Eurographics Symposium on Rendering 2009
Hendrik P. A. Lensch and Peter-Pike Sloan
(Guest Editors)

Volume 28 (2009), Number 4

Packet-based Hierarchal Soft Shadow Mapping

Baoguang Yang[1,2] Jieqing Feng[1] Gaël Guennebaud[3] Xinguo Liu[1]

1 - State Key Lab of CAD&CG, Zhejiang University
2 - S3 Graphics 3 - INRIA Bordeaux

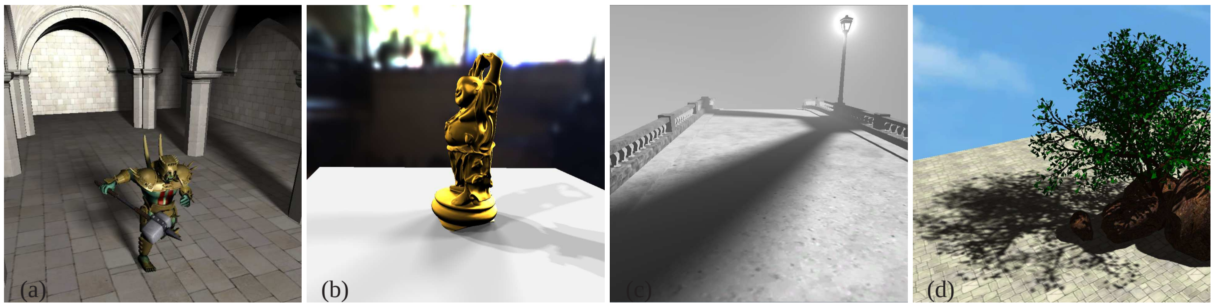

Figure 1: *Some pictures rendered using our soft shadow mapping algorithm, at 16, 24, 30, 46 frames per second respectively, without any precomputation.*

Abstract

Recent soft shadow mapping techniques based on back-projection can render high quality soft shadows in real time. However, real time high quality rendering of large penumbrae is still challenging, especially when multi-layer shadow maps are used to reduce single light sample silhouette artifact. In this paper, we present an efficient algorithm to attack this problem. We first present a GPU-friendly packet-based approach rendering a packet of neighboring pixels together to amortize the cost of computing visibility factors. Then, we propose a hierarchical technique to quickly locate the contour edges, further reducing the computation cost. At last, we suggest a multi-view shadow map approach to reduce the single light sample artifact. We also demonstrate its higher image quality and higher efficiency compared to the existing depth peeling approaches.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Picture/Image Generation

1. Introduction

Shadows are one of the most important visual effect which both increase the level of realism of a 3D scene, and help to identify spatial relationships between objects. Assuming a purely punctual light source, hard shadows can be efficiently computed using either a shadow volume [Cro77] or a shadow mapping [Wil78] based technique. However, since real world light sources have some extent, higher shadow realism is achieved by computing so called soft shadows. Their accurate evaluation requires to integrate the illumination over the visible parts of the light, which is a computationally expensive procedure. Therefore, when performance matters, a very common approximation is to assume the incoming illumination is constant over the light area. In that context, the problem of rendering soft shadows boils down to the problem of computing the percentage of visibility between a point and an extended light source.

Corresponding author: Xinguo Liu.
E-Mails: {yangbaoguang, jqfeng, xgliu}@cad.zju.edu.cn,
gael.guennebaud@inria.fr

Among the wide soft shadow literature, most promising approaches probably include the so called soft shadow volumes [AAM03, FBP08] and soft shadow mapping [GBP06, GBP07, SS07] based techniques. While the former ones usually generate more accurate results than image based techniques, they are limited to polygonal meshes and their cost highly depends on the geometry complexity. On the other hand, image based techniques are tailored for high performance, and can deal with all rasterizable geometries, thus making them particularly attractive for real time graphics applications (e.g., games). In particular, soft shadow mapping (SSM) methods based on back-projections treat the shadow map as a uniform, spatially sorted, and discrete representation of the scene. The visibility coefficient of a given 3D point is then computed by back-projecting the occluding shadow map samples onto the light source.

However, because the number of back-projected samples linearly depends on the light source area, the performance of such methods drops significantly when rendering large penumbrae. To overcome this issue, Guennebaud *et al.* [GBP07] proposed both light space and screen space strategies to locally adjust the precision according to some visual heuristics. Such strategies are not plenty satisfactory as the quality degrade in complex situations [GBP07].

As a first contribution of this paper, we present a very fast visibility integration procedure which is built upon the occluder contour extraction procedure with back-projection of Guennebaud *et al.* [GBP07]. The efficiency of our novel algorithm comes from two main ingredients. First, we propose an optimized hierarchical algorithm to extract the occluder contours from the shadow map. Our strategy reconstructs exactly the same contours as the original method, but with significantly fewer shadow map queries. Secondly, we exploit the screen space coherence of the penumbrae via a packet-based algorithm computing the visibility factors of multiple neighboring pixels at once. This approach is motivated by the excellent results obtained with packet based ray tracers [Wal04]. As in the ray tracing context, using packets amortizes the hierarchical traversal and contour extraction costs (including the respective memory reads) for a speed up factor proportional to the packet size. Overall, we observed outstanding performance gain up to a factor 20 without any accuracy loss. To our knowledge, this is the first time a packet-based method has been used in such a context.

Another well known limitation of soft shadow mapping methods is the shadow underestimation which might occur when occluders have a high depth range. Indeed, a shadow map only represents the parts of the scene seen from a single point. Therefore, some occluder parts might not be taken into account. While this issue can be addressed using multi-layer shadow maps [SS07, BCS08], such an approach is extremely costly as the number of required layers to get a significant improvement might be arbitrarily large.

As another contribution of this paper, we present and discuss a multi-view shadow map approach to reduce this single light sample artifact. In particular, we show it can produce higher image quality with respect to shadow map aliasing, and higher performance than a multi-layer approach.

2. Related Work

During the last decade, much research work has been devoted to the real-time rendering of soft shadows. In this section we will focus on the most recent and related techniques, and refer to the literature for a more complete survey of older methods [HLHS03].

