EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

High-Quality Adaptive Soft Shadow Mapping

Gaël Guennebaud[†], Loïc Barthe[‡] and Mathias Paulin[‡]
[†] ETH Zurich [‡] IRIT - Université Paul Sabatier

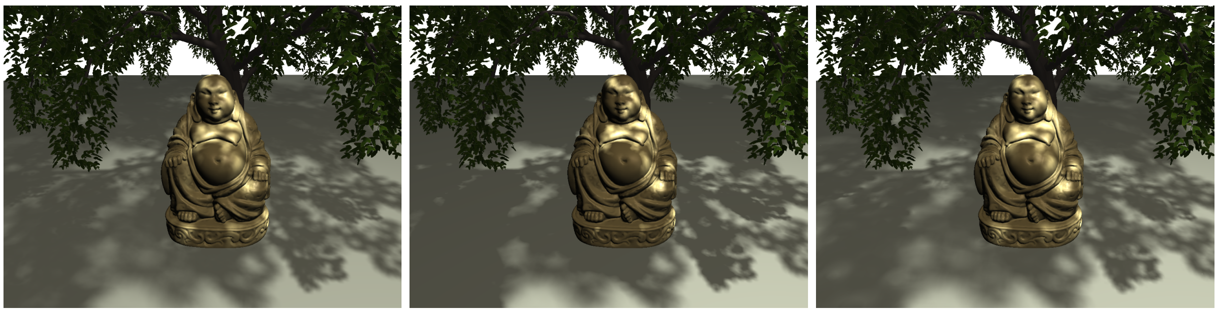
Figure 1: *A complex scene rendered with soft shadows in a 768 × 768 image. From left to right: ground truth (1024 light samples), our previous method [GBP06] and our new algorithm at 24 fps (hard shadow mapping is performed at 41 fps).*

Abstract
The recent soft shadow mapping technique [GBP06] allows the rendering in real-time of convincing soft shadows on complex and dynamic scenes using a single shadow map. While attractive, this method suffers from shadow overestimation and becomes both expensive and approximate when dealing with large penumbrae. This paper proposes new solutions removing these limitations and hence providing an efficient and practical technique for soft shadow generation. First, we propose a new visibility computation procedure based on the detection of occluder contours, that is more accurate and faster while reducing aliasing. Secondly, we present a shadow map multi-resolution strategy keeping the computation complexity almost independent on the light size while maintaining high-quality rendering. Finally, we propose a view-dependent adaptive strategy, that automatically reduces the screen resolution in the region of large penumbrae, thus allowing us to keep very high frame rates in any situation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismColor, shading, shadowing, and texture

1. Introduction

Soft shadows are among the most important lighting effects. In addition to increasing the realism of synthetic images, soft shadows simplify the identification of spatial relationships between objects without the aggressiveness of hard shadows. Over the past decade, many researchers have focused on soft shadow rendering. However, achieving high-quality soft shadows with real-time performance remains a challenging open problem. From the practical point of view, the rendering of soft shadows is equivalent to solving a visibility problem between a point and an extended light source, which can be either a surface or volume. Ideally, a soft shadow algorithm should be able to handle dynamic and complex scenes in real time, should not distinguish receivers from occluders, and should generate shadows as faithful as possible to real ones.

Recently, the two common hard shadows rendering techniques, shadow volumes [Cro77] and shadow mapping [Wil78], have been extended to support soft shadows with respectively, penumbra-wedges [AAM03, ADMAM03] and sample back-projections [GBP06]. Interestingly, these two extentions remain sufficiently close to their respective hard shadow version, so that the well known advantages and drawbacks of shadow volumes versus shadow maps can be directly generalized to them. In particular, the Guennebaud et al. soft shadow mapping (SSM) technique [GBP06] renders approximate soft shadows from a single shadow map per light source without any other assumptions or precomputation. This approach can therefore handle all rasterizable geometries and it is well suited to render both complex and dynamic scenes with real-time performance. However, the approach currently exhibits some limitations which reduce its practical use. From the quality point of view, the current back-projection method has to deal with gaps and overlapping artifacts between the shadow map samples. Owing to the complexity of the problem, only gaps are coarsely filled, increasing the overlapping error and thus leading to noticeable overestimations of the shadow. Furthermore, the performance of the method drops significantly for large penumbrae (e.g., when using large light sources or when objects are very close to a light source). To keep a high frame rate, an adaptive precision strategy is introduced but, unfortunately, it generates noticeable discontinuities between different levels of precision. Finally, the method suffers from the common single light sample approximation.

In this paper, we address all the aforementioned limitations of SSM, except for the single light source sample problem, which will be the topic of further investigations. Our contributions include a new, more accurate, visibility computation method based on an efficient contour detection procedure. Combined with radial area integration, it overcomes the gap and overlapping artifacts and reduces aliasing. This new visibility computation procedure is also more efficient, especially in the case of large penumbrae, since it only back-projects occluder contours instead of all occluder samples. Secondly, inspired by *trilinear* mipmap filtering, we propose a smoothing method that removes the discontinuities produced by the light space adaptive strategy with a negligible overhead. Finally, we propose an original screen space, view dependent, adaptive sampling strategy that automatically reduces the screen resolution in regions of large penumbrae. Complete visibility information is then reconstructed efficiently using a pull-push algorithm. This optimization allows huge acceleration, since the reduction of the screen resolution by a factor of four theoretically speeds up the computation by a factor of sixteen. As a result, we obtain a practical framework that produces realistic soft shadows with a very high frame rate, hence leaving resources available for other algorithms, enhancing the image quality and realism in real-time rendering applications, such as physics simulations and high quality material rendering.

