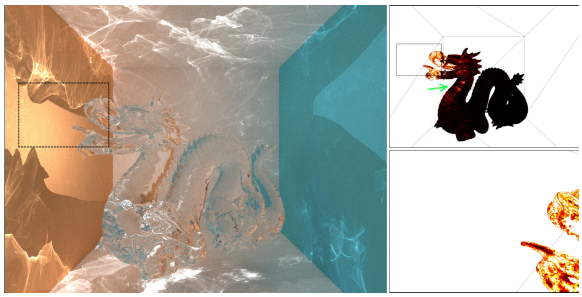


Figure 3: The rectangle depicts the camera area used for the importance pass. Right: Importance of the nearest vertex on the mesh, brighter color means higher importance.

This way, larger weight is given to larger incident angles, which preserve features around the silhouettes. Jagged edges can therefore be avoided for both visible objects and shadows, as shown later in Section 5.1. We define per-vertex importance as the area-weighted average of all neighboring triangles' importance values.

3.3. Importance Acquisition

To compute the importance values, we use a modified SPPM, which we explain next. Figure 2 illustrates the process.

Photon Pass Instead of flux, photons store the importance weight for later use. The photon simulation itself does not directly alter the mesh importance values. Contrary to traditional photon mapping, we store photons at *all* surface interactions for correct handling of reflective/refractive objects.

Hit Point Generation At each camera ray triangle hit, the mesh importance value is directly updated using Eq. (2), weighted by the hit point importance γ.

Radiance Estimation Pass All gathered photons add their importance weight to the triangle they have been deposited on. Additionally, the path of each gathered photon is iterated backward, to add the weights of all light path vertices preceding the gathered photon.

This way, our importance in Eq. (1) measures the amount of visible light that originates from a triangle. Thereby, the sum of visible photons corresponds to the visible flux, and dividing by the triangle area turns it into visible radiance. Figure 3 shows visualizations of the mesh importance. The left image gives an overview of the scene. In the lower right image, the importance distribution within a given view area after 100 iterations is shown. The upper right one shows the same importance distribution for the entire scene. The bright spots across the dragon's neck (indicated by the arrow) show triangles that were either hit by reflected hit point rays or by photon rays that reach the area seen by the camera later on.

The mesh importance (normalized by the number of iterations) converges rather fast. Figure 4 shows the mean squared change in mesh importance per importance iteration.

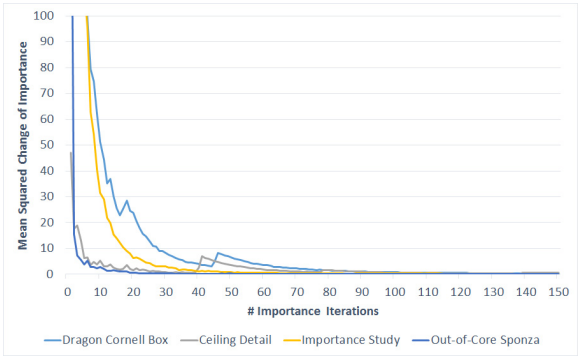


Figure 4: Mean squared importance change per additional importance iteration.

As suggested by the plot, we recommend and use 100 importance iterations in all test scenes.

3.4. Importance-based Reduction

Once the importance photon mapping is completed, an importance-driven mesh simplification can be performed. Our reduction algorithm relies on iterative edge collapses as performed in the decimation framework provided by the OpenMesh library [BSBK02], which we used in version 3.2. Figure 5 gives an example of the collapse process. All vertices search for their best collapse target and store it alongside with an error value on a heap. Valid targets are all vertices in the local ring, for which a collapse does not lead to a non-manifold or self-intersecting mesh.

Our method declares for each vertex the best collapse target as the vertex with lowest importance within the ring. To project the per-vertex importance to edge collapse error values, we sum up the importance of all vertices in the ring around the start vertex of a collapse candidate edge. The lower the resulting per-edge value, the more it is preferred for collapsing. This heuristic prefers to collapse vertices in low importance areas. The mesh is reduced iteratively by collapsing the edge with lowest error, until the desired reduction percentage is reached. After each collapse, normals and edge importance scores of all surrounding vertices are updated. The importance of the removed vertex is added to the remaining one.