2.1. Object Based Soft Shadows

In the category of object based methods, Assarsson *et al.* extended shadow volumes [Cro77] with penumbra-wedges to render soft shadows [AAM03]. For each silhouette edge, a penumbra-wedge is constructed, rasterized, and back projected onto the light source to accumulate the occluded area of the light. This method neglects the overlapping of multiple occluders leading to overestimated shadows. This effect can be partly reduced using some blending heuristics [FBP06]. A more robust approach is to generate multiple light samples, and track the occlusion of each sample using counters [LAA*05]. Forest *et al.* [FBP08] extended this last technique to make it suitable for interactive rendering of dynamic scenes.

Eisemann and Décoret [ED07] presented a GPU based visibility sampling framework generating an *influence region* for each triangle. In contrast to the previous silhouette based approaches, this method does not suffer from the *single light sample silhouette* artifacts, and requires only one bit per light sample to track the visibility. Another notable difference is that the *influence regions* are rasterized in the light space instead of the view space introducing aliasing. This drawback has been addressed by Sintorn *et al.* [SEA08] using alias free shadow maps [AL04]. Unfortunately, the performance of their approach drops as the distribution of the view samples in the light space becomes too uneven. Johnson *et al.* presented a similar work [JHH*09] on the new Larabee architecture using an analytical integration of the light visibility.

2.2. Convolution Based Soft Shadows

Image based soft shadow methods estimate the penumbrae from shadow maps [Wil78]. Owing to this additional discretization, such methods are likely to generate shadows with lower quality than object based ones. On the other hand, they are not limited to polygonal meshes, and their cost is almost independent on the scene complexity.

Assuming a single occluder parallel to the light, it has been shown that percentage closer filtering [RSC87] can be extended to produce plausible soft shadows [SS98, Fer05, ED06, ADM*08]. While such approaches lead to impressive performance, in terms of accuracy, they cannot be compared to methods based on a visibility integration.

2.3. Soft Shadow Mapping with Backprojection

In shadow mapping techniques with backprojection [AHL*06, GBP06], the complex geometry of the occluders is replaced by a simple shadow map. Each shadow map texel is considered as a rectangular *micropatch* parallel to the light source (Figure 2-a). For each view pixel, shadow map texels are classified as occluder or background by comparing their depth values to the pixel's depth. During this classification, the visibility is integrated by accumulating the area of the light occluded by each occluding *micropatch*. This is achieved by backprojecting the *micropatches* onto the light source plane, whence the name of the method. Owing to the discrete nature of the representation, this algorithm is prone to gaps (i.e.,

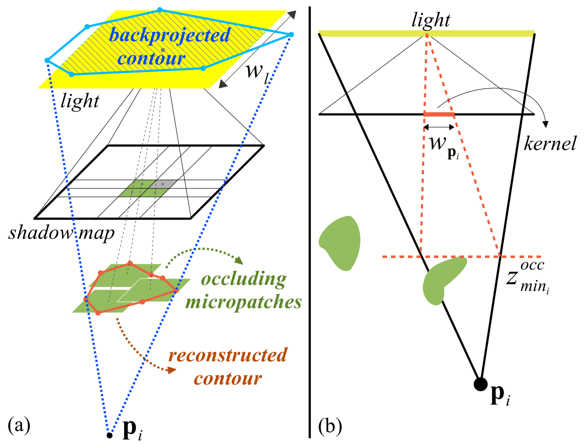

Figure 2: *(a) SSM: Micropatch interpretation versus contour reconstruction. (b) Kernel computation.*

some occluded parts of the light are not removed) and overlapping (i.e., some parts are removed more than once) artifacts [GBP06].

To overcome this problem, Schwarz and Stamminger proposed to reconstruct a continuous quad-mesh connecting the centers of the occluding samples [SS07]. Then, they performed a discrete integration using light samples. In the same vein, Guennebaud *et al.* proposed to simply reconstruct a continuous contour of the occluders [GBP07] using a marching square algorithm (Figure 2-a). Contour edges are then backprojected onto the light source to perform an analytical radial integration of the occluded light area around the light center as in the penumbra-wedge technique [AAM03]. Because of its high efficiency, we chose to base our work on this contour based SSM technique.

Since it is not conceivable to search for occluders over the whole shadow map for every view pixel, several optimization techniques have been designed. Guennebaud *et al.* showed that the search area, that we will call *kernel* (Figure 2-b), can be drastically reduced using a hierarchical shadow map (HSM) [GBP06]. A HSM is equivalent to a quadtree structure where each texel (or node) stores both the minimal and maximal depth values of its covered region. The HSM is used to quickly compute depth bounds of the current search area to iteratively refine it, and to quickly classify and prune fully lit and fully occluded pixels.

A HSM is simple to use but too conservative. In contrast, a multi-scale shadow map (MSSM) [SS07], which is a variant of the neighborhood-buffer [Déc05], allows to accurately query any power-of-two sized square via a single texture fetch, and any rectangular regions with four fetches only. Thus a MSSM yields much tighter kernels and a better classification of penumbra pixels [SS07]. In this paper we adopt this MSSM structure, and extend it to a hierarchical traversal of the kernel for further enhanced performance.

However, for a large light source, even an ideal kernel might still be arbitrarily large. By trading the quality, it is possible to bound the kernel size via the use of lower levels of the HSM [GBP06]. In order to generate smooth penumbrae, special care has to be taken at the level transitions [GBP07, SS08]. Finally, the ratio performance/quality can be further traded taking advantage of the low frequency of soft shadows [GBP07]: the visibility is computed for a fraction of screen pixels inversely proportional to an estimation of the screen space size of the penumbrae, and a continuous information is reconstructed afterwards using a non linear screen space filter [GBP07].

Unfortunately, these two last acceleration techniques require many additional passes which become expensive as the number of shadow maps increases. Moreover the heuristics controlling the precision take into account a single occluder depth. Therefore the quality significantly degrade in complex situations where many occluders with various depths interact. In contrast, our acceleration techniques do not perform any compromise on the quality while keeping a simple overall algorithm.

3. Overview

An overview of our rendering algorithm is given in Algorithm 1. From a high level point of view, it is very similar to a standard SSM technique. The main differences are in steps 6 to 8 where the visibility is computed using our very fast procedure. For the visibility integration itself, we reuse the contour extraction with backprojection technique [GBP07]. However, instead of evaluating the visibility for each view pixel independently, we propose to exploit the screen space coherence of the penumbrae by evaluating it per packet of $w \times h$ pixels (step 7, section 4.1). Packets which are not coherent enough, i.e., packets containing pixels with different contours or for which the kernels do not overlap much, are quickly detected (step 6, section 4.3) and processed pixel-wise in a second pass (step 8).