2. Related Work

We briefly review the most recent contributions in real-time soft shadow rendering. A more complete survey can be found in Hazenfratz et al. [HLHS03]. Hard shadows come from unrealistic point light sources and are classically rendered using either shadow volumes [Cro77] or shadow mapping [Wil78]. Both approaches have their respective advantages and drawbacks. The former accurately defines the shadow boundary via a geometric, but expensive, silhouette extraction, the latter requires only the fast and more generic acquisition of a depth image. The discrete nature of shadow maps, however, leads to aliasing that can be reduced by either increasing the effective shadow map resolution [FFBG01, SD02, WSP04] or by filtering the boundaries [RSC87].

Shadow volumes were recently extended with penumbra-wedges [AAM03, ADMAM03] in order to provide the simulation of extended light sources with penumbrae. This method constructs and rasters a wedge for each silhouette edge seen from the source center and, therefore, it is limited to manifold meshes having a relatively low complexity. The occluded area is radially integrated using back-projection and additive accumulation between occluders. This usually leads to overestimated shadows that can be improved using more accurate, but expensive, blending heuristics [FBP06]. Some hybrid methods [CD03, WH03] combine a geometric silhouette extraction with a shadow map. While being more efficient than the rasterization of penumbra-wedges, such approaches can only compute a coarse approximation of the external penumbrae.

Compared with methods based on an object space silhouette extraction, purely image based techniques are especially attractive, since they support any type of rasterizable geometry (e.g., meshes, point-clouds, and binary alpha-textured models) and they are less sensitive to the scene complexity. While some require the rendering of multiple shadow maps per light [ARHM00, HBS00, SAPP05], limiting their use to static scenes only, others try to keep high performance using a single light sample. However, most of these latter techniques rely on heuristics rather than visibility computations or require limitations on the scene. For instance, some are limited to planar receivers [SS98] while others incorrectly take into account the occluder's shape as well as occluder fusion [BS02], and also generate popup effects when an originally hidden shadow appears [AHT04]. Eisemann and Décoret [ED06] approximate the scene by a set of flat slices that are combined using a probabilistic approach.

The idea of soft shadow mapping (SSM) was recently introduced by Atty et al. [AHL*06] and Guennebaud et al. [GBP06]. It overcomes most of the limitations of these previous methods. Similar concepts are also presented by Aszódy et al. [ASK06] and Bavoil et al. [BCS06]. The clue is to use a single shadow map as a discretized representation of the scene, the visibility being computed by back-projection of the shadow map samples onto the light source. However, while Atty's approach [AHL*06] separates occluders and receivers and is limited to small shadow map resolutions, Guennebaud [GBP06] keeps all the advantages of standard shadow mapping and presents several optimizations. This latter approach, on which this paper improves, is summarized in the next section. Note that, the concept of back-projection was initially proposed to compute *offline* accurate soft shadows [DF94].

3. Soft Shadow Mapping Settings

Guennebaud et al.'s soft shadow mapping (SSM) technique [GBP06] computes a so-called *visibility buffer* (v-buffer) storing the percentage of light (the *visibilty factor*, $v_p \in [0,1]$) seen from each 3D point, $\mathbf{p}$, of the scene corresponding to a screen pixel. To simplify the explanation, we present the approach for a single square light source of width w_l.

At each frame, SSM first computes a common shadow map providing a discretized representation of the scene seen from the light source's center. This acquisition step requires the definition of a projection frustum having its *near plane* and its borders taken parallel to the light and at a distance z_n from the origin. Let w_n be its width and r its resolution (Figure 2a). The algorithm approximates the visibility factor v_p of a given point $\mathbf{p}$ of depth z_p by accumulating the light area occluded by each shadow map sample. Each sample s of depth z_s is interpreted as a small 3D quadrilateral parallel to the light source that is back-projected from $\mathbf{p}$ onto the light source and clipped to the light's borders. In practice, only *occluding samples* are back-projected, i.e., samples s for which $z_s < z_p$. However, because samples do not join perfectly, this approach is prone to gaps (some occluded parts of the light are not removed) and overlapping artifacts (some parts are removed several times). Owing to the problem's complexity, only gaps are coarsely filled by extending the sample to its neighbors, increasing the overlaps and thus the overestimation of the shadows. Let us define the *kernel* (called "search area" in [GBP06]) to be the squared region aligned with the shadow map's space axis, containing the subset of all potentially occluding samples, and let w_k be its width in pixels (Figure 2a).

In order to further optimize the integration step, a hierarchical version of the shadow map (HSM) is built from high to low resolution in a similar fashion to mipmapped textures. Each pixel stores both the minimum and maximum covered depth values. This HSM is very cheap to compute and it is the trick that admits all the optimizations. As a first step, it is used to compute, iteratively, a very tight approximation to the kernel size. Indeed, the subset of occluding samples is necessarily included in the pyramid defined by the light quadrilateral and the current point $\mathbf{p}$, and it is further away than the closest sample of depth z_{min} (Figure 2a). This global *min* depth value is given by the top level of the HSM. A first approximation to the kernel is therefore given by the projection onto the shadow map plane of the intersection between the pyramid and a parallel plane at z_{min}:

$$w_k = w_l \frac{z_n r}{w_n} \left(\frac{1}{z_{min}} - \frac{1}{z_p} \right) \tag{1}$$

From this first estimate, a more accurate local *min* depth value z'_{min} is obtained directly from the level of the HSM leading to a kernel size just below one pixel, i.e., the level number $\lfloor \log_2(w_k) \rfloor$. This local *min* depth value allows the kernel size to be optimized iteratively until convergence. Next, the comparison of the depth of the current point $\mathbf{p}$ with the depth bounds values of the optimized kernel allows us to efficiently and conservatively check whether $\mathbf{p}$ is fully lit, fully occluded or potentially in the penumbra. The accurate visibility computations are performed in this last case only. In spite of these optimizations, the approach has linear complexity with respect to the area of the light and it linearly depends on the number of pixels in the penumbra.