Since importance values do not take mesh quality metrics into account, some meshes may lose important details or exhibit locally flipped normals. Therefore, combinations with error quadrics are also of high interest. However, this requires a proper weighting since they operate on distances in world space, while our importance is governed by visibility through photons or camera rays. Instead we alleviate this issue by adding a constraint that prohibits collapses that lead to a rotation of triangle normals of more than a given angle α_{max}. We found that for all our test scenarios $\alpha_{max} = 60$ was sufficient to prevent mesh errors without constraining the reduction process too much.

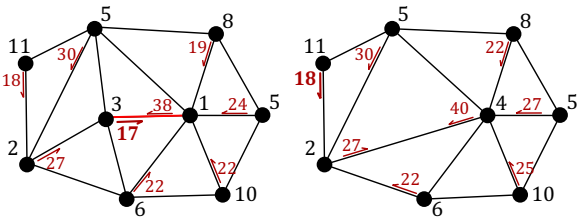


Figure 5: Example for importance-based edge collapse. Vertex importance is shown in black, chosen collapse candidates with direction are in red. *Left:* Initial mesh. *Right:* After collapse of edge with lowest importance.

3.5. Mapping Low-Poly Importance to High-Poly

To be able to render out-of-core scenes, it is necessary to compute the importance from an already simplified scene. The resulting importance data then needs to be mapped to the original, more detailed mesh to repeat the decimation using the newly acquired importance value.

Two additional bits of information need to be saved during the initial simplification to make this possible: First, each vertex of the decimated mesh saves its index in the original mesh. Second, each vertex of the original mesh saves to which vertex it was collapsed. Using this information the per-vertex importance can be projected onto the higher-detailed mesh. Every vertex of the original mesh can be connected to a vertex that still exists in the reduced mesh and can thus adopt its importance value. This leads to areas on the mesh which share the same importance value. In these areas, we rank the vertices by the number of collapses they are away from their importance source, and choose the "closest" vertex for the collapse. Simple "copying" of importance to the high resolution mesh is thereby a conservative choice.

With this, *multiple* importance passes are possible: The initial importance pass operates on a scene that was reduced by standard, view-independent techniques. Each additional pass remaps its importance as described above and the process can be repeated. We show later in Section 5.3 that multiple importance passes can improve the image quality.

3.6. Distribution of Decimation Factors

A crucial input parameter is the desired scene decimation factor D which lies between zero (maximal decimation, no vertices left) and one (no decimation, original mesh). To perform the importance-based reduction, meshes in the scene have either to be joined during the decimation process or to be decimated separately with individual decimation factors. In cases where the whole scene does not fit into memory, multiple meshes are mandatory. The factors need to account for the different sum of importance values within a mesh, since it is not desirable to apply the same decimation factor to insignificant meshes as to important ones.

Our goal is to obtain individual decimation factors d_i (for mesh index i) that depend on the ratios of the different mesh importance, but yield a vertex count V_D similar to an equal decimation D for all meshes. For this, we initialize the decimation factors with the sum of all importance values per mesh. Note that it is important to use the *unnormalized* importance values from before the triangle area division, mentioned in Section 3.4, since we are interested in the overall image contribution. Then, we perform a normalization to enforce our vertex count condition as follows:

$$\begin{aligned} V_D &= \sum v_i \cdot D \\ V_d &= \sum v_i \cdot d_i \\ d_i' &= d_i \cdot \frac{V_D}{V_d} \end{aligned} \tag{3}$$

Where v_i is the number of vertices in a given mesh and index i and V_d being the total number of vertices for a decimation using the factors d_i. Some of the resulting d_i' might be greater than one, which requires a higher vertex count than that of the input mesh. To overcome this issue, we clamp all d_i' to one and perform the normalization again. We repeat this process until all d_i' are smaller than one or a maximum iteration count is reached. Afterwards we start the decimation process, ignoring all meshes with factors larger or equal than one. For very small decimation factors it might not be possible to perform as many collapses as demanded, due to the collapse restrictions mentioned in Section 3.4 (no self-intersection and non-manifold meshes). Since this increases the final vertex count only slightly, we do not employ further special handling for these cases.

4. Details and Implementation

We implemented SPPM on the GPU in Optix 3.5 [PBD*10] using a rectified hash grid [DKHS14]. For both importance and rendering passes we use a maximal ray depth of 16 for view rays and 8 for photons.

