DOI: 10.1111/cgf.14008

COMPUTER GRAPHICS *forum*
Volume 39 (2020), number 6 pp. 155–166

Preserving Shadow Silhouettes in Illumination-Driven Mesh Reduction

F. Bethe, J. Jendersie and T. Grosch

Department of Informatics, TU, Clausthal-Zellerfeld, Germany
{florian.bethe, johannes.jendersie, thorsten.grosch}@tu-clausthal.de

Abstract

A main challenge for today's renderers is the ever-growing size of 3D scenes, exceeding the capacity of typically available main memory. This especially holds true for graphics processing units (GPUs) which could otherwise be used to greatly reduce rendering time. A lot of the memory is spent on detailed geometry with mostly imperceptible influence on the final image, even in a global illumination context. Illumination-driven mesh reduction, a Monte Carlo–based global illumination simulation, steers its mesh reduction towards areas with low visible contribution. While this works well for preserving high-energy light paths such as caustics, it does have problems: First, objects casting shadows while not being visible themselves are not preserved, resulting in highly inaccurate shadows. Secondly, non-transparent objects lack proper reduction guidance since there is no importance gradient on their backside, resulting in visible over-simplification. We present a solution to these problems by extending illumination-driven mesh reduction with occluder information, focusing on their silhouettes as well as combining it with commonly used error quadrics to preserve geometric features. Additionally, we demonstrate that the combined algorithm still supports iterative refinement of initially reduced geometry, resulting in an image visually similar to an unreduced rendering and enabling out-of-core operation.

Keywords: ray tracing, rendering, polygonal mesh reduction, modelling, Monte Carlo techniques, methods and applications

ACM CCS: I.3.5 [Computer Graphics]: Geometric algorithms, languages, and systems; I.3.7 [Computer Graphics]: Raytracing

1. Introduction

The level of realism displayed in today's production movies largely stems from physically based light transport simulation algorithms, with many improvements regarding their quality as well as acceleration on both traditional CPUs and graphics processing units (GPUs) [DKHS14]. However, the use of global illumination for realistic images comes hand-in-hand with large scenes featuring detailed geometry; Christensen *et al*. [CFS*18] give an overview of the structure and typical workload of Pixar's production renderer *RenderMan*. The memory requirements make it difficult to use GPUs for acceleration and may prove to be *out-of-core* even for main memory, inspiring the need for algorithms capable of operating on scenes not fitting into the target memory.

To tackle this, two general approaches exist: either the light transport simulation is able to work on partial scenes [BBS*09] [ENSB13] [GG14], possibly by swapping in geometry or partitioning the scene, or the scene has to be reduced in size until it fits into memory. On the level of an individual mesh, the corresponding error metric is often the Hausdorff distance, maintaining a minimal deviation between the original and reduced mesh in euclidean space. Finding the reduced mesh is a computationally intensive problem and often the global optimum is traded in exchange for feasibility; locally defined reduction criteria such as *error quadrics* [GH97] may end up in local minina, but it is always possible to compare two reduced meshes and establish their quality with respect to the Hausdorff metric.

However, once light sources and especially full global illumination are added to the equation, the definition of error shifts: while a purely geometric error metric remains interesting, it no longer coincides with the intention behind scene simplification. To capture this intent, the error metric has to take into account the possible scene-wide effects of geometry changes in a global illumination context. One possibility is to define the metric on the measure function of the camera; the RMSE of two converged renderings would be such a metric. This has multiple downsides: not only is accurately

(a) Ground truth (Path tracer) (b) Reduction as in [RGG15] (c) Our approach

Figure 1: *Reduction of ∼1.2 million triangles by 95%. Our approach preserves the meshes of shadow casting but otherwise not visible meshes.*

rendering global illumination an expensive operation, it is also a noisy one; unless the image is sufficiently converged, the root mean square error (RMSE) between two renderings may be dominated by noise. As a side effect, we are now shackled to a given algorithm, which is not inherently a disadvantage. On the contrary, geometric reduction should not retain features that have no effect on the rendering; an example of such features may be (missing) caustic paths for a path tracer. Similarly to the Hausdorff distance, directly using metrics defined on final renderings to guide geometry reduction is infeasible, requiring local metrics.

In this paper, we contribute the following:

- A path tracing-based formulation of illumination-driven mesh reduction [RGG15]
- Preservation of shadow silhouettes for point and area light by providing a definition for an object's *shadow silhouette importance* in the context of next-event estimation *(NEE)*
- An improved error metric for collapse priority by combining importance with *error quadrics* [GH97] to prevent excessive mesh degeneration

To examine the reduction quality, we test multiple scenes and observed the progression with more aggressive reduction goals. We furthermore discuss different blendings between the original metric and our extension as a quantification of image feature preference in excessive reduction scenarios. Finally, we give an overview of the reduction performance and additional overhead of our extension.

2. Related Work

As a solution to the rendering equation first formulated in the same work, Kajiya [Kaj86] introduced a Monte Carlo integration-based algorithm called *path tracing*. To alleviate some of its issues numerous improvements such as *bi-directional path tracing* [LW93] [Vea98] have been suggested, aimed at both reducing the variance introduced by its stochastic nature and its inability to effectively sample certain kinds of light paths.

A different approach has been taken by *photon mapping* [Jen96], which distributes and stores photons in a scene using spatial data structures called photon maps. These are queried in a second pass, resulting in a radiance estimate by accounting for all stored photons within a region around camera ray hits. *Progressive photon mapping* [HOJ08] and *stochastic progressive photon mapping (SPPM)* [HJ09] improve this technique by progressively shrinking the search radius, the latter being the algorithm of choice for Reich *et al.* [RGG15]. However, we target vanilla path tracing, which is still often used due to its simplicity and GPU scaling.

2.1. Importance

A variety of importance techniques used in rendering exist, of which Christensen [Chr03] gives an overview. As a general concept they see importance as an adjoint to different radiometric units. In photon mapping so-called *importons*, first mentioned by Jensen [Jen96], may guide the distribution of photons to focus on the important parts of the scene. Reich *et al.* [RGG15] use this notion of importance to guide their mesh reduction, the general idea being that geometry interacting with many contributing light paths has a high impact on the rendered image. In their implementation, they use SPPM since caustics (i.e. focused light paths) have a high influence on the light distribution in the scene and are not covered by standard, lighting-independent reduction algorithms.

2.2. Shadow algorithms

Shadow algorithms for rasterizers can be grouped into *shadow map*–based [Wil78] and *shadow silhouette*--based [Cro77] approaches; Eisemann *et al.* [ESAW16] give an overview of existing techniques. The latter generally works by extending a volume from a light source and through an object's *silhouette*, defined as the borders between faces oriented towards the light source and those facing away. Since the resulting *shadow geometry* can be quite large, hybrid approaches such as *shadow silhouette maps* [SCH03, CD04] augment shadow maps with information about the shadow silhouette to reduce aliasing.