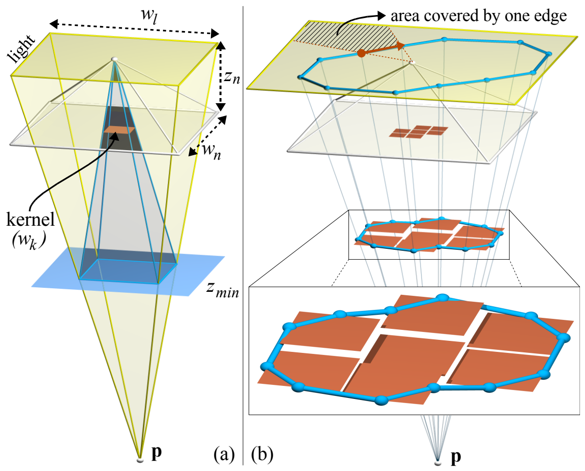

Figure 2: *(a) Shadow map parameters and computation of the kernel size. (b) Overview of our new visibility computation procedure.*

In the following section we present a new, physically plausible, visibility computation procedure that naturally overcomes the gap and overlapping artifacts. In section 5 we then present high quality adaptive light space and view dependent optimization schemes.

4. Accurate Visibility Computation

Building on the soft shadow mapping framework described in the previous section, we present a new, physically plausible, visibility computation procedure. For each visible point $\mathbf{p}$, our algorithm first detects the contour edges of the occluders seen from $\mathbf{p}$ and they are then back-projected onto the 2D light source domain, in which the occluded area is radially integrated.

4.1. Smooth Contour Detection

Our first task is the detection of the external silhouettes of the occluders seen from the current point $\mathbf{p}$. Due to our simplified representation of the scene by a shadow map, such a silhouette is actually the contour of the aggregates of adjacent occluding samples of the shadow map. More generally, because the shadow map is generated by a rasterization process, any contour that strictly contains all centers of occluding samples is valid. Thus, by analogy to the marching square algorithm (the equivalent of the marching cube algorithm [LC87] in the 2D domain), we propose a convenient contour detection algorithm that creates edges connecting two adjacent sample borders (Figure 2b). Moreover, our method allows us to grow or shrink the contour slightly using a parameter $t \in [-1, 1]$. We opted for such a scheme because it does not require extra edge segments for the connection of adjacent sample borders with different depth values, it leads to an efficient detection algorithm, and it smooths the contour, thus reducing aliasing.

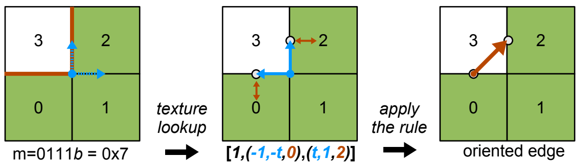

Figure 3: *Illustration of our local contour edge detection procedure. Green squares represent occluding samples. Left: the two boundaries are highlighted in red and the local frame coordinate is shown in dotted blue. Middle: the mask m is used to index a table giving the edge detection rule that includes the number of edges (in black) and the 3D coordinates of the edge, each extremity being implicitely defined by a 2D offset (in blue) and the index of the sample holder (in red). Right: the reconstructed oriented edge.*

Our algorithm is based on a set of local edge detection rules that are applied to each 2×2 block of samples intersecting the current kernel (Figure 3). For each block, a four bit mask, m, is built according to the occluding states of the four samples of the current block: 1 if it is an occluder and 0 otherwise. This mask is used to index a precomputed table storing for each case:

- the number of detected edges (0,1 or 2),
- the 2D coordinates of the extremities of the first edge (if any) defined in a local frame having its origin at the block center (shown in blue in Figure 3(left),
- the two indices (between 0 and 3) of the samples of the block holding the extremities.

The two indices of the extremities are used to read their respective depth values from the depth of the block samples, thus allowing us to reconstruct an edge with 3D coordinates. Note that the 2D coordinates defining the extremities are controlled by the global parameter $t \in [-1, 1]$. This allows us to shrink or grow the contour curve such that when $t = 0$ the contour joins the sample boundaries, as in Figure 2b and when $t = 1$ (resp. $t = -1$) the contour joins the centers of occluder (resp. background) pixels. Figure 4 illustrates four different edge detection rules. All the other rules are easily obtained by symmetry and the two trivial cases $m = 0\mathrm{x}0$ or $m = 0\mathrm{xF}$ never lead to an edge and are skipped. Note that, for the rare cases leading to two edges, it is sufficient to store the parameters of the first edge since the second is trivially obtained by *central symmetry* (i.e., negate the coordinates) and adding 2 to the index defining the sample holder. Furthermore, the sorting of the edge extremities is provided in a consistent manner, i.e., such that the occluder is always on the same side of the oriented edge. This is especially important to correctly integrate the area, as explained in the next section.

While any value of $t \in [-1, 1]$ is plausible, we recommend $t = 0$ as the most reasonable choice from a probabilistic point of view. The limit choice, $t = 1$, erodes the contour and fewer cases generate edges. Hence, choosing $t = 1$ simplifies and accelerates the whole algorithm and it is appropriately used

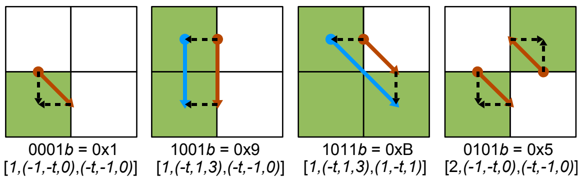

Figure 4: *Illustration of four cases with their respective masks and rules. The red arrows represent the edges for $t = 0$ and the blue ones those for $t = 1$.*

in our light space multi-resolution strategy, described in section 5.1.