The steps 7 and 8 are further optimized using a hierarchical traversal of the kernels which significantly reduces the number of visited shadow map samples (section 4.2). In particular, we based our traversal on a multi-scale shadow map (MSSM) structure that will allow us to reach higher performance than with a naive quadtree traversal. Recall that the

Algorithm 1 Overview of our rendering algorithm

```
1 Create a position buffer: render the scene from the view point
  while storing the 3D position in the light space;
2 **Subdivide the light source into $S \times S$ equal-sized sub-lights**
  **For each sub-light**
3     Render a shadow map from the sub-light center;
4     Build a multi-scale version of the shadow map [SS07];
5     Compute the kernels of all points in the position buffer,
      and remove fully lit or fully occluded view pixels [GBP06];
6     **Classify pixel packets as coherent and non-coherent**
7     **Compute visibility factors of coherent pixels**
      **by packet of $w \times h$ pixels**;
8     **Compute visibility factors of remaining pixels**;
9     Render the final image using the visibility factors;
```

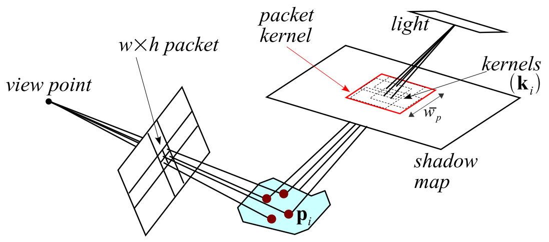

Figure 3: *Illustration of the kernels in a* 2×2 *pixel packet.*

MSSM is also used in step 5 to compute tight kernels as described in [SS07].

The last main difference is the optional step 2 which aims to reduce the single light sample artifact by subdividing large light sources into multiple smaller ones (section 5).

4. Fast Visibility Evaluation

In this section we focus on the visibility evaluation procedure corresponding to the main steps 7 and 8 of the overall algorithm. Without loss of generality, we will consider a single square light source of size w_l. As we already said, it is based on the contour extraction with backprojection method [GBP07]. In a nutshell, for each visible point $\mathbf{p}$, this algorithm associates to each shadow map sample $\mathbf{s}$ of its respective kernel a binary value $f(\mathbf{s})$ which is 1 (occluder) if its depth values is smaller than the depth of $\mathbf{p}$, and 0 otherwise (non-occluder). Then, a 2D variant of the marching cube algorithm is applied on the dual grid connecting the centers of the shadow map samples to detect and reconstruct the contour edges. Figures 2, 4 give some examples. Every time an edge has been detected, it is backprojected onto the light source to radially integrate the occluded area.

In the rest of this section we present our complementary packet-based and hierarchical optimizations of this algorithm. As we will show they respectively take advantage of the screen space and light space coherence without loss of quality.

4.1. Coherent Visibility Evaluation

Kernel coherence analysis

Consider a set $\mathbf{q}$ of neighboring view pixels. Unless this set crosses an object silhouette, their corresponding 3D points $\mathbf{p}_i$ are very close to each other, and consequently their projections $\mathbf{p}'_i$ onto the shadow map are also likely to be very close, as illustrated in Figure 3. Recall that the kernel $\mathbf{k}_i$ of a point $\mathbf{p}_i$ with depth $z_{\mathbf{p}_i}$, is a square region centered at $\mathbf{p}'_i$. Therefore, the amount of overlap of the kernels depends on their respective width $w_{\mathbf{p}_i}$:

$$w_{\mathbf{p}_i} = \alpha w_l \left(\frac{1}{z^{occ}_{min_i}} - \frac{1}{z_{\mathbf{p}_i}} \right) , \qquad (1)$$

where α is the light space to shadow map space scale factor, and $z^{occ}_{min_i}$ is the minimal depth value of the occluders inside

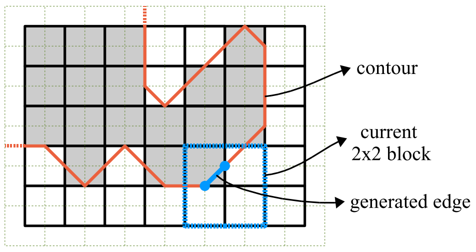

Figure 4: *Illustration of the contour extraction algorithm. The gray squares represent the occluding samples while the dashed grid corresponds to the dual grid. The current highlighted* 2×2 *block of the marching square algorithm yields the blue edge of the contour.*

the pyramid formed by the light and the point $\mathbf{p}_i$ (Figure 2-b). Since the points $\mathbf{p}_i$ are assumed to be close, the depth differences $|z_{\mathbf{p}_i} - z_{\mathbf{p}_j}|$ are very small, and the pyramidal regions containing the occluders are likely to overlap much. Thus, the difference between the $z^{occ}_{min_i}$ values is likely to be small, and so does the kernel size difference (eq. 1).

Finally, this empirically shows that there exist a strong coherence in the kernels of neighboring pixels, i.e., their respective kernels are likely to overlap very much except nearby the silhouettes as depicted in Figure 5 (left). Intuitively, the amount of non coherence of the set of points $\mathbf{q} = \{\mathbf{p}_i\}$ can be quantified by the area of the kernels $\mathbf{k}_i$ outside the reference kernel $\mathbf{k}_0$:

$$\max_i \left\{ \frac{|\mathbf{k}_i \setminus \mathbf{k}_0|}{|\mathbf{k}_i|} \right\} , \qquad (2)$$

where $|\cdot|$ denotes the area. This empirical measurement is depicted in Figure 5 (right) showing a strong kernel coherence in this scene.

Packet-wise evaluation

The initial visibility evaluation algorithm extracts the occluder contours for each visible point individually. The previous observations suggest a big optimization opportunity by evaluating the visibility per packet of $w \times h$ coherent screen pixels.

The algorithm starts by evaluating the coherence of each packet (step 6 in Algorithm 1). Non coherent packets are marked and their visibility will be computed pixel-wise in a second pass. Note that accurate coherence measurements tailored for our specific algorithm, will be derived later in section 4.3.

Our modified packet-wise visibility shader is described in Algorithm 2. This shader takes as input a single packet and returns wh visibility coefficients (using multiple render targets if $wh > 4$).

Given the input packet, we start by computing a common *packet kernel* by taking the axis aligned bounding box of the $w \times h$ pixels' kernels.