Jensen and Christensen [JC96] propose extending the photon map to incorporate shadow photons, which are stored in shadowed locations. By checking whether a mixture of regular and shadow photons exist in the merge radius, they restrict shadow rays to pixels lying in the border regions partially shadowed.

2.3. Mesh reduction

Botsch *et al*. [BKP*10] give a general overview of mesh processing algorithms. Hoppe [Hop96] introduces a representation for smooth transitions between multiple level-of-details based on *mesh optimization* [HDD*94]. Perception-guided approaches to mesh simplification have been explored by Guthe *et al*. [GBBK04] as well as Menzel and Guthe [MG10]; both ignore global illumination effects and are thus ill-suited for retaining image fidelity.

Garland and Heckbert [GH97] define an algorithm based on *quadric error metrics* which operates solely on a geometric level, ignoring any context given by surrounding meshes or the rendering algorithm. It minimizes the square of sums of the projected distance to all faces affected by a collapse.

2.4. Illumination-driven mesh reduction

We first give an overview of the existing *importance-based reduction framework* following the reduction pipeline depicted in Figure 2. It is important to note that *importance* in this context does not denote the adjoint to radiance, but rather an empirical quantity loosely based on *contribution*.

At the heart of the algorithm stands *computing importance*. To attribute visual importance to mesh geometry, illumination-driven mesh reduction [RGG15] accrues *importance*$\mathcal{I}(\mathbf{T})$ per triangle $\mathbf{T}$ whenever the light transport algorithm finds a closed path between a light source and the camera. This process is based on *SPPM*: for each photon in the merge radius, the importance of the triangle gets increased. To keep directly visible surfaces intact, viewpath vertices also add importance, regardless of whether a photon was merged. This may be repeated iteratively to reduce noise in these estimates. The importance sum $\mathcal{I}(\mathbf{M}) = \sum_{\mathbf{T}\in\mathbf{M}} \mathcal{I}_T(\mathbf{T})$ over all triangles of a mesh $\mathbf{M}$ is then used to distribute *reduction factors* inside the scene. Each mesh is decimated via edge collapse based on the importance $\mathcal{I}(V) = \frac{1}{|\text{triangles}(V)|} \sum_{\mathbf{T}\in\text{triangles}(V)} \mathcal{I}(\mathbf{T})$ of a vertex V and its neighbours, prioritizing collapses of low-importance vertices and gradients. Note that it is trivial to extend the framework to incorporate quads, a geometric primitive often preferred by artists. The *local importance estimate* is defined as

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

where x_{ij} and x_{ij-1} form a path segment and $n_{\mathbf{T}}$ is the surface normal at x_{ij}. With this, the importance of a triangle is given as

$$\mathcal{I}_{SPPM}(\mathbf{T}) = \frac{1}{area(\mathbf{T})} \sum_{i=1}^{N} \left(\gamma \sum_{j=1}^{|\bar{z}_i|} \hat{I}(z_{ij}, \mathbf{T}) + \sum_{j=1}^{|\bar{y}_i|} \hat{I}(y_{ij}, \mathbf{T}) \right), \quad (2)$$

where $\bar{z}_i$ and $\bar{y}_i$ are the i-th view and light sub-path, respectively, N is the number of all paths, j indicates the vertex on the sub-path and γ serves as a user-defined weight between viewpath and photon importance.

The accumulated per-face importance is then taken as input for the *importance mapping* step. To understand its necessity, we have to take a step back and recall the purpose of the framework: reducing a scene's geometry, which ordinarily may be *out-of-core*, to fit into main memory or VRAM, depending on the concrete application. The previous step of importance accretion however requires the scene to be renderable. To achieve this without having to resort to out-of-core rendering, Reich *et al*. proposed a pre-processing step of *initial geometric reduction*, which uses any regular view-independent technique; the only requirement is that it allows tracing an edge-collapse history. Importance computation is then performed on this pre-reduced scene and the mapping step projects importance onto all vertices in the *original* scene via the collapse history. *Importance reduction* then takes the original scene mesh by mesh and performs edge collapses, prioritizing low importance vertices/edges. Note that the number of collapses per mesh is chosen proportional to the *importance sum*$\mathcal{I}(\mathbf{M})$; this is a critical aspect to the framework's out-of-core capabilities.

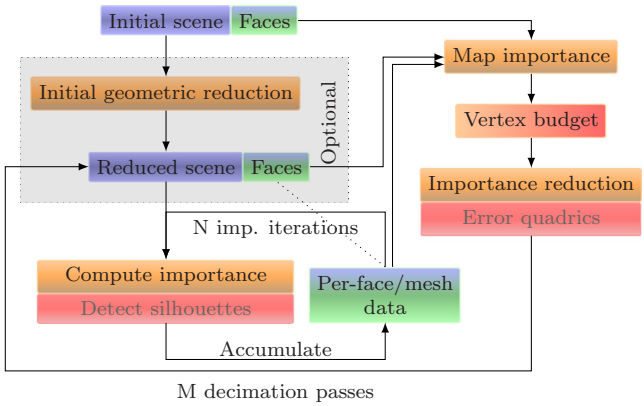


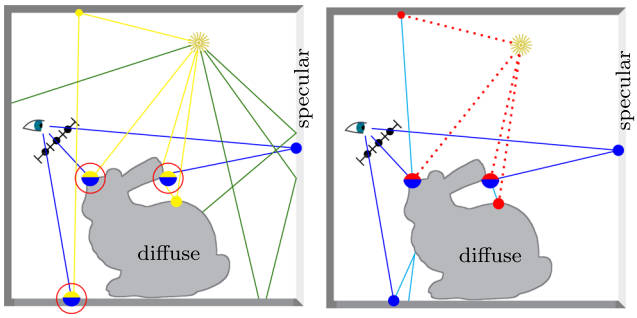
Figure 2: *Overview of the reduction pipeline. Components additional/modified to Reich* et al. *[RGG15] are marked red.*

3. Integrating Shadows into the Illumination-Driven Reduction Framework

3.1. Obtaining importance with path tracing

The approach by Reich *et al*. [RGG15] is mostly geared towards lighting scenarios involving caustics generated by complex glass objects. This reflects in the chosen light-simulation algorithm; photon mapping generally performs well for LS^+DE paths. However, in scenes mostly dominated by diffuse interactions a path tracer with next event estimation is often preferential due to its lower memory requirements, faster iteration time and ease of implementation.

To utilize the importance-based mesh reduction, we first give a notion of the importance acquisition in a way suitable for a path tracer. The original paper defines the importance computation in terms of $\hat{I}$, which scales the importance a viewpath vertex adds to a triangle with the angle between incident direction and surface normal, putting emphasis on silhouettes. We now wish to obtain a similar quantity with a path tracer. The viewpath is relatively straightforward and only requires a small adaption. SPPM classifies materials into either diffuse, glossy or specular and stops upon encountering a diffuse surface, whereas path tracing does not. Thus, we need to keep track of the BxDF sampled along the path



(a) SPPM [RGG15] (b) PT

Figure 3: *Difference in importance acquisition: Viewpath vertices ( ) increase importance for both methods directly, unless they bounced off a diffuse surface ( ). While the SPPM-based acquisition traces photons and increases importance (,) for successful merges along the photon's path ( ), the path tracing-based variant increases importance (,) upon successful next-event estimations ( ). Photons that did not get merged ( ) do not contribute anything, while for PT sub-paths after a diffuse interaction ( ) may still contribute indirect importance ( ) if later vertices contribute radiance.*

and reduce the amount of importance added; we defer the search for a proper weighting for glossy reflections to future research.