4.2. Radial Area Integration

Finally, the occluded area is radially integrated from the light center by accumulating the signed area *covered* from each contour edge, in a similar way to the one used in the penumbra-wedges technique [ADMAM03] (Figures 2b and 5a). To summarize, each edge is projected onto the 2D normalized light source space and clipped by the light's borders. The area *covered* by one edge can either be accurately computed or directly obtained from a precomputed 4D texture for animated textured light sources. Finally, this area is added or subtracted according to the sorting (clockwise or counterclockwise) of the edge extremities. Lengyel [Len05] and Forest [FBP06] give more details and efficient implementations of this step. The main difference arises in the process initialization. The integrated area has to be initialized with the full light area if the light center is visible and with zero otherwise. Even though this step is equivalent to the computation of hard shadows from the light center, it is not possible to directly use the shadow map in a standard way. Instead, we have to check carefully whether the projection $\mathbf{p}'$ of $\mathbf{p}$ onto the shadow map lies inside or outside our occluder contour. This is achieved by applying our local edge detection procedure on the four closest samples around $\mathbf{p}'$: the signs of the 2D cross products between $\mathbf{p}'$ and the contour edges (if any) give us the inside/outside relationship. Figure 5 illustrates this process and compares it to standard shadow mapping.

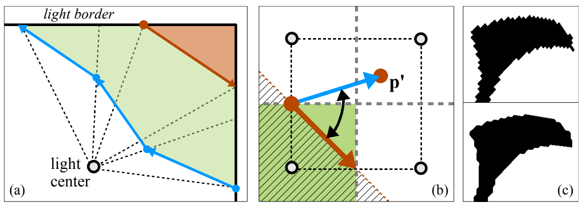

Figure 5: *(a) Radial area integration: the area covered from each edge is subtracted for the blue edges (counterclockwise ordering) and added for the red one (clockwise). (b) Computation of the hard shadow boundary matching our contour detection approach. (c) Comparison of the hard shadows produced with standard shadow mapping (top) and our contour approach (bottom).*

5. Adaptive Computations

We describe how our SSM algorithm can be considerably accelerated when taking into account the fact that large penumbrae are low frequency phenomena. Indeed, low frequencies can be rendered with less precision for the same visual quality [DHS*05]. We present both a light space and a screen space multi-resolution process. While the former aims to maintain a constant kernel size, the second automatically reduces the output resolution where the penumbra fills a large region.

5.1. Light space multi-resolution

The kernel size w_k linearly varies with respect to the width of the penumbra. Hence, as proposed by Guennebaud et al. [GBP06], light space adaptive precision can be achieved by locally using the finest HSM level that yields a kernel size (in pixels) lower than a given threshold t_k, i.e., the level lod:

$$lod = \lfloor \log_2(w_k/t_k) \rfloor \tag{2}$$

However, discontinuities occur when the level changes from one pixel to another (Figure 6a). Thus, we introduce linear filtering between the different levels, inspired by common trilinear mipmap filtering (see [SA06]).

To this end, we would need a continuous estimate of the best suited HSM level. As can be seen in equations 1 and 2, this estimate depends on the optimized local z'_{min} depth value that varies discontinuously due to the use of a *min* operator. Therefore we propose to compute a continuously varying estimate of the closest occluder depth z_{occ} using linear filtering of the occluder depth values fetched in the appropriate HSM level. Note that this average occluder depth value is only used as a hint to perform the smoothing and is not used to estimate the kernel size, which is still conservatively computed from the minimal depth value. Let $\widetilde{w_k}$ be the smooth kernel size computed using z_{occ} instead of z_{min} in equation 1. The continous level parameter λ is then:

$$\lambda = \log_2(\widetilde{w_k}/t_k) \tag{3}$$

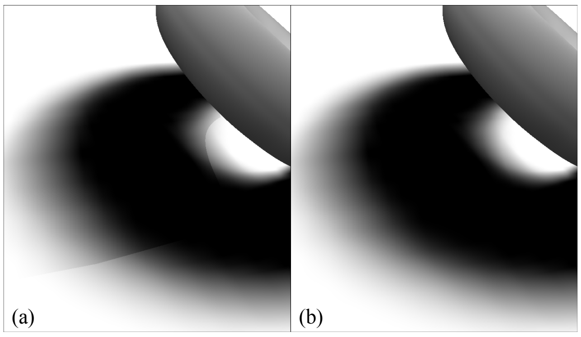

Figure 6: *(a) Light space adaptive precision exhibits discontinuities at the transition level. (b) Discontinuities are smoothed by our local filtering method.*

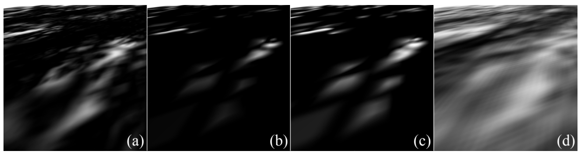

Figure 7: *Close-up view of the v-buffer of complex geometry obtained with an average of* $2k$ *shadow maps of (a)* $2k \times 2k$ *and (d)* 256×256 *pixels. The v-buffer obtained with our visibility computation on the* 256×256 *pixel level (b) without and (c) with our shrinking correction.*

Blending is then performed as for standard trilinear mipmap filtering: two visibility factors are computed using the two closest levels and the final value is linearly interpolated. In practice, instead of always computing the visibility factor twice, we reduce the blending operation to pixels which are close to a transition level. Thus, the overhead cost of this filtering is almost negligible. Let β be the width of the blending region ($\beta = 0.05$ is a typical choice). This is achieved by the following procedure:

```
lod = floor(lambda)
vp = vis(p,lod)
d = lambda - lod
if (d<beta)
  vp = lerp((1-d/beta)/2, vp, vis(p,lod-1))
else if (1-d<beta)
  vp = lerp((d/beta)/2, vp, vis(p,lod+1))
```

Ideally, each level should be computed independently by a true rasterisation of the scene. In addition to being too expensive, this may also lose the fine details of complex geometry (Figure 7d). Thus, we first use the *min* depth component of the HSM, as suggested by Guennebaud et al. [GBP06]. Then, we counterbalance the increase of the size of the foreground occluders (Figure 7b) by shrinking the occluder contours built from low levels using our edge detection algorithm with $t = 1$ (Figure 7c).