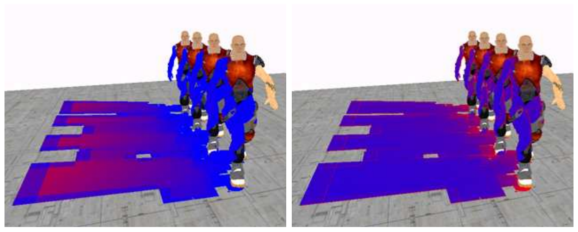

Figure 5: *Illustration of the kernel size (left) and coherence (right). Blue indicates a small value, while red indicates a large value. Pixels with zero sized kernel are rendered with shading and textures.*

Next, the common occluder contour edges are extracted by examining each block of 2×2 shadow map samples which overlaps the *packet kernel*. Since we are considering multiple shaded points at once, the computation of the status value $f(\mathbf{s})$ of a sample $\mathbf{s}$ has to be slightly modified as follows:

$$f(\mathbf{s}) = \begin{cases} 1, & if \;\; z_{\mathbf{s}} \le z_{pmin}; \\ null, & if \;\; z_{pmin} < z_{\mathbf{s}} < z_{pmax}; \\ 0, & if \;\; z_{pmax} \le z_{\mathbf{s}}. \end{cases} \quad (3)$$

where $z_{\mathbf{s}}$ is the depth value of $\mathbf{s}$, and z_{pmin}, z_{pmax} are the minimal and maximal light space depth values of the packet. Note that a status value of *null* indicates the sample partially occludes the packet, i.e., some pixels in the packet are occluded by the sample, while others are not. In practice, this means the pixels of the packet do not have the same occluder contours. When such a case arises, the packet is marked as non coherent, and its respective pixels will be processed pixelwise with the other non coherent packets.

Once the four status values of the 2×2 block have been computed, we apply the marching square rules as in the standard algorithm [GBP07]. If an edge is detected, then it has to be processed separately for each point of the packet. Indeed, this very last step, which includes the backprojection and the radial integration of the occluded area, entirely depends on the exact positions of the input points and thus it cannot be factorized. Nevertheless, we emphasize that all the rest of the algorithm, such as the expensive shadow map texture accesses and the contour extraction logic, is now performed only once per block of $w \times h$ pixels.

Algorithm 2 *Packet-based evaluation*

```
Input: a packet of $w \times h$ points
Output: $wh$ visibility coefficients
construct an extended kernel for the packet;
for each block of $2 \times 2$ samples in the extended kernel do
    if it contains a contour edge then
        construct the contour edge;
        for each point in the packet do
            back-project the edge onto the light plane;
            radially integrate the covered light area;
```

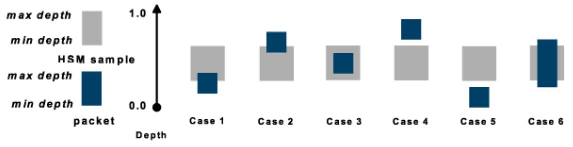

Figure 6: *Depth range relationships between a node of the MSSM and a packet. Cases 4 and 5 do not produce any occluder edges.*

4.2. Hierarchical Contour Extraction

In the above basic contour extraction procedure, all the 2×2 sample blocks in the kernel are processed one after the other. Most of them, however, do not generate any contour edge, because they are either completely inside or completely outside the occluder. To obtain further acceleration, we take advantage of this light space coherence of the contour using a hierarchical extraction procedure. This will allow us to quickly skip blocks of samples that do not produce any contour edge.

Note that such an idea has already been proposed with a micropatch interpretation of the shadow map [DU07]. In this work, the shadow map is converted to a min-max quadtree structure (i.e., a HSM), which is then traversed in a depth-first order from the top root node for every view pixel. During the traversal, branches which cannot occlude the light are quickly detected and pruned.

Our algorithm follows the same general principle but with several differences. First, instead of using a basic min-max HSM, we propose to reuse the more accurate MSSM data structure. Let r be the resolution of the shadow map. A MSSM is made of $log(r)$ levels, each containing $r \times r$ texels (*nodes*). A texel (i, j) at level l stores the minimal and maximal depth values in a neighborhood region of size $2^l \times 2^l$ centered around the texel [SS07]. The depth bounds of any power-of-two sized square region can be queried with a single texture fetch. As we will show, doing a hierarchical traversal through a MSSM is as simple as with a basic quadtree, but it will allow us to reduce the number of visited branches.

Secondly, we have to adapt the algorithm to the extraction of contours. Let us remind that the contour extraction procedure works in the dual grid and it has to examine every 2×2 sample block having a depth range overlapping the packet depth range. This means the samples, and consequently the nodes of the hierarchy, cannot be considered separately anymore. To overcome this apparent difficulty, we slightly modify the way the MSSM is built such that it matches the dual shadow map grid. Instead of initializing the first level of the MSSM with a copy of the shadow map, we initialize each texel (i, j) of the first level with the minimal and maximal depth values of the four adjacent samples (i, j), $(i+1, j)$, $(i, j+1)$, and $(i+1, j+1)$. In other words, now the first level contains the depth range of all possible 2×2 blocks of the shadow map such that there is no need to look at the neighboring MSSM texels during the traversal.

In order to reduce the number of visited nodes, instead of starting the traversal from the top root node [DU07], we start from the node of the lowest level ℓ which entirely covers the current kernel and such that its top and left sides match the top and left sides of the kernel. Intuitively, a MSSM can be seen as a set of quadtrees computed for every different offsets. Therefore, once we have selected the starting node, we have instantiated a single quadtree that we can easily follow without any overhead.

During the traversal, the depth range of the current node is compared to the depth range of the current packet yielding the six possibilities depicted in Figure 6. If the node cannot occlude the light (case 4), or completely occlude it (case 5), then the branch does not contain any contours, and it is pruned. Otherwise, we go down to the next finer level and check the four children. This procedure is iteratively repeated until the finest level is reached. To manage the recursivity, we adopt the efficient stack implementation [DU07] which is sketched in Algorithm 3.