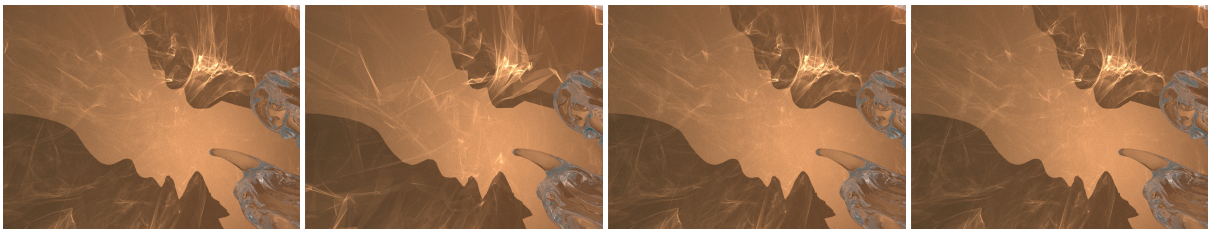
4.1. Importance Pass

Each mesh is assigned to a continuous piece of a shared importance buffer which contains all importance values. After photon and view ray path generation, we determine the photon paths that are connected to view rays (via radius query). We iterate these photons paths reversely, and increase the importance of the visited triangles, according to Eq. (2). As we use a photon buffer of pre-determined size, the path traversal is trivial since all deposits reside in consecutive memory. All view rays contribute immediately at each intersection point.

Afterwards, the importance buffer is copied to RAM for the CPU-based mesh reduction. Instead of a single buffer for the entire scene, it is imaginable to use an importance buffer per mesh. Then, photons would not only have to save an importance index, but also an index of a bindless buffer. Apart from being only available on more recent hardware, this leads to an increased memory footprint per photon.

4.2. Mesh Reduction

All reductions are performed in parallel per mesh using OpenMP. The initial, non-importance-based reductions use



(a) Reference (b) Error quadrics only. RMSE: 5.677% (c) One pass importance reduction. RMSE: 3.032% (d) Two pass importance reduction. RMSE: 3.033%

Figure 6: DRAGON BOX scene. Meshes in *(b)–(d)* are reduced to 10%. For a scene overview see Figure 3. First importance pass was obtained from the original scene.

the error quadrics of Garland and Heckbert [GH97], which are implemented in OpenMesh 3.2 [BSBK02]. For reduction steps before our importance pre-pass (as in Section 3.5), we ignore meshes with very low triangle count, such as the Cornell Box walls, to avoid reduction artifacts. When using importance-driven reduction these meshes usually have a high importance and are thus not decimated.

The OpenMesh framework is extensible by custom modules, which have callback functions for pre-collapse, getting collapse priority and post-collapse. We implemented two modules, one determining the importance-based reduction priority and one for saving a collapse history that is needed for mapping importance from a low-poly model back to its high-poly original (see Section 3.5).

In our test scenes, RAM was sufficient to hold all meshes in their original quality at once. In larger out-of-core scenes it may be necessary to load and unload meshes on demand.

5. Results

Our test system contains an Nvidia GTX670 with 2 GB VRAM, an Intel Core i7-3770 and 16 GB RAM. In all test scenes we used the SBVH [SFD09] acceleration data structure provided by Optix. Where not noted otherwise, we use for each image 10 billion photon rays, 1 million per iteration.

5.1. General Reduction Quality

One of our main objectives is the preservation of caustics under heavy scene decimation. Figure 6 shows renderings of a detail from the DRAGON CORNELL BOX. The dragon consists of 0.4 million vertices. Figure 6a depicts the detail without any reduction, Figure 6b shows the same detail with a purely quadric based reduction to 10% of the original vertex count. The initial importance pass in Figure 6c and 6d operated on the original scene. During the importance pass, hit point rays hit the visible part of the dragon mesh far more often than photons. This leads to a high preservation priority of the visible mesh part.