5.2. Screen space multi-resolution

The previous optimization speeds up the algorithm significantly and allows us to guarantee a given performance, since the kernel size is bounded. Additional significant accelerations can also be achieved by adjusting the screen resolution according to the screen space size of the penumbrae. In practice, this is done by cancelling the visibility computation of some screen pixels. The missing information is efficiently reconstructed using a pull-push algorithm.

View dependent selection

Figure 9 shows our screen space penumbra size estimation procedure. First, we compute a conservative estimation of the object space penumbra size s_{obj} using the continuous occluder depth estimation z_{occ}:

$$s_{obj} = w_l \frac{p_z - z_{occ}}{z_{occ}} \tag{4}$$

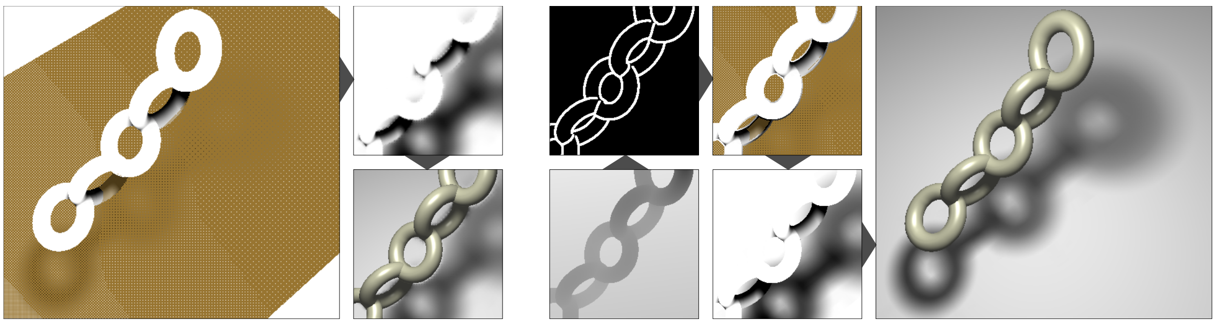

Figure 8: *Illustration of our view dependent pipeline. Left three images: the v-buffer before and after the pull-push reconstruction (skipped pixels are shown in orange). A naïve sampling leads to a wrong reconstruction. Right five images: the same process but considering the screen space depth and normal discontinuities.*

Then, the penumbra screen space size s_{scr} is conservatively computed as the smallest diameter of the projection of a disk of radius s_{obj}, centered at $\mathbf{p}$ and of normal $\mathbf{n}$, onto the screen space:

$$s_{scr} = \eta \frac{s_{obj}}{z_{p_e}} \mathbf{n}^T \mathbf{v} \tag{5}$$

where $\mathbf{v}$ is the view vector, η is the scaling factor of the camera frustum and z_{p_e} is the depth of the point $\mathbf{p}$ in the camera space. Note that the use of the surface normal allows us to increase the sampling density in view tangential regions.

The density of the selected screen pixels is then adjusted according to s_{scr}. In order to get a uniform variation of the selected pixels, we use a precomputed pattern similar to that used in dithering. An example of such a 4×4 pattern is illustrated in Figure 9 and the visibility factor of a pixel is computed if and only if the following condition is satisfied:

$$s_{scr} < s_{min} \cdot \sqrt{pattern[i\%h, j\%h]}, \tag{6}$$

where h is the pattern width, i and j are the screen space coordinates of the current pixel corresponding to $\mathbf{p}$, and s_{min} is a user-defined constant seting the minimum penumbra size at which we start to skip screen pixels. For a fully automatic process, we suggest $s_{min} = 16$. Figure 8(left) shows the result of such an adaptive selection.

The use of larger patterns allows us to obtain lower sampling densities and hence get even better performance. However, small patterns are sufficient because our estimation

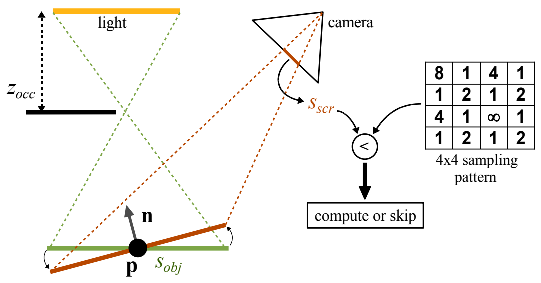

Figure 9: *Evaluation of the screen space penumbra width and an example of a 4×4 sampling pattern of the screen space.*

method only considers the closest occluder to the light, and thus it overestimates the penumbra width, specially at the fusion of a large and a small penumbra. Moreover, even a 2×2 pattern, which only reduces the sampling resolution by a factor of two, can, in the best case, increase the performance by a factor of four.

Pull-push reconstruction

The above procedure leads to a sparse, unstructured visibility buffer that may contain several gaps. In order to reconstruct the v-buffer smoothly, we use the pull-push algorithm described by Gortler [GGSC96]. A weight buffer is associated with the visibility buffer and initialized to 1 for the computed pixels and 0 for the gaps. During the pull phase, the visibility and weight buffers are iteratively reduced by accumulating the weights and averaging the visibility factors until the weight buffer is completely saturated. Then, during the push phase, the gaps are iteratively filled, from the complete lowest resolution up to the highest by linearly blending the current, incomplete, high resolution buffer with the current, complete, low resolution buffer scaled up using bilinear filtering. Gortler [GGSC96] and Grossman [GD98] provide mode details of this step.