Obtaining the importance from the photons is more complicated. Reich *et al*. [RGG15] take the number of photons within the merge radius at a viewpath segment, which is proportional to its flux. However, it also increases the importance for all triangles with which the photon has interacted previously. Instead of the flux we use the *differential irradiancedE*, since a path tracer cannot (easily) obtain the flux at a given point. We determine the direct differential irradiance dE_{NEE} at each viewpath segment via next event estimation and estimate the indirect differential irradiance dE_{trace} by tracking the path back upon termination. This presents an issue: since multiple viewpath hits of a triangle also incur multiple estimates of its irradiance, we need to store the view- and lightpath importance separately alongside the number of irradiance estimates we obtained. The final importance of a triangle is then

$$\mathcal{I}_{PT}(\mathbf{T}) = \frac{1}{area(\mathbf{T})}\Big[\gamma \sum_{i=1}^{M} \hat{I}(z_i, \mathbf{T}) + \frac{1}{M}\sum_{i=1}^{M}\left(\hat{J}(dE_{NEE}(z_i), \mathbf{T}) + \hat{J}(dE_{trace}(z_i), \mathbf{T})\right)\Big], \tag{3}$$

where M is the number of all viewpath *vertices*. $\hat{J}(E, \mathbf{T})$, similarly to $\hat{I}$, weights the differential irradiance dE with the angle between their incident direction and surface normal, respectively; see Figure 3 for an example in a simple scene. In the following, we assume all importance quantities as defined for the path tracer.

3.2. Shadow silhouettes as visual feature

While there currently is no closed-form error function for mesh reduction in the context of global illumination, we can state what it must encompass. From the perspective of path space, there are four possibilities for a path segment after a reduction happened:

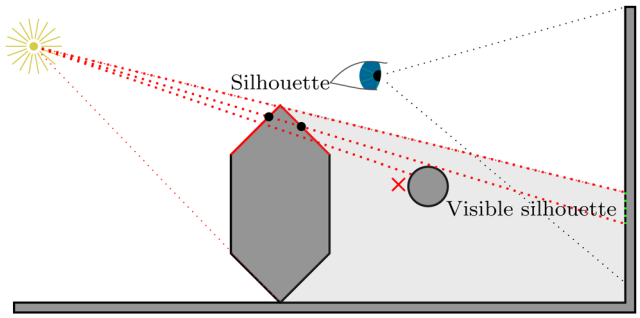


Figure 4: *Silhouette detection: Direct shadows visible from the camera check if the blocking face is part of the blocker's silhouette*

1. the segment did not intersect the affected geometry before and after reduction;
2. the segment did not intersect the affected geometry before, but does after reduction (concave surface 'popping');
3. the segment did intersect the affected geometry before and after;
4. the segment did intersect the affected geometry before, but not after reduction.

This taxonomy is not complete, as it ignores new path segments that are possible only after the reduction, but it helps understanding why the current illumination-driven framework is incomplete: its error function accounts for Possibilities 1 and 3, but not always for the remaining ones. A path segment blocked by an *occluder* does not indicate the necessity to maintain the geometric structure of said occluder according to the algorithm. While this is not an issue (and in fact, an advantage) for core shadows, it causes noticeable disruptions in the outline of shadowed image regions and may even cause the occluder to degenerate into a tetrahedron, as illustrated by the right bunny shadow in Figure 1(b). Other artefacts such as popping of concave mesh parts are partially mitigated by the use of multiple *decimation passes*.

We propose to extend the framework to specifically account for direct shadow silhouettes, the most noticeable issue. At its core, we extend a light sub-path whenever it hits an object's *silhouette*. We use the definition utilized by shadow volumes [Cro77] which states that a *silhouette edge* is the edge between two neighbouring faces, one oriented towards and the other oriented away from the light source. Note that an area light may have multiple silhouette 'rings' depending on its extent. To *detect silhouettes* we utilize the shadow rays of next-event estimations. Whenever such an estimation fails, we know that the point is not directly illuminated by a given light source. To determine whether it is a mesh's silhouette blocking the light, it is unfortunately not sufficient to check if the point is shadowed at all (usually implemented as an *any-hit* intersection test). Instead we attempt to find the first two faces between the light source and the shaded point, as depicted in Figure 4. It is then sufficient to check if the faces share vertices, in which case we have found a shadow silhouette; note that index buffer-based renderers may need to compare the vertices' positions if they use per-vertex attributes. Since the shadow silhouette may not actually be visible at our viewpath vertex, we then need a third check to see if there is blocking geometry between the backside of the silhouette and the viewpath (see Figure 5 for an example scene).

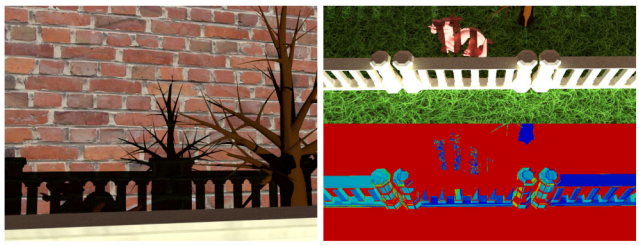


Figure 5: *Partial shadowing: importance [from red (high) to blue (low)] is only attributed to the mesh actually casting a shadow; in this case the dragon has low importance where it itself is shadowed by the fence.*

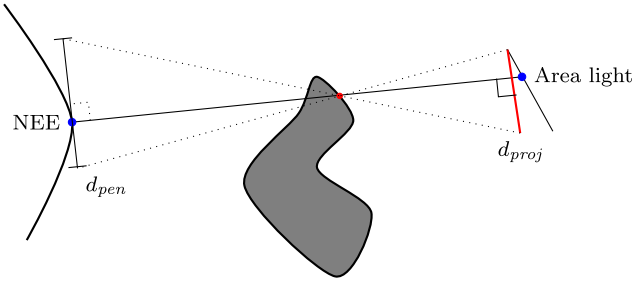


Figure 6: *Projecting the size of an area light along the shadow ray yields a rough estimate of the penumbra size.*

To integrate this into the importance framework, we define the *silhouette importance* of a triangle as

$$\mathcal{I}_{sil}(\mathbf{T}) = \sum_{i=1}^{N} s(z_i) \cdot L_{NEE}(z_i) \cdot \left(\frac{1}{1+d_{pen}}\right)^2, \tag{4}$$

where z_i is a viewpath vertex, $s(z_i)$ is the remaining *sharpness* of the viewpath at the vertex, $L_{NEE}(z_i)$ is the radiance estimate from the next-event estimation at z_i blocked by an object, and d_{pen} is the estimated penumbra size, as depicted in Figure 6. In our context, we define a heuristic for the sharpness of a viewpath as

$$s(z_i) = \prod_{j=1}^{i} \left(\frac{2}{1+e^{-0.1\cdot\rho(z_j)}} - 1\right), \tag{5}$$

where $\rho(z_j)$ is the BRDF of a prior path vertex. The overall shape ensures that we ignore silhouettes that are visible only via diffuse or glossy reflections, but has been determined entirely empirically without any theoretical basis.