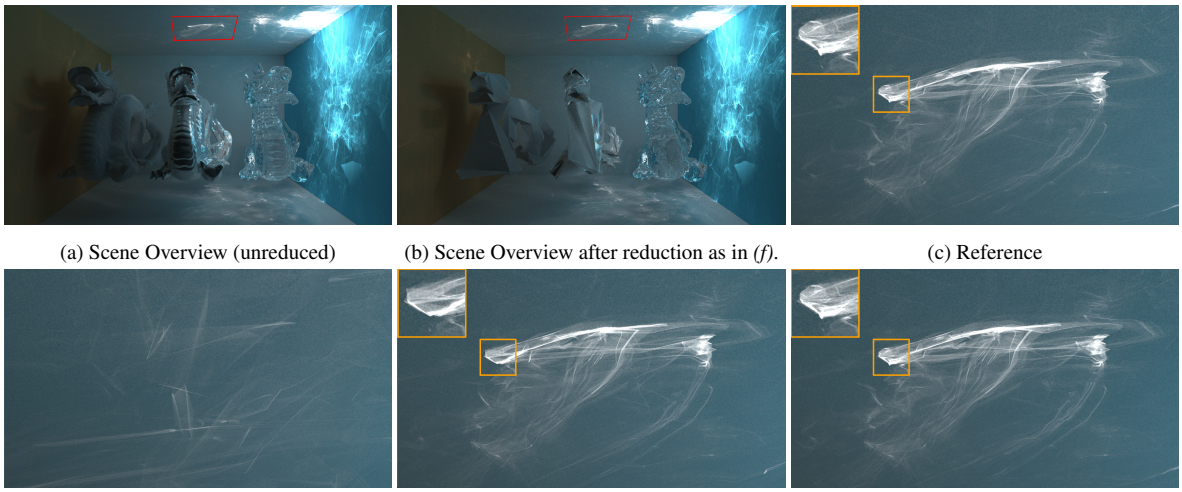
Figure 7 shows an extreme case where a caustic is cast from an off-screen object. The scene consists of three Stanford dragons, which were already used in the DRAGON CORNELL BOX scene. The left dragon is diffuse, the middle one glossy and the right one has a glass material. A single spot light illuminates the glass dragon through its maw (see Figure 7a). The red rectangle shows the view area of the camera used in images 7c–7f, with 7c as reference. In the images 7d–7f the scene was reduced to 5% of its original vertex count. The importance passes operated on an already reduced scene (as in image 7d) which means that the photon mapper did not know the original scene at any moment. The images 7d and 7f are a bit darker than the reference. This is caused by small variations in the reduction of the other two opaque dragon meshes. Our heuristic favors preservation of caustics over indirect low-frequency diffuse light. Figure 7b shows what the entire scene looks like after two importance iterations with the camera focused on the detail.

After the first importance iteration, the per-mesh decimation factor distribution algorithm (see Section 3.6) determines 13.93% for the glass dragon, 0.90% for the glossy one and 0.17% for the diffuse one on the left. For the optional second importance pass, the resulting importance distribution favors the glass dragon even more, which results in a slightly denser mesh (14.15%). However, our reduction algorithm is not able to find possible collapses below a decimation of about 0.95% which means that the actual vertex count raises slightly. While the evenly reduced scene has 65,847 vertices, the total vertex count after the first importance pass is 69,620 and 70,496 after the second. For high reduction in large scenes, the lower reduction bound may be a problem, in case there are many unimportant models that retain more vertices than anticipated. In those cases, it is advisable to replace meshes under a certain decimation threshold by proxy geometries.

Figure 8 shows the effect of the angular weight introduced in Eq. (2). Improvements can be observed foremost at the shadow silhouettes and in areas of strong refraction.

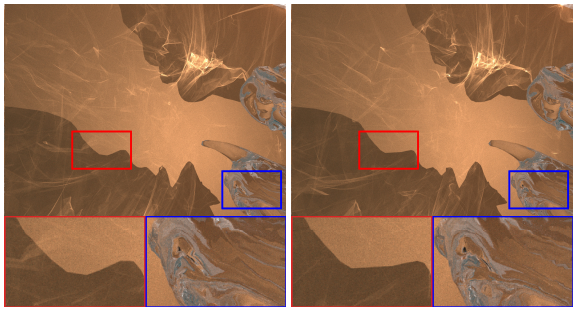
5.2. Hit Point Importance Factor

To examine the effect of different hit point importance factors we used the scene from Figure 7 with a different camera setting where indirectly illuminated parts of both the glass and glossy dragon are visible. A reference image is shown



(a) Scene Overview (unreduced)
(b) Scene Overview after reduction as in *(f)*.
(c) Reference
(d) Result with error quadrics only. RMSE: 9.063%
(e) One pass importance reduction. RMSE: 3.717%
(f) Two pass importance reduction. RMSE: 3.002%

Figure 7: The red rectangle in *(a)* and *(b)* indicates view area of following images (actual renderings, no close-ups). Meshes in *(d)–(f)* are reduced to 5%. Our approach is able to preserve caustics under drastic reduction from not directly visible objects. First importance pass was obtained from an already reduced scene (5%).