This pull-push reconstruction procedure is especially well suited for our purpose since it does not require any information about the local size of the gaps and it allows a straightforward and very efficient GPU implementation. However, because the algorithm does not consider shadow discontinuities due to depth or normal discontinuities of the receivers, the algorithm may wrongly interpolate the penumbra across two different surfaces, as illustrated in Figure 8(left). We solve this problem with a slight modification of the above adaptive sampling scheme. We perform a full sampling close to any potential discontinuity (Figure 8(right)) using a screen space discontinuity buffer computed with a contour detection filter (e.g., a thresholded Sobel filter) on the depth and normal buffers of the scene seen from the camera. Then, the discontinuity lines are increased such that their width is greater than the maximum allowed pixel spacing and the visibility factor of a pixel is evaluated if it either satisfies the equation 5 or is on a discontinuity.

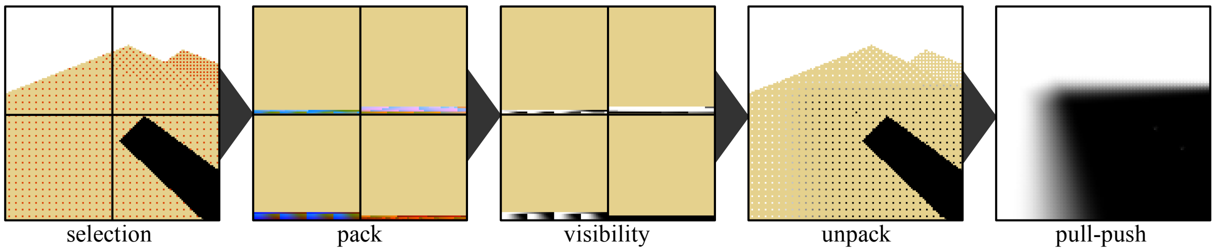

Figure 10: *Illustration of the implementation of our algorithm with sparse view dependent selection. Left: the screen space is divided into four blocks. The selected pixels are shown in red. Fully lit and fully umbra pixels are shown in white and black respectively. See section 6 for details.*

6. Implementation

Without our view dependent selection, our algorithm can be efficiently implemented as described by Guennebaud et al. [GBP06], i.e., using a deferred shading like strategy, allowing visibility computations to be made independent of the scene complexity. The selection of "in penumbra" pixels is advantageously implemented using either dynamic branching for recent GPUs (GeForce 8x00 and Radeon X1x00) or *early-z* rejection with two passes for older ones (GeForce 7x00). Because our view dependent selection algorithm generates a very sparse fragment repartition, we cannot take full advantage of dynamic branching that treats fragments per block of about 32 pixels. Unfortunately, the *early-z* rejection capability of current GPUs suffers from the same limitation and, after extensive experiments, we propose the following implementation. After computing the required buffers (the shadow map and its hierarchy, the screen space attribute buffer, and the discontinuity buffer), the complete v-buffer is computed in five passes as illustrated in Figure 10.

The first pass classifies each pixel as occluded, lit or in penumbra, and, in the latter case, applies view dependent selection. The result of this pass is written in the v-buffer. The second pass *packs* the selected penumbra pixels such that the visibility computations can be performed efficiently. The screen space is divided into several blocks of $h \times h$ pixels (e.g., 64×64) that are sent to the GPU using the `GL_POINTS` primitive, one primitive per pixel. A *geometry shader* cancels the unselected points and accumulate the others into a *transform feed-back buffer* (see [Nv0] for the details on these new features). Then, the visibility factors are computed by sending a small quadrilateral per block that minimally covers the selected points, i.e., a quadrilateral of size $h \times \frac{n_i}{h} + 1$ where n_i is the number of points of the block i. The fourth pass *unpacks* the data at their original position into the v-buffer by sending a `GL_POINTS` primitive for each selected point of each block. Finally, the pull-push algorithm reconstructs the complete v-buffer that is used in the final shading pass. In order to reduce the overhead generated by our additional passes, we perform the classification pass (pass one) per block using occlusion queries to estimate the ratio of pixels that have to be accurately processed. Then, each compact block with a ratio above 80%, for example, is directly processed by drawing a quadrilateral covering the whole block. On the other hand, a block with a ratio of zero is simply skipped.

7. Results

We have implemented our new soft shadow algorithm on a Pentium 2.80Ghz with a NVIDIA GeForce 8800-GTS graphics card, using the high level OpenGL shading language for the GPU implementation. We first study the quality and the effectiveness of our new visibility computation procedure and then analyze the advantages and the quality of our adaptive strategy.

7.1. Visibility Computation

The main advantage of our new visibility computation is its ability to generate very high quality soft shadows. Figure 1 compares our new algorithm with our previous one [GBP06] on complex geometry with a high depth complexity. While the previous gap-filling approach over-shades all the fine shadow details, our new approach remains very close to the reference image obtained by averaging the result of 1024 high resolution hard shadow maps ($2k \times 2k$). A comparison on a simpler example and with the penumbra wedge technique is given in Figure 11. We can clearly see the shadow overestimation of both previous approches and the high quality of our new technique.