The silhouette importance heuristic has been chosen for various reasons. First, the importance of a shadow silhouette has to depend on its *visibility*, that is, how much the local radiance function would change should it be altered. We approximate this by taking the would-be radiance difference in case of a successful NEE multiplied with a measure of radiance sharpness; in our tests we approximated this by manually classifying materials in diffuse, glossy and specular akin to SPPM, but more sophisticated weightings are possible. Secondly, extended light sources naturally may have multiple silhouette 'rings' depending on the exact location of the NEE. Because the visual disturbance of a soft shadow in case of reduction is lower than that of a hard one, the importance of these rings has to be weighted down. Since it is not a trivial task for a path tracer to determine whether an NEE is fully, partially or not shadowed at all, we weight the shadow importance with the inverse of the expected penumbra size at the shadowed point. For this, we assume the worst case (i.e. the shortest side of the light's primitive) and projected it onto the shadowed surface via the intercept theorem (Figure 6). This heuristic assumes that the light source is mostly symmetrical and the occluder intersection is close to the object's true shadow edge, which results in worse estimates for bar-like lights and large objects relative to the light's area.

Figure 7: *Detected shadow silhouettes (marked in white) on a mesh with ∼183k triangles at 1, 10 and 100 importance iterations. While not all silhouette triangles are found, the remapping of importance from a lower resolution mesh increases the effective number of silhouette triangles found.*

To implement the pipeline so far we require additional memory. For each face, we have to track several quantities: its viewpath importance $\hat{I}(z_i, \mathbf{T})$, irradiance $E(z_i)$ and sample counter to estimate its average irradiance, and the silhouette importance $\mathcal{I}_{sil}(\mathbf{T})$. The latter may simply be added to the viewpath importance, leaving two floats and an int per face as memory overhead. However, there are some issues unaccounted for. One issue, largely solved by more importance iterations, is: if the shadow-casting mesh has a high level of detail, it may be possible to miss the silhouette. This may happen when the projected area of a silhouette face onto the shadowed surface is smaller than the footprint of the viewpath we perform the shadow test for. Figure 7 visualizes the detected silhouette faces at increasing numbers of samples. Due to jittering the starting point on each pixel, a slightly different shadow ray is cast every iteration, increasing the likelihood of detecting the full silhouette overall.

3.3. Distribution of reduction factors

A central aspect of importance-based mesh reduction is the distribution of a vertex budget across the scene; meshes located in remote corners or not directly illuminated will generally need less vertices than those in the centre of attention. Since the pipeline so far computes this budget based on $\mathcal{I}(\mathbf{M})$*without* silhouette importance factored in, invisible but occluding meshes still get reduced too much. Merely summing $\mathcal{I}_{sil}(\mathbf{T})$ as well does not help either; the two

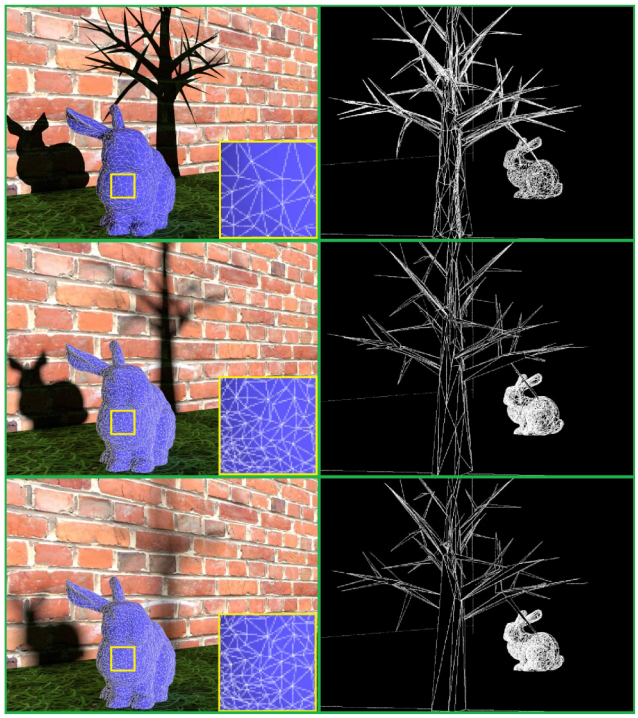


Figure 8: *Reduction by 97.5% (100 importance iterations) with different light sizes. On the left is the reduced scene blended with its wireframe, on the right the respective wireframe from a different camera angle. The soft shadow of the tree requires less faces for reasonable fidelity, leaving more for the foreground bunny.*

quantities are not directly comparable due to the chance of missing silhouette faces.

To achieve a fair distribution, we split the budgeting into two parts: the already known importance sum $\mathcal{I}(\mathbf{M})$ as well as a *shadow importance sum*

$$\mathcal{I}_{SS}(\mathbf{M}) = \delta \cdot \sum_{i=1}^{N} s(z_i) \cdot dE_{NEE}(z_i) \cdot \left(\frac{1}{1+d_{pen(i)}}\right)^2, \qquad (6)$$

where $dE_{NEE}(z_i)$ is the differential irradiance of the occluded NEE at z_i and δ is a user-defined weighting factor indicating preservation preference. The definition is similar to the silhouette importance in Equation (4), but instead of radiance we use differential irradiance similarly to $\mathcal{I}(\mathbf{M})$. This heuristic is not exact: the influence of a shadow silhouette should be proportional to its visible perimeter length, which for area lights becomes an area instead. Both are difficult to properly estimate, which is why we instead approximate it by integrating $\mathcal{I}_{SS}(\mathbf{M})$ for all occluded NEEs; this deviates more for highly nested silhouettes, but performed well in our tests.

Following from this, we have to track one additional quantity, but only *per mesh*, not per face. Figure 8 shows the effect of different light sizes on vertex distribution: note the increased detail of the visible bunny for larger penumbras of the tree due to the different budgeting.