(a) Angular weight on
(b) Angular weight off

Figure 8: Ignoring the angular weight from Eq. (2) results in missing shadow details and jagged appearance of the glass.

in Figure 9a. Using a hit point importance of $\gamma = 0$ (thus ignoring direct camera ray hits) gives the best caustics (see 9b), but results in drastically degenerated meshes, especially for the glass object. A hit point importance of $\gamma = 4$ on the other hand preserves the directly visible meshes well, but gives bad caustics (see 9e). In this scene, we achieve good results with a hit point importance $\gamma = 1$ (see 9d). However, it is important to adjust this factor proportional to the number of pixels and the photon ray count per iteration.

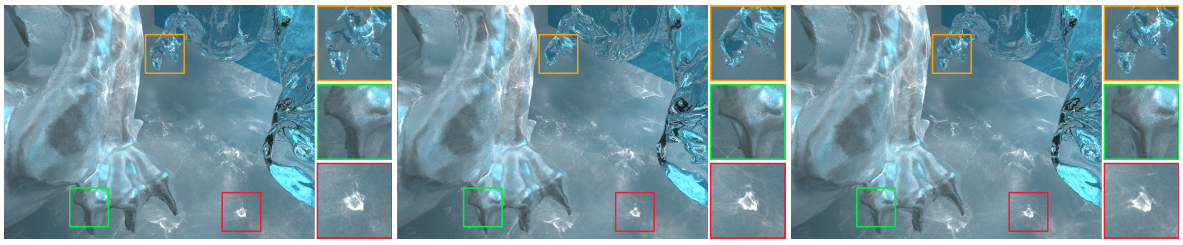
5.3. Multiple Importance Passes

We found that multiple importance passes can slightly improve the image quality if the initial importance pass operates on an already reduced scene. Figure 6d gives an example where the rendering did not profit from multiple importance passes since the initial pass was obtained from the original scene. There are both some features that are better and some that are worse preserved. The root mean squared error (RMSE) to the reference is even slightly higher. The ceiling detail in Figure 7f, for which the importance pass was based on an already reduced mesh, can profit from an additional importance pass. The caustic outlines were preserved more accurately, however the image appears darkened.

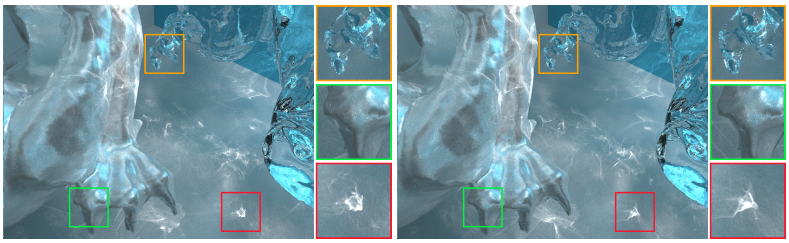
We experimented with more importance passes, but experienced similar behavior as in Figure 6d. This is because an initial reduction might remove crucial features that cannot be retained in a single importance pass, since there is no importance information from these features. To simulate such a scenario, we reduced the DRAGON CORNELL BOX to 1% and performed multiple importance passes, each with a 5% reduction output for the next stage (see Figure 10). The more importance passes are performed, the more the caustics resemble the reference image. For eight passes (Figure 10d), however, we see a similar phenomenon as for two passes in Figure 6d: Compared to less passes there are both improved and worsened features while the RMSE rises again.

5.4. Out-of-Core Scenario

Figure 11 shows renderings of an out-of-core scene with seven lights and a total vertex count of 15.8 million. Since changes in geometry are quickly noticeable in high-frequency illumination effects like caustics, we added several highly detailed glass objects: Four reduced Stanford Lucys each with 1.8 million vertices, four Stanford Buddhas each with 0.5 million vertices and 15 Stanford Dragons as used in the other scenes each with 0.4 million vertices. While raw geometry and texture data total in about 800 MB, the acceleration data structure needed for ray tracing adds

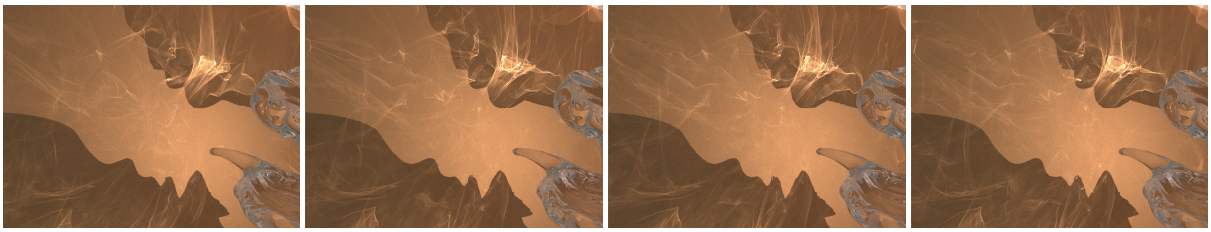


(a) Reference (b) Reduction to 5%, $\gamma = 0$ (c) Reduction to 5%, $\gamma = 0.5$



(d) Reduction to 5%, $\gamma = 1$ (e) Reduction to 5%, $\gamma = 4$

Figure 9: Comparison of different hit point importance factors γ. Images depict the same scene used as in Figure 7, but under a different camera setting. The importance pass operated on an unreduced scene.