With respect to performance, the computation of the HSM requires less than 1ms for a $1k \times 1k$ shadow map. The classification of pixels and the initialization of the integration are also negligible. The most critical part is the visibility computation, where the treatment presented in section 4 loops on the kernel pixels. With the previous back-projection and gap-filling method, occluding samples are detected in two instructions and back-projected in about eighteen vector instructions. Our new approach, requires about seven instructions to detect an edge and 32 additional instructions for the back-projection. However, in our new approach, the back-projection is performed for contour edges only. Hence, in addition of being more accurate, it is theoretically faster, especially in the presence of large uniform penumbrae. Using a CPU implementation, our approach is $4-10$ times faster. Unfortunately, we have not yet succeeded in reaching such acceleration with our GPU implementations, and we observe accelerations up to only 1.5 times.

7.2. Adaptive Precision Strategies

From the practical point of view, the smoothing of the discontinuities required by our light space adaptive strategy (section 5.1) produces an overhead of about 10%. This is

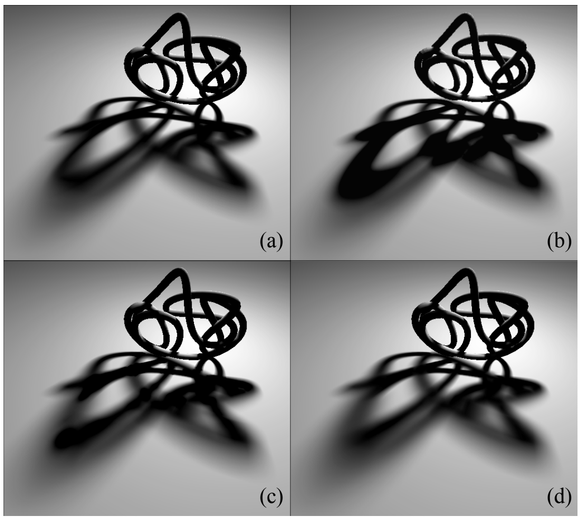

Figure 11: *(a) reference picture, (b) penumbra wedges, (c) our previous and (d) our new visibility computation.*

largely compensated by the gain provided by the multi-resolution representation. The additional passes required by our view dependent strategy (section 5.2) only depend on the screen size and are very cheap. Indeed, with a 768×768 screen resolution, both the computation of the discontinuity buffer and the pull-push reconstruction are performed within $\sim$0.8ms. The overhead due to our *pack* and *unpack* operations is $\sim$1ms and it is largely compensated by the benefit that they bring. The visual improvement and the performance obtained with both our screen and light space adaptive precision strategies are illustrated in Figure 12. As can be seen, with aggressive parameter values, the former has a tendency to fill small lit areas (Figure 12c) while the latter leads to smoother shadows (Figure 12b). Used together, reasonable settings provide very high quality and allow us to reach high performance (Figure 12d).

8. Discussion and Conclusion

We have presented a physically plausible algorithm to compute visibility between a point and an extended light. This has allowed us to generate high quality soft shadows on dynamic and complex scenes with real-time performance. Our algorithm keeps all the advantages of classical shadow mapping. In particular, it can be applied to any rasterizable geometry, and there is no distinction between occluders and receivers. Even though we have presented our approach on rectangular light sources, in practice our edge based integration procedure is compatible with textured planar light sources as well as circular or spherical lights [ADMAM03].

The error magnitude introduced by the discretization step depends on the penumbra size: the larger the penumbra, the smaller the error. This is exploited in this paper in order to improve performance by reducing accuracy in regions of large penumbrae while still maintaining high-quality shadows. On the other hand, when the light-umbra transition becomes sharp, e.g., in the case of objects in contact, the aliasing problem of shadow maps may appear. Even though the proposed contour-reconstruction method reduces aliasing, it would be interesting to integrate into our algorithm one of the many existing methods that increase the effective shadow map resolution. An alternative and simpler solution would be to replace aliasing by blur, i.e., by maintaining a minimal width of the penumbrae. To this end, it is sufficient to artificially increase the light size (per pixel) such that the kernel size is always above a given threshold value.

Central to our algorithm is the use of a single shadow map per light source as the scene representation. However, because a single depth image cannot faithfully represent a general 3D scene, our approach suffers from the common *single light sample* approximation that is discussed in previous works [AAM03, GBP06]. The largest errors occur at the penumbra fusion of occluders that have a high relative depth difference. For simple scenes, this approximation can be reduced by splitting the light sources into multiple smaller ones. Alternatively, extending our approach with layered depth images [ARHM00] could be considered. Here the challenging problems concern the fast acquisition of the multiple layers and the proper combination of their respective contributions. Nevertheless, the examples shown in this paper and in the accompanying video show that our approach adapts well to this approximation, even in the case of a scene with a high depth complexity.

Acknowledgments

This work was supported in part by the french government through the NatSIM ANR project. Contract ANR 05-MMSA-0004-01. We would like to thank the reviewers for their insightful comments and remarks. We also thank Neil Dodgson for his help in the completion of the paper. The models in Figures 1 and 12 are provided courtesy of VCG-ISTI and INRIA by the AIM@SHAPE Shape Repository.

References

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

[ADMAM03] ASSARSON U., DOUGHERTY M., MOUNIER M., AKENINE-MÖLLER T.: An optimized soft shadow volume algorithm with real-time performance. In *Proc. of Workshop on Graphics Hardware'03* (2003), ACM Press.

[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 (2006).

[AHT04] ARVO J., HIRVIKORPI M., TYYSTJÄRVI J.: Approximate soft shadows using image-space flood-fill algorithm. In *Proc. of Eurographics 2004* (2004).

[ARHM00] AGRAWALA M., RAMAMOORTHI R., HEIRICH A., MOLL L.: Efficient image-based methods for rendering soft shadows. In *Proc. of ACM SIGGRAPH 2000* (2000), ACM Press.