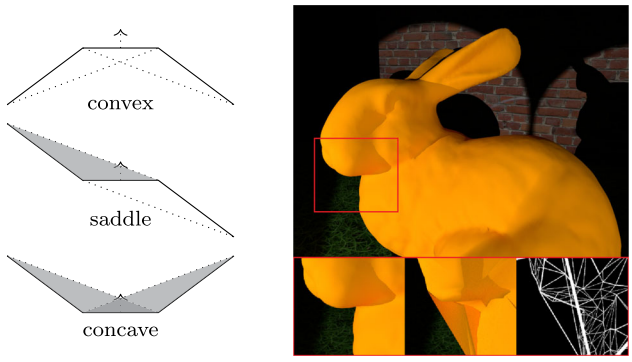


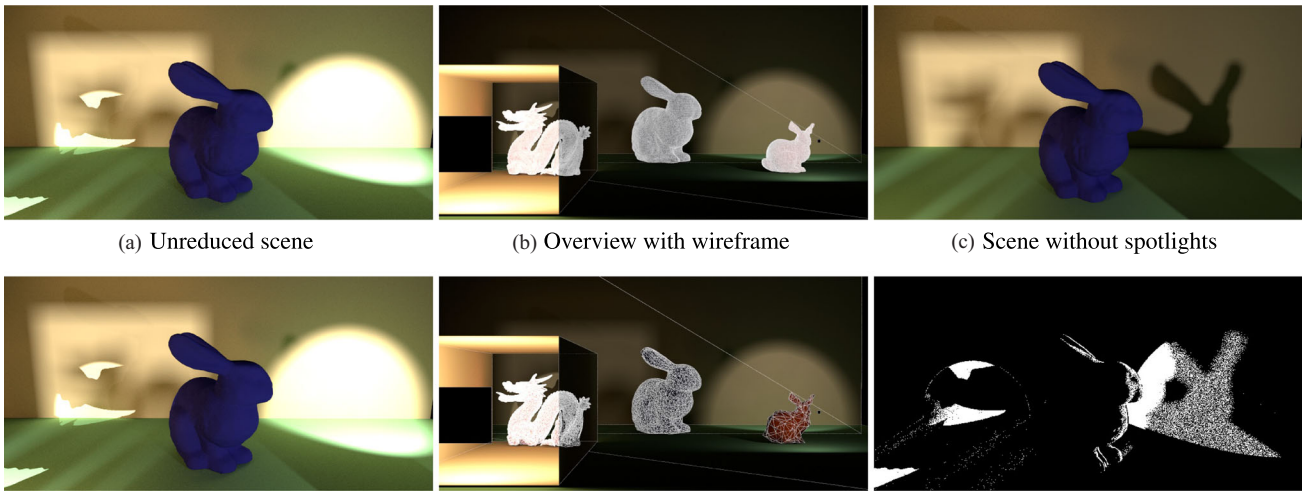
Figure 9: *Left: the three possible cases when collapsing an edge; only the convex case does not expand the mesh, which may lead to visible changes to the object's silhouette as shown on the right: a previously hidden, low importance face collapses into line-of-sight.*

The pipeline so far is still overly preserving invisible yet occluding meshes in some scenarios though, namely, when a bright light source *outshines* the shadow. To avoid this, we ignore the contribution to $\mathcal{I}_{SS}(\mathbf{M})$ as well as $\mathcal{I}_{sil}(\mathbf{T})$ when the *ratio* between $L_{NEE}(z_i)$ and $L(z_i)$ drops below 2%; this follows from the *Weber-Fechner law* [Fec58] [Web34]. It is important that this ratio has to be taken with tone-mapping applied. Figure 10 shows a more complex lighting scenario with two area light sources, which are partially outshined by two spot lights; note the largely absent silhouette importance on the bunny to the right. Unfortunately, this technique requires at least one additional NEE to estimate light from sources other than the shadowed one, which may however be used to reduce the noise level of the overall importance.

3.4. Reducing geometric errors

As noted in Section 3.2, the framework requires multiple decimation passes to suppress popping artefacts. They are a consequence of an information gap in the error metric; if multiple faces are never hit by view or light paths, then the collapse order may as well be random. Figure 9 illustrates the possible consequences for different types of local topology. This effect is most notable in non-transparent meshes with steep concavities, such as a tree with branches. A simple measure to prevent popping would be to restrict collapses to purely convex ones. However, this would restrict the decimation too much: low-importance regions may not have convex collapses at all or would need to propagate it slowly from other convex areas. To decrease the risk of incurring these artefacts in the first place, we instead propose to combine the importance-based error metric with a geometric one, which has the added benefits of more well-behaved reduction even in unproblematic unimportant mesh parts. Quadric error metrics as used by Garland and Heckbert [GH97] seem to be suitable. They define the error an edge collapse causes as the square of the distance to the planes defined by all faces affected by a collapse.

As mentioned by Reich *et al.*, combining the two in a theoretically sound way proves difficult as the quadrics error is measured in world space distances, whereas importance is derived from path



(a) Unreduced scene (b) Overview with wireframe (c) Scene without spotlights

(d) Reduced by 90% (e) Reduced scene overview (f) Areas with outshined shadows

Figure 10: *Reduction by 90% (100 importance iterations) of a scene with two spot lights which outshine the (soft) shadows of two area lights. The bottom right shows pixels where no shadow silhouette importance was accrued due to outshining light sources.*

space. However, we do not require a correct combination of the two, as our primary goal is to minimize the geometric error for parts of a mesh we already deemed unimportant. We thus propose the *geometric mean* between a vertex' ring importance as defined by Reich *et al*. and the quadric error as a heuristic for prioritizing collapses:

$$P(V, V_1) = \sqrt{V_1^T (Q(V) + Q(V_1)) V_1 \sum_{u \in N(V_1)} \mathcal{I}(u)}, \tag{7}$$

where $P(V, V_1)$ is the collapse priority (lower signifying preference) of vertex V onto V_1, $Q(V)$ is the error quadric of vertex V and $N(V)$ is the neighbourhood of vertex V. While the result itself does not relate to a known quantity, it achieved the desired result in our experiments: areas with zero geometric error may be decimated regardless of importance, while areas with low importance but high geometric error are less preferable than those with less geometric error. For regular scene sizes the term is still dominated by the importance.