(a) Single importance pass. RMSE: 4.864% (b) Two importance passes. RMSE: 4.174% (c) Four importance passes. RMSE: 3.942% (d) Eight importance passes. RMSE: 4.126%

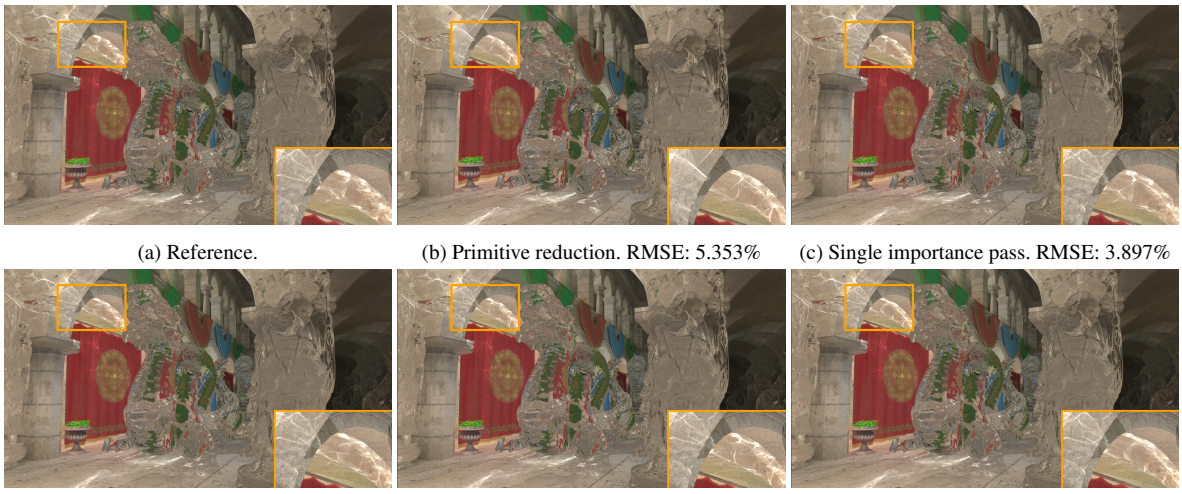
Figure 10: First importance pass was obtained from 1% reduced scene. Each additional pass, including rendering itself, operated on 5% scene. For other settings and reference image see Figure 6.

additional 1.8 GB. Thus, the scene does not fit into the graphics memory of our test system, which means that the GPU had to swap data to main memory. For the reference rendering we used the entire scene anyway, which resulted in very slow rendering timings (about 24h). Using our technique, we reduced the scene to 10% of its original vertex count (both initial reduction and importance pass). We applied decimations to glass objects only, as they are much higher tessellated than the rest of the SPONZA ATRIUM. The strongest image errors occur at the caustics at the arc above the red curtain and at the subtle caustics in the second floor, top middle in the images of Figure 11. With more importance iterations the artifacts do not change significantly in this scenario.

We expect larger scenes to perform similar, as long as the decimation algorithm is able to perform the demanded simplification ratios. Also, the initial reduction needs to contain all major mesh features, since otherwise our conservative importance remapping heuristic may not be able to retrieve those, even with multiple importance passes.

5.5. Performance and Memory Consumption

Table 1 shows absolute preparation times and compares the speed-ups in our reduced scenes compared to the original scenes. The preparation timings consist of the runtime of the first two importance passes and the subsequent reduction which takes the same time for both passes. The "Rendering Speed-Up" column shows average speed-up factors per rendering iteration. An importance iteration is slightly slower than a normal photon mapping iteration because of the scattered write into the global importance buffer. However, since there are usually many more photon iterations than importance iterations, it is not significant for total performance. Naturally, our technique excels in the out-of-core example since GPU paging is eliminated. In most cases there are still speed-ups for smaller scenes, however they remain small due to the logarithmic cost of ray tracing. Simple reduced scenes



(a) Reference. (b) Primitive reduction. RMSE: 5.353% (c) Single importance pass. RMSE: 3.897%

(d) Two importance passes. RMSE 3.796% (e) Four importance passes. RMSE 3.736% (f) Eight importance passes. RMSE 3.702%

Figure 11: Out-of-core scene with 15.8 million vertices. Rendered at 1080p with 1 billion photon rays, 114,688 per iteration. Reduction to 10% for images *(b)–(f)*. The hit point importance is set to $\gamma = 0.1$.