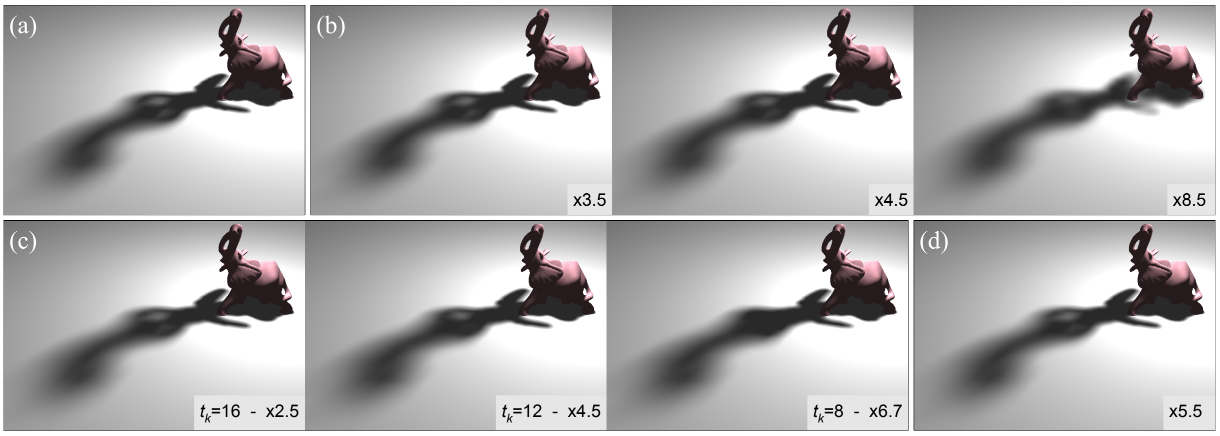

Figure 12: *(a) Soft shadows produced by our raw SSM algorithm at 31 fps for a 768×768 screen resolution. Illustration of quality versus performances for (b) our view dependent selection and (c) our light space adaptive precision. The numbers indicate the speed up of the SSM passes. These two optimizations are combined in (d) using the settings of the first pictures of (b) and (c), yielding a rate of 71 fps.*

[ASK06] ASZÓDI B., SZIRMAY-KALOS L.: Real-time soft shadows with shadow accumulation. Eurographics Short Papers, 2006.

[BCS06] BAVOIL L., CALLAHAN S., SILVA C.: *Robust Soft Shadow Mapping with Depth Peeling*. SCI Institute Technical Report UUSCI-2006-028, University of Utah, 2006.

[BS02] BRABEC S., SEIDEL H.-P.: Single sample soft shadows using depth maps. In *Proc. of Graphics Interface* (2002).

[CD03] CHAN E., DURAND F.: Rendering fake soft shadows with smoothies. In *Proc. of the 14th Eurographics workshop on Rendering* (2003), Eurographics Association, pp. 208–218.

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. In *Proc. of ACM SIGGRAPH '77* (1977), pp. 242–248.

[DF94] DRETTAKIS G., FIUME E.: A fast shadow algorithm for area light sources using backprojection. *Computer Graphics 28*, Annual Conference Series (1994), 223–230.

[DHS*05] DURAND F., HOLZSCHUCH N., SOLER C., CHAN E., SILLION F. X.: A frequency analysis of light transport. In *Proc. of ACM SIGGRAPH '05* (2005), ACM Press, pp. 1115–1126.

[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.

[FBP06] FOREST V., BARTHE L., PAULIN M.: Realistic soft shadows by penumbra-wedges blending. In *Proc. of Graphics Hardware* (2006).

[FFBG01] FERNANDO R., FERNANDEZ S., BALA K., GREENBERG D. P.: Adaptive shadow maps. In *Proc. of ACM SIGGRAPH 2001* (2001), ACM Press, pp. 387–390.

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

[GD98] GROSSMAN J. P., DALLY W. J.: Point sample rendering. In *Proc. of the 9th Eurographics Workshop on Rendering* (1998).

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *Proc. of SIGGRAPH '96* (1996).

[HBS00] HEIDRICH W., BRABEC S., SEIDEL H.-P.: Soft shadow maps for linear lights high-quality. In *Proc. of the 11th Eurographics Workshop on Rendering* (2000), pp. 269–280.

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

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. of SIGGRAPH '87* (1987), pp. 163–169.

[Len05] LENGYEL E.: Advanced stencil shadow and penumbra wedge rendering. Game Developer Comference, unpublished slides. URL: http://www.terathon.com, 2005.

[Nv0] Nvidia opengl extension specifications for the geforce 8 serie architecture. URL: http://developer.download.nvidia.com /opengl/specs/g80specs.pdf.

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. In *Proc. of SIGGRAPH '87* (1987), ACM Press, pp. 283–291.

[SA06] SEGAL M., AKELEY K.: The OpenGL Graphics system: A Specification. URL: http://www.opengl.org/registry /doc/glspec21.20061201.pdf, 2006.

[SAPP05] ST-AMOUR J.-F., PAQUETTE E., POULIN P.: Soft shadows from extended light sources with penumbra deep shadow maps. In *Proc. of Graphics interface'05* (2005).

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Proc. of ACM SIGGRAPH '02* (2002), pp. 557–562.

[SS98] SOLER C., SILLION F. X.: Fast calculation of soft shadow textures using convolution. In *Proc. of ACM SIGGRAPH '98* (1998), ACM Press, pp. 321–332.

[WH03] WYMAN C., HANSEN C.: Penumbra maps: Approximate soft shadows in real-time. In *Proc. of the 14th Eurographics workshop on Rendering* (2003), pp. 202–207.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proc. of ACM SIGGRAPH '78* (1978), pp. 270–274.

[WSP04] WIMMER M., SCHERZER D., PURGATHOFER W.: Light space perspective shadow maps. In *Proc. of the 15th EG Symposium on Rendering* (2004), Eurographics Association.