4. Evaluation

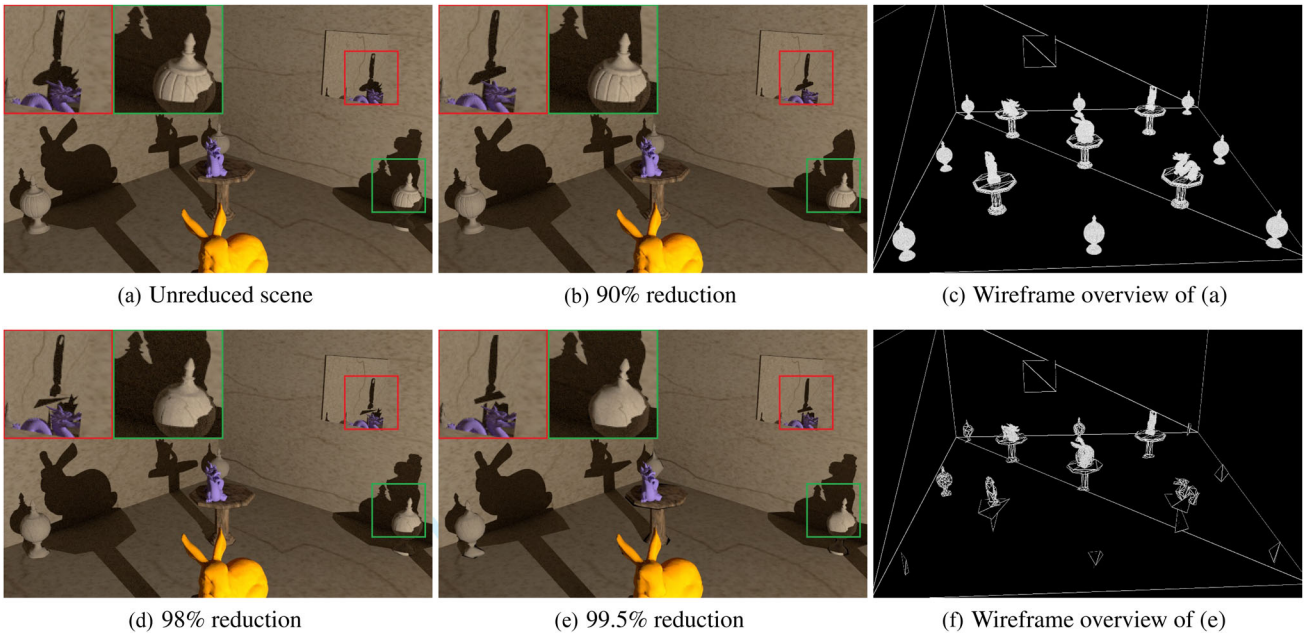
We implemented the reduction pipeline in our workgroup renderer *Mufflon* [BJGB19], which uses a two-level linear BVH as acceleration structure and OpenMP for CPU-side parallelism. All performance evaluations were executed on an AMD Ryzen Threadripper 2990WX alongside DDR4-2400 RAM. We did not include comparisons to related algorithms. Unless mentioned otherwise, we initially reduced the scene by the same amount as the single final reduction with 100 importance iterations (see Figure 2) as well as one NEE per light source (see Section 3.3). To the best of our knowledge, there is no other algorithm using information about global illumination to guide mesh reduction; image-driven simplification [LT00] only incorporates one fixed light source and does not respect global light transport at all.

4.1. Reduction quality

The main goal of our work is the preservation of shadow details in reduced scenes. To this end, Figure 11 depicts a section of the Holy Bunny scene with a total of 4.3 million triangles. Figure 11(a) shows the unreduced scene rendered with a path tracer; Figure 11(c) shows a wireframe rendering from a different angle. To evaluate the overall reduction quality, Figures 11(b)–11(e) present the result after reduction by 90%, 98% and 99.5% respectively. With 90% reduction, there are only small changes in the image, most notably the shadow cast by a table seen in the wall mirror. With higher reduction rates, artefacts become more apparent: the small figurines in the image corners lose their detailed rills along their bodies and the reflected shadow of the buddha statue becomes more coarse, exposing that the table it is resting on consists of two separate parts. Multiple reduction passes do not remedy the over-reduction as showcased by Figure 12: since the shadow area reduces, the respective mesh is attributed even less importance which exacerbates the issue. Figure 11(f) displays a wireframe rendering of the scene at 99.5% reduction. Notably, the buddha statue to the right of the bunny retains a higher polygon count than its counterpart due to the difference in visible shadow size. The figurines which are neither directly visible nor cast a visible shadow are maximally reduced, becoming tetrahedrons.

4.2. Weighting importance sources

To assess the influence of the importance weights δ and γ from Equations (3) and (6), Figure 13 depicts the scene Hairy Yeahright reduced by 95%. Figure 13(a) shows the unreduced scene, while Figure 13(b) shows the scene after reduction with error quadrics only. The decimation is clearly visible on both the visible hair and its shadow. Figure 13(c) shows the same scene after importance reduction with weights $\gamma = \delta = 1$, leaving the shadow largely intact but overly reducing parts of the head and tail of the statue. The hairball receives ∼40% of the illumination and view importance as well



(a) Unreduced scene (b) 90% reduction (c) Wireframe overview of (a)

(d) 98% reduction (e) 99.5% reduction (f) Wireframe overview of (e)

Figure 11: *Reduction progression of Holy Bunny scene; initial decimation was performed to the same level as the final reduction*

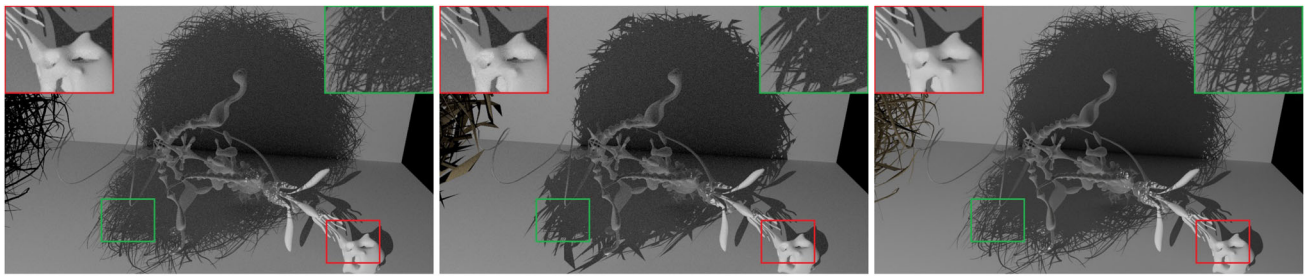
(a) 1 pass (b) 2 passes (c) 3 passes

Figure 12: *Close-up of the mirror in Figure 11 after 1, 2 and 3 reduction passes. Multiple reduction passes do not improve the quality of an over-reduced diffuse or occluding mesh.*

as ∼97% of the shadow importance, resulting in a vertex distribution of roughly 85% to 15% between the hairball and the statue. The same settings were used to reduce the scene in Figure 13(d), however here we did not weight the importance with error quadrics to prioritize collapses. It is apparent that the occluding hairs which are not part of its silhouette collapse too much, leaving visible gaps in the shadow. The statue also suffers from ill-suited collapses especially at sharp edges, which may be partially mitigated by tuning the normal-flip prevention used by Reich *et al*. In Figure 13(e) we put higher emphasis on directly visible geometry, using $\gamma = 100$ and obtaining a vertex distribution of 46% to 54%. This visibly improves the statue and hair geometries, while simultaneously worsening the shadow outline in some places. To contrast this, we used $\delta = 100$ in Figure 13(f), placing higher emphasis both locally on the shadow silhouettes and also attributing a higher vertex budget to the hairball. While the shadow silhouette improves slightly, the statue clearly worsens and noticeably disturbs the image impression as a result of the vertex budget distribution of 97% to 3%. Leaving the weighting equal for all importance parts resulted in balanced reductions for all tested scenes; increasing the weight for one part generally means worsening another aspect of the image.