Scene	reduction to	Preparation			Rendering Speed-Up		
		1st Imp.	2nd Imp.	Reduct.	Simple	1 Pass	2 Passes
DRAGON CORNELL BOX (Fig.6)	10%	55 *s*	59 *s*	11 *s*	1.10×	1.07×	1.07×
DRAGON CORNELL BOX (Fig.10)	5%, 1% init	29 *s*	52 *s*	11 *s*	—	1.23×	1.20×
CEILING DETAIL (Fig.7)	5%, 5% init	44 *s*	67 *s*	14 *s*	1.28×	1.05×	1.05×
IMPORTANCE STUDY (Fig.9d)	5%	79 *s*	78 *s*	13 *s*	—	1.02×	1.02×
OUT-OF-CORE SPONZA (Fig.11)	10%, 10% init	120 *s*	198 *s*	78 *s*	10.06×	8.56×	8.85×

Table 1: *Preparation*: Absolute duration of importance acquisition and subsequent reduction (constant for both importance passes). *Rendering Speed-Up*: Rendering time speed-up compared to unreduced scene (using same photon/hit point ray count).

are faster because many ray paths in our illumination-driven mesh reduction have a similar complexity in comparison to the original scene. However, they are not able to preserve caustics for the same reduction rate.

The importance buffer stores a value for each mesh vertex, which increases the memory consumption during the importance pass. For the decimation, importance is projected to the original mesh using the collapse history. For out-of-core scenes this increases the overall VRAM load during the importance pass. The peak RAM consumption of our application during the decimation process of the out-of-core scene was about 5.3GB. While the multi-threaded importance preparation and mesh reduction took only 4.7 minutes (single importance pass) in the out-of-core test, this process may be more significant for larger scenes.

6. Conclusions and Future Work

We introduced a novel importance-based mesh reduction method for photon mapping-based rendering. Our algorithm enables high mesh reduction with only slight loss of image quality. This reduces memory demands, which not only benefits rendering performance, but also makes it possible to render scenes which were previously considered out-of-core.

We stored the importance per vertex. Further investigation is needed to compare this with other storage methods like textures or volumes. As additional improvement, it would be possible to replace parts of the scene with proxy geometry, depending on their importance value. This is especially useful for larger out-of-core scenes containing many objects. Possibly, an initial importance pass could operate entirely on proxy geometry and refine/replace the meshes iteratively. Our work did not examine how texture information can be reduced using our new importance measure, which we consider an orthogonal problem. We improved the rendering of static scenes from a fixed viewpoint. It would be interesting to study how an initially high-tessellated scene could be optimized for a range of different viewpoints and probably even varying illumination conditions. In this light, temporal coherence is another aspect to study, including coherent distribution of a (possibly) fixed triangle budget, and simplification per frame or for a group of frames. We expect further improvement by casting shadow photons from blockers and connecting them to hit points to explicitly preserve visible shadow edges. Our experiments show that the "visible radiance" importance heuristic sometimes misleads the simplification, leading to slightly larger RMSE in the image af-

ter multiple importance passes. Finding an optimal criterion that directly measures the "visible change" in the image after geometry simplification is ultimately what we strive for in further research. A fast evaluation of such an error criterion is challenging since it requires quick retrieval of all paths that go through the two triangles that participate in the edge collapse, as well as all paths that would go through the new triangles. We would also like to explore how our technique integrates into other Monte Carlo based methods, e.g., bidirectional path tracing.

Acknowledgements

We thank Christian Rössl for his insights on mesh reduction and the anonymous reviewers for their helpful comments. The Dragon, Lucy and Buddha models are courtesy of the Stanford University. The Crytek SPONZA scene was created by Frank Meinl. The work was partially supported by the German Research Foundation (DFG) grant GR 3833/3-1.