Compared to a standard HSM, the main advantage of the MSSM is that the instantiated quadtree much better matches the given kernel, thus allowing to significantly reduce the number of visited nodes. As an example, let us consider a 8×8 kernel where no branch gets pruned. Then, a HSM will need to visit between 21 $(= 1+2^2+4^2)$ and 57 nodes to reach the leaves, while a MSSM will always visit only 21 nodes. Of course, this example with a power-of-two sized kernel is a best case scenario for the MSSM, and in the worst case both data structures perform equally well. As another advantage, reusing the MSSM which has be used to compute tight kernels, prevents the need to compute and store a second data structure.

Finally, we emphasize that a hierarchical approach is much more effective at the extraction of contours rather than micropatches [DU07]. Indeed, in our case, we can not only prune branches which does not belong to an occluder part, but also branches which entirely belong to an occluder, and thus discard many more samples as illustrated in Figure 7. Furthermore, the combination of a hierarchical traversal with a packet-wise evaluation is particularly efficient as the later allows to better amortize the cost of the former.

Algorithm 3 *Hierarchical contour extraction*

```
get the node n covering the current kernel;
i=-1; // index of current child
loop
    if ++i==4 then
        pop (i,n.level) from the stack; // and update node n
    end if
    if node n and packet depth ranges overlap then
        if n.level==0 then
            extract and process contour edge;
        else
            push (i,n.level) onto the stack;
            goto first child; // i=-1; n.level–; etc.
        end if
    end if
```

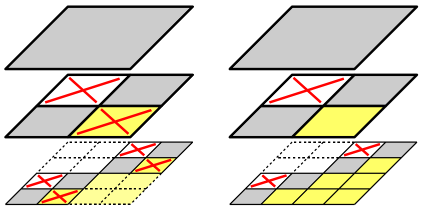

Figure 7: *Comparison of the pruning capability of our contour based hierarchical approach, and the micropatches based approach [DU07]. Occluder nodes are shown in yellow, overlapping ones in grey, and empty ones in white. Skipped nodes are marked with a red cross.*

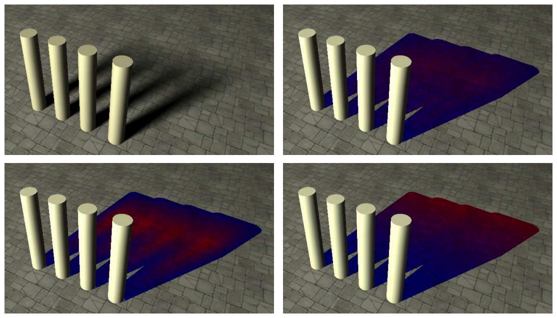

Figure 8: *Illustration of the number of visited shadow map samples per pixel using our technique (top-right), [DU07] (bottom-left), and only a MSSM (bottom-right).*

4.3. Coherence Measurement

In sub section 4.1, we gave an intuitive and general definition of a coherent packet. In this section we strive to take into account the specificities of our algorithm to derive more accurate coherence measurements. Recall that the goal is to quickly determine for each packet whether a packet-wise or pixel-wise evaluation is better suited (step 6 in Algorithm 1).

As a first condition, we want a coherent packet to have non diverging contours, i.e., all of its points yield the same contour. Let us assume a uniform distribution of the shadow map samples in the depth range of the current kernel. Then, according to equation (3), the probability P_{null} that a given sample of the kernel yields a diverging contour, i.e., the probability it has a *null* status, corresponds to the ratio between, 1) the intersection of the packet and kernel depth ranges, and 2) the kernel depth range:

$$P_{null} = \frac{max\{min\{z_{pmax},\bar{z}_{max}\} - max\{z_{pmin},\bar{z}_{min}\},0\}}{\bar{z}_{max}-\bar{z}_{min}} \quad (4)$$

where $\bar{z}_{min}$ and $\bar{z}_{max}$ respectively denote minimal and maximal depth values of the kernel. Let $\bar{w}_p$ be the width of the kernel, then the probability $P_d(\mathbf{q})$ that a given packet $\mathbf{q}$ has different contours is given by:

$$P_d(\mathbf{q}) = 1-(1-P_{null})^{\bar{w}_p^2} \, . \quad (5)$$

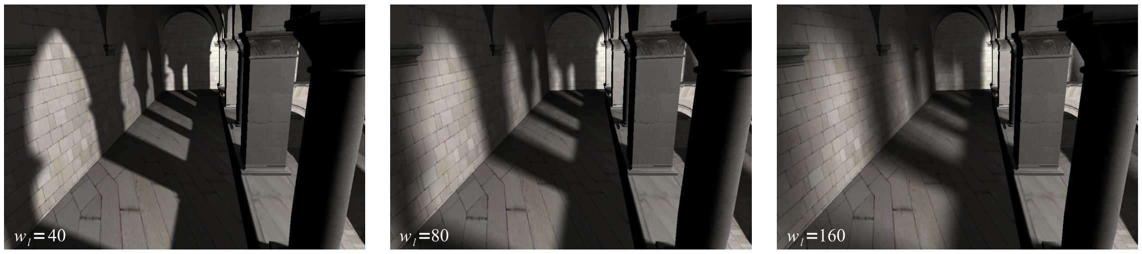

Figure 9: *Soft shadows rendered under increasing size of square lights. The numbers correspond to the light size.*

Finally, a packet will be classified as non coherent if its diverging probability $P_d(\mathbf{q})$ is greater than a given threshold T_d. In practice we found the choice of $T_d = 0.8$ yields very good results with a very low percentage of false positives and false negatives.

In addition to the previous condition, we must ensure that the respective kernels sufficiently overlap such that a packet-wise evaluation will pay off. In order to determine which is the best strategy, we propose to compare the estimations of their respective costs using an empirical cost model. Let Q_n, Q_e, and Q_v be the cost to process a node, to extract an edge, and to accumulate the visibility respectively. Let $\overline{w}_p$ and w_i be the kernel width of the packet and of the i^{th} point of the packet respectively. In addition, let us assume the number of extracted contour edges is equal to the width of the processed kernel. Then for the packet strategy, the number of visited nodes is $log(\overline{w}_p)\overline{w}_p$, the edge extraction cost is $Q_e\overline{w}_p$, and since the visibility accumulation cannot be factorized, its cost for a packet is $Q_v wh\overline{w}_p$. The cost of the pixel-wise strategy is obtained by summing the processing cost of each pixel of the packet. Under these assumptions, the cost ratio between the two strategies is given by:

$$\frac{Q_n log(\overline{w}_p)\overline{w}_p + (Q_e + Q_v wh)\overline{w}_p}{\sum_i Q_n log(w_i) w_i + (Q_e + Q_v) w_i} \qquad (6)$$

Therefore, a packet with a ratio greater than one will be marked as non coherent. In practice, these three parameters are implementation dependent. We fixed $Q_n = 1$, and automatically tuned the values of the other two by running a two dimensional optimization algorithm on the average rendering time of a typical scene. This step has only to be done once.