4.3. Performance

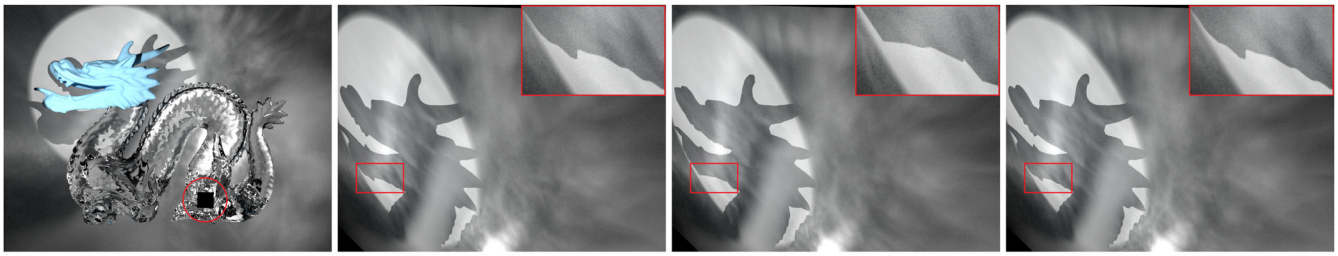
To evaluate the overhead of our method, Table 1 shows the execution times of our reduction steps when reducing scenes by 90%, 95% and 98% of their vertices, respectively; for a general performance overview of the illumination-driven mesh reduction see [RGG15]. The third column shows the render time of 100 samples per pixel for a path tracer on the original scene; it should be noted that 100 path tracing iterations are not nearly enough to produce converged renderings. The fourth column contains the execution times of the importance gathering with 100 iterations for the pre-reduced scenes. Unlike in [RGG15], importance iterations may incur significant overhead dependent on the amount of visible shadows and the complexity of occluding geometry, peaking for Hairy Yeahright with its many small hairs. The majority of the overhead stems from additional intersection tests and BVH traversals necessary for silhouette detection; for this reason, Sponza and Lucy features the lowest ratio—its spotlights reduce the amount of shadow tests necessary. Overall there is no clear trend when comparing the overhead for different reduction levels. The fluctuations in run-time between them are likely caused by BVH split decisions. The fifth and sixth columns specify the absolute time necessary to decimate the scene initially and with importance. The additional cost for incorporating importance lies between 11% and 30%, depending again on the geometric complexity of the scene instead of its size. Here, local validity criteria may prevent collapses, as observed with the hairs in Hairy Yeahright; this causes longer collapse chains of single vertices, reducing the impact of the importance overhead.



(a) Unreduced scene (b) Error quadrics only (c) $\gamma = \delta = 1$

(d) $\gamma = \delta = 1$ without error quadrics (e) $\gamma = 100$, $\delta = 1$ (f) $\gamma = 1$, $\delta = 100$

Figure 13: *Different configurations for Hairy Yeahright: all but (a) were reduced by 95% initially, (c)–(f) additionally underwent one decimation pass. The weights affect whether shadows or directly visible objects are preferred if both cannot be preserved.*



(a) Scene overview (b) Unreduced scene (c) Importance only (RMSRE 0.16) (d) Full pipeline (RMSRE 0.14)

Figure 14: *Comparison between reduction with Reich* et al.*'s [RGG15] pipeline for path tracing (c) and our extension (d). The scene features the Stanford Dragon with diffuse head and glass body in front of a spotlight and a small area light (circled). Each image was rendered with a path tracer at 200,000 samples per pixel. The reduced versions were trained with 4000 importance iterations and reduced by 80%.*

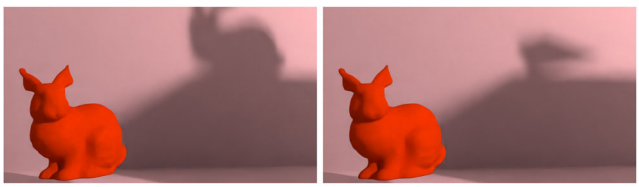
4.4. Interaction with illumination-driven reduction and failure cases

Since we amended the original reduction pipeline, it is reasonable to discuss the possible effects on the reduction result in the context of caustic preservation. Figure 14 shows a comparison between the reduction pipeline with and without shadow silhouette preservation with equal importance weighting. Note that we used a relatively large area light source to get somewhat converged results. Both versions keep the caustics mostly intact, but do not fully preserve them. This is due to the poor convergence of caustics in path tracing and thus the poor importance convergence. However, the shadow silhouette of the dragon's head is visibly disfigured in Figure 14(c). Generally, the preservation of caustics will suffer when preserving shadow silhouettes at the same time if the allotted vertex budget does not allow both to remain intact. It is difficult to predict which feature is deemed less important and depends on the brightness of both the shadow and caustics.

Another factor to consider is the influence of error quadrics. In certain edge cases they may hinder optimal reduction: most notably when only a very small part of a mesh has an observable effect on the image, for example, a small light source right in front of a diamond attached to a necklace with a rough surface. Our error will preserve the overall structure of the necklace as well, which may impact the quality of caustics if not a sufficiently high vertex budget has been allotted. Note that this particular case may be avoided by splitting necklace and diamond into two meshes. In all other cases where reduction occurs despite high importance, a low image error can no longer be guaranteed anyway.

Table 1: *Performance comparison of our approach between different scenes rendered at a resolution of 800×600 with 100 importance iterations/samples per pixel. Initial decimation is equal to total reduction. All timings are given as* process *time.*

	Complexity (vertices/faces) Memory (geometry only) Importance overhead	Reduction	Unreduced PT	Importance pass	Initial /	importance decimation
Holy Bunny	2.13M / 4.26M	90%		314.4s	78.8s	109.7s
	127.8MB	95%	186.2s	316.9s	83.1s	110.7s
	25.5MB	98%		303.1s	80.4s	109.8s
Dragon Tree	592K / 1.18M	90%		340.8s	23.7s	28.6s
	35.5MB	95%	237.9s	332.3s	24.2s	30.7s
	7.1MB	98%		372.5s	24.9s	30.2s
Hairy Yeamright	1.85M / 3.67M	90%		1187.7s	50.4s	56.5s
	110.6M;B	95%	465.9s	1331.3s	50.2s	57.2s
	22.2MB	98%		1802.3s	50.1s	57.5s
Fence	603K / 1.21M	90%		472.0s	37.2s	47.1s
	35.1MB	95%	256.1s	392.9s	38.3s	48.3s
	7.2MB	98%		428.8s	37.6s	49.1s
Sponza and Lucy	11.8M / 23.69M	90%		3706.9s	385.7s	502.7s
	710.7MB	95%	2924.5s	3325.4s	443.1s	530.5s
	142.1MB	98%		3325.4s	507.4s	598.3s



(a) Ground truth (b) Reduced by 95%

Figure 15: *Failure case: two bunnies, one directly visible and one only visibly via its indirect shadow from a brightly lit wall part.*

Concluding the evaluation, we draw attention to scenarios our contributions do not cover. We cannot detect or preserve *indirect* shadows caused by brightly lit patches or caustics, as Figure 15 demonstrates. This is a fundamental limitation of our approach, which may be solved by exploring bidirectional rendering techniques such as shadow photons.