References

[BBS*09] BUDGE B. C., BERNARDIN T., STUART J. A., SENGUPTA S., JOY K., OWENS J. D.: Out-of-core data management for path tracing on hybrid resources. *Computer Graphics Forum (Proc. Eurographics) 28*, 2 (2009), 385–396. 1

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. AK Peters, 2010. 2

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: Openmesh: A generic and efficient polygon mesh data structure. In *OpenSG Symposium* (2002). 4, 6

[CB04] CHRISTENSEN P. H., BATALI D.: An irradiance atlas for global illumination in complex production scenes. In *Proc. Rendering Techniques* (2004), pp. 133–142. 3

[Chr03] CHRISTENSEN P. H.: Adjoints and importance in rendering: An overview. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2003), 329–340. 3

[CLF*03] CHRISTENSEN P. H., LAUR D. M., FONG J., WOOTEN W. L., BATALI D.: Ray differentials and multiresolution geometry caching for distribution ray tracing in complex scenes. *Computer Graphics Forum (Proc. Eurographics) 22*, 3 (2003), 543–552. 2

[DKHS14] DAVIDOVIČ T., KŘIVÁNEK J., HAŠAN M., SLUSALLEK P.: Progressive light transport simulation on the GPU: Survey and improvements. *ACM Trans. Graph. 33*, 3 (2014), 29:1–29:19. 1, 2, 5

[FMH05] FRADIN D., MENEVEAUX D., HORNA S.: Out-of-core photon mapping for large buildings. In *Proc. Eurographics Symposium on Rendering* (2005), pp. 65–72. 3

[GBBK04] GUTHE M., BORODIN P., BALÁZS A., KLEIN R.: Real-time appearance preserving out-of-core rendering with shadows. In *Proc. Rendering Techniques* (2004), EGSR, pp. 69–79. 2

[GG14] GÜNTHER T., GROSCH T.: Distributed out-of-core stochastic progressive photon mapping. *Computer Graphics Forum 33*, 6 (2014), 154–166. 1, 2, 3

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proc. Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, pp. 209–216. 2, 3, 6

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (2012), 192:1–192:10. 2

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 28*, 5 (2009), 141:1–141:8. 2

[HJ10] HACHISUKA T., JENSEN H. W.: Parallel progressive photon mapping on GPUs. In *SIGGRAPH Asia Sketches* (2010), ACM, pp. 54:1–54:1. 2

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 27*, 5 (2008), 130:1–130:8. 2

[HPJ12] HACHISUKA T., PANTALEONI J., JENSEN H. W.: A path space extension for robust light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (2012), 191:1–191:10. 2

[Jen95] JENSEN H. W.: Importance driven path tracing using the photon map. In *Proc. Rendering Techniques*. Springer, 1995, pp. 326–335. 3

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Rendering Techniques' 96*. Springer, 1996, pp. 21–30. 2, 3

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proc. Computer Graphics and Interactive Techniques* (1986), SIGGRAPH '86, pp. 143–150. 2

[KGH*14] KŘIVÁNEK J., GEORGIEV I., HACHISUKA T., VÉVODA P., ŠIK M., NOWROUZEZAHRAI D., JAROSZ W.: Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4 (2014), 103:1–103:13. 1

[KSS98] KLEIN R., SCHILLING A., STRASSER W.: Illumination dependent refinement of multiresolution meshes. In *Proc. Computer Graphics International* (1998), pp. 680–687. 2

[KW00] KELLER A., WALD I.: Efficient importance sampling techniques for the photon map. In *Proc. Vision Modelling and Visualization* (2000), pp. 271–279. 3

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. *Proc. Compugraphics* (1993), 145–153. 2

[MG10] MENZEL N., GUTHE M.: Towards perceptual simplification of models with arbitrary materials. *Computer Graphics Forum (Proc. Pacific Graphics) 29*, 7 (2010). 2

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: a general purpose ray tracing engine. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 66:1–66:13. 2, 5

[PP98] PETER I., PIETREK G.: Importance driven construction of photon maps. In *Proc. Eurographics Workshop on Rendering* (1998), Springer-Verlag, pp. 269–280. 3

[RV93] RUSHMEIER P., VEERASAMY: Geometric simplification for indirect illumination calculations. In *Proc. Graphics Interface* (1993), pp. 227–236. 2

[SFD09] STICH M., FRIEDRICH H., DIETRICH A.: Spatial splits in bounding volume hierarchies. In *Proc. High Performance Graphics* (2009), pp. 7–13. 6

[SW00] SUYKENS F., WILLEMS Y. D.: Density control for photon maps. In *Proc. Eurographics Workshop on Rendering* (2000), Springer Wien, pp. 23–34. 3

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proc. Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, pp. 65–76. 2