Finally, we emphasize that the two above heuristics influence only the performance. In particular, if a non-coherent packet is misclassified as coherent, then it will be caught during the packet contour extraction (section 4.1) and processed pixel-wise in a second step.

5. Multi-view shadow maps

Using a single shadow map as a representation of the whole scene may miss some occluders, and shadows are underestimated [GBP06]. To reduce this limitation, it has been suggested to use multi-layer shadow maps computed from a depth peeling procedure [BCS08, SS07].

Alternatively, we suggest a multi-view shadow map approach where the input light is divided into $S \times S$ equally sized sub-lights. The visibility factors are computed individually for each sub-light, and then they are averaged to get the visibility factor of the whole light. Note that for each sub-light we have to construct its own shadow map and MSSM structure. Of course, in practice only one shadow map and one MSSM are allocated and reused using a multi-pass implementation.

Without additional information about the target application, it is difficult to establish which method performs better at reducing the single shadow map artifact. Indeed, it is very easy to design examples where one method outperforms the other, and conversely (e.g., see Figure 10). On the other hand, as illustrated in Figure 11, increasing the number of multi-view shadow maps, increase the effective resolution of the scene seen from the light source, thus reducing the aliasing artifacts at shadow edges. In contrast, a depth peeling approach renders all shadow map layers from the center of the light, and thus the effective resolution is bounded to the resolution of a single shadow map.

Furthermore, we argue that regarding performance, multi-view shadow maps are very efficient as they do not increase the number of depth sample queries. Indeed, let n be the number of shadow map texels contained in the kernel of a given point $\mathbf{p}$. According to equation (1), when a single shadow map is used, we have:

$$n = (w_{\mathbf{p}})^2 = (\alpha w_l)^2 \left(\frac{1}{z_{min}^{occ}} - \frac{1}{z_{\mathbf{p}}} \right)^2 . \qquad (7)$$

With a multi-layer approach, assuming m layers are used, the number of visited samples n_{ml} would naively be $n_{ml} = mn$. Though not mentioned in their paper, Schwarz and Stamminger's approach [SS07] can be further optimized by com-

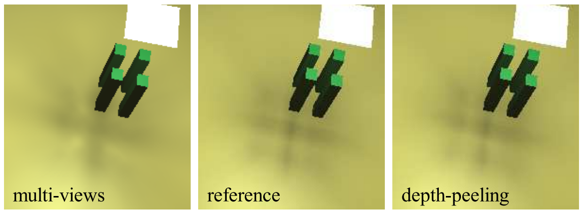

Figure 10: *Example of a tricky case for the multi-view approach with 4 sub-lights.*

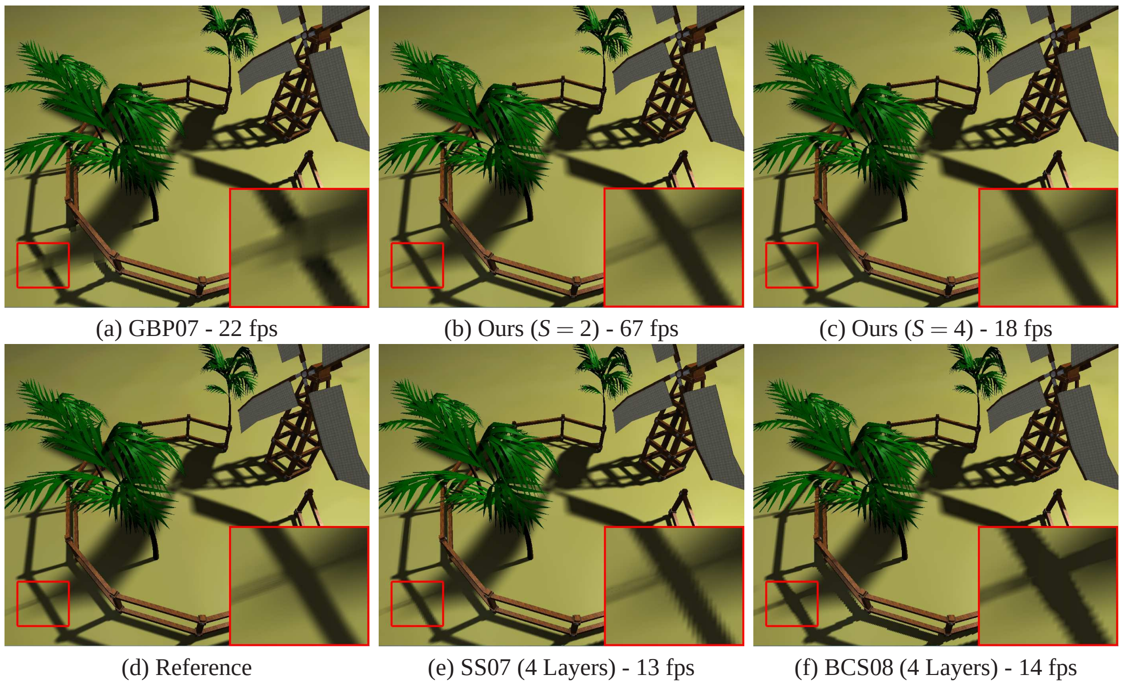
(a) GBP07 - 22 fps (b) Ours ($S = 2$) - 67 fps (c) Ours ($S = 4$) - 18 fps
(d) Reference (e) SS07 (4 Layers) - 13 fps (f) BCS08 (4 Layers) - 14 fps

Figure 11: *Comparison of various techniques addressing the single shadow map artifact. The reference image has been computed from the average of 1024 high resolution hard shadow maps. Every single shadow map is of* 1024×1024*, and we implemented all methods to use a MSSM to compute the kernels.*

puting a different MSSM for each layer. Let $z^{occ}_{min_j}$ be the minimal occluder depth for the layer j. Then, the optimized number of visited pixels is:

$$n_{ml} = (\alpha w_l)^2 \sum_{j=1..m} \left(\frac{1}{z^{occ}_{min_j}} - \frac{1}{z_{\mathbf{p}}} \right)^2 . \qquad (8)$$