5. Conclusion and Future Work

We extended the illumination-driven mesh reduction [RGG15] to preserve shadows by detecting visible shadow parts and their silhouettes. We transitioned the original approach to path tracing and combined it with error quadrics to minimize geometric errors in uniformly important areas of a mesh. We have shown that this combination does not affect the quality of the original method.

As demonstrated in Figure 7, it remains a challenge to reliably detect the entire silhouette we wish to retain. Possible avenues for improvement include an extended intersection test to find close-by faces as well as spreading importance to neighbouring faces when encountering a silhouette, although neighbourhood information is not readily available in index buffer-based renderers. An issue especially for caustic-heavy scenes is the lack of indirect shadow preservation, which makes it less suited for bidirectional renderers. Photon-based algorithms may explore the option of *shadow photons* as introduced by Jensen [JC96] as an alternative means to detect silhouettes capable of detecting indirect shadows as well, but would inherit the weaknesses of photon maps. Our estimation of penumbra size is also not applicable to environment maps, for which a classification into distinct area lights similarly to the approach by Annen *et al.* [ADM*08] may be used. Generally, perception-based features such as shadow transitions are influenced by post-processing as well; tone-mapping in particular has a large say in whether parts of a scene with little illumination contain noticeable details or not and should influence the error metric. To conserve memory in large scenes, objects that occur multiple times are often *instanced*, meaning that the geometric data reside in memory only once; during rendering, a transformation matrix is then applied for each instance. The current approach reduces the mesh equally for all instances, leading to wasted reduction potential. In a similar vein, removing meshes or instances below a certain importance could prove beneficial for scenes with many objects. To render animated sequences, the reduction should be made temporally coherent. An issue for reduction algorithms in general are *non-manifold* meshes.

Acknowledgements

The hairball model is by courtesy of NVIDIA, Yeahright by Keenan Crane, Crytek Sponza by Frank Meinl; the Lucy, Dragon, Buddha and Bunny models are by courtesy of Stanford University. The decorative models were created by Dosch Design. The work was supported by the German Research Foundation (DFG) grant GR 3833/3-2.

References

[ADM*08] Annen T., Dong Z., Mertens T., Bekaert P., Seidel H.-P., Kautz J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics 27*, 3 (2008), 1–8.

[BBS*09] Budge B., Bernardin T., Stuart J. A., Sengupta S., Joy K. I., Owens J. D.: Out-of-core data management for path tracing on hybrid resources. *Computer Graphics Forum 28* (2009), 385–396.

[BJGB19] Bethe F., Jendersie J., Gu F., Brüll F.: Mufflon. https://github.com/TU-Clausthal-Rendering/Mufflon, 2019.

[BKP*10] Botsch M., Kobbelt L., Pauly M., Alliez P., Lévy B.: *Polygon Mesh Processing*. AK Peters/CRC Press, Boca Raton, FL, 2010.

[CD04] Chan E., Durand F.: An efficient hybrid shadow rendering algorithm. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques* (2004), EGSR'04, Eurographics Association, p. 185–195.

[CFS*18] Christensen P., Fong J., Shade J., Wooten W., Schubert B., Kensler A., Friedman S., Kilpatrick C., Ramshaw C., Bannister M., et al.: Renderman: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics 37*, 3 (2018).

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

[Cro77] Crow F. C.: Shadow algorithms for computer graphics. *SIGGRAPH Computer Graphics 11*, 2 (1977), 242–248.

[DKHS14] Davidovič T., Křivánek J., Hašan M., Slusallek P.: Progressive light transport simulation on the GPU: Survey and improvements. *ACM Transactions on Graphics 33*, 3 (2014), 1–19.

[ENSB13] Eisenacher C., Nichols G., Selle A., Burley B.: Sorted deferred shading for production path tracing. In *Proceedings of the Eurographics Symposium on Rendering* (2013), EGSR '13, Eurographics Association, p. 125–132.

[ESAW16] Eisemann E., Schwarz M., Assarsson U., Wimmer M.: *Real-Time Shadows*. AK Peters/CRC Press, Boca Raton, 2016.

[Fec58] Fechner G. T.: Über ein wichtiges psychophysisches Grundgesetz und dessen Beziehung zur Schätzung der Sterngrössen. *Ges. Wissensch., Math.-Phys. Kl* (1858).

[GBBK04] Guthe M., Borodin P., Balázs A., Klein R.: Real-time appearance preserving out-of-core rendering with shadows. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques* (2004), EGSR'04, Eurographics Association, p. 69–79.

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

[GH97] Garland M., Heckbert P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., p. 209–216.

[HDD*94] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: *Mesh optimization*. Tech. rep., Washington University Seattle, Department of Computer Science and Engineering, 1994.

[HJ09] Hachisuka T., Jensen H. W.: Stochastic progressive photon mapping. *ACM Transactions on Graphics 28*, 5 (2009), 1–8.

[HOJ08] Hachisuka T., Ogaki S., Jensen H. W.: Progressive photon mapping. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 1–8.

[Hop96] Hoppe H.: Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, Association for Computing Machinery, p. 99–108.

[JC96] Jensen H., Christensen N.: Efficiently rendering shadows using the photon map. In *Proceedings of Compugraphics'95* (1996), Eurographics m.fl.

[Jen96] Jensen H. W.: Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (1996), Springer-Verlag, p. 21–30.

[Kaj86] Kajiya J. T.: The rendering equation. *SIGGRAPH Computer Graphics 20*, 4 (1986), 143–150.

[LT00] Lindstrom P., Turk G.: Image-driven simplification. *ACM Transactions on Graphics (ToG) 19*, 3 (2000), 204–241.

[LW93] Lafortune E. P., Willems Y. D.: Bi-directional path tracing. In *Proceedings of Compugraphics'93* (1993), pp. 145–153.

[MG10] Menzel N., Guthe M.: Towards perceptual simplification of models with arbitrary materials. *Computer Graphics Forum 29*, 7 (2010), 2261–2270.

[RGG15] Reich A., Günther T., Grosch T.: Illumination-driven mesh reduction for accelerating light transport

simulations. *Computer Graphics Forum 34*, 4 (2015), 165–174.

[SCH03] Sen P., Cammarano M., Hanrahan P.: Shadow silhouette maps. *ACM Transactions on Graphics 22*, 3 (2003), 521–526.

[Vea98] Veach E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998.

[Web34] Weber E. H.: *De pulsu, resorptione, auditu et tactu: annotationes anatomicae et physiologicae, auctore*. prostat apud CF Koehler, 1834.

[Wil78] Williams L.: Casting curved shadows on curved surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1978), SIGGRAPH'78, Association for Computing Machinery, pp. 270–274.