Even though we have $z^{occ}_{min_j} \geq z^{occ}_{min}$, since for $j = 1$ these two quantities are equal, it is clear that for $m > 1$ we have $n_{ml} > n$. On the other hand our multi-view shadow map approach subdivides the light source into $S \times S$ sub-lights, and so the number of visited shadow map samples is:

$$n_{mv} = (\alpha \frac{w_l}{S})^2 \sum_{k=1..S^2} \left(\frac{1}{z^{occ}_{min_k}} - \frac{1}{z_{\mathbf{p}}} \right)^2 . \qquad (9)$$

Let us assume each sub shadow map view covers the same region as the initial main view. Then $z^{occ}_{min_k} = z^{occ}_{min}$ holds for each sub-view, and we have $n = n_{mv}$. This shows that in contrast to multi-layers, our multi-view approach does not increase the total number of visited samples, thus proving its higher efficiency. Of course, this theoretical result only holds for the visibility computation part, and both approaches still require the computations of multiple shadow maps and MSSM structures which are significantly expensive for complex scenes.

We acknowledge that the possibility of using multi-view shadow maps has already been mentioned [GBP06, SS07]. However, to the best of our knowledge, it has never been studied in detail for its high efficiency and quality, nor compared to depth peeling approaches.

6. Results and Discussions

We have implemented our soft shadow mapping algorithm using DirectX 10.0, and HLSL shader mode 4.0. The results presented in this section were measured using a 1.8GHz CPU with a NVIDIA GTX280 graphics card. Note that in this section, results reported as [GBP07] actually correspond to our own algorithm implementation with the packet and hierarchical optimizations disabled. In particular, unlike the original method, we included the computation of tighter kernels using the MSSM, but disabled the adaptive precision strategies as they reduce the quality.

Packet size choice

The most important parameter of our algorithm is certainely the size of the packets. Figure 12 shows average performance results obtained to render the scene of Figure 1-a using three different image resolutions, and six packet size configurations. These results were obtained without the hierarchical optimization. As it can be expected, larger image resolutions give better results with larger packets. Indeed, smaller packet sizes can not fully utilize the higher coherence given by a larger image, while larger packets may break the coherence. Since increasing the size of the light increases the screen space size of the penumbrae, large light sources suggest the use of larger packets.

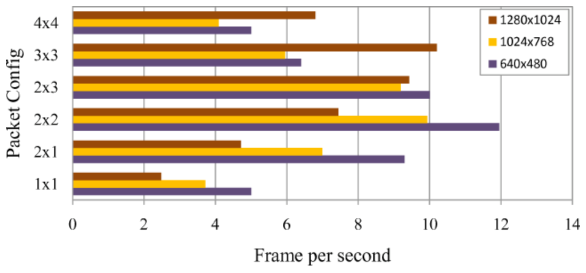

Figure 12: *Effect of the packet sizes on the performance for different image resolutions.*

Of course, the choice of the packet size also depends on many other criteria such as the geometry complexity. In addition, larger packets usually need more GPU registers that negatively affects its parallelism, preventing a full utilization of the GPU computation power. For all these reasons, for our current implementation and our test GPU, we found conservative packet sizes such as 2×2 or 3×3 packets to be good default choices.

Performance

Figure 13 shows the effect on the performance of our packet based and hierarchical traversal optimizations. These results were obtained with a single shadow map for the scene in Figure 9, using various light sizes, a 1280×1024 buffer, and 3×3 packets. As can be seen, our packet optimization always speeds up the computation even in the case of small penumbrae which exhibits a small coherence. Our hierarchical optimization yields another significant acceleration factor, from 1.5 to 4 times faster. This makes our overall algorithm an order of magnitude faster for large light sources. The percentage of coherent packets among all the non trivial packets for light sizes of 40, 80, and 160 (Figure 9) are 52%,68% and 88% respectively.

Our algorithm is very efficient at rendering low frequency shadows, or dealing with multiple light sources. For instance, Figure 1-b has been rendered using 9 textured light sources to coarsely simulate the environment lighting. Using 512×512 shadow maps, this scene is rendered at 24 FPS using our packet-based hierarchal method, that is ten times faster than Guennebaud *et al.*'s method [GBP07].

A hard case scenario for our method is a scene with many fine details breaking the coherence, such as Figure 1-d. Nevertheless, even in such a case, we found our algorithm still performs twice as fast as [GBP07] for a render buffer of 1280×1024 pixels, and 2×2 packets.

Figure 8 shows that with our hierarchical traversal algorithm, the number of visited shadow map samples is significantly reduced compared to only using a MSSM [SS07], or compared to the previous hierarchical method [DU07].

Figure 1-c shows a scene with an omni-directional light source (handled with 6 shadow maps) closely surrounded by an object. In such a case, standard SSM methods exhibit very

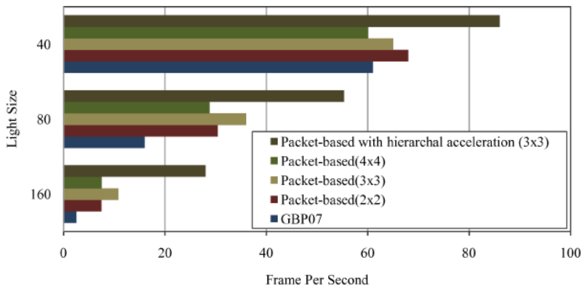

Figure 13: *Performance comparison of our different optimizations.*

low performance because the minimal occluder depth value z^{occ}_{min} in eq. (1) and Figure 2 is extremely small, and yields huge kernels. On the other hand our hierarchical approach allows to quickly prune very large parts of the initial kernel, and thus maintain real-time rates with a speed up of a factor 20.

Discussions

In our current system, the size of the packets has to be fixed once for all. As we saw, optimal packet sizes essentially depend on the screen space frequency of the penumbra. Therefore, significant speed enhancement could probably be obtained by extending our algorithm to automatically select the best packet size, not only for each frame, but also locally for each pixel of the screen.

Higher performance could also be obtained by improving the GPU parallelism. In particular, we suggest the use of a prefix-scan [HSO07] algorithm to pack the selected packets or pixels before processing them. Moreover, even though we presented our algorithm for current GPUs, we believe that our packet-based and hierarchical approaches are also very well tailored for other massively multi-core architectures such as Intel's Larrabee.

Also note that further acceleration can also be obtained by adopting the light space adaptive precision strategy of Guennebaud et al. [GBP07]. In practice, it suffices to stop the recursive traversal at a given level selected on a per packet basis. In that respect, the two approaches appear to be complementary.

When the number of practical lights increase, for instance, using multiple sub-lights, and/or omnidirectional light sources represented by six orthogonal sub-lights, the rendering time can become largely dominated by the rendering of the scene in the shadow maps and the construction of the MSSM. In order to get further overall speed improvements, future research should not only focus on the visibility computation part, but also on improving the construction of multiple shadow maps and multiple MSSMs, or investigate novel faster data structures.

From the quality point view, our method suffers from the same limitations than the original contour based SSM. In particular, it is important to recall that noticeable aliasing artifacts occur when the penumbrae become sharp, and/or, in the case of distant objects. To overcome this limitation, one could investigate the combination of SSM with, e.g., parallel-split shadow maps [ZSXL06] which allow to refine the shadow map where more accuracy is needed.

7. Conclusion

We have presented a packet-based hierarchical soft shadow mapping method for rendering convincing soft shadows in dynamic scenes with outstanding real time performance. To this end, we showed the SSM kernels exhibit a very high coherence. That motivated us to design a packet-based algorithm amortizing floating point operations and texture fetches across multiple screen pixels. To further enhance the performance we proposed a hierarchical method which quickly prunes large blocks of trivial samples. We showed this later optimization allowed us to get rid of the huge performance penalty of previous SSM methods when dealing with occluders close to the light source. At last, we presented and discuss multi-view shadow maps as a solution to the single shadow map artifacts. We showed its superior performance and antialiasing capability than multi-layer methods.

Acknowledgments

This work is supported by the NSFC (No. 60873046 and 60603078), the 973 Program of China (No. 2006CB303102), NSF of Zhejiang Province (No. R106449), Microsoft Research Asia, and the Bird associated team (INRIA-DREI). We thank Louis Bavoil for his support regarding his depth-peeling technique. The original Sponza Atrium model is made by Marko Dabrovic, RNA studio, www.rna.hr.

References

[AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Trans. Graph. 22*, 3 (2003), 511–520.

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph. 27*, 3 (2008), 1–8.

[AHL*06] ATTY L., HOLZSCHUCH N., LAPIERRE M., HASENFRATZ J.-M., HANSEN C., SILLION F.: Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum 25*, 4 (dec 2006), 725–741.

[AL04] AILA T., LAINE S.: Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004* (2004), Eurographics Association, pp. 161–166.

[BCS08] BAVOIL L., CALLAHAN S. P., SILVA C. T.: Robust soft shadow mapping with backprojection and depth peeling. *journal of graphics tools 13*, 1 (2008), 19–30.

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. *SIGGRAPH Comput. Graph. 11*, 2 (1977), 242–248.

[Déc05] DÉCORET X.: N-buffers for efficient depth map query. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3 (2005).

[DU07] DMITRIEV K., URALSKY Y.: Soft shadows using hierarchical min-max shadow maps. In *Game Development Conference* (March 2007).

[ED06] EISEMANN E., DÉCORET X.: Plausible image based soft shadows using occlusion textures. In *Proc. of the Brazilian Symposium on Computer Graphics and Image Processing* (2006), IEEE.

[ED07] EISEMANN E., DÉCORET X.: Visibility sampling on gpu and applications. *Computer Graphics Forum (Proceedings of Eurographics 2007) 26*, 3 (2007).

[FBP06] FOREST V., BARTHE L., PAULIN M.: Realistic soft shadows by penumbra-wedges blending. In *ACM SIGGRAPH/Eurographics symposium on Graphics hardware* (2006), pp. 39–46.

[FBP08] FOREST V., BARTHE L., PAULIN M.: Accurate Shadows by Depth Complexity Sampling. *Computer Graphics Forum, Eurographics 2008 Proceedings 27*, 2 (2008), 663–674.

[Fer05] FERNANDO R.: Percentage-closer soft shadows. 35.

[GBP06] GUENNEBAUD G., BARTHE L., PAULIN M.: Real-time soft shadow mapping by backprojection. In *Eurographics Symposium on Rendering* (2006).

[GBP07] GUENNEBAUD G., BARTHE L., PAULIN M.: High-quality adaptive soft shadow mapping. *Computer Graphics Forum 26* (2007), 525–533.

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. *Computer Graphics Forum 22*, 4 (dec 2003), 753–774.

[HSO07] HARRIS M., SENGUPTA S., OWENS D. J. D.: Parallel prefix sum (scan) with cuda. In *GPU Gems 3*, Hubert Nguyen (Ed.), 851-875, 2007.

[JHH*09] JOHNSON G. S., HUNT W. A., HUX A., MARK W. R., BURNS C. A., JUNKINS S.: Soft irregular shadow mapping: fast, high-quality, and robust soft shadows. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), ACM, pp. 57–66.

[LAA*05] LAINE S., AILA T., ASSARSSON U., LEHTINEN J., AKENINE-MÖLLER T.: Soft shadow volumes for ray tracing. *ACM Transactions on Graphics 24*, 3 (2005), 1156–1165.

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. In *Proc. of SIGGRAPH* (1987), pp. 283–291.

[SEA08] SINTORN E., EISEMANN E., ASSARSSON U.: Sample-based visibility for soft shadows using alias-free shadow maps. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2008) 27*, 4 (June 2008), 1285–1292.

[SS98] SOLER C., SILLION F. X.: Fast calculation of soft shadow textures using convolution. In *Proc. of SIGGRAPH* (1998), pp. 321–332.

[SS07] SCHWARZ M., STAMMINGER M.: Bitmask soft shadows. *Computer Graphics Forum 26* (2007), 515–524.

[SS08] SCHWARZ M., STAMMINGER M.: Quality scalability of soft shadow mapping. In *GI '08: Proceedings of graphics interface 2008* (Toronto, Ont., Canada, Canada, 2008), Canadian Information Processing Society, pp. 147–154.

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Saarland University, 2004.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph. 12*, 3 (1978), 270–274.

[ZSXL06] ZHANG F., SUN H., XU L., LUN L. K.: Parallel-split shadow maps for large-scale virtual environments. In *Proc. of the Virtual reality continuum and its applications* (2006), ACM press, pp. 311–318.